МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»

ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Представление и использование знаний в интеллектуальных автоматизированных системах»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024г.

Лабораторный практикум

по курсу «Представление и использование знаний в автоматизированных системах»

Цель практикума в том, чтобы помочь студентам овладеть навыками практического представления знаний в интеллектуальных системах. Пособие содержит материалы по разделам дисциплины «Представление и использование знаний в автоматизированных системах», которые рассматриваются на практических занятиях, и основано на образовательном контенте, используемом авторами в учебном процессе ЮФУ в ходе профессиональной подготовки магистров.

Термин «искусственный интеллект» (ИИ) появился в 1956 г. и до конца XX в. являлся модным и часто используемым и иногда не к месту. В настоящее время он используется крайне осторожно, а для некоторых является синонимом несбывшихся надежд. Это произошло, на мой взгляд, в основном из-за двух причин: первая – завышенные первоначальные ожидания (с этого начинают почти все новые открытия и идеи); вторая – ИИ является исследовательской областью компьютерной науки и был обречен на то, что все его достижения быстро становилось практическими приложениями и отчуждались в компьютерные приложения, драйверы различных устройств и опции «гаджетов».

Так, уже ставшие обыденными распознавание сотовыми телефонами голосовых команд, сканирование и распознавание текста, в том числе и рукописного, «проговаривание» текста электронными устройствами, автоматическое распознавание автомобильных номеров, машинный (подстрочный) перевод текста, нахождение лиц при автоматических режимах цифрового фотографирования, «интеллектуальные» бытовые устройства некоторое время назад являлись нерешенными задачами ИИ.

Получены и куда более серьёзные результаты, которые нашли своё приложение в советующих и экспертных системах, системах диагностики, управления, проектирования, поддержки принятия решений и т.д. Настоящее учебное пособие ориентировано не только на некоторые теоретические аспекты систем ИИ, но и на решение связанных с ними практических задач для систем, основанных на знаниях. Приведены также задания на лабораторные работы, примеры и алгоритмы решения таких задач. В их числе описаны и элементы реально работающих программных комплексов.

В качестве основных инструментов лабораторного практикума использовались средства из таблицы 1. В качестве системы контроля версий использовался git совместно с GitHub.

Таблица 1 – Используемое в данной работе ПО

Язык	Python
Пакет библиотек	Tkinter, NumPy, Mathplotlib 3.0
Среда разработки	JetBrains PyCharm 2019.2.4
Дополнительные средства	git

Python – (в русском языке распространено название пито́н) – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Tkinter – (от англ. Tk interface) – кроссплатформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows), написанная Стином Лумхольтом (Steen Lumholt) и Гвидо ван Россумом. Входит в стандартную библиотеку Python.

NumPy – библиотека с открытым исходным кодом для языка программирования Python. Возможности:

- поддержка многомерных массивов (включая матрицы);
- поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.

Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB.

Matplotlib – библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

Matplotlib написан и поддерживался в основном Джоном Хантером и распространяется на условиях BSD-подобной лицензии. Генерируемые в различных форматах изображения могут быть использованы в интерактивной графике, в научных публикациях, графическом интерфейсе пользователя, веб-приложениях, где требуется построение диаграмм. В документации автор

признаётся, что Matplotlib начинался с подражания графическим командам MATLAB, но является независимым от него проектом.

Библиотека Matplotlib построена на принципах ООП, но имеет процедурный интерфейс pylab, который предоставляет аналоги команд MATLAB.

JetBrains PyCharm – интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

JavaScript (аббр. JS) – мультипарадигменный язык программирования. Поддерживает объектноориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

Vue.JS – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

JetBrains WebStorm – интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы IntelliJ IDEA.

WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями. Важным преимуществом интегрированной среды разработки WebStorm является работа с проектами (в том числе, рефакторинг кода JavaScript, находящегося в разных файлах и папках проекта, а также вложенного в HTML). Поддерживается множественная вложенность (когда в документ на HTML вложен скрипт на Javascript, в который вложен другой код HTML, внутри которого вложен Javascript) – то есть в таких конструкциях поддерживается корректный рефакторинг.

GitHub – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

Лабораторная работа №1. Разработка и исследование «темпорального процессора» для информационно- аналитических систем

Цель работы

Приобрести навыки представления и использования темпоральных компонентов знаний в информационных и управляющих системах.

Постановка задачи

Научиться представлять темпоральные атрибуты знаний; разрабатывать алгоритмы поиска темпоральных характеристик в реляционных базах данных; разрабатывать интерфейсы пользователей для имитационных моделей; разрабатывать алгоритмы анализа темпоральных отношений для различных позиций наблюдения (наблюдателя).

Теоретическая часть

1. **Основные положения представления темпоральных характеристик процессов**

Время рассматривается исследователями в многочисленных аспектах: философском (логическом, онтологическом), естественнонаучном (физическом, психологическом) и даже этическом. Направление «течения» времени также вызывает философские споры, как и многие другие аспекты времени. Если учитывать, что все больше событий из будущего становятся достоянием настоящего и далее прошлого, то можно предположить, что время «течет» (или мы движемся) в будущее, и значит, из прошлого через настоящее. Конечно, интуитивно мы считаем неизменными положение компонентов в тройке, изменяется только содержимое каждой компоненты. Существование фатального «будущего» требует признания того, что события и процессы находятся в «будущем» в той же последовательности, в какой они произойдут в «настоящем» и будут «храниться» в прошлом. Если мы каким-либо образом «управляем» событиями и процессами в течение времени своей жизни, то проще представить будущее в виде хаотического множества событии, которых мы можем избежать, ускорить или замедлить их наступление и течение. Однако этот хаос частично упорядочен, и события и процессы «сведены» в подмножества в соответствии с возможностями, определяемыми законами бытия (биологическими, физическими, социальными...) и ограничениями (сословными, национальными, религиозными...). Каждое из этих событий и каждый из процессов может характеризоваться некоторой оценкой вероятности наступления и оценкой «удаленности» от «настоящего». Это справедливо только в том случае, если мы признаем существование будущего как временного «хранилища» всех событии и процессов.

Если «содержание» «будущего» составляют еще не произошедшие события «прошлого», то «настоящее» вообще существует благодаря объединению результатов работы органов чувств. «Настоящим» человек называет самое

близкое «прошлое». И в самом деле, то, что можно отнести к категории «сейчас», уже произошло и, естественно, относится к «прошлому». Итак, «будущее», скорее всего, не существует, а если и существует, то состоит из элементов «прошлого», «настоящее» есть тоже хоть и ближайшее, но скорее «прошлое», т.е. то, о чем можно реально рассуждать, это только о «прошлом» и его подмножествах в виде «настоящего» и «будущего». Для рассмотрения времени как свойства некоторых сущностей известны следующие подходы:

1) рассматривать время наряду с другими атрибутами без учета его характерных свойств;

2) рассматривать время «через призму» его характерных свойств.

Используя первый подход, можно считать, что есть некоторый терминальный (ни от чего не зависящий) процесс– «хранитель» времени, результатом которого является значение некоторых атрибутов-носителей текущего времени. В свою очередь, атрибуты-носители текущего времени могут быть носителями абсолютного поясного времени, носителями оперативного и относительного времени, а также носителями времени в пределах хранимых отрезков времени. Сложные технические и технологические системы содержат подсистемы «единого времени» (СЕВ), обеспечивающие синхронизацию событий и процессов. Примеры таких систем являются АСУ ТП (автоматизированные системы управления технологическими процессами), в них существуют многочисленные иерархические процессы, распределенные в пространстве и времени. Однако для обозначения времени используются обычные переменные или поля тегов. Время используется как атрибут событий или как уставное значение для контролируемых процессов. Временная шкала, используемая для представления трендов, обычно масштабируется. Но существуют приложения, в которых время играет первостепенную роль. В частности, для выявления некоторых закономерностей в сложных процессах требуется анализ временных компонентов событий.

Временные диаграммы используются в основном как средство представления синхронизации процессов чаще всего в не программируемых цифровых и аналоговых устройствах. Это преимущественно детерминированные процессы. Для анализа сложных процессов с целью выявления возможных отношений между событиями были бы полезны сведенные вместе временные тренды нескольких событий. Для этого необходимо выявить возможные отношения, которые могут существовать между временными компонентами. Такие событийные или «временные» тренды должны быть доступны как экспертному, так и автоматизированному или автоматическому анализу.

Исторические тренды представляют собой отображение изменения значений некоторого атрибута в течение отрезка времени, не включающего текущий момент времени. Текущие тренды включают в качестве последнего по времени значения атрибута его значение в настоящим момент времени. Тренды будущего представляют либо предполагаемые значения некоторого информационного атрибута в будущем, либо запланированные изменения значения управляющего

атрибута. Основными свойствами времени, порожденными представлениями человека о времени, многие авторы объявляют следующие [Домбровский]:

1) структура времени (линейная, ветвящаяся, циклическая);
2) свойства шкалы времени (непрерывность, дискретность);
3) обусловленность моментов времени (детерминизм, индетерминизм).

Некоторые авторы выделяют иные свойства времени: упорядоченность, возможность определить отношение «раньше/позже» между событиями/процессами, течение времени, универсальность времени, необратимость, не фиксированность будущего. Что же собой представляет шкала времени? Любая шкала (особенно в метрическом ее понимании) является переносом представления человека о пространстве на другие атрибуты, воспринимать которые непосредственно с помощью органов чувств человек не может.

Для иллюстрации терминов, которые человек использует при описании темпоральных компонентов, представим группы терминов (темпоральные теории) событий и процессов на рис. 2. Событие представляется только одной точкой на временной шкале– временем, когда оно произошло, а процесс–двумя: временем начала и окончания. Будем использовать следующие группы терминов: «раньше–позже», «прошлое–настоящее–будущее», «было–есть–будет». Фактически на рисунке представлены темпоральные связи между терминами и по ним можно восстановить следующие высказывания человека относительно представленных событий, например:

1. «Событие 2 (всегда) «будет» «позже» события 1»;
2. «Событие 2 «есть» в «будущем»»;
3. «Событие 2 (когда-нибудь) «будет» в «настоящем», но чуть «позже»;
4. «Событие 2 (когда-нибудь) «будет» в «прошлом»».

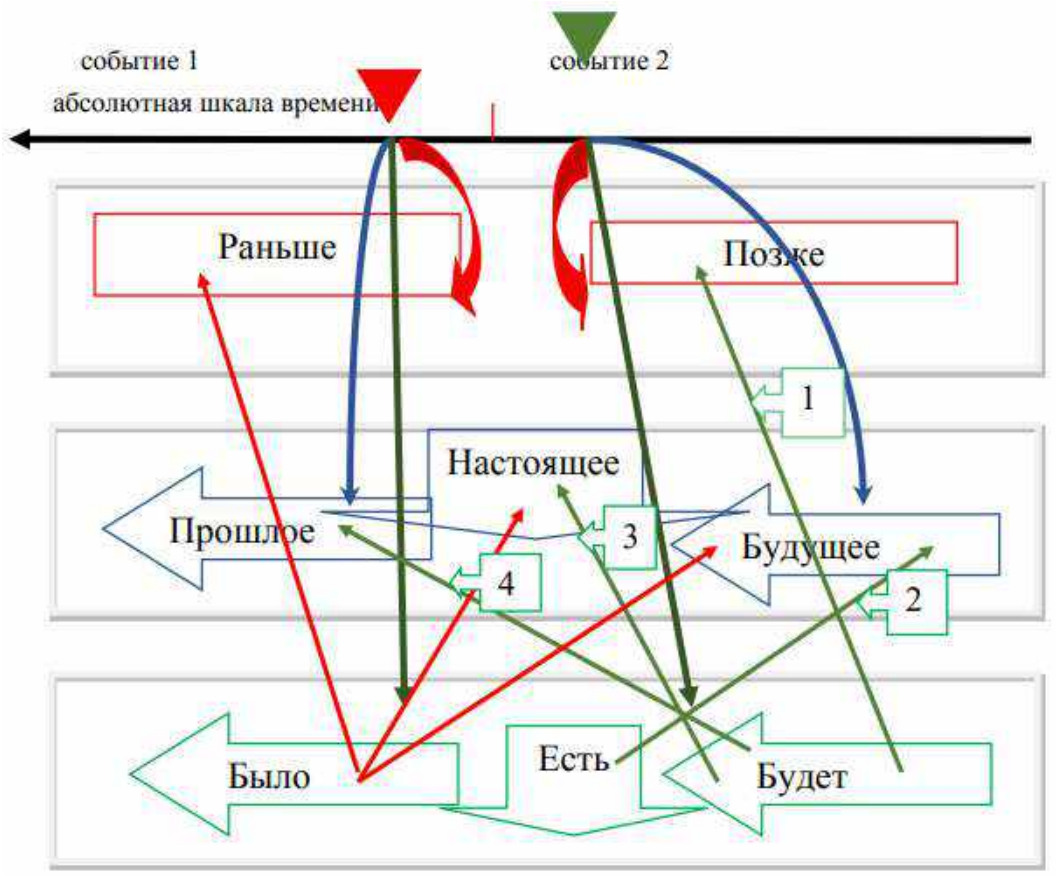


Рис.1. Позиционирование событий на временных шкалах

В случае с пространством линейная шкала – это уложенные друг за другом одинаковые (эталонные) линейные элементы самого пространства. А само измерение заключается в подсчете элементов, которые можно разместить между двумя точками. Со временем дела обстоят еще сложнее. Само время мы измеряем с помощью событий или процессов, которые обладают стабильностью по отношению к данному их свойству. Время, как таковое, может не существовать, а лишь являться некой родственной общностью свойств разнотипных сущностей. Сравнивая идентичные события или процессы, мы используем категорию времени для того, чтобы их отличить: одно из них уже произошло, другое еще происходит (длится). Поскольку существование времени само по себе считается спорным, мы будем рассматривать его как свойство или атрибут некоторого события или процесса.

Темпоральные отношения могут быть формализованы в конечное множество отношений либо представлять высказывания на естественном языке. Объектами темпоральных отношений могут быть события или процессы. Под событиями будем понимать факт или результат. Событие принято характеризовать одним отсчетом времени (темпором) – временем, когда оно произошло. Процесс принято характеризовать моментом (темпором) начала и окончания и, как следствие, длительностью.

2. Темпоральные отношения

Определение события как результата действий единичного субъекта [Кандрашина и др.] не совсем полно, поскольку исключает из рассмотрения действия двух или нескольких субъектов, объектов и систем. Событием может быть не обязательно (только) смена состояний объекта, но и интересующее нас сочетание текущих значений атрибутов, характеризующих объект. Иногда принадлежность события объекту(ам) и субъектам трудно определить, или она нас не интересует. Некоторые элементы события/процесса могут быть описаны нечеткими темпорами, т.е. такими, границы которых нечетко определены. В этом случае следует рассматривать отношения четкого и нечеткого или отношения двух или нескольких нечетких темпоров (τi, τj). Ниже приведены два типа элементарных темпоральных высказываний:

1. ERτ – между событием и темпором.
2. τiRτj – между двумя темпорами.

Для практических целей полезно использовать составные темпоральные высказывания, которые представляют совокупность элементарных темпоральных высказываний, объединенных при помощи логических связок. Рассмотрим отношения между интервальными темпорами. Точечные темпоры можно рассматривать как интервальные с совпадающими началами и окончаниями и нулевом длительностью. Все рассматриваемые отношения будут двухместными. В основе рассматриваемых отношений лежит группа интервально-временных отношений ECT [Кандрашина и др.] темпоральной логики. Темпоры можно сравнивать по длительности и по взаимному расположению. Два темпора могут находиться в следующих отношениях:

– (rts) – последовательны с паузой, если интервал первого процесса закончился раньше, чем начался интервал второго (они не одновременны, и первый интервал раньше второго);

– (rtsn) – последовательны без паузы;

– (rtes) – пересекаются;

– (rtel) – вложенные с примыканием к началу;

– (rter) – вложенные с примыканием к окончанию;

– (rte) – вложенные без примыканий;

– (rtU) – отношение несравнимости темпоров.

Будем считать, что два темпора находятся между собой в отношении rtU, если:

1) ничего не известно о процессах, которые характеризуются этими темпорами, что свидетельствовало бы о других темпоральных отношениях между этими темпорами;

2) необходимо предпринять некоторые действия, чтобы темпоральные отношения стали более определенными.

Создание темпоральной СУБД

Исследователи темпоральных баз данных рассматривали несколько способов создания темпоральной СУБД. Вначале рассматривалась возможность создания темпоральных СУБД «с нуля», то есть самостоятельная реализация некоторой

темпоральной модели. Однако реляционные СУБД быстро прогрессировали, расширялась их функциональность, поэтому создание «урезанной» темпоральной СУБД было нецелесообразным, а ресурсов и возможности повторной реализации всех средств реляционной СУБД (при создании темпоральной СУБД) просто не было. Отсутствовала и соответствующая потребность со стороны пользователей. Поэтому вскоре разработчики стали рассматривать различные способы дополнения и расширения обычных реляционных СУБД поддержкой темпоральной модели данных. Почти все такие способы сводились к созданию некоторого функционального блока, отвечающего за разбор темпоральных запросов, подмену их некоторыми реляционными вычислениями, а потом обратное преобразование в требуемое темпоральное представление для возвращения результатов пользователю. Основным отличием был уровень «вмешательства» в реляционную СУБД, а также степень интеллектуальности данного блока темпоральных преобразований (рис. 2). Сравнение различных вариантов реализации промежуточного слоя представлено, например, в [TJS98].

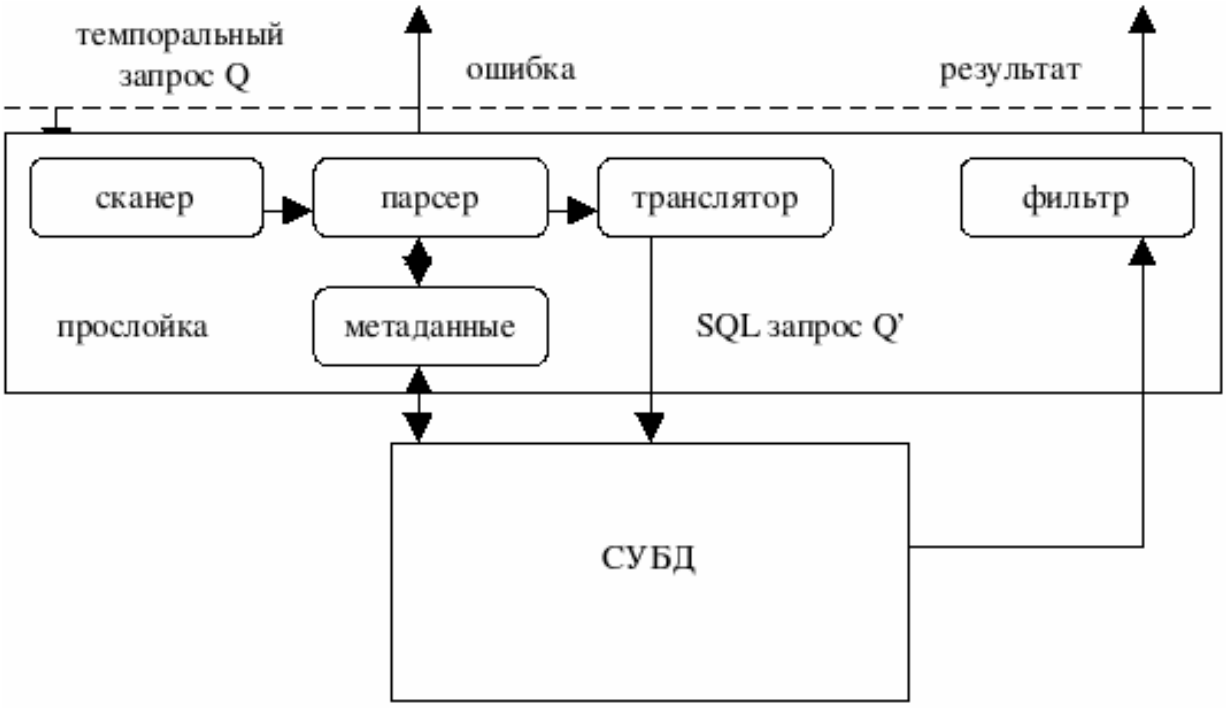


Рис. 2. Общий вид многоуровневой архитектуры темпоральной СУБД

Рассмотрим три крайних случая: преобразование на уровне ядра реляционной СУБД, преобразование на уровне пользовательского приложения и использование независимого промежуточного proxy-уровня. Первый способ предоставляет максимум возможностей по расширению синтаксиса языка баз данных, обеспечению различных проверок, а также оптимизации. Он также обеспечивает полную прозрачность для всех пользовательских приложений, а возможно, и для администратора БД. Однако он доступен только разработчикам СУБД.

В отличие от простого использования приложением реляционной СУБД для работы с темпоральными данными второй способ предполагает выделение некоторых модулей или библиотеки (в пользовательском приложении), отвечающих за преобразование именно темпоральных запросов. То есть основное приложение использует некоторую темпоральную абстракцию, а не реляционную БД в чистом виде, а потом интерпретирует результаты запросов. Подобная абстракция (пусть и на уровне приложения) позволяет сократить число ошибок и отделить бизнес-логику приложения от технической составляющей хранения данных. В [Sno99] подробно описано, как можно создавать темпоральные приложения в рамках реляционной СУБД.

Третий способ подразумевает создание между пользовательским приложением и СУБД некоторого промежуточного «черного ящика» (будем называть его proxy-уровнем), который может быть реализован в виде драйвера, сервиса, внешней библиотеки и т.п. Для пользовательского приложения этот proxy-уровень должен работать как темпоральная СУБД. С другой стороны, для СУБД этот proxy-уровень является обычным реляционным приложением. Поэтому proxy-уровень оказывается прозрачным как для пользовательского приложения, так и для СУБД, а также не требует внесения изменений в их исходный код. Исходя из того, что между proxy-уровнем и СУБД интенсивность обмена данными может оказаться на порядок выше, чем между proxy-уровнем и приложением (с учетом различных дополнительных оптимизаций), предпочтительно размещать его как можно ближе к СУБД.

Рассмотрим преимущества и недостатки представленных способов. Основным недостатком первых двух подходов является необходимость в изменении кода СУБД или приложения, и поэтому они доступны, в первую очередь, самим разработчикам. Одним из недостатков второго и третьего способа является невозможность сильно влиять на внутреннее представление и хранение информации в СУБД, оптимизацию доступа к ней и т.п. Одним из значительных преимуществ всех трех способов является возможность использовать уже существующие в СУБД модули интерпретации и обработки, лишь дополняя их разбором и преобразованием только темпоральных конструкций и элементов. Дополнительным преимуществом третьего способа можно назвать относительную независимость от конкретной СУБД, если не используются какие-либо специфические особенности и конструкции. В большинстве случаев на работоспособность proxy-уровня не влияет обновление версии СУБД, так как синтаксис SQL-запросов останется прежним.

Язык запросов к темпоральной базе данных

Для работы с темпоральными базами данных необходимо было разработать язык запросов к ним, который должен был бы стать расширением языка запросов SQL к реляционным базам данных. Рассмотрение языка запросов начнем с конструкций выборки, предложенных авторами языка TSQL2 (SQL/Temporal).

Выше говорилось о темпоральных базах данных, однако реляционная модель предполагает, что данные хранятся в таблицах, и база данных состоит из набора

таких таблиц. Поэтому уточним понятия. Во-первых, поскольку практически любая база данных содержит данные, связанные с промежутками времени, ее можно было бы назвать темпоральной. Однако, говоря о темпоральной СУБД, подразумевают, что интерпретацией подобных данных занимается сама система, и поэтому принято считать, что темпоральная база данных – это база данных, содержащая связанные со временем данные, интерпретацией которых занимается СУБД, являющаяся, таким образом, темпоральной СУБД[14]. С другой стороны, под управлением темпоральной СУБД вполне может находиться обычная реляционная база данных. Более того, для части таблиц темпоральная поддержка может быть включена, а для других таблиц – нет. Поэтому далее будем рассматривать отдельную таблицу, на время забывая о возможных ее связях с другими таблицами.

При построении модели языка запросов к темпоральной базе данных ставились следующие цели. Во-первых, при переходе к темпоральной модели желательно расширить все три компонента модели данных: структуру данных, операции и ограничения целостности. Во-вторых, любая корректная конструкция в исходном языке запросов должна остаться корректной в новом языке, и семантика этих конструкций должна остаться прежней; например, результат, возвращаемый запросом, должен быть таким же. То есть необходимо обеспечить восходящую совместимость языка запросов и базы данных. Кроме того, желательно обеспечить темпоральную восходящую совместимость: необходимо, чтобы после добавления в систему темпоральной поддержки все унаследованные конструкции продолжали работать так же, как и раньше. Из этого следует, что все существующие приложения не должны почувствовать переход от обычной реляционной СУБД к темпоральной СУБД ([BBJ+97]). Более того, последующее добавление темпоральной поддержки в какую-нибудь конкретную таблицу не должно отразиться на корректности выполнения запросов. С другой стороны, подобное добавление должно позволить использовать темпоральные запросы в новых программах, заметно облегчая работу программистам и администраторам системы.

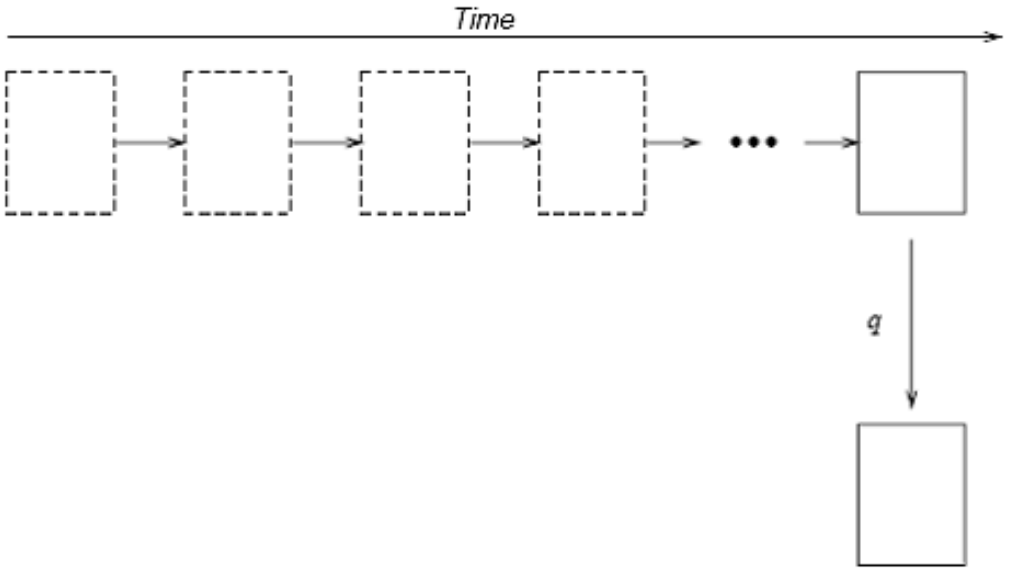


Рис. 3. Работа с таблицей без темпоральной поддержки

Рассмотрим таблицу, содержащую информацию о заработной плате сотрудников. До добавления темпоральной поддержки таблица содержала только актуальную информацию (на текущий момент времени). В таком случае у приложения имеется возможность узнать текущее значение любых элементов данных, но предыдущее значение будет потеряно после любого изменения. После удаления записи вообще невозможно будет сказать, например, сколько получал данный сотрудник перед увольнением. Такая ситуация существует при использовании традиционной реляционной модели (рис. 3). Предположим, что позже было принято решение о необходимости хранить историю значений заработной платы сотрудников. Решить это можно было бы несколькими путями. Во-первых, можно создать специальную таблицу с историей зарплаты, но это могло бы быть достаточно странным, так как уже есть таблица «заработных плат». Более того, пришлось бы модифицировать код приложения, чтобы при удалении и изменении значений происходило автоматическое обновление данных и во второй таблице. Второй способ – добавление специальных столбцов, отражающих интервал актуальности конкретной записи, но при этом также потребуется доработка приложения, и логика многих запросов может стать не слишком очевидной. Наиболее естественным было бы добавить для данной конкретной таблицы темпоральную поддержку. В этом случае все ранее разработанные запросы будут корректно работать (с текущим состоянием), но можно сформулировать новые темпоральные запросы, которые позволят узнать историю изменений (рис. 4).

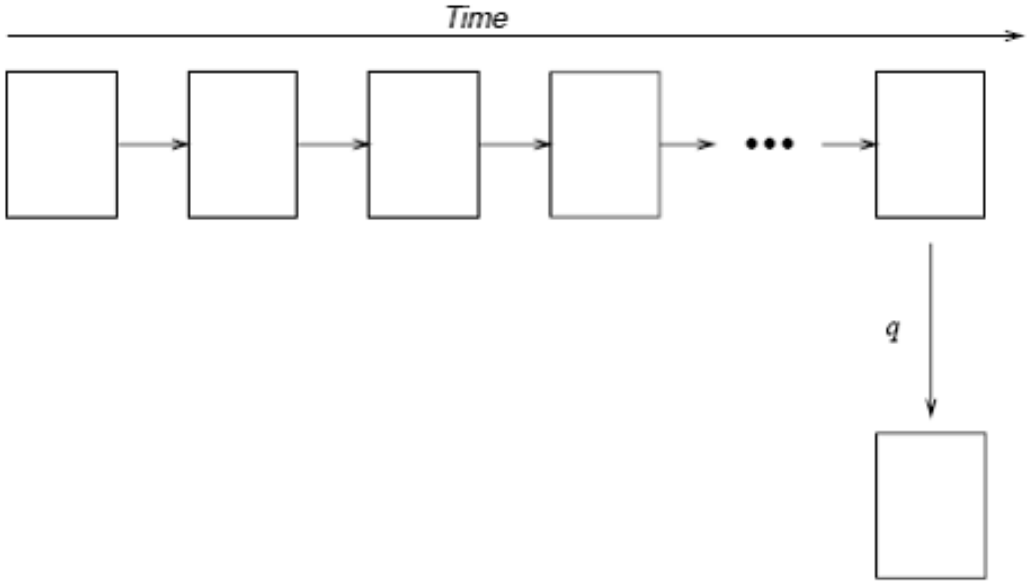


Рис. 4. Работа реляционных приложений с темпоральной таблицей

Какие же запросы могут быть сформулированы к таблице с темпоральной поддержкой? Во-первых, это запрос значений на какой-нибудь момент времени в прошлом, то есть создание среза истинности фактов на произвольную дату. Например, для обычного реляционного запроса «какую зарплату сейчас получает каждый из сотрудников?» можно легко сформулировать его темпорального двойника «какую зарплату получал каждый из сотрудников в указанную дату?» В этом случае результат запроса останется в рамках реляционного представления. С другой стороны, вполне естественным оказывается запрос «когда и какую зарплату получал каждый из сотрудников?». Здесь уже в результатах запроса появляется линия времени. Один из традиционных способов представления подобных результатов опирается на интервальное действительное время, то есть к обычному реляционному представлению результатов добавляется еще один столбец, содержащий интервалы истинности фактов. Алгоритм формирования результатов подобных запросов можно упрощенно представить следующим образом: для каждого момента времени[17] вычисляется реляционный подзапрос «какую зарплату получает каждый из сотрудников», после чего к общему результату добавляются результаты этих подзапросов с учетом интервалов истинности. Подобная семантика «последовательной» интерпретации реляционных запросов называется *последовательной*, см. рис. 5.

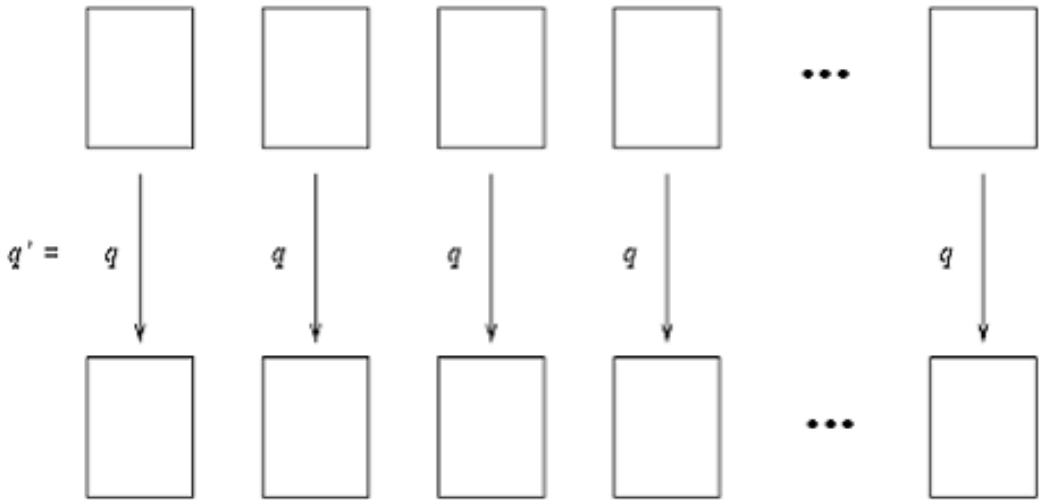


Рис. 5. Обработка "последовательных" запросов к темпоральной таблице

Однако на основе этой семантики невозможно сформировать запрос, который требует «сравнения» нескольких последовательных моментов времени. К таким запросам относится большинство запросов, включающих агрегационные функции «во времени», например, «вывести среднюю заработную плату сотрудника за все периоды времени» (отметим, что результат будет представлен обычной реляционной таблицей) или «вывести периоды, когда на оплату труда сотрудников уходила максимальная сумма». Предусмотреть все подобные типы запросов невозможно, и также нельзя ограничивать выразительные возможности языка, поэтому была предложена конструкция запросов *произвольного* доступа к темпоральным данным (*Non-Sequenced Queries*), которые предоставляют возможность самостоятельно сформулировать необходимый запрос, накладывая ограничения на системный темпоральный столбец. См. рис. 6.

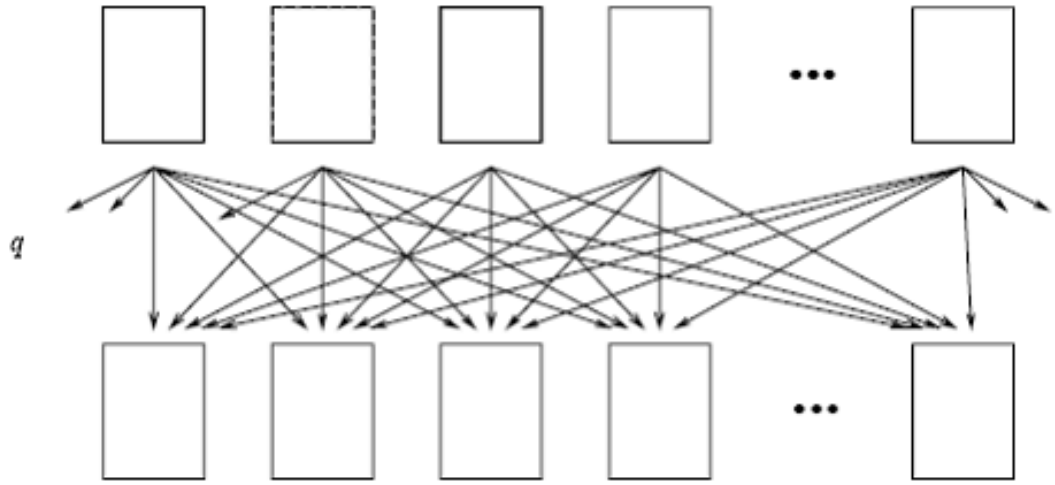


Рис. 6. Обработка "произвольных" запросов к темпоральной таблице

Таким образом, при наличии темпоральной поддержки в системе можно выделить четыре уровня использования запросов (для конкретной таблицы): 1 –

темпоральная поддержка для таблицы отсутствует, 2 – использование реляционных запросов к таблице с темпоральной поддержкой, 3 – использование последовательных запросов и 4 – использование произвольных темпоральных запросов. Подобная классификация верна как для таблиц с поддержкой действительного времени, так и для тех, для которых поддерживается транзакционное время. Поскольку эти линии времени являются ортогональными и независимыми, в системе с темпоральной поддержкой получается шестнадцать вариантов использования запросов на выборку. На основе запросов на выборку можно формулировать темпоральные ограничения целостности, которые будут проверяться при операциях обновления базы данных. Для операций модификации и удаления записей можно накладывать аналогичные ограничения в их разделах WHERE. С другой стороны, обработка запросов на обновление для транзакционного и действительного времени будет принципиально отличаться. Например, в разделе SET невозможно задавать системные поля транзакционного времени. Поэтому полностью корректная реализация поддержки транзакционного времени возможна только при реализации темпоральных операций внутри ядра СУБД.

Представление результатов темпоральных запросов

Выше уже упоминались два подхода к представлению информации в темпоральной базе данных: интервальный, когда принимается во внимание период истинности факта, и точечный, когда для каждого отдельного момента времени рассматриваются все факты, истинные в этот момент. При реализации темпоральной СУБД и проектировании языка запросов требуется выбрать один из этих вариантов представлений. При хранении данных в базе данных наиболее компактным является интервальное представление. При представлении результатов запросов также многое говорит в пользу интервального представления. Но при формулировке многих запросов удобнее сравнивать моменты времени, а не оперировать интервалами истинности отдельных фактов, поэтому точечное представление получает значительное преимущество.

Операции над множествами очень удобны для теоретических выкладок и формализации, но трудно реализуемы в системе при значительном объеме данных. Поэтому интервалы времени принято разделять на такие подинтервалы, чтобы для каждого факта они не пересекались, но дополняли друг друга до исходного интервала, а любые два интервала разных фактов либо не пересекались, либо совпадали. При таком представлении истинность фактов не будет меняться ни для одного из получившихся интервалов, что, например, позволит корректно использовать полный вложенный перебор по интервалам и фактам при получении результата запросов. После получения результата в виде набора интервалов истинности для каждого факта необходимо выполнить обратную процедуру – объединение пересекающихся или примыкающих друг к другу интервалов. Два данных преобразования называются *распаковкой* и *упаковкой*. Эти операции могут применяться и во

время промежуточных вычислений. Фактически, перед каждой операцией над темпоральными данными можно выполнять распаковку, а потом – упаковку.

Специальное значение «сейчас»

Слово «сейчас» означает «в настоящее время, в данный момент». В запросах на языке SQL довольно часто используются конструкции CURRENT_TIMESTAMP, CURRENT_DATE и CURRENT_TIME, которые заменяются соответствующими константами в момент выполнения запросов, поэтому в самой базе данных хранятся лишь конкретные значения дат и времени. Так как язык запросов к темпоральной базе данных является расширением языка SQL, данные конструкции в нем также могут использоваться как синонимы «динамических» констант. Однако для работы с фактами, которые истинны в настоящий момент, но могут меняться с течением времени, необходимо использовать дополнительное специальное значение *СЕЙЧАС*. Оно используется при задании верхней границы интервала истинности факта, хранимого в таблице с темпоральной поддержкой ([CDI+97]). Пусть в базе данных хранится информация о периоде работы сотрудника в компании. Тогда до 2 апреля для сотрудника, пришедшего на работу 1 апреля, промежутком времени его работы в компании является один день 1 апреля. На следующий день промежуток его работы в компании изменится на интервал с 1 по 2 апреля. Через день интервал примет вид 1 – 3 апреля и так далее. Одним из путей решения подобной проблемы могло бы быть ежедневное обновление базы данных, изменяющее верхнюю границу интервала на корректное значение, но это нереальный вариант. Поэтому при выполнении запросов можно использовать подобную динамическую подстановку[20]. Для этого можно хранить динамическую константу *СЕЙЧАС*, обозначающую зависимость факта от текущего времени. В приведенном примере это бы означало, что человек работает в компании с 1 апреля по *СЕЙЧАС*. В качестве представления *СЕЙЧАС* можно использовать одно из значений домена TIMESTAMP или пустое значение NULL. При выборе значения для представления *СЕЙЧАС* необходимо гарантировать, чтобы это значение не использовалось с каким-либо иным смыслом. Фактически, требуется сделать выбор из значения NULL и максимального или минимального значения типа TIMESTAMP. При использовании *СЕЙЧАС* в запросах необходимо принимать во внимание, что его следует заменять текущим временем на этапе выполнения. При работе над системой TimeDB [Tim07] исследовалась производительность системы для различных вариантов представлений. Было продемонстрировано, что представление *СЕЙЧАС* с помощью минимального значения типа TIMESTAMP всегда самое медленное. Представление в виде NULL было наиболее эффективным при большем проценте пересечений с текущим временем. Однако представление в виде NULL-значения всегда приводило к максимальному количеству чтений диска. На основе этого анализа был сделан выбор в пользу максимального значения типа TIMESTAMP.

Использование некоторого специального значения *СЕЙЧАС* не является единственно возможным решением. Например, можно разделить все факты на

два класса, в одном из которых хранятся факты, для которых интервал уже точно известен, а в другом – те, у которых верхняя граница еще не известна. Если хранить эти факты в разных таблицах, то не потребуется вводить специальное значение *СЕЙЧАС*, однако несколько усложнится выборка, а таблицы с поддержкой и транзакционного, и действительного времени придется разбивать на четыре подтаблицы.

Выше речь шла о представлении значения *СЕЙЧАС* в системе, однако важна и его интерпретация. Например, тот факт, что сотрудник работает с 1 апреля по *СЕЙЧАС*, не подтверждает, но и не опровергает, что на следующей неделе он также будет работать. Но для другого сотрудника может быть известно, что он работает только до конца этой недели, и поэтому в отчетах за данную неделю о сотрудниках, которые будут работать на следующей неделе, второй сотрудник фигурировать не должен, а про первого сказать ничего нельзя. Поэтому корректная интерпретация значения *СЕЙЧАС* возможна лишь для прошлого. При работе с событиями будущего необходимо проявлять осторожность и точно понимать, что и каким образом должно быть получено в качестве результата задаваемого запроса.

Разрежение таблиц с темпоральной поддержкой

Если говорить о таблицах с темпоральной поддержкой, то причиной снижения производительности может стать постоянно возрастающий объем хранимых данных. Это, в первую очередь, относится к таблицам с темпоральной поддержкой транзакционного времени, так как в таких таблицах данные физически не удаляются из системы, а лишь добавляются (в том числе и при модификации записей). С другой стороны, через определенный промежуток времени оказывается, что часть данных устарела, и поэтому их было бы желательно физически удалить из системы, так как они не только занимают место, но и снижают производительность при выборках. Но подобное физическое удаление подвергает риску общую целостность хранящихся данных. Поэтому фундаментальным требованием является возможность контролирования физическое удаление данных, что приводит к операции, которая называется *разрежением* (*vacuuming*) таблиц с темпоральной поддержкой. Для обеспечения корректности выполнения разрежения необходимо иметь возможность указывать в запросе данные, которые должны быть удалены, и те, которые должны быть оставлены. Очевидно, что все «активные» строки должны остаться в таблице при любом ее разрежении. В простейшем случае отбрасываются все те кортежи, которые были удалены до определенного момента времени – происходит срез истории.

Но подобное разрежение является не совсем корректной операцией. Например, пусть мы хотим сделать выборку состояния на какой-нибудь момент времени *A*, предшествующий аргументу среза *B* (*A* < *B*), когда в таблице находились некоторые строки *X* и *Y*, первая из которых была удалена до момента времени *B*, а вторая – нет. Тогда в качестве результата запроса будет выдана строка *Y*, но

не *X*. Многие запросы могут начать неправильно работать, и к ним, скорее всего, будут относиться все запросы, использующие агрегатные функции.
Чтобы обезопасить себя от части подобных проблем, можно предложить специальный вариант среза, когда нижняя граница интервала транзакционного времени для всех строк становится равной моменту времени среза, если этот момент принадлежал исходному интервалу. Но и в этом случае искажается информация о времени истинного появления строки в системе. Поэтому разрежение темпоральных таблиц не является однозначным решением, и в каждом конкретном случае необходимо отдельно рассматривать все плюсы и минусы.
Отметим, что с точки зрения работы с системой и формулировки запросов нет необходимости в разрежении, однако переполнение базы данных «устаревшими» сведениями может негативно сказываться на производительности. От негативных последствий проблемы зависимости поведения приложений при проведении разрежения можно частично избавиться, если вместо реального удаления данных выполнить их перенос на другой доступный носитель. Например, данные из одной базы переносятся в другую, расположенную на другом сервере, причем в исходной базе остается информация об этом переносе. В этом случае при необходимости обращения к истории запрос может быть перенаправлен или объединен с результатом выборки из другой базы данных.
Проблема с «разрежением» таблиц относится к транзакционному времени, но она также рассматривается и с точки зрения действительного времени. Например, если речь идет о хранении некоторых промежуточных версий какого-либо документа, то может оказаться, что интерес представляет лишь окончательная версия за январь прошлого года, а все промежуточные версии можно удалить. В этом случае можно также воспользоваться разрежением, но выбор доступных методов гораздо шире, например, можно оставить каждый третий документ или не больше одного документа в месяц, а само разрежение применить к документам прошлого года.

Проектирование темпоральных баз данных

В настоящее время существуют проверенные методы и инструментарий, используемые при проектировании реляционных баз данных. При работе с темпоральными базами данных необходимо учитывать темпоральную специфику и использовать соответствующие средства. Рассмотрим простой вопрос: сколько в базе данных должно быть таблиц с темпоральной поддержкой? Если для действительного времени ответ на данный вопрос достаточно прост (количество таблиц определяется потребностью приложений), то при наличии транзакционного времени он становится нетривиальным, так как от ответа от него зависит объем накапливаемой истории. При этом не всегда требуется именно поддержка транзакционного времени, а вполне может подойти и простой журнал, реализованный вне базы данных. При проектировании таблиц с поддержкой действительного времени можно использовать обычные

инструменты, дополняя таблицы интервалами (парой значений) времени, однако в этом случае количество связей между таблицами заметно увеличивается. Более того, невозможно корректно описать работу со специальным значением *СЕЙЧАС*. При проектировании темпоральной базы данных следует также помнить, что все «обычные» реляционные ключи таблиц будут неявно расширяться верхней границей интервала транзакционного и/или действительного времени. Ограничения целостности также должны формулироваться с учетом этих обстоятельств. Если темпоральная поддержка используется не для всех таблиц, то возникает вопрос, связанный с выборкой данных из нескольких таблиц на какой-либо момент прошлого: что делать с совместными выборками из таблицы с поддержкой транзакционного времени и из таблицы без подобной поддержки? Вполне возможно, что результат выборки будет неверен, так как информация из обычной таблицы могла быть уже удалена или изменена, а в соответствующей темпоральной таблице остались внешние связи. Таким образом, если разрешить подобные выборки, то могут возвращаться некорректные результаты. Если же такие выборки запретить, то становится непонятно, как работать с потенциально константными данными, например, названиями дней недели, которые могут храниться в таблице без темпоральной поддержки. Принудительное добавление темпоральной поддержки для всех таблиц также не является лучшим выходом, так как усложняет алгоритм работы СУБД и требует дополнительных ресурсов.

ACID-свойства темпоральных транзакций

Основными свойствами традиционных транзакций, поддерживаемыми в реляционных СУБД, являются ACID (атомарность, целостность, изолированность и продолжительность). При создании темпоральной СУБД чрезвычайно важно перенести подобные свойства и на темпоральные транзакции. ACID-свойства темпоральных SQL-транзакций могут быть сохранены за счет отображения каждой темпоральной транзакции в одиночную реляционную SQL транзакцию. Альтернативные реализации темпоральных транзакций, при которых proxy-уровень отображает темпоральную SQL-транзакцию в несколько обычных SQL-транзакций, могут привести к неразрешимым проблемам, если во время выполнения получится так, что одна из реляционных SQL-транзакций будет зафиксирована, а вторая – нет, что должно означать неудачу всей темпоральной SQL-транзакции, а не отдельной ее части. Выполнить же откат транзакции, зафиксированной в базе данных, может оказаться просто невозможно, так как изменения уже могут быть использованы параллельно работающими транзакциями. Поэтому для поддержки параллельно работающих темпоральных транзакций необходимо реализовывать их в виде одиночных SQL-транзакций, иначе не удастся обеспечить поддержку ACID-свойств.

Кроме этого, недостаточно требовать только то, чтобы каждая темпоральная SQL транзакция отображалась в одну SQL-транзакцию. Для каждой транзакции необходимо обеспечить, чтобы в ее пределах транзакционное время было

константно, но тогда встает вопрос, как его выбирать, ведь условия на специальное значение *СЕЙЧАС* должны проверяться с тем же значением константы. Здесь возможны два основных варианта: транзакционное время выбирается в самом начале или в самом конце выполнения транзакции. Первый вариант являеется некорректным, так как параллельная транзакция может внести изменения в таблицы, которые еще только понадобятся данной транзакции. Поэтому наиболее удачным выбором является момент времени непосредственно перед фиксацией транзакции после того, как получены блокировки на все затрагиваемые объекты. Но и здесь возможны проблемы, если транзакция получит одно значение транзакционного времени, а будет зафиксирована чуть позже, так что между этими двумя моментами будет благополучно выполнена некорректная выборка. Подобная проблема может возникнуть и при параллельном выполнении двух различных транзакций.
Один из способов, который сможет гарантировать корректную поддержку ACID-свойств темпоральных транзакций, состоит в реализации дополнительного механизма блокировок на proxy-уровне. Отметим, что восстановление базы данных, которое является важной частью работы СУБД, является прозрачным для конечного пользователя. При использовании многоуровневой архитектуры для конечного пользователя промежуточный слой ничем не отличается от СУБД, поэтому при проектировании прослойки можно полностью положиться на механизмы восстановления, реализованные в СУБД. Также нет причин, по которым они стали бы работать быстрее или медленнее.

Эффективность при работе с темпоральной СУБД
Одним из наиболее важных вопросов при использовании баз данных является эффективность работы приложений с соответствующей СУБД. Для повышения скорости обработки запросов в реляционных СУБД традиционно применяются индексы для выборки данных из таблиц, а также статистики и различные эвристики при выборе алгоритма соединения данных из нескольких таблиц. При разработке темпоральных СУБД значительное внимание уделялось именно вопросам производительности, так как в общем случае небольшая доля «активных» данных из постоянно растущего объема информации в базе данных приводила к резкому спаду производительности при интенсивном использовании и/или недостаточном внимании при разработке приложения.
Наиболее очевидным, а также доступным способом оптимизации темпоральных приложений является использование индексов. Например, при добавлении дополнительных специальных столбцов для интервалов транзакционного времени необходимо включить верхний предел во все уникальные индексы. С другой стороны, необходимость постоянно разделять данные на «активные» и «устаревшие» требует их разделения на ранней стадии обработки, что также может быть обеспечено с помощью индексов. Аналогичная ситуация возникает при соединении темпоральных таблиц, так как оно часто проводится именно по специальным темпоральным столбцам.

Однако встает вопрос: должны ли темпоральные столбцы располагаться до или после остальных столбцов (обычного реляционного) индекса? Если они будут идти после всех обычных столбцов, то система получится почти «реляционной», так как большинство операций вначале будет выполняться именно на основе реляционных условий, а лишь затем будет применяться ограничение по времени, а значит, все проблемы с огромным объемом информации ложатся на традиционное реляционное ядро СУБД. Если же темпоральные столбцы будут располагаться перед обычными, то объем обрабатываемых данных будет напрямую зависеть лишь от наличия ограничений на темпоральные столбцы; например, при попытке получить историю отдельного кортежа при отсутствии темпоральных ограничений системе придется просмотреть историю всех кортежей (точнее, полностью весь индекс). Кроме того, подобный индекс будет довольно часто перестраиваться при изменениях данных темпоральных столбцов. С другой стороны, единственным изменением кортежа в таблице с темпоральной поддержкой транзакционного времени является установка вместо *СЕЙЧАС* точной верхней границы интервала (в момент удаления или изменения кортежа).

Описанные выше проблемы касались выборок из одной темпоральной таблицы. Наибольший же интерес представляют выборки из нескольких таблиц и соответствующее соединение результатов. До сих пор разрабатываются различные алгоритмы и предлагаются эвристики для решения задачи эффективного соединения нескольких таблиц в традиционной реляционной СУБД. Частично они позволяют решить проблему производительности при соединении таблиц, но вся оптимизация возлагается на СУБД. Одновременно с этим следует помнить, что данные хранятся в интервальном представлении, поэтому соединение будет либо строиться на нестрогих неравенствах, либо алгоритм будет выполняться темпоральным ядром СУБД с учетом упаковки и распаковки темпоральных интервалов. Отсюда следует, что доступным путем является оптимизация преобразований различных темпоральных запросов в реляционные запросы и обратно с учетом построенных индексов. С другой стороны, в большинстве случаев разработчики приложений не смогут получить значительный выигрыш в эффективности, если будут сами формулировать реляционные запросы, а не пользоваться автоматическими преобразованиями, но первый путь сложнее и чреват появлением ошибок.

Таким образом, у создателей прототипов темпоральных СУБД не было широкого выбора методов оптимизации, а единственным действенным способом являлось расширение индексов или использование промежуточных алгоритмов, но за счет увеличения потока данных между proxy-уровнем и реляционной СУБД. При этом исследователи предлагали различные расширения стандартных подходов к индексированию реляционных баз данных и хранению данных.

Алгоритм программы

Алгоритм программы можно разбить на несколько частей:

1 Генерация события (начальных и конечных точек события);

2 Определение отношения между событиями;

3 Определение позиции курсора на экране, с дальнейшей проработкой прошло событие или нет.

1.1 Генерация события (начальных и конечных точек события)

Генерация события происходит следующим образом

1 Берется промежуток одного месяца;

2 В промежутке выбираются даты конца и начала события по нормальному закону распределения от начального дня месяца до конечного, при этом начало событие всегда меньше, чем конец события;

3 Далее 2 даты формируют кортеж, который представляет собой событие.

Определение отношения между событиями

Определение отношения между событиями происходит с помощью продукционных правил на примере “если события последовательны без паузы, то такие события относятся к классу rtsn”. Все возможные варианты отношений событий описаны в пункте Теоретическая часть. Блок схема представлена на рисунке 2, где y1 событие 1, y2 событие 2, [0] – начальная точка, [1] – точка окончания события

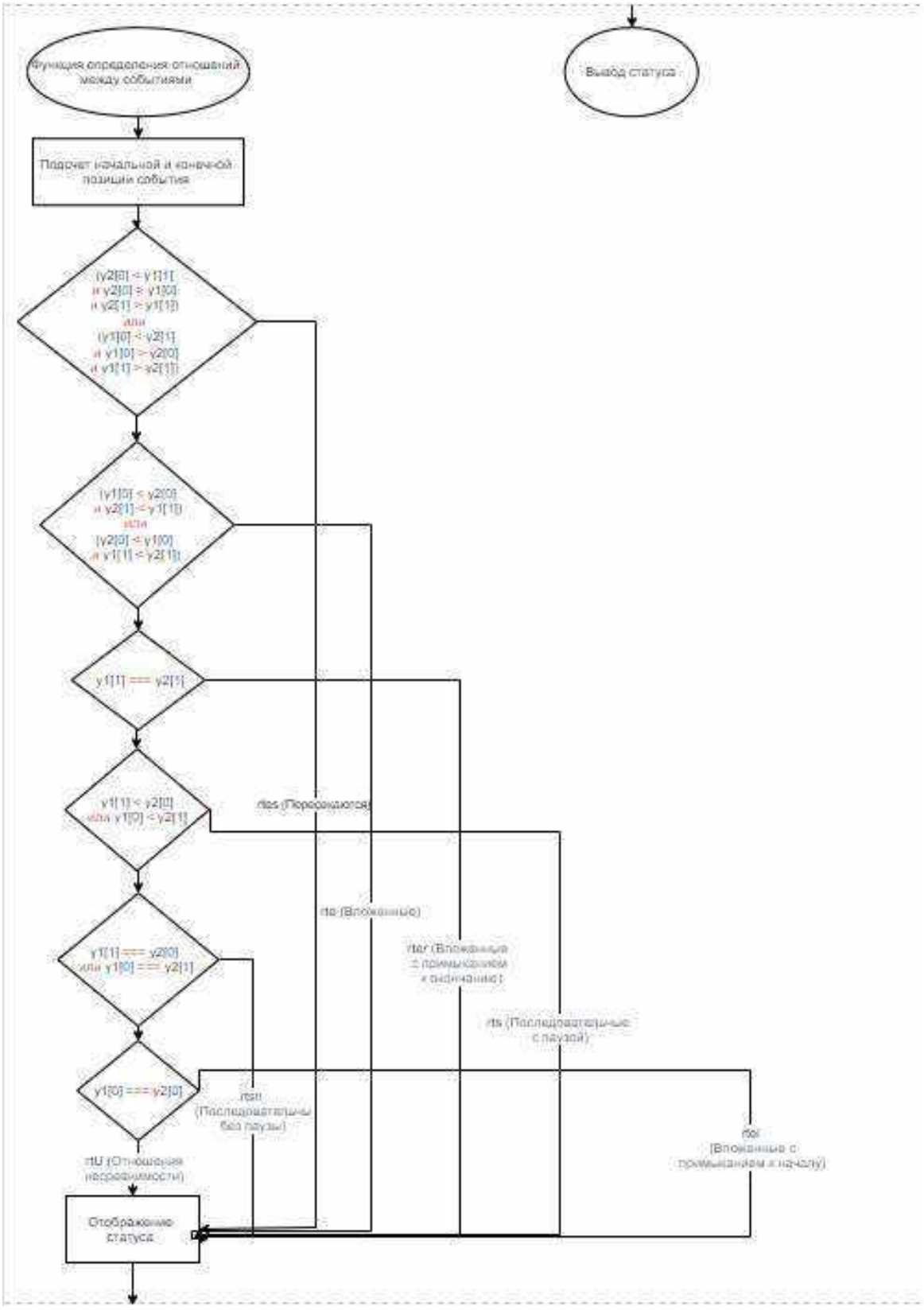


Рис. 7. Блок схема выбора типа для события

Определение позиции курсора на экране, с дальнейшей проработкой прошло событие или нет

Если растягивать на весь экран нашу машину времени, то размер событийной машины будет меняться и будет очень сложно предсказать расстояние от

пикселя до пикселя, которое будет ответственно за 1 день. В связи с этим было решено дать абсолютное значение ширины для событийной машины. И посчитать количество пикселей ответственное за 1 день. Итого получим следующую формулу:
Кол-во пикселей за 1 день = (Абсолютное значение ширины – Ширина названий событий) / Количество дней в месяце

Экранные формы пользовательского интерфейса
Основная форма разработанного приложения представляет собой веб-страницу браузера, представленную на рис. 8.

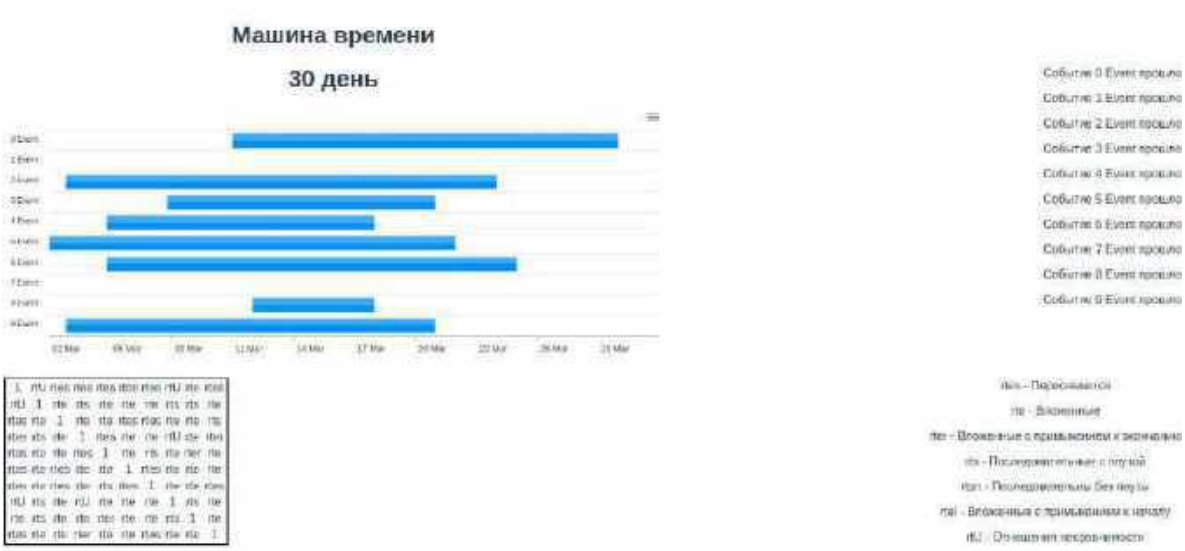


Рис. 8. Основная экранная форма разработанного приложения

Пройдемся по основным элементам данной экранной формы:

1) Окно отображения взаимных парных темпоральных отношений процессов (см. рис. 9):

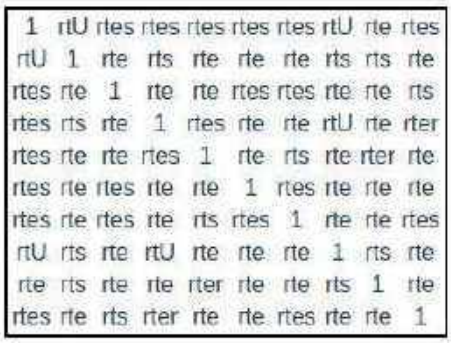


Рис. 9. Окно отображения взаимных парных темпоральных отношений процессов

2) Комментарии с расшифровкой обозначений темпоральных отношений (см. рис. 10):

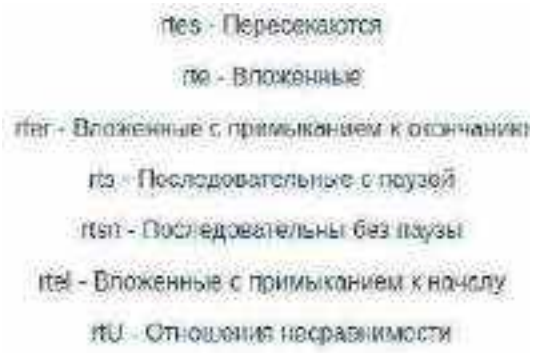


Рис. 10. Комментарии с расшифровкой обозначений темпоральных отношений

3) Лог событий (см. рис. 11):

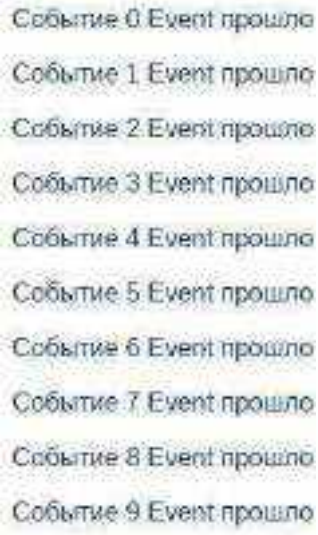


Рис. 11. Лог событий

4) Временная шкала (Окно отображения процессов в реальном масштабе времени) (см. рис. 12):

Рис. 12. Временная шкала

Лабораторная работа №2. Разработка и исследование редактора функций принадлежности лингвистических переменных для представления экспертных знаний в информационно-аналитических системах

Цель работы

Приобрести навыки представления и использования нечетких атрибутов как компонентов знаний в информационных и управляющих системах.

Постановка задачи

1) Научиться представлять нечетко определенные атрибуты знаний;
2) Разработать алгоритмы создания и редактирования компонентов лингвистических переменных в базах знаний;
3) Научиться разрабатывать интерфейсы пользователей для редактора лингвистических переменных;
4) Разработать форматы представления значений лингвистических переменных.

Теоретическая часть

1. Представление атрибутов в виде лингвистических переменных

Нечеткие атрибуты формализуются с помощью нечетких переменных. Нечеткой переменной называется тройка объектов $< \alpha, A, \tilde{C}(\alpha) >$, где α – имя некоторой переменной, А – базовое множество или область определения нечеткой переменной, $\tilde{C}(\alpha) = \{\mu C(\alpha)/\alpha\}$ – нечеткое подмножество множества А, описывает область значений нечеткой переменной, $\mu(\alpha) \in [0; 1]$ – значения функции принадлежности. То есть, если А-некоторое дискретное числовое множество, $\tilde{C}(\alpha) = \{\mu C(\alpha)/\alpha\}$ фактически представляет собой множество точек на плоскости с координатами (αì,μì(αì). А для случая если нечеткое множество определено на непрерывном числовом множестве, это множество точек представляет некоторую функцию μì(αì).

Для некоторых атрибутов предметной области могут потребоваться описания лингвистических значений. Лингвистические значения атрибутов формализуются с помощью лингвистических переменных. Лингвистической переменной называется тройка $< \beta, A, T(\beta) >$, гдеβ – имя лингвистической переменной, А – базовое множество или область определения лингвистической переменной, $T(\beta$ – терм-множество лингвистической переменной β, $T(\beta) = \{< \gamma i$, $\tilde{C}\ i\ (\alpha) >\}$, γi – имя терма,$\tilde{C}\ i\ (\alpha) = \{\mu Ci\ (\alpha)/\alpha\}$ – нечеткое множество, описывающее область значений терма γi.

Интерпретацию понятия нечеткого множества $\tilde{C}\ i\ (\alpha) = \{\mu Ci\ (\alpha)/\alpha\}$, используемого при задании нечетких и лингвистических переменных, необходимо давать исходя из процессов, лежащих в основе формирования значений описываемых атрибутов. Существует несколько способов построения функций принадлежности по результатам экспертного опроса. Лингвистическая переменная задаётся на предметной шкале посредством определения функций принадлежности для всех термов. Преобразование числового значения

некоторой (чаще всего физической) величины в некоторое векторное значение, отражающее отношение эксперта или группы экспертов к величине этого значения в рамках конкретной решаемой задачи. Например, одно и то же значение скорости для разных автомобилей и для разной дорожной обстановки может оказаться и «большим», и «маленьким». Использованные термины «большой» и «маленький» представляют результаты измерений на некоторой не метрической шкале. Эти термины будем называть термами. Совокупность всех термов принятых для нашего «словесного измерения» будем называть терм-множеством или множеством термов.

После того как мы определили, что конкретное значение скорости является «большим» («и» указывает ударение) или «маленьким», возникает естественный вопрос, насколько «большой» или «маленькой» является эта величина.

Если все эксперты утверждают, что значение является «большим», то будем считать, что некая величина, определяющая соответствие величины терму, будет равна 1. Эту величину можно называть достоверностью. А определить конкретное её значение можно, разделив, например, число экспертов, согласных с утверждением «значение величины «X» является «большим» для решаемой задачи», на общее число участвующих в опросе экспертов. В данном случае используемую методику можно назвать методом голосования. В противоположном случае если ни один из экспертов не высказался в пользу того, что значение величины является «большим», используя уже известную процедуру, получим достоверность высказывания о том, что выбранное значение скорости является «большим», равна 0. Предметная шкала, как правило, является осью абсцисс. Отрезок предметной шкалы с ненулевыми значениями функции принадлежности называется областью определения выбранного терма. При построении функций принадлежности лингвистической переменной могут возникнуть некоторые ошибки, связанные с неправильной постановкой вопросов для экспертов, с самой организацией опроса или с обработкой экспертных данных. Эти ошибки в дальнейшем могут привести к неоднозначности или противоречивости получаемых решений, а также невозможности получения таковых.

2. Требования к виду функций принадлежности лингвистической переменной

1) **Требование к упорядоченности термов** означает, что функции принадлежности на предметной шкале должны размещаться в последовательности, в которой понимают эксперты возрастание значения представляемой величины. Например, «очень малое», «малое», «среднее», «большое», «очень большое». Методика проверки выполнения этого требования в автоматическом режиме сложна и в данной работе не приводится.
2) **Требование к виду «крайних» функций принадлежности лингвистической переменной или требование к значениям функций принадлежности для граничных точек области определения ЛП:**

$$\mu t1\ (x = 0) = \mu tmax(x = xmax) = 1$$

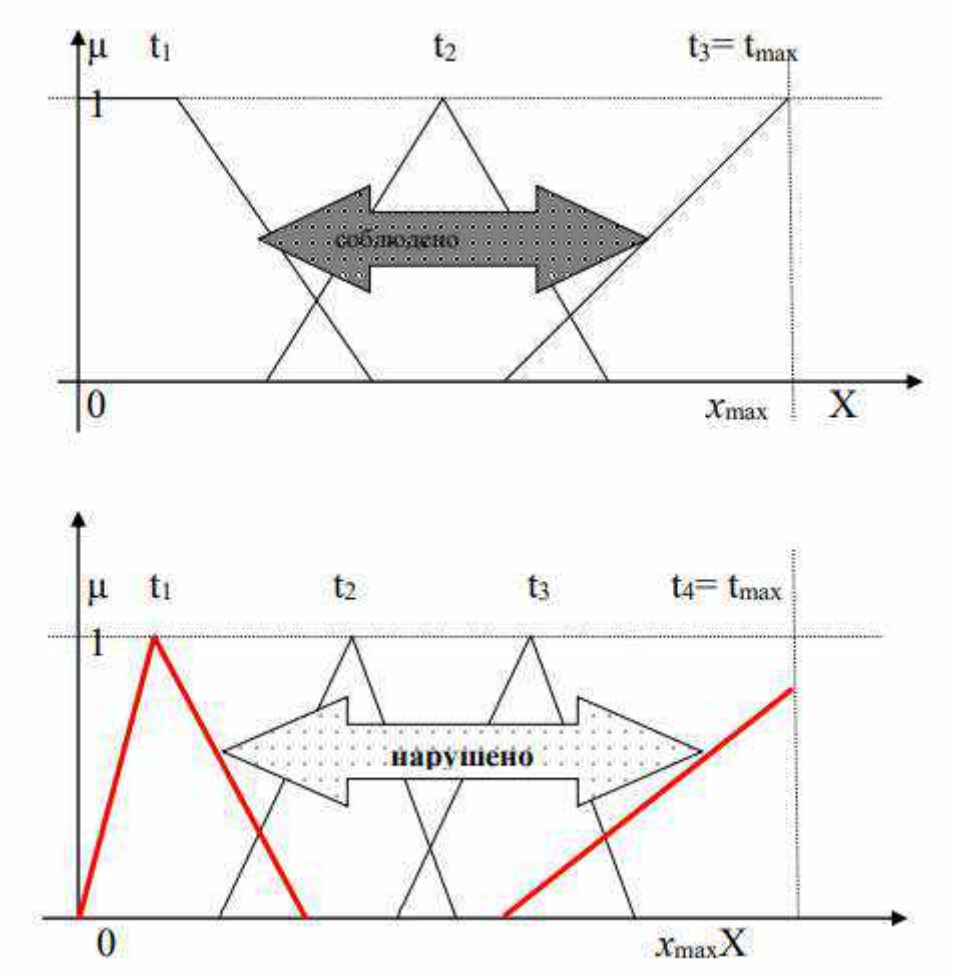


Рис. 13. Вид крайних функций принадлежности

3) **Требование к полноте покрытия предметной области областями определения функций принадлежности лингвистической переменной:**

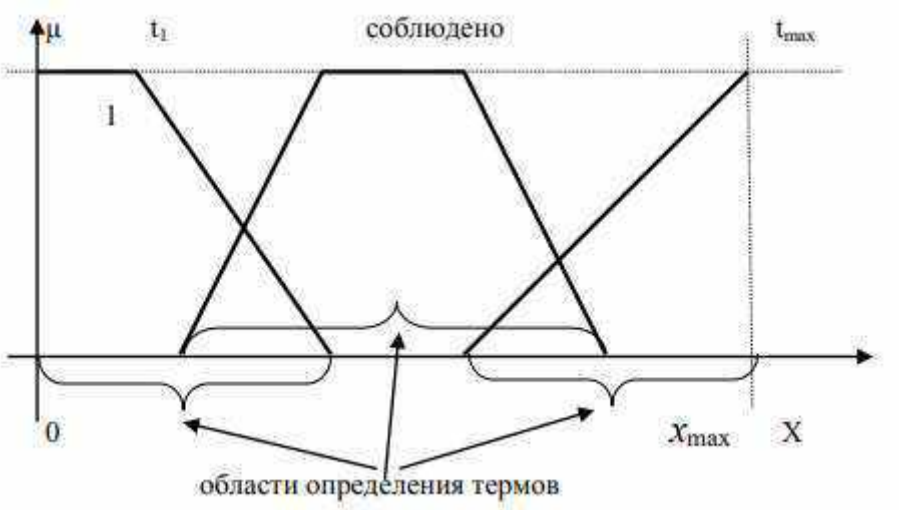


Рис. 14. Полнота покрытия предметной области

Это требование означает, что для любой точки предметной шкалы X должен существовать хотя бы один терм, значение функции принадлежности которого будет отлично от нуля: xiX, tj|tj (xi) > 0,под tj будем понимать не имя терма, а функцию его принадлежности.

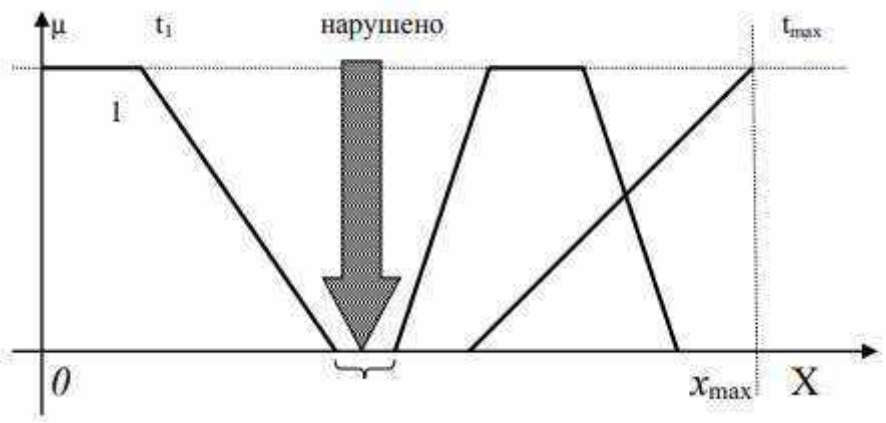


Рис. 15. Нарушение требования полноты

4) **Требование к разграничению понятий, описанных функциями принадлежности термов лингвистической переменной,** означает, что ни какие два терма не должны иметь единичные значения функций принадлежности ни для какого значения предметной шкалы:

$$t_i,t_j \mid x_k X|t_i(x_k) = t_j(x_k) = 1$$

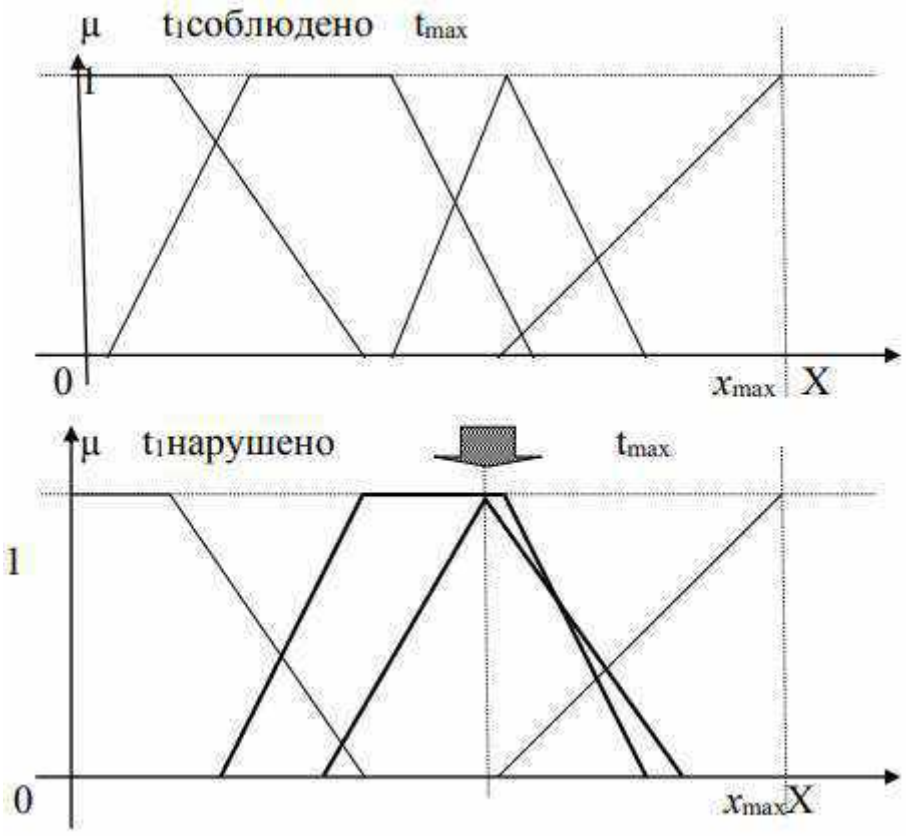


Рис. 16. Разграничение понятий

5) **Наличие типового элемента**. Для каждого терма должно существовать хотя бы одно значение на предметной шкале со значением достоверности, равным единице:

$$\text{tixkX|ti (xk)} = 1.$$

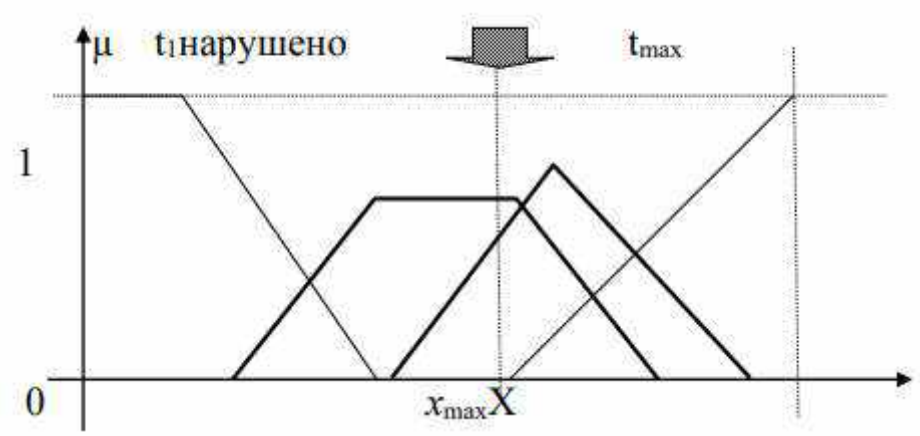


Рис. 17. Наличие типового элемента

6) **Ограничение предметной шкалы**. Такое требование вводится потому, что многие алгоритмы на лингвистических переменных основаны на вычислении площадей под модифицированными функциями принадлежности. На всех предыдущих рисунках показано ограничение предметной шкалы *xmax*.

Алгоритм программы

Алгоритм отрисовки представления нечеткого терма представлен ниже на рис. 18.

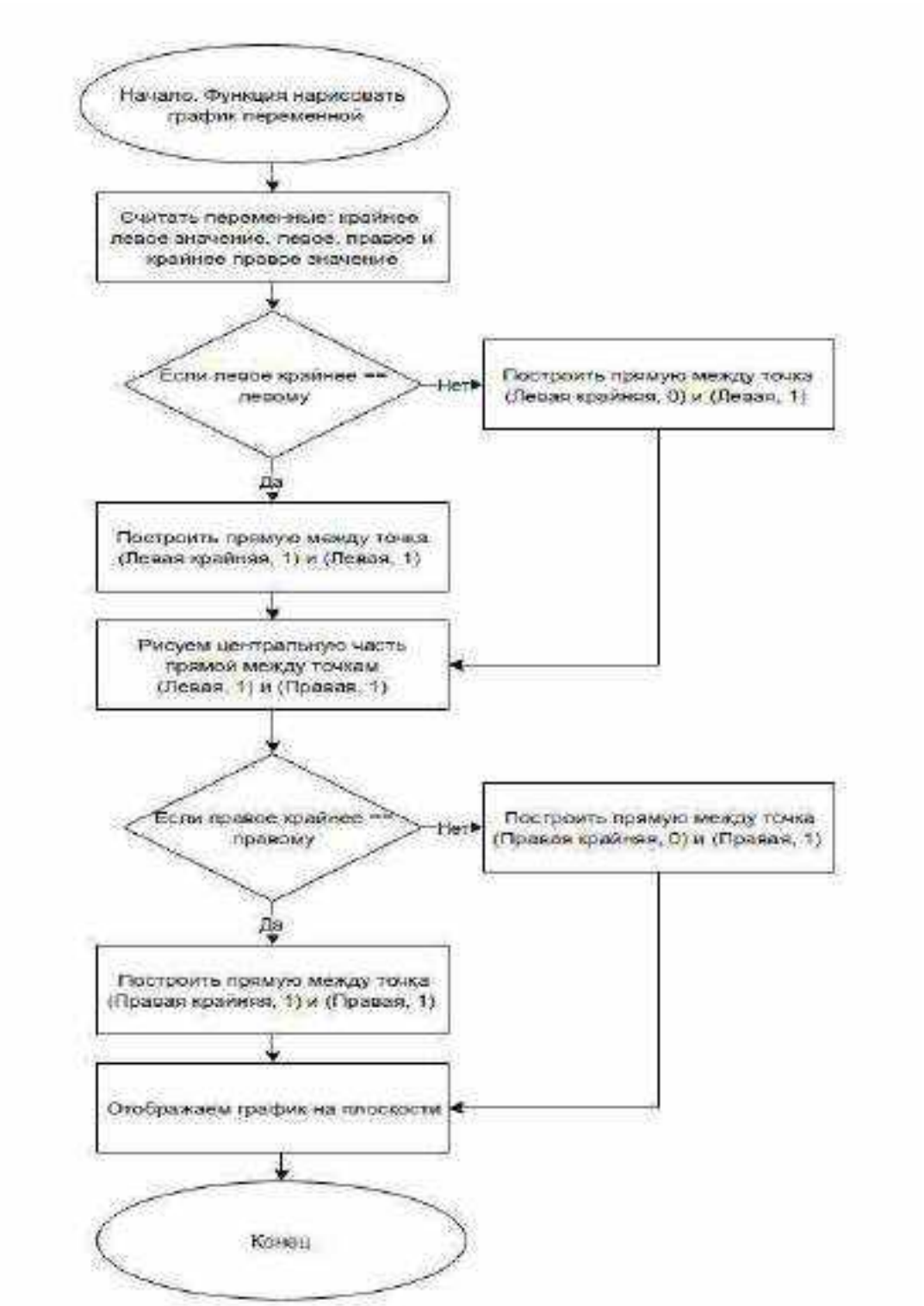


Рис. 18. Алгоритм представления нечеткой терма

Алгоритм проверки нечеткой переменной представлен ниже на рисунке 19.

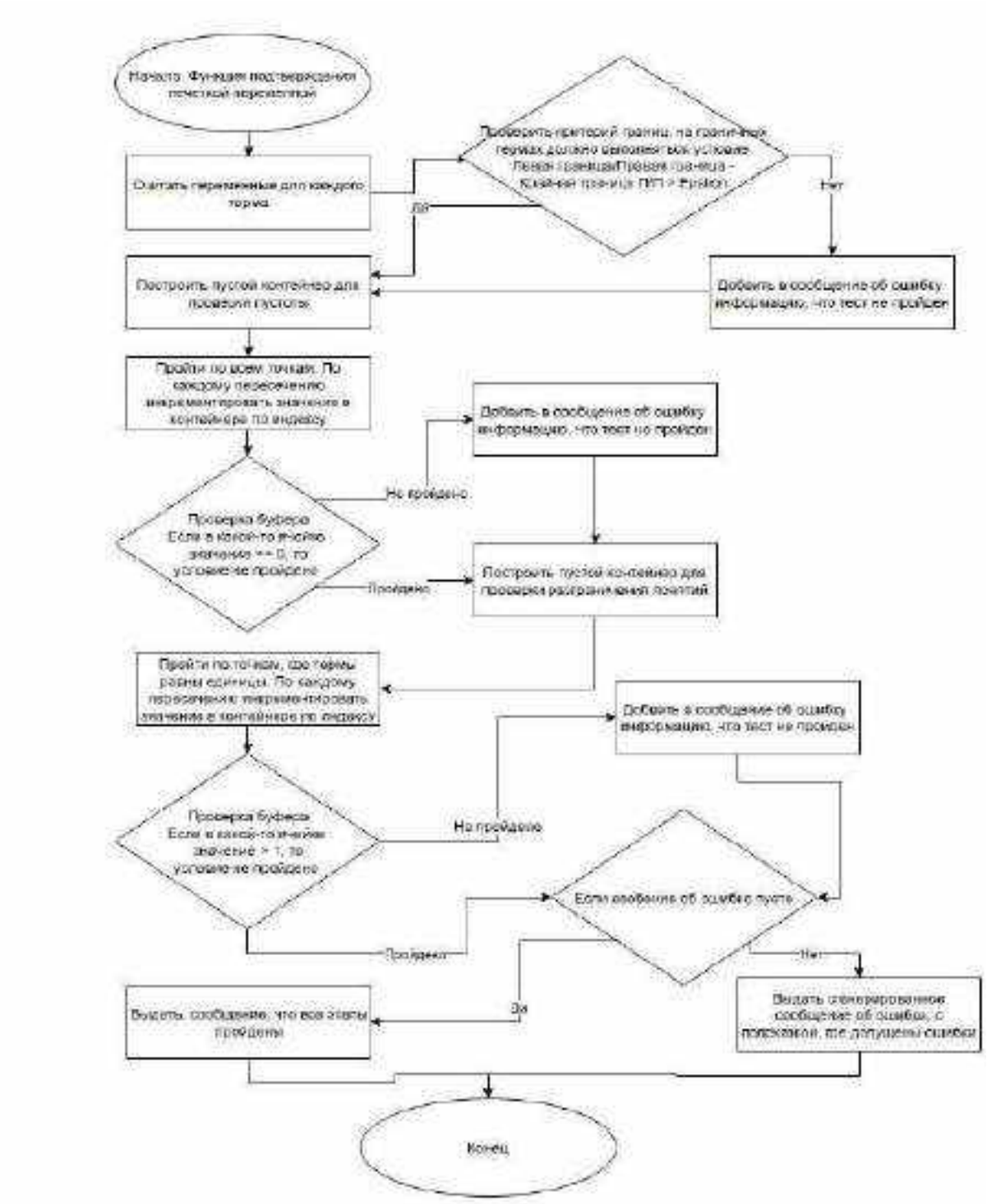


Рис. 19. Алгоритм проверки нечеткой переменной

Алгоритм общей генерации представлен ниже на рисунке 20.

Рис. 20. Алгоритм общей генерации

1. Экранные формы пользовательского интерфейса

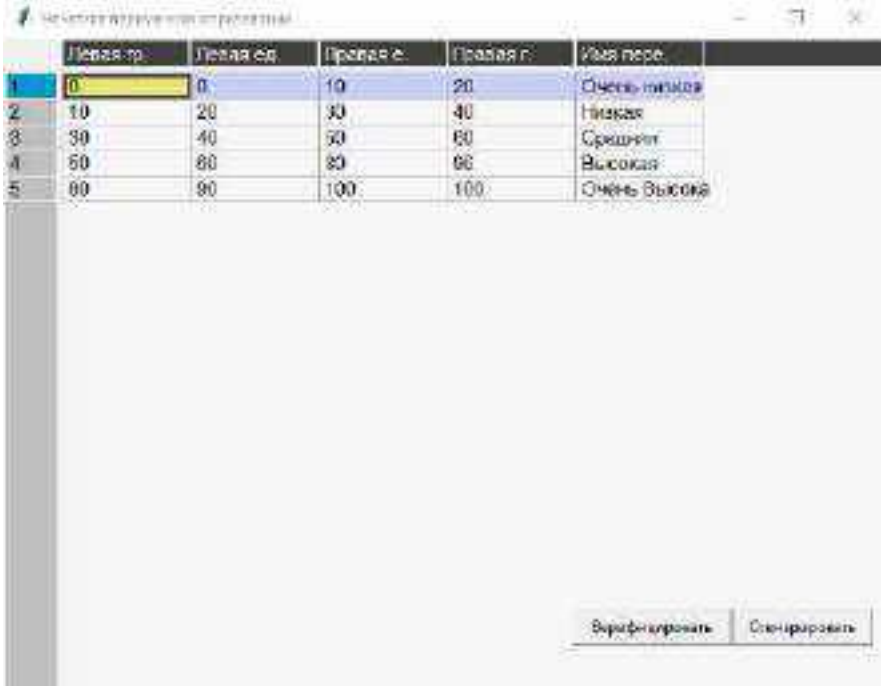


Рис. 21. Первоначальный экран редактора с заполненной таблицей

На рис. 21, 22 и 23 представлена верификация введенных данных на соответствие заданным программой требованиям к виду функций принадлежности ЛП.

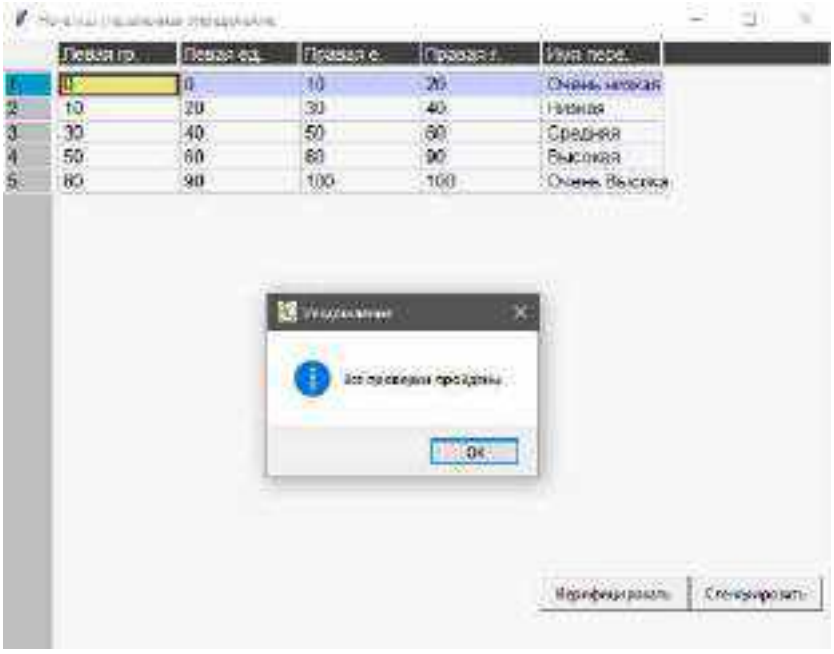


Рис. 22. Уведомление о соответствии данных требованиям

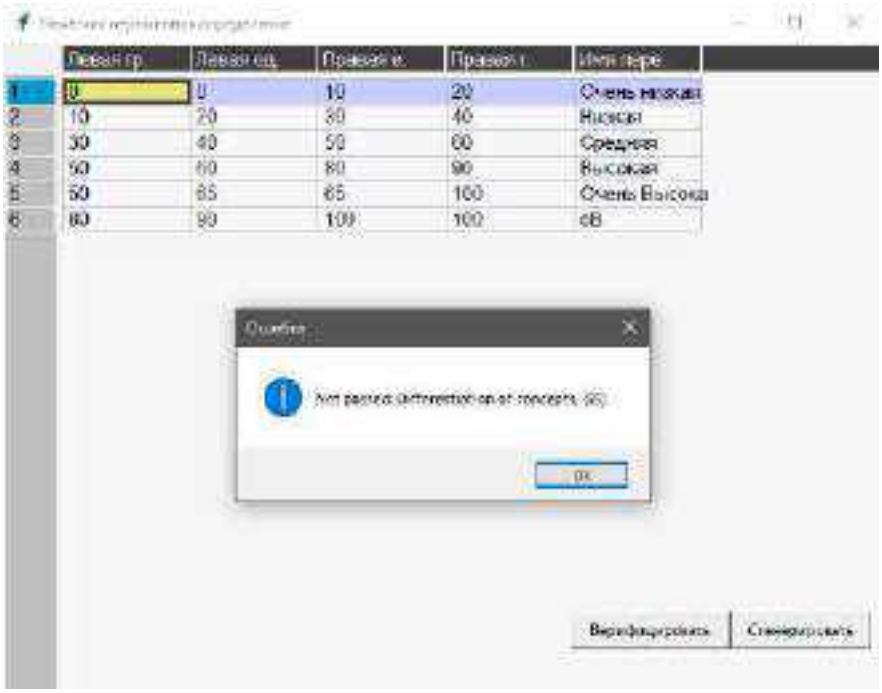


Рис. 23. Уведомление о ошибке – данные не соответствуют требованиям

Редактор автоматически строит функции принадлежности трапециевидной формы (см. рис. 24). После получения необходимого описания лингвистической переменной можно сохранить полученную редакцию ЛП. Лингвистическая переменная сохраняется в отдельном файле с именем лингвистической переменной.

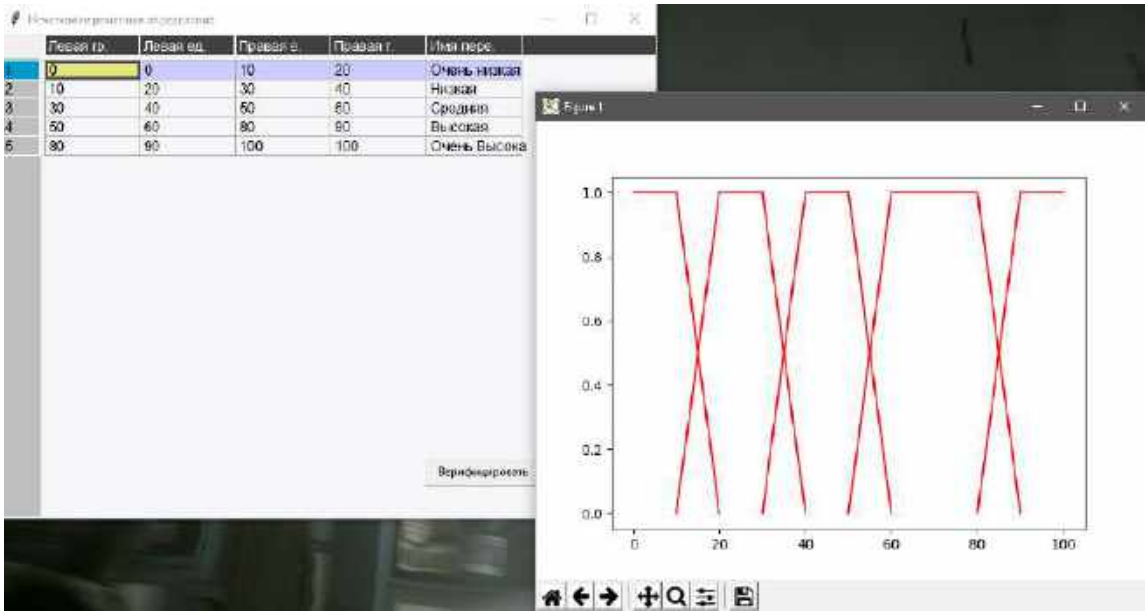


Рис. 24. Предъявление функций принадлежности по введенным значениям

2. Руководство пользователя

Установка: Для установки зависимостей необходимо установить Python3 и пакетный менеджер pip. Для установки зависимостей выполните в терминале(консоли) следующую команду из директории с проектом:

pip3 install -r req.txt

После установки зависимостей приложения для запуска используйте следующую команду также из директории с проектом:

python3 main.py (Linux) или python main.py (Windows)

Теперь можно пользоваться приложением. Для добавления или удаления строк в таблицу используйте контекстное меню, которое откроется при нажатии на правую кнопку мыши в окне приложения. Также из контекстного меню можно загрузить/сохранить и модифицировать уже готовую таблицу.

Для построения графика нечеткой переменной используйте экранную клавишу “Сгенерировать”, для проверки переменной “Верифицировать”.

Лабораторная работа №3. Разработка и исследование редактора правил для продукционной системы

Цель работы

Разработать редактор атрибутов ЭС, редактор правил, редактор объяснения результатов вывода, приобрести навыки представления и использования знаний как компонентов информационных и управляющих системах.

Постановка задачи

1) Научиться представлять атрибуты знаний;
2) Разработать алгоритмы создания и редактирования компонентов баз знаний;
3) Научиться разрабатывать интерфейсы пользователей для редактора правил;
4) Разработать форматы представления атрибутов и правил в информационных системах.

Теоретическая часть

Продукционные системы для представления знаний

Продукционные системы (системы с продукционными правилами) считаются частным случаем систем, основанных на правилах. Само правило состоит из двух частей. Разные авторы их определяют как: предпосылка – последствия; посылка – следствие; антецедент – консеквент; антецедента – консеквента; часть «ЕСЛИ» – часть «ТО»; часть «IF» – часть «THEN». «ЕСЛИ» и «ТО» являются служебными словами. В общем случае продукция имеет следующий вид:

<Идентификатор продукции>
ЕСЛИ (служебное слово) <Имя информационного атрибута 1>
<Связка> (атрибута и значений–равно, больше, меньше, …)
<Значение информационного атрибута 1>
И/ИЛИ (логические операции) <Имя информационного атрибута 2>
<Связка> (атрибута и значений–равно, больше, меньше, …)
<Значение информационного атрибута 2>
И/ИЛИ
…….
ТО <Имя описательного атрибута или атрибута управления>
<Связка> <Значение информационного атрибута>

После применения правила к имеемым данным могут быть получены новые знания (отсюда названия продукционные правила), либо будет предложено выполнить некоторое действие или получено некоторое описание.

Применение продукции может дать самостоятельный результат (одношаговый вывод) или результат может быть промежуточным и использоваться как данные для другой продукции (многошаговый вывод).

Существуют различные архитектуры продукционных систем. Наиболее известная – с рабочей памятью. В рабочей памяти размещаются факты, управляющие последовательностью применения правил. Применение правила меняет состояние рабочей области. Процесс продолжается пока не будет достигнута некоторая цель, например, будут означены некоторые терминальные атрибуты. Таким образом, продукционная система может состоять из: базы знаний – место хранения правил; рабочей области – область размещения фактов или значений информационных атрибутов; системы логического вывода – система поиска и применения подходящих правил; – системы объяснения логического вывода.

При создании экспертной системы выделяют предметную область, которая представляет собой совокупность технических, программных и прочих средств, посредством которых осуществляется функционирование в конкретной профессиональной области или даже решение отдельных её задач; методические, нормативные, технологические материалы, в которых описан и регламентируется процесс профессиональной деятельности; персонал, непосредственно принимающий участие в профессиональной деятельности, выделяются отдельные специалисты, признанные профессиональным сообществом и владеющие знаниями и навыками решения конкретных профессиональных задач, превосходящими знания и навыки основной массы специалистов предметной области. Необходимо также организационное решение и организационная воля улучшить качество решения профессиональных задач посредством распространения знаний и навыков экспертов через использование такого инструмента, как экспертные системы. Следующим доводом к разработке и использованию экспертных систем является особая требовательность к качеству и надёжности выбора решения. В этом случае для исключения «человеческого фактора» используются знания и навыки эксперта для контроля за лицом, принимающим решения (ЛПР), возможно, за самим экспертом. И ещё одним доводом является агрессивность внешней среды или другая причина невозможности присутствия человека при принятии решения или нецелесообразность (малая доля компонентов, для которых необходимо участие экспертных знаний и навыков или невозможность присутствия человека из-за малой размерности носителя…). Исходя из побудительных причин создания экспертных систем, их можно разделить на автоматизированные системы (с участием человека) и автоматические. Экспертные системы могут иметь различные внутренние структуры, однако, функционально можно выделить редактор знаний, исполняемый модуль (саму экспертную системы) и процессор объяснения результатов. Редактор знаний позволяет последовательно следить за созданием и заполнением базы знаний экспертной системы, изменять и дополнять структуры и содержимое БЗ при обнаружении неполноты и противоречивости. Атрибуты могут быть описаны переменными, принимающими строковые значения. Как правило, конкретная ЭС использует один из способов представления знаний экспертов. Выбор способа представления знаний зависит от многих критериев, например:

синтаксической, семантической и семиотической близости экспертным представлениям о предметной области, понятности для экспертов (после пользованием демонстрационным прототипом), необходимости объяснения предлагаемого решения, числом слоёв иерархии выявленных знаний, ресурсов вычислительного устройства (быстродействия, памяти…), наличия поддержки пакетов автоматизации проектирования. Если же есть необходимость в ЭС использовать в малых объёмах дополнительный способ представления к основному выбранному, его можно эмитировать выбранным основным способом. Например, продукцию можно представить при помощи фреймов. Приведём несколько основных систем представления знаний, это: предикатные системы, семантические сети, продукционные системы, фреймовые сети, системы, основанные на правдоподобных рассуждениях, нечёткие системы. Все эти системы в приложении к поиску приемлемого решения предназначены для сокращения пространства перебора вариантов. Нечеткие системы в «чистом виде» могут использоваться только для описания элементарных атрибутов или оценивания их значений. Для построения советующих систем на их основе необходимо дополнительно использовать продукции, предикаты или фреймы.

Экранные формы пользовательского интерфейса

Основной целью редактора является наполнение БЗ продукционными правилами, в которых представлены знания экспертов о предмете экспертизы. В нашем редакторе правила составляются из описанных атрибутов и их значений в пункте редактора «Создание переменных» (см. рис. 25) и «Определение значения переменных» (см. рис. 26).

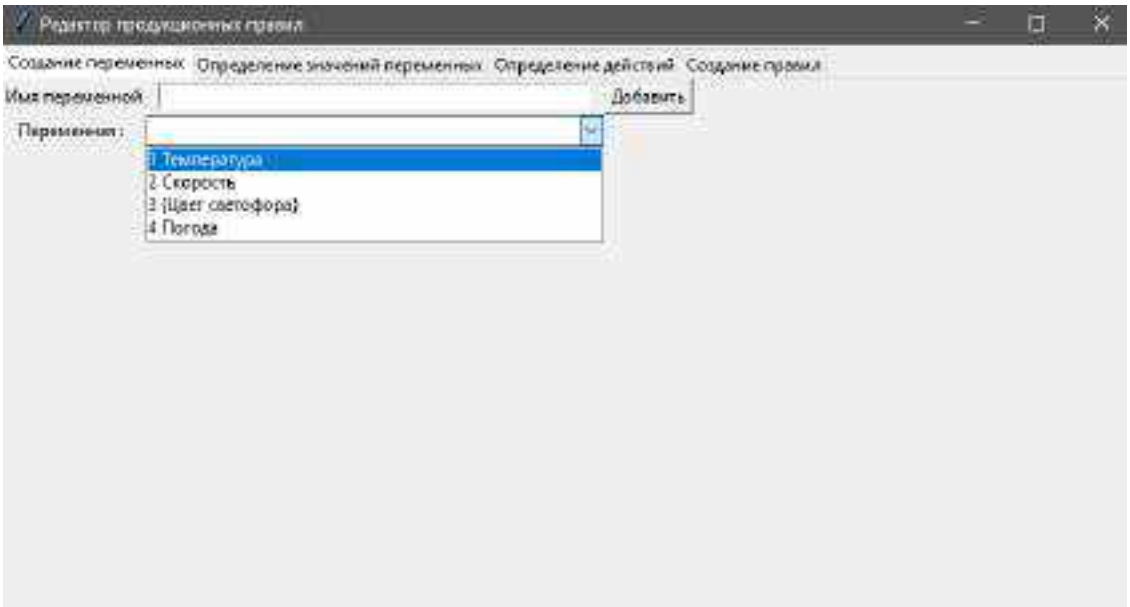


Рис. 25. Экранная форма “Создание переменных”

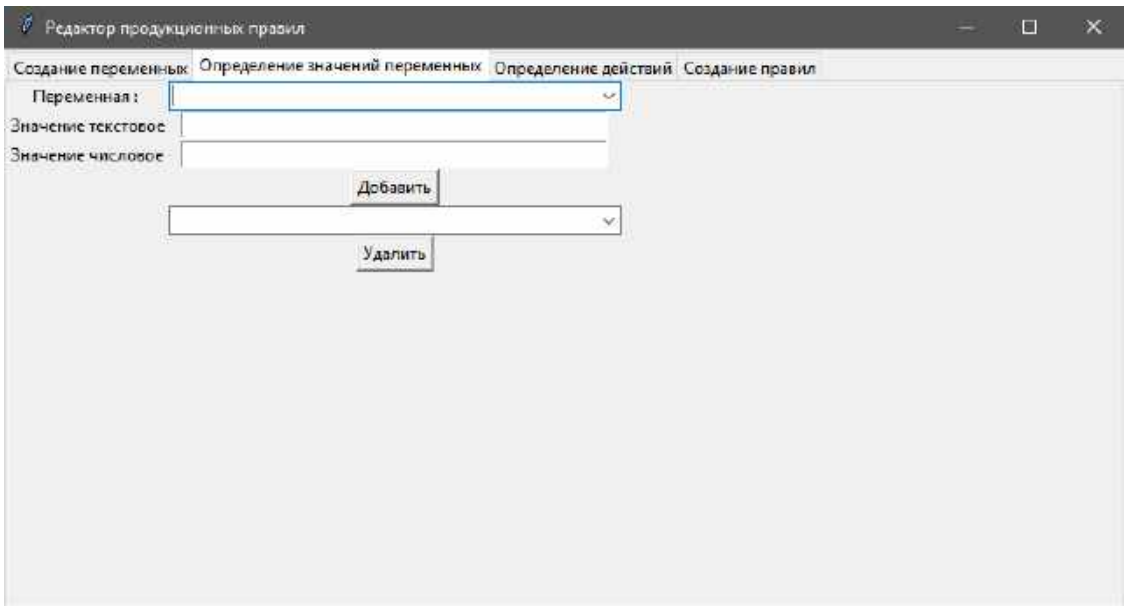


Рис. 26. Экранная форма “Определение значения переменных”

Пример экранной формы с одним продукционным правилом приведен на рис. 27.

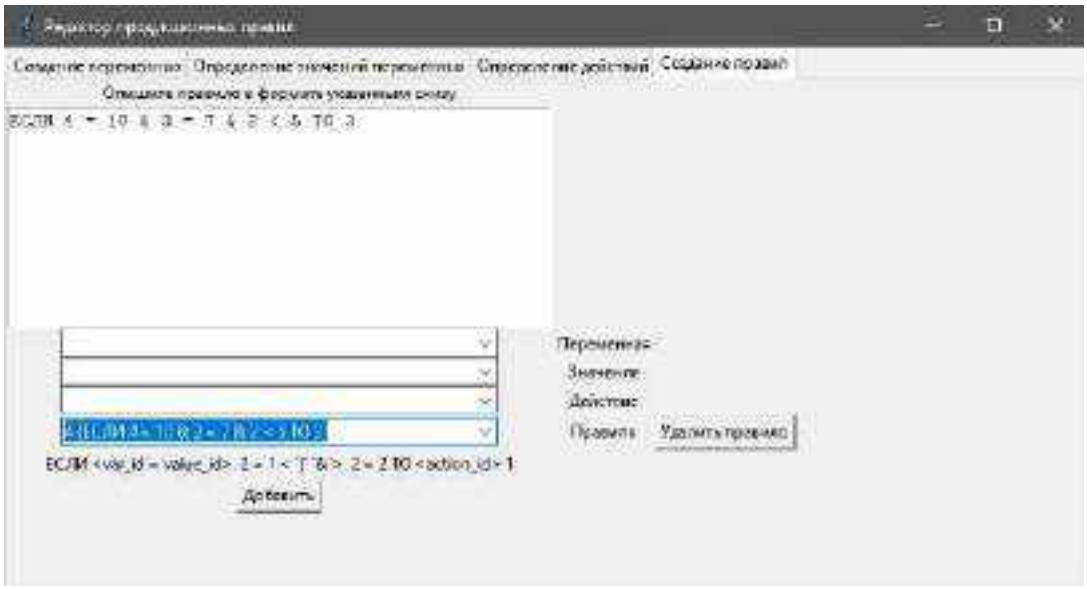


Рис. 27. Экранная форма “Создание правил”

После служебных слов «ЕСЛИ» и «ТО» следуют атрибуты антецедента и консеквента и их значения. Пользователь выбирает из предложенного ему списка атрибутов и из всего множества значений атрибута. Различные атрибуты и их значения в антецеденте разделены связкой «И». Консеквент же имеет только одно значение одного атрибута. На рис. 28 представлена иллюстрация определения действия для конкретного продукционного правила.

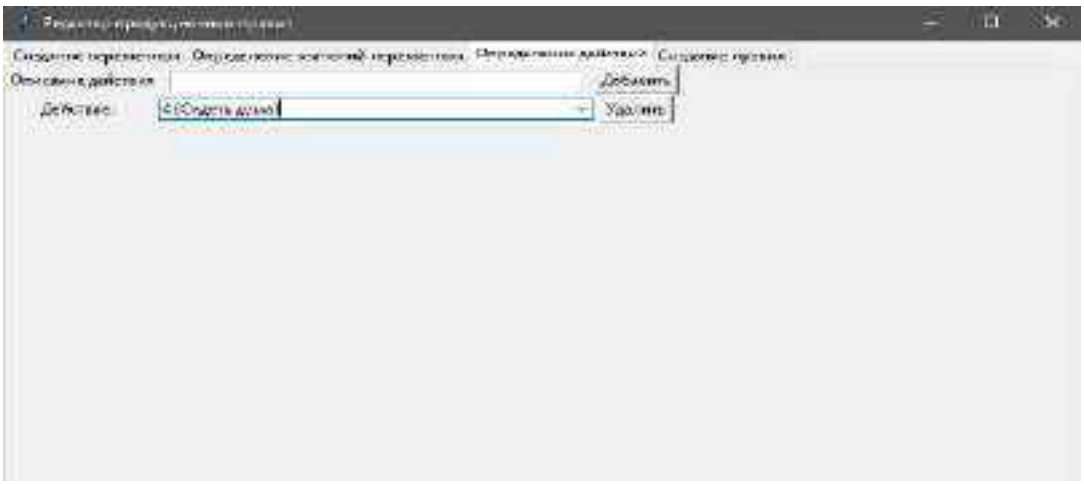


Рис. 28. Экранная форма “Определение действий”

Лабораторная работа №4. Разработка и исследование нечеткого регулятора

Цель работы

Объединить в одном проекте редактор правил, редактор лингвистических переменных и нечеткий регулятор.

Постановка задачи

1) Научиться представлять атрибуты знаний;
2) Разработать алгоритмы создания и редактирования нечётких регуляторов;
3) Разрабатывать интерфейсы пользователей нечеткого регулятора;
4) Разработать форматы представления атрибутов и правил в информационных системах.

Алгоритм программы

Алгоритм поиска значений переменной по нашей БД представлен на рисунке 29. Алгоритм добавления и удаления элементов в и из БД представлены на рисунках 30 и 31.

Рис. 29. Алгоритм поиска значений переменной

Рис. 30. Общий алгоритм добавления

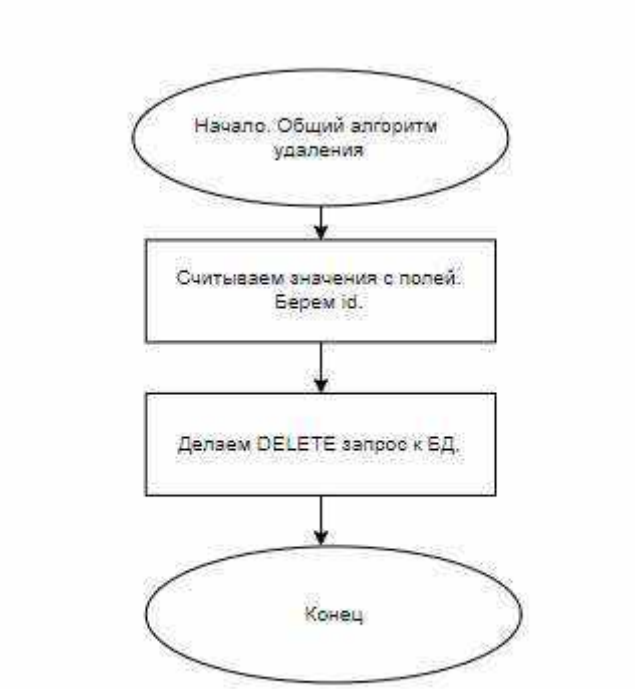


Рис. 31. Общий алгоритм удаления

Экранные формы пользовательского интерфейса

Экранный интерфейс разбит на 3 составные части

- Создание переменных (см. рис. 32), механизм работы взят из 2 лабораторной работы;
- Создание правил (см. рис. 33), механизм работы взят из 3 лабораторной работы;

- Поиск решения (см рис. 34);

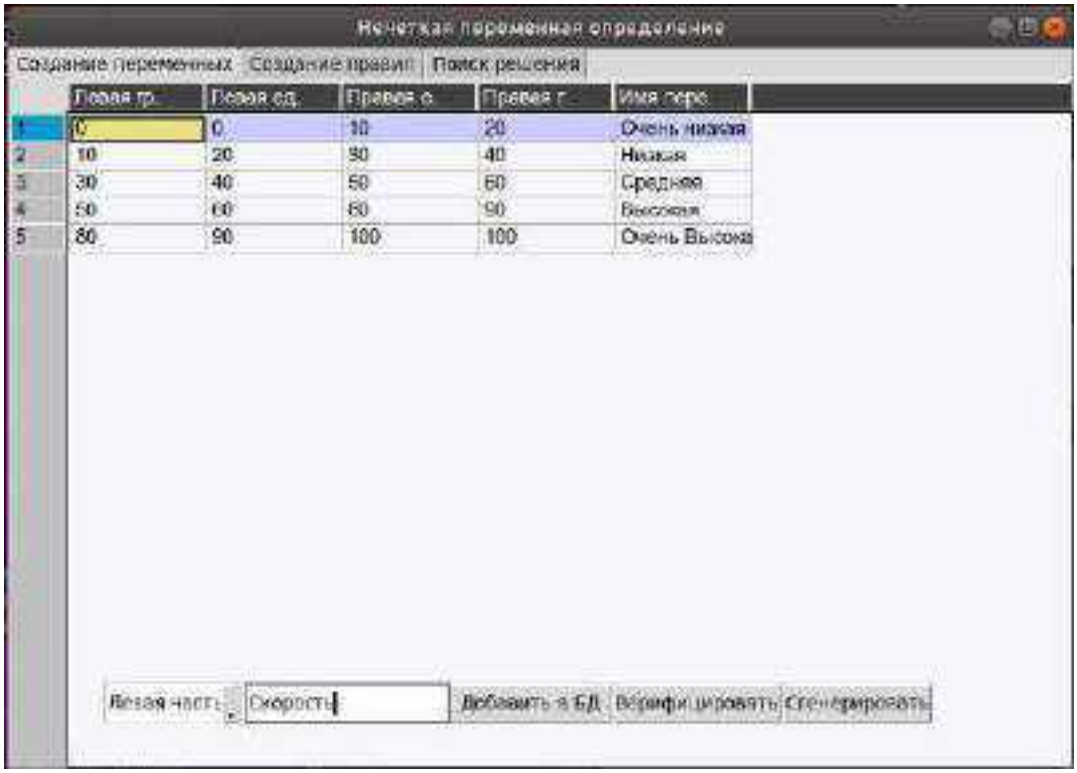


Рис. 32. Экранная форма “Создание переменных”

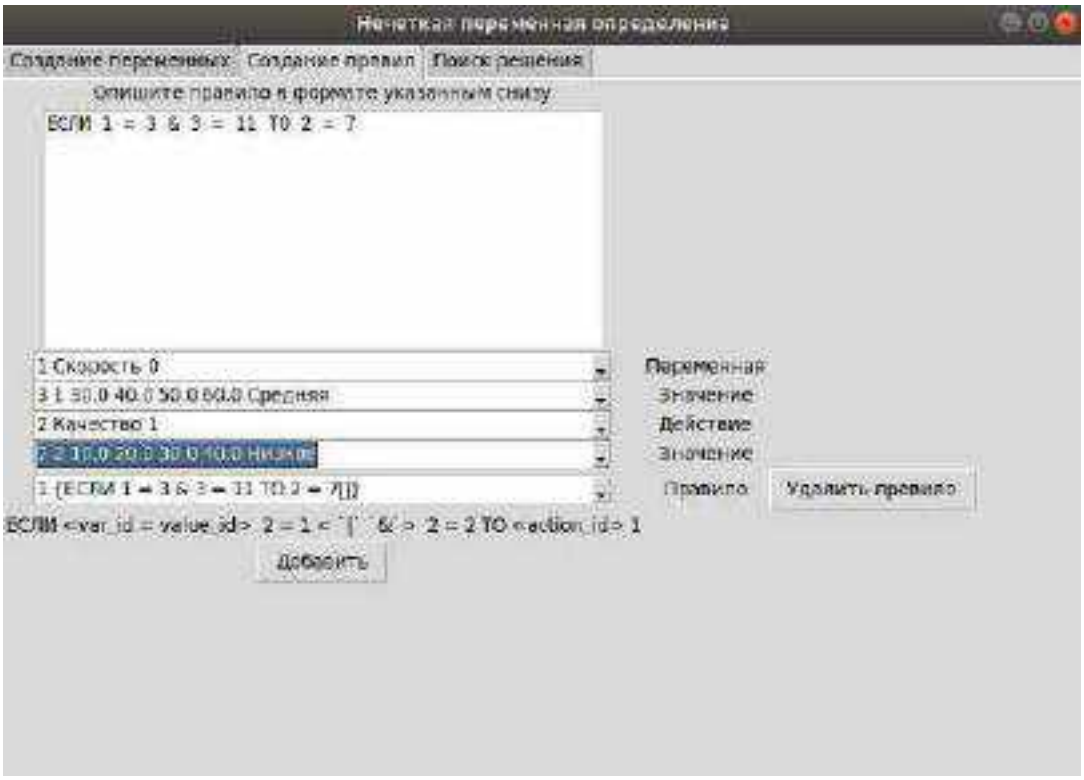


Рис. 33. Экранная форма “Создание правил”

После служебных слов «ЕСЛИ» и «ТО» следуют атрибуты антецедента и консеквента и их значения. Пользователь выбирает из предложенного ему списка

атрибутов и из всего множества значений атрибута. Различные атрибуты и их значения в антецеденте разделены связкой «И». Консеквент же имеет только одно значение одного атрибута.

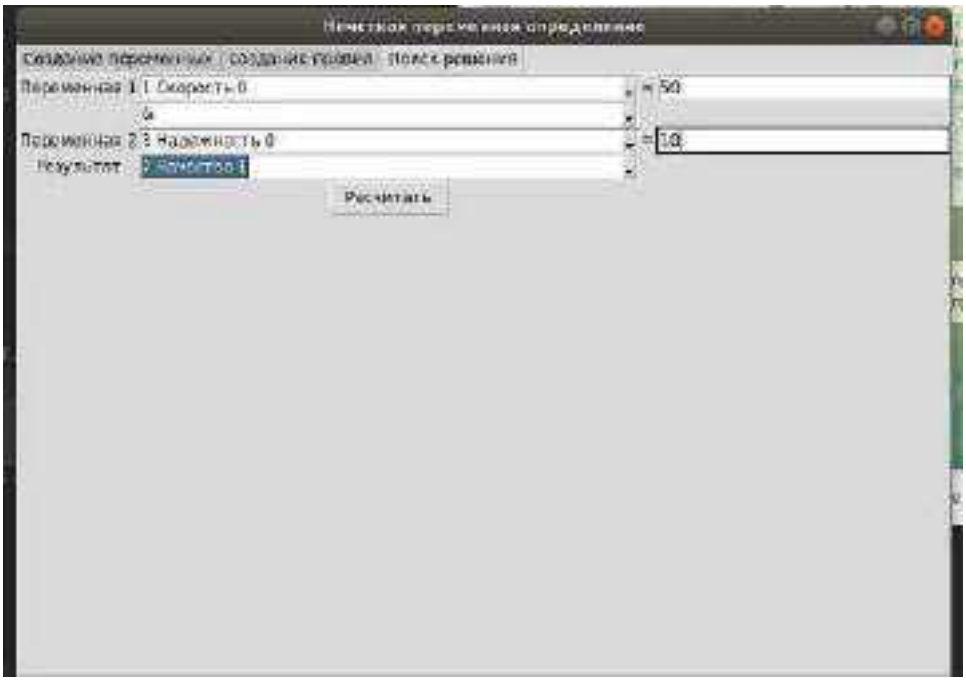


Рис. 34. Экранная форма “Поиск решения”

Найденное решение представлено на рис. 35

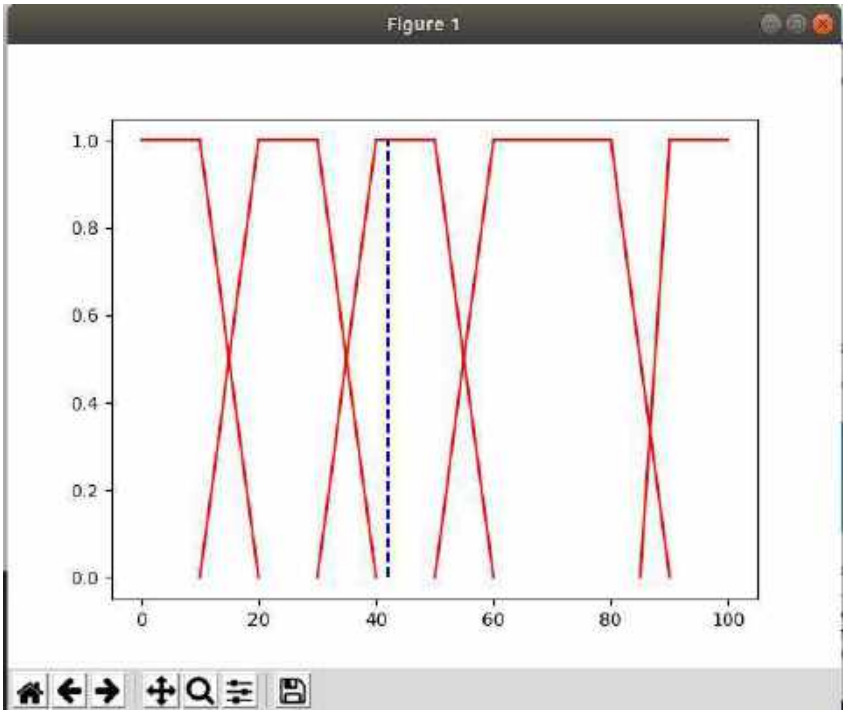


Рис. 36. Экранная форма “Найденное решение”

ЗАКЛЮЧЕНИЕ

Без знаний искусственный интеллект не может существовать в принципе. База знаний – ключевая компонента систем искусственного интеллекта. Поэтому при создании интеллектуальных систем особенное внимание уделяется моделям представления знаний. На сегодняшний день разработаны разнообразные модели знаний. Каждая из них обладает своими плюсами и минусами, и поэтому для каждой конкретной задачи необходимо выбрать именно свою модель. От этого может зависеть не столько эффективность выполнения поставленной задачи, сколько возможность ее решения вообще. При таком подходе не ставится вопрос об адекватности используемых в компьютере моделей представления знаний тем моделям, которыми пользуется в аналогичных ситуациях человек, а рассматривается лишь конечный результат решения конкретных задач. Проблема представления знаний заключается в несоответствии между сведениями о зависимостях данной предметной области, имеющимися у специалиста, методами, используемыми им при решении задач, и возможностями формального представления такой информации в ЭВМ. Часто проблема для эксперта осложняется трудностями по формулированию в явном виде имеющихся у него знаний. Большинство исследователей искусственного интеллекта рассматривают задачу разработки моделей представления знаний как задачу программной реализации концепции баз знаний. Это означает, что модели представления знаний должны обладать всеми свойствами, присущими знаниям. Модели представления знаний различаются по идеям, лежащим в их основе, с точки зрения математической обоснованности. В данном практикуме рассмотрена только часть известных моделей – эмпирические модели, включающие продукционные правила, фреймы, семантические сети, а также нейросети и эволюционные вычисления, относящиеся к бионическому направлению в искусственном интеллекте. Хочется выразить надежду, что после изучения практикума у слушателя сложились убеждения в полезности применения интеллектуальных систем, сформировались представления о моделях знаний.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Родзин С.И. Искусственный интеллект: Учеб. пособие. – Таганрог: Изд-во ТТИ ЮФУ, 2009. – 200 с.
2. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологии. – М.: Изд-во МГТУ им. Баумана, 2005. – 304 с.
3. Джексон П. Введение в экспертные системы. – М.: Изд. Дом «Вильямс», 2001. – 624 с.
4. Джонс М.Т. Программирование искусственного интеллекта в приложениях. – М.: ДМК Пресс, 2006. – 312 с.

5. Зозуля Ю.И. Интеллектуальные нейросистемы. – М.: Радиотехника, 2003. – с.
6. Курейчик В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений. – М.: Физматлит, 2012. – 260 с.
7. Липатова С.В. Сборник задач по курсу «Интеллектуальные информационные системы». – Ульяновск: УлГУ, 2010. – 64 с.
8. Рассел С., Норвинг П. Искусственный интеллект: современный подход. – М: Изд. дом «Вильямс», 2006. – 1408 с.
9. Родзин С.И., Ковалев С.М. Информационные технологии: интеллектуализация обучения, моделирование эволюции, распознавание речи. – Ростов-на-Дону: Изд-во СКНЦ ВШ, 2002. – 224 с.
10. Частиков А.П. и др. Разработка экспертных систем. Среда CLIPS. – СПб.: БХВ-Петербург, 2003. – 608 с.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Прикладные аспекты искусственного интеллекта»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024г.

Практическое задание № 1
«Классификация и регрессия посредством обучения с учителем»

1. Цель работы

Изучить методы решения задач классификации и регрессии. Ознакомиться с некоторыми наиболее распространенными методиками проведения операции обучения с учителем, предназначенными для решения задач классификации и регрессии. Изучить предложенные методы на простых примерах анализа данных о доходах и прогнозировании цен.

В процессе выполнения практического задания студенты должны ознакомиться стеоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляет задача классификации данных;
- изучить методы решения задачи классификации и понять отличие методов обучения с учителем и без него;
- изучить методы предварительной обработки данных;
- научиться использовать для решения задачи классификации данных методы на основе логистической регрессии и опорных векторов;
- изучить метод решения задачи классификации данных с помощью «наивного байесовского классификатора»;
- узнать, что такое задача регрессии (линейная и полиномиальная);
- научиться решать задачу линейной регрессии для простой и множественной регрессии;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Задача классификации

Задача классификации данных предполагает разбиение исходного множества данных на заданное число подмножеств (классов) с учетом заданного критерия оценки.

Применительно к методам машинного обучения задачу классификации можно

интерпретировать как задачу идентификации категории, к которой относится рассматриваемый объект (точка) данных. Модель решения классификации строится на основании обучающей выборки, содержащего объекты данных и им соответствующие метки. Например, необходимо определить содержит ли рассматриваемая фотография изображение лица человека. Мы создаем обучающую выборку, состоящую из двух классов: *face* и *no-face*. Затем мы обучаем нашу модель с помощью имеющегося набора данных. После завершения процесса обучения модель можно использовать для построения умозаключений. Алгоритм классификации данных стремится определить наиболее подходящий критерий для разбиения данных на заданное количество классов.

При этом важно правильно подобрать размер обучающей выборки. Она должна быть достаточно представительной, чтобы можно было делать необходимые ***обобщения*** для того, чтобы определить критерий разбиения данных на классы. При недостаточном размере выборки алгоритм будет испытывать проблемы при обработке неизвестных данных из-за эффекта ***переобучения*** модели.

Эффективная модель классификации значительно упрощает поиск и извлечение данных. Они могут применяться в системах распознавания лиц, идентификации спама, рекомендательных механизмах и т.п.

2.2. Обучение с учителем и без учителя

Одним из направлений искусственного интеллекта является машинное обучение. Методы машинного обучения можно условно разделить на две категории:обучение с учителем и обучение без учителя.

Обучение с учителем — это процесс создания модели машинного обучения, основанной на маркированных (помеченных) обучающих данных. Предположим, мы планируем разработать систему, которая будет **автоматически** прогнозировать доход человека исходя из заданного набора характеристик, таких как возраст, образование, место жительства и т.п. *Для* этого нам необходимо создать базу данных, содержащую подробные сведения о людях, и пометить эти характеристики (признаки) соответствующими маркерами. Это нужно для того, чтобы показать алгоритму классификации, какие параметры соответствуют тому или иному уровню

доходов. На основании этого соответствия алгоритм будет учиться рассчитывать уровень дохода человека, используя предоставленные параметры.

Обучение без учителя — это процесс создания модели машинного обучения, при котором не используются маркированные тренировочные данные в отличие от предыдущего метода. При этом из-за отсутствия меток алгоритму приходится обнаруживать отношения между данными, исходя исключительно данных. То есть при решении, например, задачи классификации мы не можем точно указать критерий такого разделения. Следовательно, при обучении без учителя алгоритм должен самостоятельно структурировать заданный набор данных путем разбиения на группы с использованием способа, который представляется наилучшим из всех возможных. В этом практическом задании мы рассмотрим методы классификации, основывающиеся на обучении с учителем.

2.3. Предварительная обработка данных

При решении практических задач мы обычно имеем дело с большими объемами необработанных исходных данных. Для успешной работы алгоритма машинного обучения необходимо определенным образом отформатировать исходные данные, т.е. выполнить предварительную подготовку данных. Рассмотрим несколько различных методов предварительной обработки данных:

- бинаризация - применяется в тех случаях, когда необходимо преобразовать числовые значения в булевы;
- исключение среднего - методика предварительной обработки данных, когда из ***векторов признаков*** исключаются средние значения, чтобы каждый признак центрировался на нуле. Целью данной операции состоит в том, чтобы исключить из рассмотрения смещение значений в векторах признаков;
- масштабирование. В векторе признаков каждое значение может меняться в некоторых случайных пределах. Поэтому важно масштабировать признаки для работы алгоритма машинного обучения;
- нормализация - заключается в приведении значений в векторе признаков к единой шкале измерения. В машинном обучении могут использоваться различные формы нормализации. Мы будем рассматривать два наиболее распространенных из

них, в которых значения изменяются так, чтобы их сумма была равна единице. **L1-*нормализация***, использует метод наименьших абсолютных отклонений и обеспечивает равенство единице суммы абсолютных значений в каждом ряду. **L2-*нормализация***, использует метод наименьших квадратов и обеспечивает равенство единице суммы квадратов значений.

В общем метод L1-нормализации считается более надежным по сравнению с L2-нормализацией, поскольку он менее чувствителен к выбросам.

2.4. Кодирование меток

В процессе решения задачи классификации данных мы имеем дело с множеством меток. Ими могут быть слова, числа или другие объекты. При использовании функций машинного обучения, входящих в библиотеку sklearn языка Python по умолчанию, считается, что метки должны иметь числовой формат. Однако часто в качестве меток используются слова, поскольку в таком виде они лучше всего воспринимаются человеком так легче отслеживать соответствия. Для преобразования слов в числа необходимо использовать кодирование. Под ***кодированием меток*** подразумевается процесс преобразования словесных меток в числовую форму.

2.5. Логистический классификатор

Логистическая регрессия — это методика, применяемая для объяснения отношений между входными и выходными переменными. Входные переменные считаются независимыми, выходные - зависимыми. Зависимая переменная может иметь фиксированный набор значений. Эти значения соответствуют классам задачи классификации.

Нашей целью является рассмотрение отношений между независимыми и зависимыми переменными путем оценки вероятности принадлежности зависимой переменной к тому или иному классу. Логистическая функция — это ***сигмоида***, используемая для создания функций с различными параметрами. Она тесно связана с построением *обобщенной линейной модели*, когда мы пытаемся построить прямую линию и расположить по отношению к группе точек таким образом, чтобы

минимизировать ошибку. Вместо *линейной* регрессии мы применяем логистическую регрессию, что позволяет упростить решение задачи.

2.6. Наивный байесовский классификатор

Наивный байесовский классификатор — это простая модель решения задачи классификации, основанная на использовании теоремы Байеса. Как известнотеорема Байеса позволяет оценить вероятность наступления события с учетом сопутствующих обстоятельств. Такая модель создается посредством присваивания меток классов экземплярам задачи, представленным в виде векторов значений признаков. При этом предполагается, что значение любого заданного признака не зависит от значений других признаков. Эго предположение о независимости рассматриваемых признаков и определяет термин «наивный» в названии модели.

В случае использования наивного байесовского классификатора считается, что каждый из признаков вносит независимый вклад в конечный результат, оценивающий вероятность наступления события.

Для проверки полученных результатов используем следующие оценочные параметры:

- ***качество*** - соотношение числа правильных прогнозов к общему числу рассмотренных точек данных;
- ***точность*** - соотношение числа правильно предсказанных значений данного класса к общему числу предсказанных значений;
- ***полнота*** - соотношение числа правильно предсказанных значений данного класса к общему числу его фактических значений.

Оценку работы модели производим путем сравнения предсказанных значений с истинными метками. Однако данный метод является не вполне надежным, для получения более точной оценки нужно выполнить перекрестную проверку.

В основе перекрестной проверки лежит идея разделения обучающей выборки на обучающий и проверочный наборы, чтобы не использовать одни и те же данные при обучении и тестировании.

2.7. Матрица неточностей

Матрица неточностей — это таблица, используемая для оценки эффективности классификатора. Обычно она путем анализа полученных результатов и их сравнения с ожидаемыми значениями. В процессе анализа мыопределяем число неверно отнесенных классов, т.е. точность работы алгоритма. В процессе конструирования матрицы неточности мы имеем дело с различными метриками. Рассмотрим случай бинарной классификации, когда выходная переменная может принимать два возможных значения: 0 и 1.

- **Истинно положительные результаты** – когда предсказанный результат совпадает с ожидаемым и равен 1.
- **Истинно отрицательные результаты** – когда результаты также совпадают и равны 0.
- **Ложноположительные результаты** — это случаи, когда предсказанный результат равен 1, а ожидаемый равен 0. Такая ситуация известна как ошибка 1-го рода.
- **Ложноотрицательные результаты** —случаи, когда предсказанныйрезультат равен 0, а ожидаемый равен 1.. Такая ситуация известна как ошибка 2-го рода.

В зависимости от специфики решаемой задачи может потребоваться настройка алгоритма для того, чтобы уменьшить долю ложноположительных или ложноотрицательных результатов. Например, в системах биометрической идентификации личности очень важно избежать ложноположительных результатов, чтобы злоумышленники не смогли получить доступ к конфиденциальной информации.

2.8. Метод опорных векторов

Метод (машина) опорных векторов (Support Vector Machine - SVM) — этометод классификации с помощью гиперплоскости, разделяющей классы. Гиперплоскость представляет собой N-мерную версию прямой линии. Применительно к задаче бинарной классификации маркированного обучающего набора данных, SVM находит оптимальную гиперплоскость, которая делит данные на два класса. Мы рассмотрим в данном практическом задании именно задачу

бинарной классификации. В этом случае мы имеем дело с точками и прямыми, расположенными на двумерной плоскости.

Можно провести множество прямых линий, разделяющих данные классы точек, но нам нужно выбрать ту, которая наилучшим образом разделяет множества на два класса. Лучшим вариантом в таком случае можно считать линию, которой соответствуют максимальные расстояния, на которые точки удалены от разделяющей их линии. Точки, расположенные на пунктирных линиях, называются ***опорными векторами***. Расстояние между двумя пунктирными линиями называется ***максимальным зазором***.

В дальнейшем описанный метод бинарной классификации можно использовать и для решения задач классификации с N классами.

2.9. Понятие регрессии

Регрессия — это процедура оценки того, как соотносятся между собой входные и выходные переменные. Следует отметить, что выходные переменные представляют собой вещественные числа, т.е. существует бесконечное число результирующих возможностей. Эго вступает в противоречие с требованием задачи классификации, что количество выходных классов должно быть фиксированным. Входные переменные называют независимыми переменными (или предикторами), а выходные - зависимыми (или критериальными) переменными. При этомдопускается взаимная корреляция между входными переменными.

Целью является выявление взаимозависимостей между выходными и входными переменными. Регрессионный анализ позволяет выяснить, как изменяется значение выходной переменной, при изменении значений некоторой части входных переменных. Если, например, имеет место линейная зависимость, говорят олинейной регрессии. Иногда линейной регрессии оказывается недостаточно для анализа взаимосвязей между входными и выходными переменными. В этом случае необходимо использовать полиномиальную регрессию, в которой зависимость между входными и выходными переменными носит полиномиальный характер. С вычислительной точки зрения такой подход достаточно сложен, но обеспечивает более высокую точность. Выбор вида регрессии для выявления указанных

соотношений определяется конкретикой задачи. Регрессию часто используют для прогнозирования цен, экономических показателей и т.п.

3. Практическая часть

3.1. Предварительная обработка данных

Чтобы привести данные к форме, приемлемой для алгоритмов машинного обучения, мы должны предварительно подготовить их и преобразовать в нужный формат.

1. Создать новый файл Python и импортировать необходимые пакеты.

2. Задать начальный набор данных.

3. Для бинаризации входных данных используем встроенный метод, установив для этого желаемое пороговое значение. Добавляем необходимый текст в уже существующий файл Python. После выполнения процедуры бинаризации на экран будут выводиться преобразованный набор входных данных.

4. Выполнить процедуру исключения среднего. Для этого добавим новый фрагмент программы в существующий файл. В результате выполнения процедуры будут вычислены и выведены на экран величина среднего значения и стандартное отклонение для входных данных до и после выполнения процедуры.

5. Выполняем процедуру масштабирования. Добавляем новый фрагмент программы в существующий файл. В результате выполнения этой процедуры на экран выводится набор данных, после выполнения масштабирования. Каждая строка масштабирована так, чтобы максимальным значением было 1, а все остальные значения определялись относительно него.

6. Для выполнения нормализации данных добавим новый фрагмент программы в существующий файл. Будет выполнена нормализация данных двумя вышеупомянутыми методами L1-нормализация и L2-нормализация. В результате на экране отобразятся два преобразованных набора данных.

3.2. Кодирование меток

1. Для моделирования процедуры кодирования создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем метки.

3. Задаем объект кодирования и проводим его обучение.

4. Получаем отображение слов в числовые значения.

5. Для проверки корректности работы процедуры кодирования задаем случайный набор упорядоченных меток. Затем выполняем процесс декодирования

6. В результате на экране должна появиться информация, позволяющая оценить корректность работы использованного алгоритма.

3.3. Модель логистической регрессии

Работу модели логистической регрессии рассмотрим на примере решения задачи классификации. Для работы понадобится программный пакет Tkinter. В случае необходимости его можно найти по ссылке: https://docs.python.org/2/library/tkinter.html.

1. Для моделирования логистической регрессии создаем новый файл Python и импортируем необходимые пакеты.

2. Зададим набор входных данных с использованием двумерных векторов и соответствующих меток.

3. Обучаем модель, используя помеченные данные.

4. Решаем задачу классификации набора данных, задаем параметры визуализации результатов работы модели для графического вывода полученных границ классов.

5. Изменяя значение штрафа на неточности классификации (параметр C) подбираем оптимальное значение параметра, соответствующее наилучшей настройке алгоритма к полученному набору данных. При этом важно не допустить переобучения модели из-за чересчур больших значений параметра C.

3.4. Наивный байесовский классификатор

1. Для создания модели наивного байесовского классификатора создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, каждая строка которого содержит значения, разделенные запятой. Загружаем данные из этого файла.

2. Зададим модель наивного байесовского классификатора. В данной работе используем *гауссовский* наивный байесовский классификатор, в котором предполагается, что значения, ассоциируемые с каждым классом, соответствуют закону распределения Гаусса.

3. Выполняем обучение модели на заданной обучающей выборке.

4. Решаем задачу классификации заданного набора входных данных. Для оценки качества работы модели вычисляем значения оценочных параметров: качество, точность и полнота. Сравниваем предсказанные значения с истинными метками, а затем визуализируем результат.

5. Разобьем обучающую выборку на обучающий и проверочный наборы в пропорции 8 к 2 соответственно. Затем выполним обучение модели наивного байесовского классификатора на этих данных.

6. Вычислим качество классификатора и визуализируем результаты. Для этого воспользуемся встроенными функциями для вычисления качества, точности и полноты модели на основании тройной перекрестной проверки.

7. После выполнения программы на экран выводятся числовые значения оценочных параметров и графическое отображение результатов классификации.

3.5. Матрица неточностей

1. Рассмотрим пример построения матрицы неточностей. Для этого создаем новый файл Python и импортируем необходимые пакеты. Задаем наборы фактических (калибровочных) и предсказанных меток классов.

2. Строим матрицу неточностей, используя заданные метки и визуализируем ее.

3. Выводим отчет о результатах классификации. Отчет содержит оценки эффективности классификатора по отношению к каждому из классов.

Белому цвету соответствуют более высокие значения, черному - более низкие, на что указывает расположенная справа градиентная шкала. В идеальном случае все диагональные квадраты были бы белыми, а все остальные - черными, что соответствовало бы 100% качеству.

3.6. Классификация данных о доходах с помощью метода опорных

векторов

Рассмотрим решение задачи классификации данных с помощью машины опорных векторов на примере разбиения определенного круга физических лиц на две категории в соответствии с уровнем их доходов. Прогноз необходимо сделать на основе заданного набора входных данных (атрибутов). Целью является классификация по критерию соответствия заданному уровню доходов (например, по размеру прожиточного минимума). Поскольку в данном примере мы будем иметь дело с данными смешанного типа, т.е. данные представлены как в символьном, так и в числовом виде, необходимо будет выполнить предварительную обработку данных.

1. Для создания модели задачи бинарной классификации создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, содержащий начальные данные о физических лицах и их доходах. Загружаем данные из этого файла.

2. Проводим предварительную обработку данных для начала работы алгоритма. Каждая строка данных отделяется от следующей с помощью запятой, что требует соответствующего разбиения строк. Последним элементом каждой строки является метка. В зависимости от этой метки мы будем относить данные к тому или иному классу.

3. Преобразуем исходный список в массив array, чтобы его можно было использовать в качестве входных данных функций sklearn. Если атрибутпредставляет собой символ, то его необходимо соответствующим образом кодировать. Если атрибут – число – оставляем его без изменения.

4. Воспользуемся моделью классификации на основе машины опорных векторов и начинаем обучение модели.

5. Для оценки эффективности и корректности работы модели выполняем перекрестную проверку. Для этого разбиваем обучающую выборку на обучающий и тестовый наборы в пропорции 8 к 2 и запускаем решение для тренировочных данных. Вычисляем значение параметра F-мера.

6. После проверки используем модель для случайно выбранного элемента набора данных. Изменяем в программном коде значения основных параметров алгоритма (F-мера, точность, полнота), тестируем его работу на нескольких наборах

значений параметров и сравниваем полученные данные.

3.7. Создание модели регрессии для одной переменной

1. Для создания модели регрессии для одной переменной задачи создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, содержащий исходные данные. В качестве разделителя используется запятая.

2. Разделяем имеющиеся данные на обучающий и тестовый наборы.

3. Проводим обучение модели линейной регрессии на соответствующем наборе данных.

4. Используя обученную модель, решаем задачу для тестового набора данных. Определяем качество решения задачи регрессии на основе сравнения полученных результатов с известными.

5. После завершения программы на экран выводятся числовые значения оценочных параметров и графическое отображение результатов работы программы.

3.8. Создание модели многомерной регрессии

1. Для создания модели многомерной регрессии создаем новый файл Python и импортируем необходимые пакеты. В данном примере будем решать задачу полиномиальной регрессии. Источником данных служит текстовый файл, содержащий исходные данные. В качестве разделителя используется запятая.

2. Разделяем имеющиеся данные на обучающий и тестовый наборы.

3. Проводим обучение модели линейной регрессии на соответствующем наборе данных.

4. Используя обученную модель, решаем задачу для тестового набора данных. Определяем качество решения задачи регрессии на основе сравнения полученных результатов с известными.

5. После завершения программы на экран выводятся числовые значения оценочных параметров и графическое отображение результатов работы программы.

3.9. Использование модели регрессии на основе машины опорных векторов

для решения задачи оценки стоимости недвижимости

Рассмотрим применение модели регрессии с помощью машины опорных векторов для получения прогноза цен на недвижимость на основе заданного набора атрибутов.

1. Для создания модели регрессии создаем новый файл Python и импортируем необходимые пакеты. Загружаем набор данных с ценами на жилье.

2. Для более объективной оценки случайным образом перемешиваем данные и разбиваем их на обучающий и тестовый наборы в соотношении 8 к 2.

3. Задаем модель регрессии на основе машины опорных векторов и проводим ее обучение. Необходимо эмпирическим путем подобрать значение параметра C (штраф за ошибки) при котором модель будет выдавать наиболее точный прогноз.

4. Выберем случайным образом объект из входного набора данных и получим для него прогноз.

5. Путем сравнения полученных результатов с известными, сравним корректность построенной модели.

4. Порядок выполнения практического задания

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения практического задания результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическое задание № 2
«Распознавание образов на основе методов обучения без учителя»

1. Цель работы

Изучить различные методы, относящиеся к категории методов обучение без учителя. Ознакомиться с некоторыми наиболее распространенными методами и моделями решения задачи кластеризации. Научиться практическому использованию изученных методов для решения простых задач оценки состояния рынков, финансового прогнозирования и т.д.

В процессе выполнения практического задания студенты должны ознакомиться стеоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляет обучение без учителя;
- узнать, что такое задача кластеризации;
- изучить основные методы решения задачи кластеризации;
- изучить метод кластеризации данных с помощью метода k-средних;
- научиться проводить оценку количества кластеров с помощью алгоритма сдвига среднего;
- изучить метод оценки качества кластеризации с помощью силуэтных мер;
- понять, что собой представляют смешанные гауссовские модели;
- модель классификации на основе смешанных гауссовских моделей;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Обучение без учителя

В предыдущей работе мы рассматривали случай, когда в процессе обучения алгоритм использует наборы помеченных данных. На практике часто приходится оперировать данными, не имеющими меток. В этом случае применяются методы ***обучения без учителя***, которые используют для решения задачи различные критерии сходства.

2.2. Задача кластеризации и методы ее решения

Задача ***кластеризации*** относится к числу наиболее популярных задач обработки данных, решаемых с помощью методов обучения без учителя. В этом случае мы имеем входное множество данных, которые необходимо каким-то образом распределить по нескольким группам (***кластерам***), причем число этих групп заранее неизвестно. Для оценки «близости» данных и их отнесения к тому илиному кластеру могут использоваться различные критерии, например, евклидово или хеммингово расстояние.

Задача в данном случае состоит в том, чтобы выбрать оптимальный критерий (параметр) наилучшим образом отражающий принадлежность или непринадлежность анализируемого объекта к той или иной группе. Из-за отсутствия универсальных критериев и методик эту задачу необходимо решать в каждый раз заново, исходя из условий конкретной задачи.

Разработано достаточно большое число методов решения задачи кластеризации. Одним из них является метод ***k-средних***. Предполагаем, чтоколичество кластеров известно и равно ***k***, разделяем имеющиеся данные на ***k*** групп. На каждой итерации алгоритма происходит обновление положений ***центроидов*** (центров тяжести кластеров). Процесс продолжается до тех пор, пока все центроиды не займут оптимальные положения.

Важным моментом в работе алгоритма, оказывающим существенное влияние на качество решения, является задание начального расположения центров. Для этого могут использоваться различные стратегии. В стандартном варианте алгоритма начальное расположение центров задается случайным образом. Существуют модифицированные варианты, когда для задания начального расположения используются различные эвристики. Например, задание такого варианта расположения центров кластеров, чтобы они находились на максимально возможном удалении друг от друга. Затем в процессе работы алгоритма происходит пошаговое улучшение текущего варианта расположения за счет отнесения каждой точки к ближайшему кластерному центру.

После того, как все точки входного набора отнесены к какому-либо кластеру первая итерация завершается. После чего, исходя из полученной конфигурации

кластеров, снова определяется положение центров кластеров и процесс повторяется. После того, как центры кластеров примут устойчивое положение можно считать работу алгоритма завершенной, а текущее положение центров кластеров будет представлять решение задачи кластеризации.

2.3. Метод сдвига среднего

Метод сдвига среднего - алгоритм, используемый для решения задач кластеризации. Он называется ***непараметрическим***, поскольку в нем не используются какие-либо допущения относительно базового распределения данных.

В данном методе мы предполагаем, что положение объектов данных внутри обучающей выборки данных соответствует закону распределения вероятности. Отсюда следует, что кластеры будут образовываться вокруг точек максимума базового распределения. То есть число кластеров в данном случае будет равно числу максимумов (пиков) функции распределения. Целью является определение центров этих кластеров.

Для каждого объекта данных определяется некая окрестность для которой определяется центр. По мере появления новых точек центры кластеров изменяют свое положение, приближаясь к центру кластера, к которому она относится. Движение происходит от областей с меньшими значениями вероятности к областямс высокими значениями. Из-за такого движения центров кластеров от средних к максимальным значениям вероятности метод получил свое название – метод сдвига среднего. Как и в предыдущем случае, алгоритм продолжает работу до тех пор, покацентры кластеров не перестанут смещаться.

2.4. Использование силуэтных оценок

В реальных задачах мы имеем дело с огромным количеством неструктурированных данных, поэтому нужны дополнительные способы качества кластеризации.

Силуэтная мера - это интегральная характеристика связности и разделения кластеров данных. Она дает оценку соответствия исследуемого объекта данных тому или иному кластеру. Силуэтная оценка вычисляется для каждой точки данных

по следующей формуле:

$$SSc = (p - q) / max(p, q)$$

где p - среднее расстояние до точек ближайшего соседнего кластера, а q - среднее расстояние до точек в текущем кластере.

Данный параметр может принимать значения в интервале от -1 до 1. Если параметр принимает положительные значения, близкие к +1, это означает, что данная точка имеет сходство с другими объектами данного кластера. Если же параметр отрицательные значения, то наоборот. Таким образом, если в результате силуэтной оценки получено слишком много точек с отрицательными значениями, тонеобходимо попытаться получить новый вариант разбиения для другого числа кластеров.

2.5. Смешанные гауссовские модели

В смешанных моделях предполагается, что данные управляются несколькими законами распределения. Комбинация таких компонентов создают многомодальную функцию распределения, которая и становится смешанной моделью. Например, если распределения подчиняются закону Гаусса, то модель называется гауссовской моделью.

Если мы хотим получить качественную репрезентативную модель, то должны учесть все возможные вариации в пределах исследуемой выборки. В таком случае мы можем использовать отдельные модели для моделирования отдельных вариаций, а затем объединить их в смешанную модель. Смешанные модели обеспечивают получение более высокой точности и гибкости моделирования базовых распределений данных. Они также сглаживают возможные пропуски в наборах данных.

Смешанная гауссовская модель - это параметрическая модель, представленная в виде взвешенной суммы компонентных гауссовских функций. Параметры GMM оцениваются на основе обучающих данных с использованием таких алгоритмов, как ***EM-алгоритм*** (принцип максимума правдоподобия) или ***MAP-алгоритм*** (принцип максимума апостериорной вероятности).

2.6. Модель распространения сходства

Модель кластеризации на основе ***распространения сходства*** не требует предварительного задания числа кластеров. Мы задаем критерии сходства, которые должен использовать алгоритм. Элементы кластеров попарно обмениваются сообщениями двух категорий, содержащими информацию о ***пригодности*** и ***доступности*** этих элементов к использованию в качестве образцов. Сообщения первой категории отсылаются элементами кластера потенциальным образцам и указывают на то, насколько хорошо точка данных подходила бы для того, чтобы быть элементом кластера данного образца. Сообщения второй категории отсылаются в обратном направлении и указывают на то, насколько хорошо они подошли бы для того, чтобы служить образцом. Этот процесс продолжается до тех пор, пока алгоритм не сойдется к оптимальному набору образцов. В силу своей общности и простоты реализации этот метод широко применяется в различныхобластях.

3. Практическая часть

3.1. Решение задачи кластеризации методом *k*-средних

1. Для создания модели кластеризации методом k-средних, создаем новый файл Python и импортируем необходимые пакеты. Рассмотрим пример задачи кластеризации для двумерных данных. В качестве источника данных используем текстовый файл, каждая строка которого содержит два числа, разделенные запятой. Загружаем данные из этого файла.

2. Задаем необходимые значения параметров алгоритма: число кластеров, число итераций, метод задания начального расположения центров кластеров и параметры визуализации для отображения получаемого распределения. Для более точной привязки точек набора данных, их местоположения в пространстве удобно использовать координатную сетку.

3. Выполняем обучение модели кластеризации методом k-средних на входном наборе данных. Затем, используя обученную модель, решаем задачу кластеризации для заданного набора данных.

3.2. Оценка количества кластеров с использованием метода сдвига среднего

1. Для оценки числа кластеров в заданном наборе данных с помощью алгоритма сдвига среднего создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл. Загружаем данные из этого файла.

2. Для начала работы задаем значение параметра «ширина окна входных данных». Ширина окна - это параметр оценки плотности распределения ядра данных, который влияет на скорость сходимости алгоритма и число кластеров.

3. Обучаем модель кластеризации на основе модели сдвига среднего, используя заданную оценку ширины окна.

4. Находим центры всех кластеров. Определяем число кластеров.

5. После завершения программы на экран выводятся числовые значения оценочных параметров и графическое отображение результатов работы программы.

3.3. Оценка качества кластеризации с помощью силуэтных оценок

1. Для оценки качества кластеризации с помощью силуэтных оценок создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, каждая строка которого содержит два числа, разделенных запятой. Загружаем данные из этого файла.

2. Выполняем инициализацию переменных. Задаем массив values, содержащий список значений. Выполняем цикл по всем значениям, создавая модель k-средних на каждой итерации. Для оценки степени близости объектов данных используем евклидово расстояние.

3. В результате работы алгоритма получаем силуэтную оценку для текущей задачи кластеризации. Выполняем кластеризацию на других наборах данных. Определяем с помощью силуэтных оценок лучший вариант решения и число кластеров. Результаты выводятся на экран.

3.4. Классификация данных на основе гауссовской смешанной модели

1. Для построения модели классификации на основе смешанной гауссовской

модели создаем новый файл Python и импортируем необходимые пакеты. Для анализа мы будем использовать набор данных из предлагаемой в задании библиотеки.

2. Разбиваем входной набор данных на обучающий и тестовый наборы данных в пропорции 8 к 2. Задаем число подмножеств (классов).

3. Создадим модель классификации на основе смешанной гауссовской модели. Для начала работы модели задаются значения необходимых параметров: число классов разбиения, число итераций алгоритма, тип используемых комбинаций.

3. Проводим обучение алгоритма классификации на основе смешанной гауссовской модели для тренировочного набора.

4. Выполняем нормализацию собственных векторов модели и настраиваем параметры визуализации результатов классификации границы для получения необходимой точности изображения

5. Вычисляем результат классификации для обучающего и тестового наборов и выводим на экран полученные графики и числовые значения.

3.5. Нахождение подгрупп на фондовом рынке с использованием модели распространения сходства

1. Рассмотрим работу модели распространения сходства на примере определения похожих моделей поведения групп среди участников фондового рынка. В качестве критерия будем использовать изменение котировок на момент открытия и закрытия биржи. Для оценки качества кластеризации с помощью силуэтных оценок создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных будем использовать набор данных из предлагаемой в задании библиотеки.

2. Вычисляем разницу между котировками на момент открытия и закрытия биржи. Выполняем нормализацию данных.

3. Создаем и обучаем модель кластеризации на основе распространения сходства.

4. Вычисляем результат работы модели и выводим на экран полученные графики и числовые значения.

Полученные результаты позволяют определить различные группы участников фондового рынка за исследованный период.

3.6. Сегментирование рынка на основе моделей совершения покупок

1. Рассмотрим работу модели совершения покупок на примере оценки сегментирования потребительского рынка. В качестве критерия будем использовать изменение котировок на момент открытия и закрытия биржи. Данные о поведении потребителей содержатся в файле формата csv, поэтому его можно напрямую импортировать в программу и преобразовать затем в соответствующий массив. Целью является определение имеющихся стереотипов поведения покупателей и построение прогноза развития продаж в этом сектора торговли. Для оценки сегментирования потребительского рынка на основе модели совершения покупок создаем новый файл Python и импортируем необходимые пакеты.

2. Выполняем оценку параметра «ширина окна входных данных».

3. Проводим обучение данных на основе модели сдвига среднего.

4. Определяем центры каждого кластера, число кластеров и выводим на экран полученные данные.

5. Выводим на экран количество и центры кластеров.

4. Порядок выполнения работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическое задание № 3 «Обучение с подкреплением»

1. Цель работы

Изучить метод обучения с подкреплением. Понять разницу между обучением с подкреплением и обучением с учителем. Научиться практическому использованию изученных методов для решения практических задач.

В процессе выполнения работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляет обучение с подкреплением;
- узнать, в чем состоит отличие обучения с подкреплением и обучения с учителем;
- изучить примеры использования методов обучения с подкреплением для решения практических задач;
- изучить структуру процесса обучения с подкреплением;
- научиться создавать программную среду и агента обучения.

2. Теоретическое введение

2.1. Понятие обучения с подкреплением

Концепция обучения имеет фундаментальное значение для искусственного интеллекта. Мы хотим добиться того, чтобы машины могли обучаться и самостоятельно действовать после обучения. Для человека обучение происходит в процессе наблюдения и взаимодействия с окружающей средой. Взаимодействуя со своим окружением, мы стараемся собрать как можно больше информации о происходящих вокруг нас событиях, изучаем причинно-следственные связи между наступлением определенного события и его возможных последствиях, изучаем какие действия необходимо предпринять необходимого результата, накапливаем знания об окружении и учимся реагировать на различные события.

Таким образом, можно сказать, что человек обучается в процессе своего взаимодействия с окружающей средой для того, чтобы предпринимать действия, способные обеспечить достижение заданной цели.

Обучение с подкреплением - это процесс, в ходе которого программный агент, подобно человеку обучается, исследует, что нужно сделать для достижениязаданной цели, и какая последовательность действий позволяет получить наилучший результат в данной ситуации. При этом, в отличие от других видовобучения агент не получает никакой информации о том, какие действия должны быть предприняты для достижения определенных результатов. Он должен самостоятельно, методом «проб и ошибок» определить ту последовательность действий, которая позволяет добиться наилучшего (оптимального) результата.

2.2. Обучение с подкреплением и обучение с учителем

Несмотря на некоторое внешнее сходство эти два подхода к обучению имеют принципиальное отличие. Так, в процессе обучения с учителем агент может использовать помеченные данные, полученные извне. А при обучении с подкреплением агент должен сам добывать необходимую информацию о своем окружении, учиться на своем опыте взаимодействия с окружающей средой.Очевидно, что разница между этими установками весьма существенная. Методы обучения с подкреплением могут эффективно использоваться для решения тех задач, в которых недостаточно входных данных и их нужно находить уже в процессе работы.

При этом в организации процесса обучения с подкреплением существует масса нерешенных вопросов. Одним из важнейших вопросов является нахождение и поддержание баланса в процессе обучения. Как определить оптимальную глубину просмотра возможных вариантов действий, каким образом выбрать оптимальную стратегию изучения окружающей среды? Ответ на эти и другие вопросы существенно влияет на эффективность обучения и скорость достижения поставленных целей. Например, если ставить приоритетной задачей получение наибольшей выгоды, то следует отдавать предпочтение хорошо знакомым действиям с известным результатом. В то же время, чтобы достичь наилучшихрезультатов агент должен пробовать выполнять новые действия, результат которых еще не известен.

Рассмотрим круг задач, для решения которых могут эффективно применяться

методы обучения с подкреплением.

Наиболее подходящий пример – это шахматы. Чтобы определить наилучший ход, игроки должны учесть множество факторов, при этом число возможных ходов настолько велико, что невозможно выполнить их полный перебор. Длятрадиционных методов обучения необходимо задавать большое количество правил для описания всех возможных ситуаций, в случае же использования обучения с подкреплением эта проблема снимается, поскольку существующий агент обучается самостоятельно в процессе игры.

Еще одна перспективная область применения методов обучения с подкреплением – это робототехника. В процессе жизнедеятельности роботу приходится постоянно принимать различные решения в незнакомой обстановке. Робот должен самостоятельно делать выбор возможного действия и использование в данном случае методов обучения с подкреплением дает ему такую возможность.

2.3. Процесс обучения с подкреплением

В общем случае одну итерацию процесса обучения с подкреплением можно описать в виде такой последовательности:

- Проверка текущего состояния для получения сведений об окружающей среде и сравнение его с заданным набором состояний.
- Выбор оптимальной стратегии из существующего набора вариантов на основании имеющей информации и текущего состояния окружающей среды.
- Совершение действия, предусмотренного выбранной стратегией.
- Получение реакции от окружающей среды на совершенное действие (получение подкрепления или вознаграждения).
- Сохранение информации о полученном вознаграждении посредством создания бинарного отношения "состояние - действие".

Таким образом, системы обучения с подкреплением выполняют одновременно несколько задач: обучение методом «проб и ошибок», изучение окружающей среды и использование реакции этой среды для корректировки и определения своих следующих шагов.

3. Практическая часть

3.1. Создание окружения

1. Для создания агентов обучения с подкреплением мы будем использовать пакет OpenAI Gym. Создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем основную функцию и анализируем входные аргументы. Создадим отображение входных аргументов на имена окружений, определенных в пакете OpenAI Gym.

3. Создаем окружение на основании входного аргумента и устанавливаем его состояние.

4. Выполняем заданное число итераций алгоритма.

5. После запуска программы откроется окно, в котором отображается обратный маятник, движущийся вправо от центра.

6. Изменяем значения аргумента и выполняем новый прогон программы.

7. Результат работы модели будет отображаться на экране.

3.2. Создание агента обучения

1. Создадим агента, способного к обучению для достижения поставленной цели. Создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем функцию и выполняем анализ входных аргументов. Задаем параметры визуализации результатов работы алгоритма.

3. Создаем отображение входных аргументов на имена окружений, определенных в пакете OpenAI Gym.

4. Выполняем заданное число итераций алгоритма. Затем выполняем сброс значений параметров и вновь запускаем выполнение алгоритма.

5. Проводится оценка результатов наблюдения на данной итерации и, в зависимости от полученной оценки предпринимается одно из набора доступных действий.

6. Оцениваем результаты выполнения выбранного действия и проверяем, достигнута ли поставленная цель. Если нет, то повторяем описанный набор действий снова.

7. После завершения работы алгоритма на экране отображаются численные значения параметров алгоритма и результаты его работы.

4. Порядок выполнения

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическое задание № 4
«Глубокое обучение и сверточные нейронные сети»

1. Цель работы

Изучить различные модели, архитектуру и принципы работы искусственных нейронных сетей. Ознакомиться с некоторыми наиболее распространенными подходами к решению задач с использованием нейронных сетей. Научиться практическому использованию моделей нейронных сетей для решения задач построения линейной регрессии и распознавания изображений.

В процессе выполнения работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляют нейронные сети;
- узнать, чем отличаются традиционные и сверточные нейронные сети;
- изучить архитектуру и принципы работы сверточных нейронных сетей;
- научиться решать задачу линейной регрессии с помощью с однослойной нейронной сети типа перцептрон;
- изучить типы слоев сверточных нейронных сетей;
- научиться решать задачу распознавания изображений с использованием однослойной нейронной сети типа перцептрон;
- научиться решать задачу распознавания изображений с использованием многослойной сверточной нейронной сети;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Понятие сверточных нейронных сетей

Прежде чем перейти к рассмотрению принципов организации и работы сверточных нейронных сетей рассмотрим кратко, что собой представляют искусственные нейронные сети. Нейронные сети состоят из нейронов, характеризующихся значениями весов и смещениями. Эти значения задаются вначале, в момент создания модели нейронной сети, и в процессе ее обучения

изменяются (настраиваются) с целью улучшения работы нейросети. Каждый нейрон имеет входы и выходы. На вход нейрона поступает определенный набор данных, которые затем в подвергаются некой обработке и преобразуются в выходные данные.

Нейронная сеть может иметь разное число слоев. Простейшие модели (типа перцептрона) имеют один слой. Нейронные сети, состоящие большого количества слоев, принято называть глубокими нейронными сетями, а научное направление, занимающееся исследованиями таких сетей, называется глубоким обучением.

При использовании обычных нейронных сетей входные данные преобразовываются одномерный массив, который распространяется от входа нейронной сети через все слои к выходу. При этом каждый следующего слоя нейрон связан со всеми нейронами предыдущего слоя. Выходной слой сети формирует результирующий сигнал.

Такая структура определяет основной недостаток обычных нейронных сетей – все данные подаваемые на вход нейросети преобразуются в одномерный массив независимо от их исходной структуры. Такой подход приемлем для обычных данных, но отрицательно влияет на качество получаемых результатов при решении задач распознавания изображений.

Исправить это положение призваны исправить сверточные нейронные сети, которые работать с двумерными структурами. Например, нам необходимо распознать изображения в черно-белых тонах с градацией серого. Эти изображения можно представить в виде двумерных массивов данных, используя при этом информацию о пространственном расположении пикселей. Обычные нейронные сети не имеют возможностей использовать эту информацию, теряя из-за этого множество базовых закономерностей (шаблонов).

Сверточные нейронные сети также состоят из нейронов, имеющих веса и смещения. Задача нейронной сети заключается в преобразовании необработанных данных, поступающих на вход нейросети к корректному виду на выходе. Сверточные нейронные сети отличаются от обычных типом используемых слоев и способом обработки входных данных. Предполагается, что входные данные являются изображениями, что позволяет извлекать специфические только для них

свойства.

2.2. Архитектура сверточных нейронных сетей

В сверточных нейронных сетях структура изображений учитывается явным образом. Нейроны в таких сетях организованы в трех измерениях: ширина, высота и глубина. Каждый нейрон текущего слоя соединен с небольшим фрагментом выхода предыдущего слоя, что можно сравнить с применением к входному изображению фильтра размером N □ N, что способствует сокращению размерности задачи по сравнению полносвязной моделью, когда каждый нейрон соединен со всеми нейронами предыдущего слоя.

Поскольку одиночный фильтр не в состоянии уловить все нюансы изображения такая операция выполняется M раз. Эти M фильтров действуют в качестве экстракторов признаков. Более глубокие слои сети способны извлекать более высокоуровневые признаки.

2.3. Типы слоев сверточных нейронных сетей

Обычно используются следующие типы слоев.

- Входной слой принимает необработанные данные изображений в том виде, в каком они есть.
- Сверточный слой вычисляет свертки между нейронами и различными фрагментами входного изображения.
- Слой скорректированных линейных блоков (выпрямитель) применяет функцию активации к выходному сигналу предыдущего слоя. Обычно используют функции наподобие max(0, *x*). Этот слой необходим для добавления нелинейности в сеть, чтобы она допускала обобщение на любой тип функций.
- Объединяющий слой семплирует выходные данные предыдущего слоя для получения структуры меньшей размерности. Такое объединение способствует сохранению наиболее существенных частей информации по мере ее прохождения.
- Полносвязный слой вычисляет выходные оценки. Результирующий выходной сигнал имеет размеры 1□1□*L*, где *L* - количество классов в тренировочном наборе данных.

По мере удаления от входного слоя и приближения к выходному изображение трансформируется из пиксельных значений в окончательные оценки классов. Точность и надежность модели зависят от многих факторов: типа слоев, глубины сети, взаимного расположения слоев различного типа внутри сети, выбора функций активации для каждого слоя, тренировочных данных и т.п.

3. Практическая часть

3.1. Создание модели линейной регрессии на основе перцептрона

1. Рассмотрим задачу создания модели линейной регрессии на основе нейронной сети типа перцептрон. Мы будем использовать библиотеку TensorFlow. Эго популярный набор средств глубокого обучения, который широко применяется для построения различных прикладных систем. Создаем новый файл Python и импортируем необходимые пакеты. В процессе выполнения работы мы будем генерировать отдельные объекты данных и проверять, насколько они будут соответствовать используемой модели.

2. Задаем объем генерируемых данных и значения необходимых параметров. Для простоты выбираем в качестве рабочей модели линейную модель: $y = mx + c$.

3. Задаем уровень шума для получения различных вариантов данных, вычисляем значение выходного параметра y и после получения набора данных разделяем его на входной и выходной.

4. С помощью генератора случайных чисел с равномерным законом распределения генерируем значения весов. Величины смещений пока принимаем равными нулю.

5. Используя переменные TensorFlow записываем уравнения и определяем функцию потерь, которую буде использовать в процессе обучения.

6. Формируем блок оптимизации значений функции потерь на основе алгоритма градиентного спуска.

7. Выполняем инициализацию переменных и начинаем процесс обучения нейросети.

8. В процессе решения на экране монитора будут отображаться данные и графики, отражающие текущее состояние модели и изменения, происходящие под

воздействием обучения. В случае правильной настройки параметров по мере увеличения числа пройденных итераций значение функции потерь должно улучшаться.

9. Строим график сгенерированных данных и накладываем на него для сравнения выбранную линейную модель.

3.2. Создание классификатора изображений на основе однослойной нейронной сети

1. Рассмотрим задачу построения однослойной нейронной сети с использованием библиотеки TensorFlow и создания модели классификации изображений на ее основе. Создаем новый файл Python и импортируем необходимые пакеты. В качестве входных данных используем файл содержащий изображения рукописных цифр. Целью работы является создание модели, способной корректно идентифицировать каждую цифру из заданного набора.

2. Выбираем функцию и выполняем анализ входных аргументов.

3. Извлекаем данные из файла с изображениями. Выбираем прямое кодирование, то есть каждый объект данных будет характеризоваться массивом данных длины n, где g – соответствует выбранному числу классов. Для указания на соответствие некоторому классу используем бинарный индекс: 1 для выбранного класса и 0 для всех остальных.

4. Преобразуем хранящиеся в файле изображения в одномерный массив и создаем входной слой нейронной сети. Создаем однослойную нейронную сеть с весами и смещениями. Число нейронов в выходном слое будет равно 10, по числу образцов чисел, хранящихся в файле.

5. Выбираем функцию для обучения нейронной сети, функцию потерь и метод оптимизации на основе алгоритма градиентного спуска.

6. Инициализируем все переменные и начинаем процесс обучения. На каждой итерации сначала выбирается очередной фрагмент изображения, затем выполняется оптимизация изображения и алгоритм переходит к выбору следующего фрагмента.

7. После завершения процесса обучения вычислим точность процесса распознавания, используя для этого тестовый набор данных.

3.3. Создание классификатора изображений на основе сверточной нейронной сети

1. Анализ результатов, показанных нейронной сетью типа перцептрон, использованной в предыдущем примере показали ее недостаточную эффективность. Поэтому попробуем добиться лучших результатов на том же наборе входных данных, используя для решений той же задачи сверточную нейронную сеть. Создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем виды функций, необходимых для успешной работы модели:

- функцию анализа входных аргументов;
- функцию создания значений весов в каждом слое;
- функцию создания значений смещений в каждом слое;
- функцию создания слоя на основе входной формы;
- функцию выполнения 2D-свертки.
- функцию, выполняющую операцию объединения;
- основную функцию.

3. Извлекаем данные из файла с изображениями, анализируем входные аргументы.

4. Создаем входной слой с 784 нейронами. Будем использовать сверточную нейронную сеть, использующую двумерную структуру изображений. Поэтому преобразуем параметр *x* в четырехмерный тензор, второе и третье измерения которого соответствуют размерам изображения.

5. Создаем первый сверточный слой, который будет извлекать 32 признака для каждого фрагмента изображения размером 5□5.

6. Выполняем свертку изображения с использованием тензора весов, вычисленного на предыдущем шаге, а затем добавляем тензор смещений. После этого применяем к полученному результату функцию активации типа "выпрямитель". Применяем оператор max_pooling размерностью 2□2 к результату, полученному на предыдущем шаге.

7. Создаем второй сверточный слой, вычисляющий 64 признака для каждого из блоков размером 5□5. После чего повторяем набор действий писанных на

предыдущем шаге. В результате выполнения вышеописанных действий размер изображения уменьшается до значений 7□7.

8. Создаем полносвязный слой с 1024 нейронами. Преобразуем результат, полученный на предыдущем слое. Для этого умножаем результат предыдущего слоя на тензор весов текущего слоя, добавляем тензор смещений и применяем к полученному результату функцию активации типа "выпрямитель".

9. Создаем исключающий слой для того, чтобы не допустить переобучения сети. Создаем механизм заполнения значений, определяющих вероятность сохранения текущего результата в случае исключения соответствующих нейроновиз слоя.

10. Определяем считывающий слой с десятью выходными нейронами, соответствующими десяти различным классам (цифрам) во входном наборе данных.

11. Вычисляем результат, определим функцию потерь и функцию оптимизатора. Определяем способ вычисления точности.

12. Выполняем инициализацию переменных и начинаем процесс обучения модели нейросети.

13. Выполняем процедуру оптимизации для текущего блока.

14. Задаем параметры визуализации результатов работы алгоритма. По завершении процесса обучения определяем точность распознавания, используя тестовый набор данных.

15. Выполняем анализ полученных результатов и сравнение текущих результатов с результатами работы однослойной нейронной сети на том же наборе входных данных.

4. Порядок выполнения работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания к практической работе.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическая работа № 5 «Предсказательная аналитика на основе ансамблевого обучения»

1. Цель работы

Изучить методы ансамблевого обучения. Ознакомиться с наиболее распространенными методиками организации ансамблевого обучения. Изучить возможности использования методов ансамблевого обучения для решения практических классификации и прогнозирования.

В процессе выполнения практической работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляет ансамблевое обучение;
- изучить методы создание моделей с помощью ансамблевого обучения;
- понять, что такое деревья решений;
- научиться создавать модели задачи классификации данных методы на основе деревьев решений;
- узнать, что такое случайные и предельно случайные леса;
- изучить методы построения моделей классификации и регрессии на основе случайных и предельно случайных лесов;
- изучить метод выбора оптимальных обучающих параметров с помощью сеточного поиска;
- научиться решать дисбаланса классов;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Понятие ансамблевого обучения

Ансамблевое обучение – это процесс построения множества моделей и нахождения такой комбинации отдельных моделей, которая позволяет улучшить результаты, по сравнению с теми, которые показывает каждая из этих моделей по отдельности. В качестве элементов такого ансамбля могут использовать модели решения различных задач, например модели классификации, регрессии,

кластеризации и т.д.

Эффективность ансамблевого обучения обусловлена тем, что этот метод позволяет минимизировать суммарный риск выбора неудачной модели. Поскольку при данном подходе обучение происходит на наборе разнородных данных, мы можем в перспективе получать хорошие результаты, такая обобщенная модель оказывается более точной, чем ее отдельные компоненты. Также можно отметить, что разнообразие получаемых результатов достигается за счет использования разных обучающих параметров. То есть каждая модель использует разные правила логического вывода и, если эти модели согласованы между собой, то результат является корректным.

2.2. Деревья принятия решений

Дерево принятия решений - это иерархическая структура, состоящая из вершин и ветвей. В результате можно разделить исходный набор данных на ветви с возможностью принятия решений на каждом иерархическом уровне. Полный вариант решения можно получить в процессе движения по дереву решений сверху вниз. Процесс принятия решений начинается с самой верхней (корневой) вершины дерева. Вершины дерева представляют собой правила, которые определяются исходя из соотношений между входными данными и целевыми метками в тренировочном наборе.

Идея использования дерева решений состоит в том, чтобы на каждом иерархическом уровне при движении от корневой вершины вниз мы на шаг приближаемся к итоговому результату.

2.3. Случайные и предельно случайные леса

Случайный лес – это частный случай ансамблевого обучения, когда отдельные модели ансамбля создаются с использованием деревьев решений. В процессе создания деревьев могут использоваться случайные подмножества тренировочных данных, что обеспечивает разброс данных между различными деревьями решений. Полученный ансамбль используется для прогнозирования результата.

Основным достоинством случайных лесов является то, что для них не

случается переобучения. При построении дерева решений его узлы последовательно расщепляются, и для них выбираются наилучшие пороговые значения, позволяющие уменьшать энтропию (неопределенность) на каждом уровне. В процессе расщепления узлов учитываются не все признаки, характеризующие данные входного набора. Вместо этого выбирается наилучший способ расщепления узлов, основанный на текущем случайном поднаборе рассматриваемых признаков. В***предельно случайных лесах*** случайным образом задаются не только значения признаков, но и пороговые значения. Эти случайно генерируемые значенияуменьшают вариативность модели, поэтому использование предельно случайныхлесов обычно приводит к более гладким границам принятия решений.

Иногда при работе с N-мерными массивами возможно разделить данные по критерию их важности и за счет этого исключить из рассмотрения некоторые второстепенные. Это позволит нам уменьшить сложность и размерность решаемой задачи. Для вычисления относительной важности отдельных признаков данных в нашей работе мы используем модель регрессии "AdaBoost". Данная модельпозволяет выбирать объекты из набора данных на основе, используя некоторое распределение их весов. Это распределение на каждой итерации работы модели изменяется. В процессе работы выделяются и отслеживаются объекты, которые были классифицированы неверно. Им присваиваются большие значения весовых коэффициентов. На каждой итерации решение принимается на основании взвешенного большинства голосов.

3. Практическая часть

3.1. Создание модели классификации на основе дерева принятия решений

1. Для создания модели классификации на основе дерева принятия решений, создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, каждая строка которого значения, разделенные запятой. Первые два значения соответствуют входным данным, последнее - целевым меткам. Загрузим данные из этого файла.

2. Разделяем входные данные на два класса в соответствии с имеющимися метками. Разбиваем набор входных данных на обучающий и тестовый наборы.

3. Создаем модель классификации на основе данных обучающего набора. Задаем значения необходимых параметров определяющих максимальную глубину дерева принятия решений.

4. Запускаем процесс решения и выводим на экран полученные значения для тестового набора.

5. Выполняем оценку работу модели при решении задачи классификации. Для чего используем значения таких параметров как точность классификации, полнота (т.е. процент числа определенных элементов по отношению к ожидаемому количеству) и F-мера, которая представляет собой гармоническое среднее показателей точности и полноты. Данный параметр позволяет получить наиболее сбалансированную оценку качества работы модели.

3.2. Создание модели классификации на основе случайных и предельно случайных лесов

1. Рассмотри примеров создания модели классификации на основе случайных и предельно случайных лесов. Для этого создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, каждая строка которого содержит значения, разделенные запятой, всего три класса данных. Первые два значения соответствуют входным данным, последнее - целевым меткам. Загружаем данные из этого файла.

2. Определяем синтаксический анализатор (парсер) аргументов, который определяет тип используемой модели классификации, исходя из значения задаваемого входного параметра.

3. Выбираем функцию оценки и набор входных аргументов.

4. Разбиваем исходные данные на обучающий и тестовый наборы.

5. Определяем значения параметров алгоритма: число деревьев, число уровней (глубина) в каждом дереве, значение для генератора случайных чисел. В зависимости от выбранного значения входного параметра используется модель классификации на основе случайного или предельно случайного леса.

6. Производим обучение модели.

7. Вычисляем результаты работы модели на тестовом наборе данных и задаем

параметры визуализации полученных результатов.

8. Выполняем анализ полученных результатов и сравнение результатов работы модели на основе случайного и предельно случайного леса.

3.3. Оценка мер достоверности прогнозов

При выводе результатов на экран монитора, там также отображаются значения вероятностей, отражающих уровень доверия по каждому классу данных. В модели классификации имеется встроенный метод, который можно использовать вычисления и оценки значений уровня доверия для различных классов данных.

1. Для изучения данной возможности выполним визуализации для набора данных из предыдущего примера.

2. Для каждой точки вычисляем вероятность ее принадлежности каждому из трех классов, после чего выбираем тот класс, которому соответствует самый высокий уровень доверия.

3.4. Обработка дисбаланса классов

1. Качество работы модели классификации определяется качеством используемых данных. Для эффективной работы модели необходимо чтобы набор включал в себя равное количество данных для каждого класса. Однако соблюсти это требование не всегда возможно. В этом случае необходимо учитывать возможный дисбаланс. Рассмотрим работу этой процедуры на конкретном примере. Для этого создаем новый файл Python и импортируем необходимые пакеты. В качестве источника данных используем текстовый файл, каждая строка которого содержит значения, разделенные запятой, в наборе выделены два класса данных. Первые два значения соответствуют входным данным, последнее - целевым меткам. Загрузим данные из этого файла.

2. Разбиваем исходные данные на обучающий и тестовый наборы.

3. Определим числовые параметры модели на основе предельно случайных лесов, управляющие оценкой и обработкой возможного дисбаланса.

4. Создаем модель классификации и производим ее обучение на наборе тренировочных данных.

5. Вычисляем показатели эффективности работы модели и выводим на экран монитора отчет о результатах классификации.

3.5. Нахождение оптимальных обучающих параметров с помощью сеточного поиска

1. Для улучшения процесса выбора оптимальных параметров задачи классификации иногда используют сеточный поиск. Рассмотрим, каким образом это происходит на примере. Для этого создаем новый файл Python и импортируем необходимые пакеты. Входные данные для работы импортируем из имеющегося файла.

2. Выделим в исходных данных 3 класса и разделим данных на обучающий и тестовый наборы.

3. Зададим сетку значений параметров для тестирования модели. В процессе тестирования мы фиксируем значение одного из параметров задачи, изменяя в то же время значения остальных параметров. Процедура повторяется для каждого из параметров. Таким образом, определяем значения параметров соответствующие оптимальной комбинации.

4. Для каждого параметра выполняем сеточный поиск и обучаем модель на конкретной комбинации параметров.

5. Выводим на экран полученную оценку для каждой комбинации параметров и общий отчет о результатах работы модели для различных оценочных значений.

3.6. Вычисление относительной важности признаков

1. Рассмотрим работу алгоритма определения относительной важности отдельных признаков на примере. Для этого создаем новый файл Python и импортируем необходимые пакеты. В качестве исходных данных используем предлагаемый файл. Для реалистичности перемешаем случайным образом имеющийся набор данных.

2. Разделяем имеющийся набор данных на обучающий и тестовый наборы.

3. Выбираем модель регрессии AdaBoost и выполняем обучение модели.

4. Выполняем оценку эффективность данного метода.

5. Выполняем нормализацию полученных значений и проводим относительной важности параметров.

3.7. Прогнозирование интенсивности дорожного движения с помощью классификатора на основе предельно случайных лесов

1. Рассмотрим работу модели классификации на основе предельно случайных лесов для решения задачи прогнозирования интенсивности дорожного движения. В качестве исходных данных используем предлагаемый файл, каждая которого строка содержит строковые значения, разделенные запятой. Значения в строке отформатированы следующим образом: день недели, время суток, место на дороге, двоичная переменная показывающая начался ли час пик (да или нет), число проезжающих транспортных средств.

Задача состоит в получении прогнозного решения о возможном объеме трафика на основании предоставленной информации. Для решения данной задачи создаем модель регрессии на основе предельно случайных лесов. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Содержащиеся в наборе нечисловые переменные нуждаются в кодировании. При этом для каждого признака, нуждающегося в кодировании, необходимо предусмотреть отдельный алгоритм.

3. Разбиваем данные на обучающий и тестовый наборы.

4. Выполняем обучение модели регрессии на основе предельно случайных лесов.

5. Вычисляем показатели эффективности работы модели на тестовых данных.

6. Выбираем параметры визуализации работы модели и выводим на экран монитора отчет о результатах работы.

4. Порядок выполнения работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания к работе.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическая работа № 6 «Создание рекомендательных систем»

1. Цель работы

Изучить принципы работы и подходы к созданию рекомендательных систем. Ознакомиться с некоторыми наиболее распространенными методами и алгоритмами построения рекомендательных систем. Изучить предложенные методы на простых примерах построения рекомендательных систем на основе анализа предпочтений.

В процессе выполнения практической работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляют рекомендательные системы;
- понять, что собой представляет обучающий конвейер;
- изучить метод извлечение ближайших соседей;
- изучить методы создания модели классификации с использованием метода K ближайших соседей;
- научиться производить вычисление оценок сходства;
- изучить метод решения задачи классификации данных с помощью «наивного байесовского классификатора»;
- узнать, что такое коллаборативная фильтрация и научиться использовать этот метод для поиска пользователей с похожими предпочтениями;
- научиться применять изученные методы на примере построения рекомендательной системы для подбора фильмов;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Обучающий конвейер

Обычно системы машинного обучения строятся на модульной основе. Конкретная конечная цель достигается за счет формирования подходящих комбинаций отдельных модулей. В библиотеке scikit-learn содержатся функции, позволяющие объединять различные модули в единые конвейерные цепочки. Нам остается лишь указать нужные модули вместе с соответствующими параметрами.

Далее на основе этих модулей создается конвейер, который обрабатывает данные и тренирует систему.

Конвейер может формироваться из модулей, выполняющих различные функции, такие как отбор признаков, предварительная обработка данных, построение случайных лесов, кластеризация и т.п.

2.2. Классификация методом K ближайших соседей

Для создания выводов в рекомендательных системах широко распространен метод ближайших соседей. Идея метода состоит в выборе точек набора данных, расположенных наиболее близко к рассматриваемой точке.

На этом принципе построена модель классификации, в которой разбиение объектов на классы выполняется на основе принципа ближайших соседей. Алгоритм находит в обучающем наборе множество точек, являющихся ближайшими по отношению к исследуемой точке; для назначения класса проводится процедура "голосования". После чего для каждого объекта входного набора выбирается тот класс, которому соответствует наибольшее число "голосов".

2.3. Вычисление оценок сходства

При построении рекомендательных систем очень важную роль играет выбор способа сравнения различных объектов, входящих в набор данных. Для этого используются оценки сходства. Оценка сходства дает представление о том, в какой степени два объекта могут считаться аналогичными друг другу. Для этой цели часто используют оценки двух типов: евклидовы и по Пирсону. В основу евклидовой оценки положено евклидово расстояние между двумя точками данных. Евклидово расстояние не ограничено по величине. Поэтому мы берем соответствующее значение и преобразуем его таким образом, чтобы новое значение находилось в диапазоне от 0 до 1. Если евклидово расстояние между двумя объектами велико, то соответствующая евклидова оценка должна иметь небольшую величину, поскольку низкая оценка указывает на малую степень сходства между объектами.Следовательно, евклидово расстояние обратно пропорционально евклидовой оценке.

Оценка сходства по Пирсону - это математическая мера корреляции двух объектов. Для ее вычисления используют ковариацию двух объектов и их индивидуальные стандартные отклонения. Значения этой оценки могут изменяться в пределах от -1 до +1. Оценка +1 указывает на высокую степень сходства объектов, тогда как оценка -1 - на большие различия между ними. Оценка 0 свидетельствует об отсутствии корреляции между двумя объектами.

2.4. Метод коллаборативной фильтрации

Термин ***коллаборативная фильтрация*** относится к процессу идентификации шаблонов поведения объектов набора данных с целью принятия решений относительно нового объекта. В контексте рекомендательных систем метод коллаборативной фильтрации используют для прогнозирования предпочтений нового пользователя на основании имеющейся информации о предпочтениях других пользователей с аналогичными вкусами.

Составляя прогнозы для предпочтений индивидуальных пользователей, мы используем имеющуюся совместную информацию о предпочтениях других пользователей. Именно поэтому данный метод фильтрации называется коллаборативным.

В данном случае основное допущение заключается в том, что если два человека дают одинаковые рейтинговые оценки одному и тому же объекту или явлению, то их оценки других объектов также будут примерно одинаковыми. Как правило, коллаборативную фильтрацию применяют, когда имеют дело с наборами данных большого размера.

3. Практическая часть

3.1. Создание обучающего конвейера

1. Рассмотрим пример создания конвейера для выбора наиболее важных K признаков из входных данных и их последующей классификации с использованием модели на основе предельно случайного леса. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Используя встроенную функцию из программного пакета scikit-learn

генерируем маркированные выборочные данные для процессов обучения и тестирования. Числовые значения в каждом векторе признаков генерируются с использованием генератора случайных выборок. Каждая точка данных включает шесть информативных признаков и не содержит ни одного избыточного.

3. Создаем конвейер путем объединения блока селекции признаков, который выбирает k "лучших" признаков и блока классификации на основе предельно случайного леса.

4. Задаем числовые значения параметров блоков. Эти значения могут изменяться в процессе работы.

5. Выполняем обучение конвейера, используя сгенерированные перед этим выборочные данные.

6. В результате получаем прогнозные результаты для всех входных значений. Вычисляем оценку на основе маркированных тренировочных данных.

7. Выводим на экран монитора отчет о результатах работы конвейера.

3.2. Извлечение ближайших соседей

1. Рассмотрим пример нахождения ближайших соседей заданной точки данных. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем набор точек и число ближайших соседей для сравнения.

3. Выбираем тестовую точку, для которой будем извлекать k ближайших соседей. Задаем параметры визуализации входных данных.

4. Создаем модель и выполняем ее обучение на основе метода K ближайших соседей на наборе входных данных.

5. После завершения процесса обучения применяем обученную модель для извлечения ближайших соседей относительно тестовой точки.

6. Выбираем параметры и выполняем визуализацию результатов работы модели, выводим на экран тестовую точку и ее ближайших соседей.

3.3. Создание модели классификации методом K ближайших соседей

1. Рассмотрим пример создания модели классификации с использованием метода K ближайших соседей. Для этого создаем новый файл Python и импортируем

необходимые пакеты. Входные данные загружаем из предоставленного текстового файла. Каждая строка текстового файла содержит значения, разделенные запятой, данные разбиты на четыре класса.

2. Визуализируем входные данные, используя четыре маркера различной формы.

3. Задаем число точек ближайших соседей, которое будет использовать алгоритм, а также шаг сетки для визуализации границ.

4. Создаем модель классификации методом K ближайших соседей.

5. Выполняем обучение модели на тренировочных данных.

6. Решаем задачу классификации для всех точек набора.

7. Задаем параметры визуализации результатов классификации.

8. Для оценки эффективности работы модели выбираем тестовую точку. Извлекаем K ближайших соседей тестовой точки с помощью обученной модели классификации на основе K ближайших соседей.

9. Настраиваем параметры визуализации результатов работы модели. Выводим на экран монитора предсказанный результат.

3.4. Вычисление оценок сходства

1. Рассмотрим пример вычисления оценок сходства. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Создадим парсер для обработки входных аргументов. Ими будут служить имена двух пользователей и тип оценки, которая будет использоваться для вычисления степени сходства.

3. Создаем функцию оценки сходства на основе евклидова расстояния. Для этого используем значение квадрата разности рейтинговых оценок.

5. Выбираем функцию для оценки сходства по критерию Пирсона.

4. Выбираем переменную для отслеживания фильмов, получивших рейтинговую оценку, и извлекаем такие фильмы. В случае отсутствия таких фильмов, вычисление оценки сходства не производится.

6. Вычисляем значения параметров, необходимых для оценки сходству на основе евклидова расстояния и по критерию Пирсона.

7. Выбираем основную функцию и анализируем входные аргументы.

8. Вычисляем оценку сходства на основании входных аргументов.

9. Настраиваем параметры визуализации результатов работы модели. Выводим на экран монитора полученные результаты.

10. Изменяем значения параметров и их сочетания и повторяем вышеописанную последовательность действий. Сравниваем новые результаты с полученными ранее.

3.5. Поиск пользователей с похожими предпочтениями методом коллаборативной фильтрации

1. Рассмотрим пример решения задачи поиск пользователей с похожими предпочтениями методом коллаборативной фильтрации. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем функцию для парсинга входного аргумента «имя пользователя».

3. Задаем функцию для поиска пользователей, аналогичных указанному. Для оценки сходства используем критерий Пирсону. Вычисляем на основе критерия Пирсона сходство между указанным пользователем и всеми остальными пользователями в наборе данных.

4. Выполняем сортировку полученных оценок по убыванию. Выбираем n первых позиций из данного списка.

5. Определяем основную функцию и анализируем входные аргументы, чтобы извлечь имя пользователя.

6. Находим первых трех пользователей, аналогичных пользователю, указанному в качестве входного аргумента.

7. Выводим на экран монитора имена пользователей вместе с оценками сходства.

3.6. Создание рекомендательной системы для выбора фильмов

1. Рассмотрим работу рекомендательных систем на примере создания рекомендательной системы для просмотра фильмов. В качестве исходных данных используем предложенный файл, содержащий информации о пользователях и

оценках, данных ими различным фильмам. Задача состоит в том, чтобы создать систему, которая могла выдавать рекомендации конкретному пользователю, относительно подбора фильмов для просмотра. Таким образом, модель должна уметь находить похожих по предпочтениям пользователей в имеющемся набореданных и использовать информацию об их предпочтениях для формирования соответствующей рекомендации. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем функцию для парсинга входного аргумента: «имя пользователя».

3. Выбираем функцию, которая будет получать рекомендации для указанного пользователя. Если информация об указанном пользователе отсутствует в наборе данных, генерируется исключение.

4. Определяем переменные для сравнения и вычисляем оценку сходства между указанным пользователем и всеми остальными пользователями в наборе данных. Если оценка сходства меньше 0, переходим к следующему пользователю.

5. Добавляем список фильмов, имеющих оценки, но еще неоцененных указанным пользователем.

6. Вычисляем взвешенную рейтинговую оценку для каждого элемента отфильтрованного списка. В случае отсутствия подходящих фильмов рекомендация не формируется.

7. Выполняем нормализаций оценочных значений на основании взвешенных оценок.

8. Выполняем сортировку оценок и формируем рекомендации.

4. Порядок выполнения работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания к практической работе.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическая работа № 7 «Обработка естественного языка»

1. Цель работы

Изучить методы и подходы к решению задачи обработки естественного языка. Ознакомиться с основными понятиями и операциями по обработке текста. Ознакомиться с некоторыми известными моделями обработки текстов. Изучить предложенные методы на простых примерах обработки фрагментов текста.

В процессе выполнения практической работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляет процедура токенизации текстовых данных;
- изучить методику преобразования слов в их базовые формы с помощью стемминга и лемматизации;
- понять методику разбиения текста на блоки;
- изучить методы формирования терм-документной матрицы с помощью модели Bag of Words;
- научиться создать модель построения прогнозов по категориям;
- изучить метод построения алгоритма анализа грамматических родов;
- узнать, что такое коллаборативная фильтрация и научиться использовать этот метод для поиска пользователей с похожими предпочтениями;
- научиться применять методы сентимент-анализа и тематического моделирования для получения оценочных прогнозов в процессе обработки текстов;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

2.1. Введение и установка пакетов

Обработка естественного языка является важной частью современных информационных систем. Основной задачей в области обработки естественного языка является разработка алгоритмов, которые позволяли бы компьютерам распознавать неструктурированный текст и понимать живую речь. Для понимания смысла конкретных предложений большое значение имеет контекст. Человек для

понимания контекста в таких случаях использует свои накопленные знания, искусственные системы же пока этого делать не умеют.

Для преодоления этой проблемы разрабатываются различные приложения на основе методов машинного обучения. Для работы таких приложений требуются огромные массивы текста, методы обучения алгоритма для выполнения набора необходимых задач, таких как категоризация текста, сентимент-анализ или тематическое моделирование. При этом алгоритмы учатся обнаруживать повторяющиеся шаблоны во входном тексте и извлекать содержащийся в нем смысл.

При работе с текстом происходит разбиение его на для проведения анализа. И здесь мы сталкиваемся с токенизацией. ***Токенизация*** - это процесс разбиения входного текста на отдельные фрагменты, такие как слова или предложения. Эти элементы называются токенами (лексемами). В зависимости от того, что мы хотим сделать, мы можем определить, собственные методы для разбиения текста на множество токенов.

В процессе анализа текста полезно извлекать корневые формы различных слов. Это позволяет получать полезные данные, помогающие анализировать входной текст. Одним из способов обеспечения этого является стемминг. Целью стемминга является приведение слов в их различных формах к общему корню. В основном это эвристический процесс, заключающийся в отсечении окончаний слов для выделения их корневых форм.

В данной работе мы будем использовать три разных алгоритма для выполнения процедуры стемминга. Различия между ними заключаются в степени строгости, используемой при получении базовой формы.

Наименее строгим из них является стеммер Портера, а наиболее строгим - стеммер Ланкастера. Стеммеры ведут себя по-разному. Результаты, полученные с помощью стеммера Ланкастера, немного сбивают с толку, поскольку он слишком урезает слова. В то же время этот алгоритм демонстрирует высокую скорость. В большинстве случае неплохо использовать стеммер Сноуболла, который обеспечивает разумный компромисс между быстродействием и строгостью.

Часто корневые формы, полученные с помощью процедуры стимминга не

имеют смысла. Для исправления данной ситуации используют другой способ. ***Лемматизация*** - еще один способ выделения корневых форм слов. Данная процедура выполняется с использованием словаря и морфологического анализа слов. Корневые формы слов получаются в данном случае путем удаления окончаний. Такую базовую форму любого слова называют ***леммой***. Необходимо также отметить, что конечный результат зависит от того, является слово глаголом или существительным.

В процессе обработки текстов входные данные обычно разбивают на блоки (чанки) для последующего анализа. Соответственно такую процедуру обычно называют чанкингом. При этом, в отличие от процедуры токенизации, при чанкинге не задаются никакие ограничения, а результирующие блоки обязаны иметь смысл. Условия разбиения текста на блоки могут меняться в зависимости от специфики решаемой задачи.

Одним из перспективных методов с точки зрения возможностей использования в машинном обучении является преобразование текста в числовую форму. Для этого используют алгоритм ***Bag of Words*** (мешок слов). Этот алгоритм создает словарь, состоящий из всех слов, которые встречаются в документе, и строит модель с помощью терм-документной матрицы. При этом мы сохраняем только счетчик слов, не учитывая их порядок следования и грамматические детали.

Терм-документная матрица – это таблица, которая содержит значения счетчиков различных слов, встречающихся в документе. Используя терм-документную матрицу, мы можем представить текстовый документ в виде взвешенной комбинации различных слов. Можно также устанавливать пороговые значения и выбирать наиболее значимые слова. В некотором смысле мы получаем гистограмму всех слов в документе, совокупность которых будет служить вектором признаков. Этот вектор признаков используется для классификации текста.

Модели прогнозирования категорий используются для предсказания категории, к которой принадлежит заданный элемент текста. Их часто используют при классификации текста для категоризации текстовых документов. Поисковые механизмы часто используют этот инструмент для поиска результатов по их релевантности.

Для построения такой модели прогнозирования мы будем использовать статистику использования слов в исследуемом тексте, что позволяет получить оценку важности данного слова.

Для анализа текста используют два параметра. Первый параметр это частота использования слова в тексте, она определяется как результат деления значения частоты появления исследуемого слова в тексте на величину равную общему количеству слов в тексте.

Второй параметр позволяет оценить уникальность слова в исследуемом документе. Его можно определить как отношение количества документов, содержащих данное слово, к общему количеству документов в коллекции.

Одной из задач, представляющих интерес в связи с обработкой текста, является идентификация грамматического рода. Для решения данной задачи можно эвристическим образом задать вектор признаков. Примером подобной эвристики может служить правило для анализа последних букв имени. Если последние буквы имения гласные: «а», «я» то, вероятнее всего, это женское имя. Имена, заканчивающиеся на согласные буквы, как правило, мужские.

Сентимент-анализ - это процесс определения тональности заданного фрагмента текста. Его, например, можно использовать для того, чтобы определить, является ли отзыв о кинофильме положительным или отрицательным. Это одно из наиболее популярных применений обработки естественного языка. В зависимости от конкретной задачи мы можем расширять количество категорий. Обычно эту методику используют для того, чтобы определить отношение людей к конкретному продукту, торговой марке или теме.

Для создания модели сентимент-анализа мы использовать в данной работе алгоритм наивной байесовской классификации. Мы извлекаем из текста все уникальные слова и создаем словарь. Затем разделяем данные на тренировочный и тестовый наборы и выполняем обучение модели классификации по тональности отзывов (положительные или отрицательные). В процессе работы выделяем наиболее информативные слова, указывающие на положительные или отрицательные отзывы.

Тематическое моделирование - это процесс идентификации шаблонов в

текстовых данных, которые соответствуют некоторой теме. Если текст содержит несколько тем, то эта методика может быть использована для идентификации и разделения тем в пределах входного текста, при этом не требуется наличие помеченных данных. С учетом огромных объемов текстовых данных, генерируемых в Интернете, тематическое моделирование приобретает особую важность, поскольку обеспечивает возможность суммаризации этих данных.

Латентное размещение Дирихле - это метод тематического моделирования, его основная идея состоит в том, что каждый фрагмент документа представляет собой комбинацию различных тем. Такое сочетание помогает идентифицировать данный текст в большом документе. Предполагается, что документы генерируются случайным образом на основе этих тем.

3. Практическая часть

Для изучения методов обработки естественного языка мы будем использовать пакет Python Natural Language Toolkit. Чтобы получить доступ ко всем наборам данных, предоставляемых в пакете, мы должны загрузить их.

В данной работе мы также будем использовать библиотеку средств семантического моделирования gensim. Для правильной работы пакета gensim нам может понадобиться установить пакет pattern.

3.1. Токенизация текстовых данных

1. Рассмотрим работу процедуры токенизации. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем входной текст, который будем использовать для токенизации.

3. Выполняем разбивку входного текста с помощью процедуры токенизации предложений.

4. Выполняем разбивку входного текста с помощью процедуры токенизации слов.

5. Выполняем разбивку входного текста с помощью процедуры токенизации слов и пунктуации.

3.2. Преобразование слов к базовой форме с помощью стемминга

1. Рассмотрим работу процедуры преобразования слов к базовой форме с помощью стемминга. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем входные слова.

3. Создаем объекты список имен стеммеров для их отображения в виде таблицы и соответствующим образом форматируем выходной текст.

4. Выполняем итеративный стемминг слов на трех различных моделях стеммеров: Портера, Ланкастера и Сноуболла.

5. Выводим полученные результаты на экран монитора.

3.3. Преобразование слов в их корневые формы с помощью лемматизации

1. Рассмотрим работу процедуры преобразования слов к базовой форме с помощью лемматизации. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. В данном разделе будем использовать тот же набор слов, что и при реализации процедуры семминга, чтобы можно было сравнить результаты.

3. Создаем модель процедуры лемматизации. Создадим список имен для их отображения в виде таблицы и соответствующим образом форматируем выходной текст.

4. Выполняем итеративную лемматизацию слов для существительных и глаголов.

5. Выводим полученные результаты на экран монитора.

3.4. Разбиение текстовых данных на информационные блоки

1. Рассмотрим работу процедуры разбиения текстовых данных на несколько блоков. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем функцию для операции разбиения, задаем значения параметров.

3. Выполняем операцию разбиения на блоки, используя входной параметр.

4. Выводим результат работы процедуры разбиения в виде списка.

3.5. Определение частотности слов с помощью модели Bag of Words

1. Рассмотрим работу процедуры определения частотности слов с помощью модели Bag of Words. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. В качестве входных данных используем имеющий файл.

3. Задаем число слов в каждом блоке.

4. Разбиваем входной текст на блоки.

5. Формируем терм-документную матрицу, содержащую сведения о значениях счетчиков каждого слова. Для этого используем метод CountVectorizer, которыйимеет два входных параметра: минимальная и максимальная частота появления каждого слова в документе.

6. Формируем словарь из отдельных слов, извлеченных на предыдущем шаге.

7. Генерируем отображаемые имена.

8. Выводим полученную терм-документную матрицу.

3.6. Создание модели прогнозирования категорий

1. Рассмотрим работу модели прогнозирования категорий. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем категории, которые будут использоваться для тренировки.

3. Формируем тренировочный набор.

4. Вычисляем значения счетчиков слов с помощью процедуры CountVectorizer.

5. Определяем выборку входных предложений, которые будут использованы для тестирования.

6. Выполняем обучение модели наивной байесовской классификации на наборе тренировочных данных.

7. Преобразуем входные данные, используя процедуру векторизации значений счетчиков.

8. Выполняем прогнозирование результата, используя вектор данных.

9. Выводим на экран монитора результаты для каждой категории входных тестовых данных.

3.7. Создание модели анализа грамматических родов

1. Рассмотрим работу модели анализа грамматических родов. Для этого создаем новый файл Python и импортируем необходимые пакеты. Загружаем данные из предложенного файла. Эти данные содержат маркированные мужские и женские имена.

2. Выбираем функцию, извлекающую последние N букв из входного слова.

3.Задаем начальное значение для генератора случайных чисел и перемешиваем данные.

4. Создаем выборку имен, которые будут использоваться для тестирования.

5. Определяем пороговые значения параметров, которые будут использоваться в процессе обучения и тестирования.

6. Разделяем данные на тренировочный и тестовый наборы.

7. Создаем модель наивного байесовского классификатора на примере тренировочного набора данных.

8. Вычисляем точность работы модели классификации.

9. Выполняем прогнозирование результатов для каждого имени из входного тестового списка.

10. Выводим полученные данные на экран монитора.

3.8. Создание модели сентимент-анализа

1. Рассмотрим работу модели сентимент-анализа. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем функцию, которая создает объект словаря на основании входных слов и возвращает его.

3. Задаем функцию и загружаем помеченные данные с отзывами о фильмах.

4. Выполняем разбиение данных на тренировочный и тестовый наборы в пропорции 8 к 2.

5. Разделяем векторы признаков для тренировочного и тестового наборов.

6. Задаем число объектов данных, используемых для тренировки и тестирования.

7. Выполняем обучение модели наивной байесовской классификации на

тренировочном наборе.

8. Выбираем первые N наиболее информативных слов.

9. Определяем выборку предложений, используемых для тестирования.

10. Выполним анализ тональности по выборочным данным и прогнозируем результаты.

11. Сохраняем элемент с наибольшим значением вероятности и предсказанный выходной класс.

12. Выводим результаты работы модели на экран монитора.

3.9. Тематическое моделирование с использованием латентного размещения Дирихле

1. Рассмотрим работу процедуры тематического моделирования. Для этого создаем новый файл Python и импортируем необходимые пакеты. Файл входных данных содержит 10 предложений, разделенных символами новой строки.

2. Задаем функцию, обрабатывающую входной текст и переходим к процедуре токенизации текста.

3. Выполняем процедуру стемминга токенизированного текста, а также удаляем стоп-слова из входного текста.

4. Выводим полученный список стоп-слов на экран монитора.

4. Порядок выполнения практической работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания к практической работе.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

Практическая работа № 8 «Создание систем распознавания речи»

1. Цель работы

Изучить методы и подходы к созданию систем распознавания речи. Ознакомиться с основными приемами работы с голосовыми сигналами. Ознакомиться с некоторыми известными моделями визуализации различных аудиосигналов. Изучить предложенные методы на простых примерах обработки звуковых сигналов.

В процессе выполнения практической работы студенты должны ознакомиться с теоретическими положениями и изучить на практике методы решения следующих задач:

- понять, что собой представляют современные системы распознавания речи;
- изучить методику работы со звуковыми сигналами;
- изучить методы визуализации аудиосигналов;
- научиться создать модель преобразования аудиосигналов в частотные интервалы;
- изучить метод генерирования аудиосигналов;
- научиться использовать метод синтезирования тонов;
- научиться применять методы извлечения речевых признаков и распознавания голосовых команд для построения простейших систем распознавания речи;
- научиться применять изученные методы для решения практических задач.

2. Теоретическое введение

Распознавание речи - это процесс распознавания слов, произносимых человеком. Голосовые сигналы улавливаются микрофоном и система пытается их распознать. Прежде чем приступить к анализу голосовых сигналов, очень важно понять их природу. Эти сигналы представляют собой смесь различных сигналов. Существует множество аспектов речи, вносящих свой вклад в эту сложность. Сюда входят эмоции, акцент, язык общения, посторонние шумы и многое другое.

Распознавание речи является важным аспектом в организации человеко-машинного взаимодействия. Исследователи работают над различными аспектами и

приложениями распознавания речи, такими как распознавание произносимых слов, идентификация личности говорящего, распознавание эмоций, идентификация акцента и т.п. В данной работе мы рассмотрим проблему понимания произносимых слов.

Реальные сигналы представляют собой непрерывные волны, откуда следует, что мы не можем их сохранить в том виде, как они есть. Мы должны семплировать сигнал с определенной частотой и преобразовать его в дискретное цифровое представление. Чаще всего речевые сигналы дискретизируют с частотой 44100 Гц. Это означает, что в течение одной секунды речевой сигнал разбивается на 44100 частей и значения, соответствующие различным отметкам времени, сохраняются в выходном файле. Мы сохраняем значение аудиосигнала каждую 1/44100 долю секунды. В таком случае мы говорим, что частота дискретизации аудиосигнала составляет 44100 Гц. В случае выбора достаточно высокой частоты дискретизации аудиосигнала у людей, слышащих его, создается впечатление, будто он непрерывный.

Аудиосигналы состоят из смеси синусоидальных волн с различными частотами, фазами и амплитудами. Анализируя частотные компоненты, можно идентифицировать множество характеристик. Любой аудиосигнал характеризуется своим распределением по частотному спектру. Чтобы преобразовать сигнал из дискретного временного представления в частотную область, мы должны использовать математический аппарат преобразований Фурье.

Как только сигнал преобразован в частотное представление, мы должны убедиться в том, что его можно использовать в качестве вектора признаков. И здесь на помощь приходят коэффициенты MFCC. MFCC - это инструмент для извлечения частотных характеристик из заданного аудиосигнала.

В процессе извлечения частотных признаков аудиосигнала система MFCC прежде всего извлекает спектр мощности. Затем она извлекает признаки, используя гребенки фильтров и дискретное косинусное преобразование.

Системы распознавания речи получают аудиосигналы на входе и распознают произносимые слова. Для этой задачи мы используем скрытые марковские модели. Аудиосигнал - это временной ряд данных, являющийся примером последовательных

данных. Исходное предположение заключается в том, что выходные результаты генерируются системой, проходящей через серию скрытых состояний. Нашей задачей является выяснение того, что собой представляют эти скрытые состояния, чтобы в нашем сигнале можно было идентифицировать слова.

3. Практическая часть

3.1. Визуализация аудиосигналов

1. Рассмотрим работу процедуры визуализации аудиосигналов. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Загружаем входной аудиофайл и обрабатываем его, используя метод file.read. Этот метод возвращает два значения: частоту дискретизации и аудиосигнал.

3. Выводим данные о форме сигнала, типе данных и длительности аудиосигнала.

4. Выполняем нормализацию сигнала.

5. Извлекаем первые 50 значений из массива numpy для построения графика.

6. Задаем параметры визуализации и выводим на экран монитора график аудиосигнала.

3.2. Преобразование аудиосигналов в частотные интервалы

1. Рассмотрим работу процедуры преобразования аудиосигналов в частотные интервалы. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Загружаем входной аудиофайл и обрабатываем его, используя метод wavefile.read. Этот метод возвращает два значения: частоту дискретизации и аудиосигнал.

3. Выполняем нормализацию сигнала.

4. Извлекаем длину и половинную длину сигнала.

5. Применяем к сигналу преобразование Фурье.

6. Выполняем нормализацию частотного сигнала.

7. Выполняем корректировку преобразованного сигнала для четных и нечетных случаев.

8. Определяем мощность сигнала в децибелах.

7. Задаем параметры визуализации и выводим на экран монитора график аудио сигнала.

3.3. Генерирование аудиосигналов

1. Рассмотрим работу процедуры генерирования аудиосигналов. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Задаем имя выходного аудио файла, а также основные аудио параметры, такие как длительность, частота дискретизации, частота звука, минимальное и максимальное значения.

3. Генерируем аудио сигнал, используя установленные параметры.

4. Добавляем в аудио сигнал шум.

5. Выполняем нормализацию и масштабирование сигнала.

6. Сохраняем сгенерированный аудио сигнал в выходном файле.

7. Выводим 200 полученных значений для построения графика. Строим график аудио сигнала.

3.4. Синтезирование звуков для генерации музыки

Рассмотрим пример синтезирования музыки посредством склеивания различных звуков. Для генерации музыки мы используем стандартные звуки, такие как A (ля), C (до), G (соль), F (фа) и т.д. Таблицу соответствий между частотой и стандартными звуками можно найти по ссылке: http://www.phy.mtu.edu/~suits/notefreqs.html.

1. Рассмотрим работу процедуры синтезирования звуков для генерации музыки. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Выбираем функцию для генерации звука на основании входных параметров.

3. Создаем аудио сигнал, используя указанные параметры.

4. Задаем основную функцию с помощью определения имен выходных аудиофайлов.

5. Генерируем звук F (фа) длительностью 3 секунды и извлекаем соответствующую частоту звука.

6. Генерируем звук, используя функцию синтезатора звука.

7. Записываем сгенерированный аудиосигнал в выходной файл.

8. Генерируем последовательность звуков путем задания длительности каждого звука в секундах.

8. Формируем аудиосигнал на базе последовательности звуков.

9. Определяем частоту и длительность для каждого звука.

10. Генерируем звук, используя функцию синтезатора звука и присоедяем его к основному выходному сигналу.

11. Сохраняем основной выходной сигнал в выходном файле.

3.5. Извлечение речевых признаков

Для извлечения MFCC-признаков мы будем использовать пакет python_speech_features. В целях упрощения в состав файлов примеров включенапапка features, содержащая файлы, необходимые для работы с этим пакетом. Для генерации различных аудиосигналов можно воспользоваться пакетом NumPy.

1. Рассмотрим пример извлечения MFCC-признаков. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Прочитаем входной аудиофайл и извлечем первые 10000 семплов для анализа.

3. Извлекаем MFCC-признаки и выводим значения параметров MFCC.

4. Извлекаем свойства гребенки фильтров и выводим значения параметров для гребенки фильтров.

5. Задаем параметры визуализации и строим графики MFCC-признаков.

3.6. Распознавание произносимых слов

Для построения нашей системы распознавания речи мы используем пакет hmmlearn.

Для тренировки нашей системы распознавания речи нам нужен набор данных с аудиофайлами для каждого слова. В состав предлагаемых файлов примероввключена папка data, содержащая все необходимые файлы. Набор входных данных содержит семь различных слов. С каждым словом ассоциирована папка, и каждая

папка содержит 15 аудиофайлов. Мы используем 14 папок для тренировки и одну для тестирования.

Для каждого слова мы создадим скрытую марковскую модель. Все эти модели необходимо сохранить и когда нам потребуется распознать слово в неизвестном аудиофайле, мы сможем обратиться к сохраненным моделям и выбрать ту из них, которая дает наибольшую оценку.

1. Рассмотрим пример создания системы распознавания речи. Для этого создаем новый файл Python и импортируем необходимые пакеты.

2. Определим функцию для набора входных аргументов. Мы должны указать входную папку, которая содержит аудиофайлы, необходимые для обучения нашей системы распознавания речи.

3. Определяем класс для тренировки, тип ковариации и тип скрытой марковской модели.

4. Инициализируем переменную для хранения модели для каждого слова.

5. Выбираем модель обучения модели, на основе заданных параметров.

6 Выбираем метод оценки входных данных.

7. Выбираем функцию, обеспечивающую построение модели для каждого слова в тренировочном наборе данных.

8. Проводим анализ входного каталога, формируем метку, а также выполняем инициализацию переменной для хранения тренировочных данных.

9. Создаем список файлов для тренировки моделей.

10. Считываем аудио сигнал из текущего файла.

11. Извлекаем MFCC-признаки и присоединяем точку данных к переменной X.

12. Выполняем инициализацию скрытой марковской модели и выполняем ее обучение на тренировочных данных.

13. Сохраняем модель для текущего слова.

14. Задаем функцию тестирования тренировочного набора данных.

16. Формируем MFCC-признаки.

17. Задаем переменные для хранения максимальной оценки и выходной метки.

18. Выполняем вычисления и по результатам выбираем лучшую модель.

19. Вычисляем итоговую оценку и сравниваем ее значение с максимальной

оценкой.

20. Выводим на экран монитора полученные результаты.

4. Порядок выполнения практической работы

4.1. Изучить теоретическое введение.

4.2. Последовательно выполнить все задания к практической работе.

4.3. Сохранить созданные файлы с программы и полученные в процессе выполнения работы результаты.

4.4. Оформить отчет по работе.

4.5. Ответить на контрольные вопросы.

5. Содержание отчета

5.1. Название и цель работы.

5.2. Полученные (сгенерированные) варианты исходных данных.

5.3. Описание использованных моделей и алгоритмов.

5.4. Примеры решений.

5.5. Скриншоты с результатами выполненных заданий.

5.6. Выводы по проделанной работе.

5.7. Список использованной литературы.

5.6. Листинг программы.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Программное и аппаратное обеспечение информационных систем»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024 г.

Лабораторная работа № 1.
Применение CASE-средств для проектирования и разработки программного и аппаратного обеспечения ИС и АС.

Цель занятия:
Изучить виды и типы CASE-средств поддержки процесса разработки программного обеспечения, освоить принципы применения CASE-средств на практике, выбрать тему и реализовать первую часть индивидуального проекта в рамках лабораторной работы.

В результате выполнения лабораторной работы студент получает комплект документации, схемы программного обеспечения, выполненные при помощи CASE-средств.

Средства выполнения практической работы:
Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.

Результаты освоения материала:
В результате проведения занятия студент должен:
Знать:

- классификацию CASE-средств и основы их применения
- современные инструменты поддержки процесса разработки программного обеспечения, относящиеся к классу CASE-средств

Уметь:

- применять на практике CASE-средства поддержки процесса разработки программного обеспечения

Методика проведения занятия:
Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.

5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями.

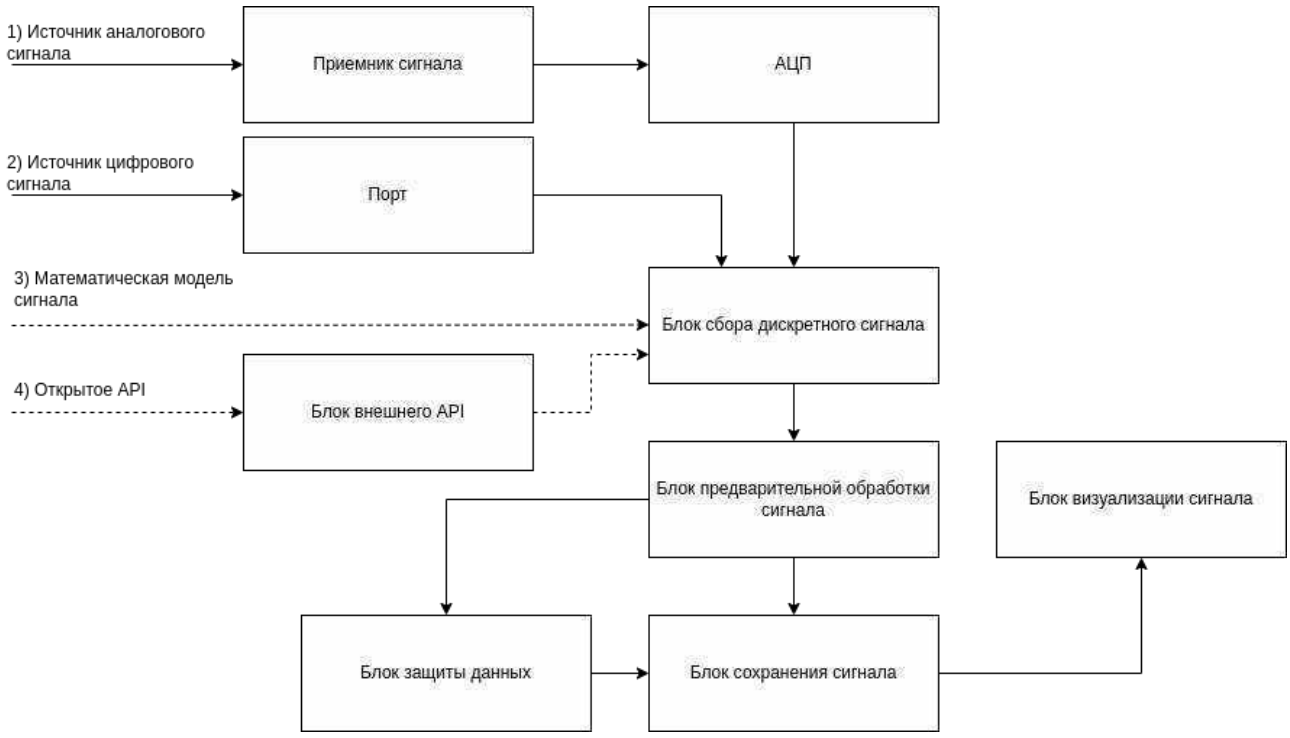
Содержание практического занятия:
В рамках практического занятия подробно рассматриваются и обсуждаются следующие ключевые вопросы

1. Основные группы CASE средств.
2. CASE средства верхнего уровня.
3. CASE средства нижнего уровня.
4. Интегрированные CASE средства.
5. Факторы, которые нужно принимать во внимание при выборе CASE-средств.
6. Методологии разработки программных проектов.
7. CASE-средства для проектирования программного обеспечения.
8. CASE-средства проектирования баз данных.
9. CASE-средства управления проектами и ролями пользователей.
10. CASE-средства моделирования.
11. Инструменты документирования проектов.
12. Основы применения UML-диаграмм.
13. Инструменты для построения UML-диаграмм.

В общем случае, каждому студенту предлагается разработать систему приема данных от аппаратного источника данных с предварительной обработкой, сохранением в базу данных и защитой информации.

Это общее задание, а частные случаи варьируются относительно варианта, а также желания и возможностей студентов.

Схематично такую систему можно изобразить следующим образом.



Студенту формируется следующее задание:

1. Формирование идеи для серии Лабораторных работ № 1–4 в виде комплексного приложения или аппаратно-программной платформы.
2. Выделение основных технических требований к разрабатываемому Приложений.
3. Оформление идеи и технических требований при помощи системы облачного документооборота (формирование комплекта документации).
4. Разработка структурной схемы Приложения с применением CASE-средств.
5. Разработка функциональной схемы Приложения с применением CASE-средств.
6. Формирование структуры модулей Приложения с применением CASE-средств (UML-диаграмм).
7. Формирование структуры базы данных Приложения с применением CASE-средств (UML-диаграмм).
8. Формирование этапов разработки Приложения и составление списка решаемых задач.
9. Оформление списка задач при помощи систем Управления проектами.

Для облачного документооборота предлагается использовать OneDrive или Google Drive, но выбор может быть изменен со стороны студента.

Для составления UML-диаграмм предлагается использовать бесплатный инструмент Draw.io, который существует в виде отдельного приложения, в виде онлайн-ресурса и даже в виде расширения для Google Drive.

Для оформления структуры базы данных предлагается использовать бесплатный ресурс dbdiagram.io или StarUML.

Для Управления проектами предлагается использовать бесплатный вариант JIRA или его аналоги типа Trello.

В идеальном случае, для получения сигналов и их предварительной обработки предполагается использовать аппаратные комплекты разработчиков типа:

1. Arduino.
2. Raspberry Pi.
3. STM SDK.

Но, так как зачастую у студентов отсутствует подобная плата комплекта разработчика или финансовые возможности ее приобретения или, чаще всего, знания в программировании аппаратуры и в области цифровой схемотехники, то студентам предлагается эмулировать процесс получения входных данных одним из следующих способов:

1. Математически.
2. При помощи файлов или базы данных с отсчетами или измерениями.
3. При помощи внешних сетевых API (открытых источников).

В первом случае, для генерации входных сигналов предлагается использовать аддитивную модель системы гармонических сигналов, которые можно генерировать на основе различных амплитуд и частот следующим образом:

X = A1*sin(2*pi*F1) + A2*sin(2*pi*F2) + … + Ak*sin(2*pi*Fk).

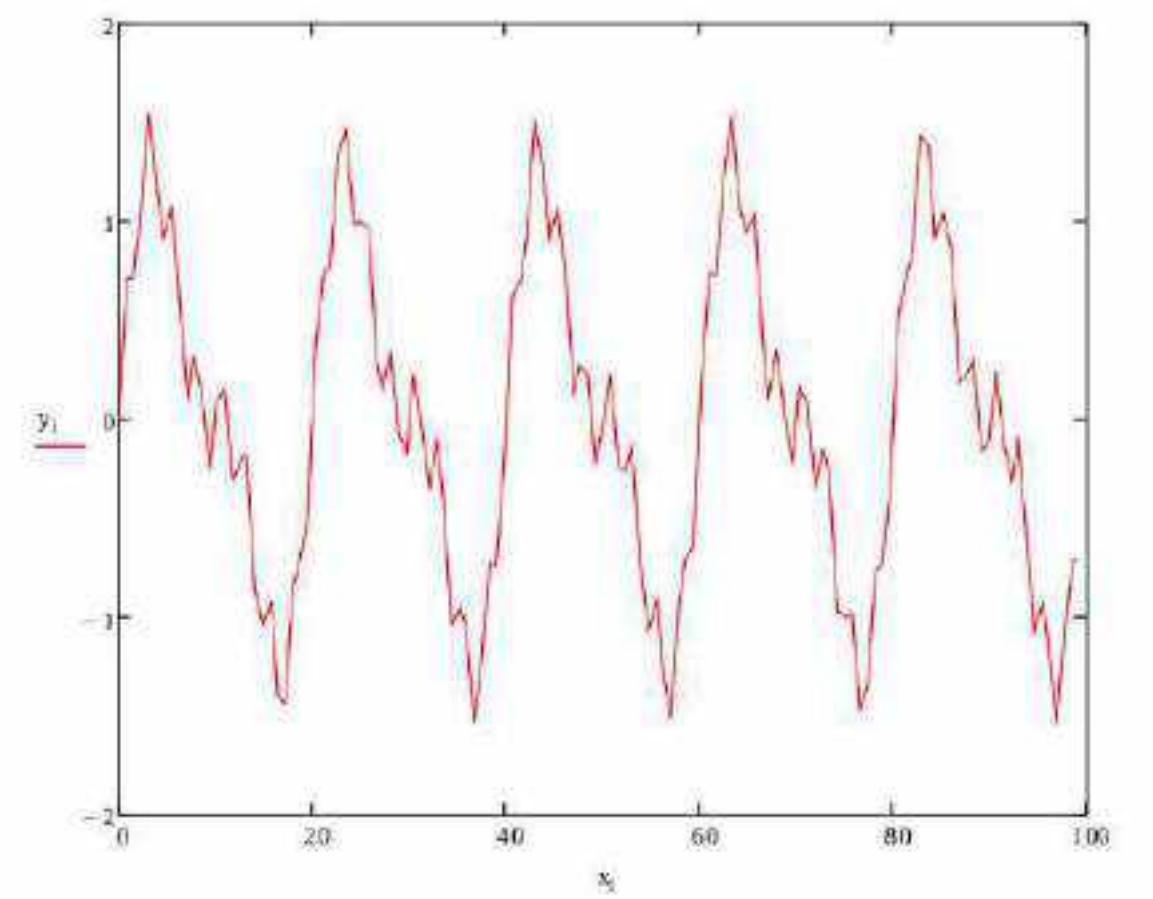
Здесь присутствует вектор амплитуд гармоник A и вектор частот гармоник F. Длины этих векторов одинаковы и равны k.

В результате сложения этих гармоник получается наложение сигналов разной силы и частоты, что позволяет имитировать источник случайных сигналов и получать разные формы.

Например, при следующей комбинации:

$yi \coloneqq \sin 2\pi\ 0{,}05xi + 0{,}5 \sin 2\pi\ 0{,}1xi + 0{,}25 \sin 2\pi\ 0{,}4xi$,

получается достаточно «чистый» сигнал



Имея такие дискретные данные, мы получаем симуляцию внешнего источника данных и процесса дискретизации.

На выходе этого процесса у нас есть массивы готовых цифровых данных, которые можно передавать на следующие этапы работы системы и использовать для выполнения задания к лабораторной работе.

Критерии оценивания:

Максимальное количество баллов за практическую работу: 10 баллов.
9-10 баллов– обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

7-8 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

6 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-5 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 2.
Разработка программной части ИС и АС.

Цель занятия:
Изучить методы сопровождения исходного кода, создания и использования репозиториев методы получения сигналов и данных, методы предварительной обработки информации перед применением бизнес-логики приложения.

В результате выполнения лабораторной работы студент получает навыки создания репозиториев исходного кода, приема данных и написания программ для предварительной обработки принятых данных.

Средства выполнения практической работы:
Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Установленные средства разработки нативного или кроссплатформенного программного обеспечения для языков JavaScript, Python, Pascal, C++, C# или Java по выбору студента.

Результаты освоения материала:
В результате проведения занятия студент должен:
Знать:

- способы современной разработки программного обеспечения с применением нативных или кроссплатформенных инструментов;
- современные инструменты поддержки процесса разработки программного обеспечения, относящиеся к классу CASE-средств.

Уметь:

- применять на практике CASE-средства разработки программного обеспечения, нативные и кроссплатформенные инструменты разработки.

Методика проведения занятия:
Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.

3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями.

Содержание практического занятия:
В рамках практического занятия подробно рассматриваются и обсуждаются следующие ключевые вопросы

1. Нативные программные средства, инструменты и библиотеки.
2. Кроссплаатформенные программные средства, инструменты и библиотеки.
3. Принципы выполнения программного кода на уровне микропроцессора ЭВМ.
4. Способы компиляции программ.
5. Принципы работы компиляторов.
6. Принципы работы интерпретаторов скриптовых языков программирования.
7. Способы кросскомпиляции приложений.
8. Использование виртуальных машин и технологий виртуализации при разработке программного обеспечения.
9. Технология работы с языком Си.
10. Технология работы с языком Pascal.
11. Технология работы с веб-стеком для разработки программного обеспечения.
12. Технология работы со стеком Java.
13. Технология работы со стеком C#.
14. Типовая структура проекта.
15. Использование систем контроля версии для управления проектами.

В общем случае, каждому студенту предлагается разработать программную реализацию приема данных от аппаратного источника данных или ее модели с предварительной обработкой полученных данных.

Это общее задание, а частные случаи варьируются относительно варианта, а также желания и возможностей студентов, как и в первой лабораторной работе.

Студенту формируется следующее задание:

1. Доработка задания из первой лабораторной работы, если это необходимо.
2. Доработка структурной схемы программного обеспечения, если это необходимо.

3. Расширение структурной схемы блока приема сигнала от датчика в зависимости от выбранного варианта источника:
   - аппаратный аналоговый датчик,
   - аппаратный цифровой датчик.
   - математическая модель датчика с генератором цифрового сигнала,
   - внешнее API (в сети Интернет).
4. Создание и инициализация репозитория исходного кода в сети Интернет.
5. Реализация блока приема данных от датчика.
6. Реализация буфера принятого сигнала от датчика.
7. Реализация блока дискретизации или синхронизации выборки данных в соответствии с заданной частотой выборки.
8. Реализация блока предварительной обработки данных от датчика согласно модели цифровой фильтрации или усреднения (скользящие средние).

Для облачного документооборота по-прежнему предлагается использовать OneDrive или Google Drive, но выбор может быть изменен со стороны студента.

Для составления или редактирования UML-диаграмм по-прежнему предлагается использовать бесплатный инструмент Draw.io, который существует в виде отдельного приложения, в виде онлайн-ресурса и даже в виде расширения для Google Drive.

Для оформления или редактирования структуры базы данных по-прежнему предлагается использовать бесплатный ресурс dbdiagram.io или StarUML.

Для Управления проектами по-прежнему предлагается использовать бесплатный вариант JIRA или его аналоги типа Trello.

В идеальном случае, для получения сигналов и их предварительной обработки (как и в Лабораторной работе № 1) предполагается использовать аппаратные комплекты разработчиков типа:

1. Arduino.
2. Raspberry Pi.
3. STM SDK.

Если у студентов не появляется возможность использовать физические устройства, то им предлагается эмулировать процесс получения входных данных одним из следующих способов (как это было описано в руководстве к Лабораторной работе № 1):

1. Математически.
2. При помощи файлов или базы данных с отсчетами или измерениями.
3. При помощи внешних сетевых API (открытых источников).

Если в Лабораторной работе 1 предлагается выбрать способ получения исходных данных, то в данной Лабораторной работе 2 требуется уже реализовать блок приема данных из внешнего источника.

Напомним, что большинство студентов может выбрать математическую модель сигнала, поэтому приведем здесь способ обработки таких данных.

Для генерации входных сигналов можно использовать аддитивную модель системы гармонических сигналов, которые можно генерировать на основе различных амплитуд и частот следующим образом:

X = A1*sin(2*pi*F1) + A2*sin(2*pi*F2) + … + Ak*sin(2*pi*Fk).

Здесь присутствует вектор амплитуд гармоник A и вектор частот гармоник F. Длины этих векторов одинаковы и равны k.

В результате сложения этих гармоник получается наложение сигналов разной силы и частоты, что позволяет имитировать источник случайных сигналов и получать разные формы.

Предлагается выбрать вектора амплитуд и частот на базе следующих вариантов.

Вариант	**Вектор A**	**Вектор F**
1	1, 0.5, 0.25	0.1, 0.1, 0.4
2	2.5, 1.5, 0.35	0.55, 0.2, 1.4
3	2, 1.5, 0.75	0.5, 0.7, 2.4
4	1.5, 0.5, 1.25	0.75, 1.1, 1.64
5	1, 5.5, 1.75	1.5, 0.3, 2.2

В качестве алгоритма обработки данных можно использовать разные варианты. Давайте рассмотрим применение скользящих средних для сглаживания сигналов.

Скользящие средние

В основе применения скользящего среднего лежит следующая формула:

EMA = (P * alpha) + (EMAprev * (1 - alpha))

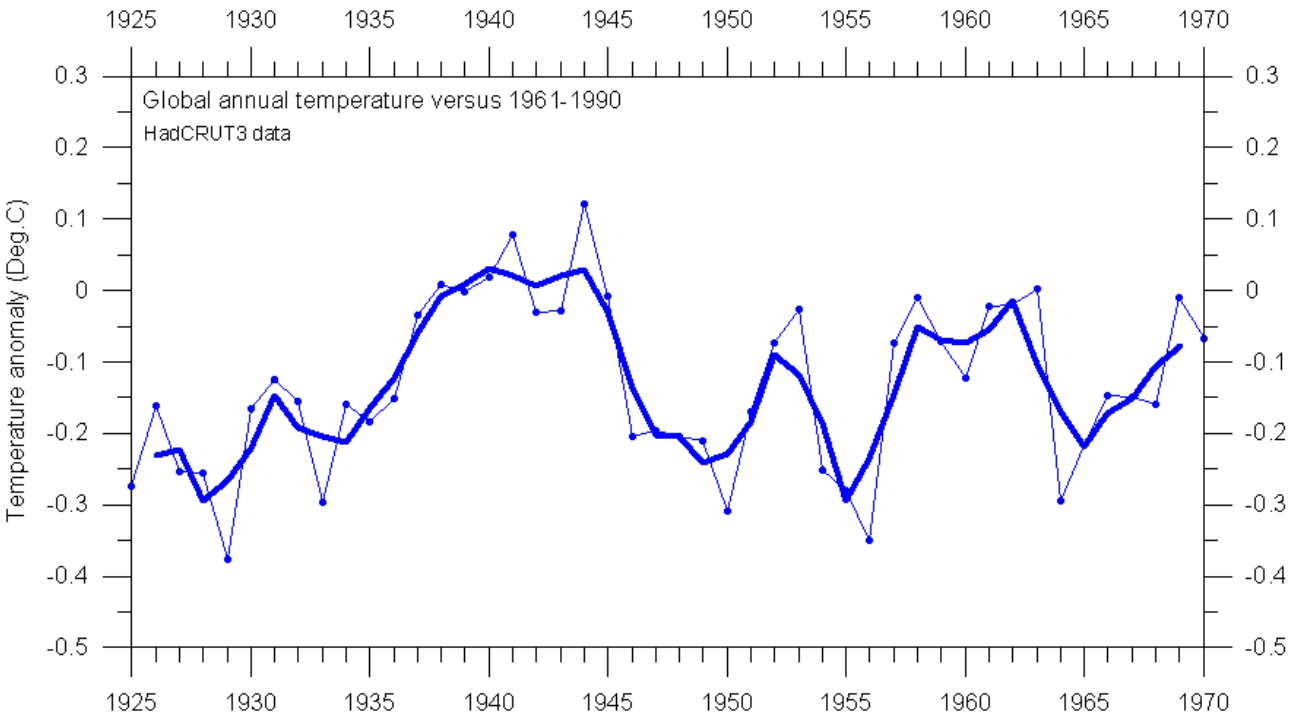
P — текущее значение сигнала
alpha — сглаживающий фактор, например 2 / (1 + N)
N — количество периодов сглаживания

Период N для EMA выбирается экспериментальным путем исходя из исторических данных.

Скользящие средние применяются для предиктивной аналитики или для сглаживания сигналов, - мы вполне можем взять их для примера алгоритма предварительной обработки.

Тем более, что одним из вариантов исходного сигнала для первой лабораторной работы мы предлагаем использовать данных о курсах валют или значения валютных пар, полученных из внешних источников типа API.

Пример использования скользящих средних с температурными данными.



Цифровой фильтр.

Это альтернативный вариант обработки информации. Цифровые фильтры — это сложная тема, но мы можем упростить материал, используя готовую формулу:

Yn = SUM Xn-j * Bj

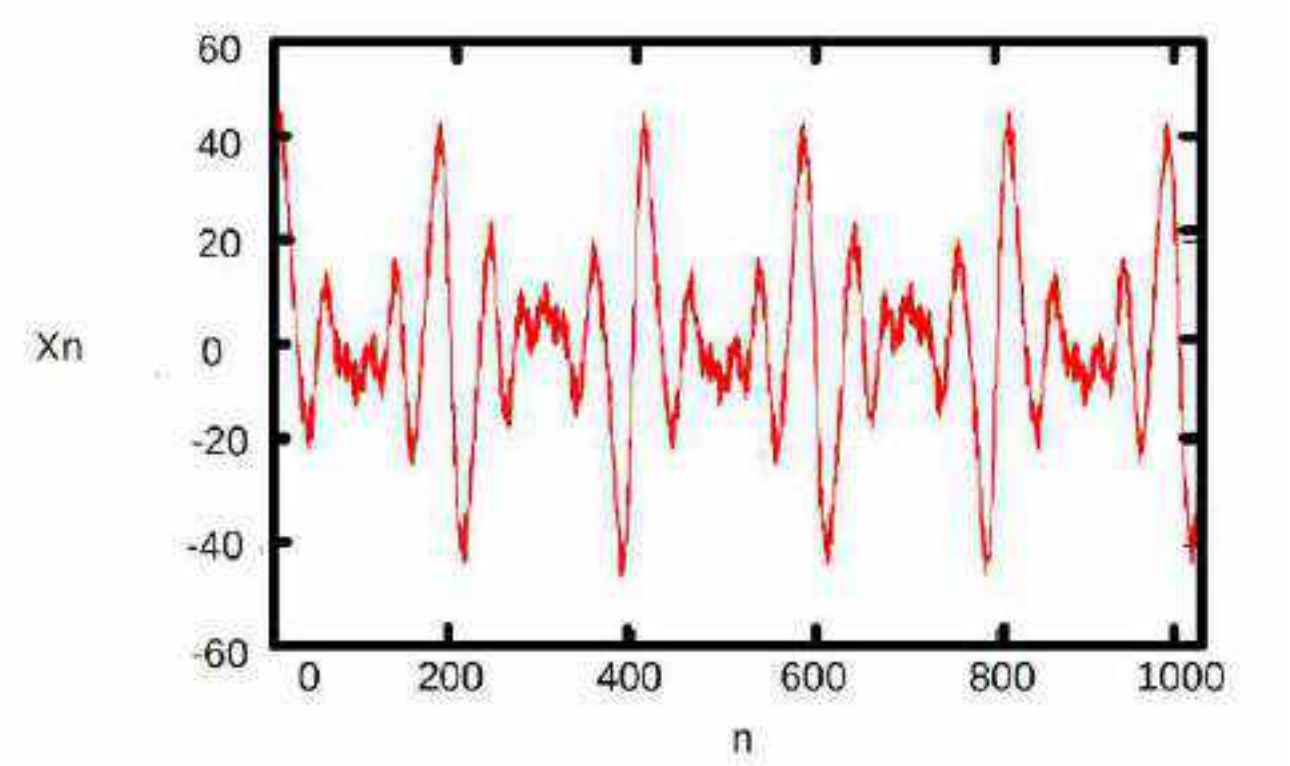
Yn — это значение выходного сигнала в текущей точке

Xn-j — это значение входного сигнала, отстоящего от текущей точки на j позиций в прошлое

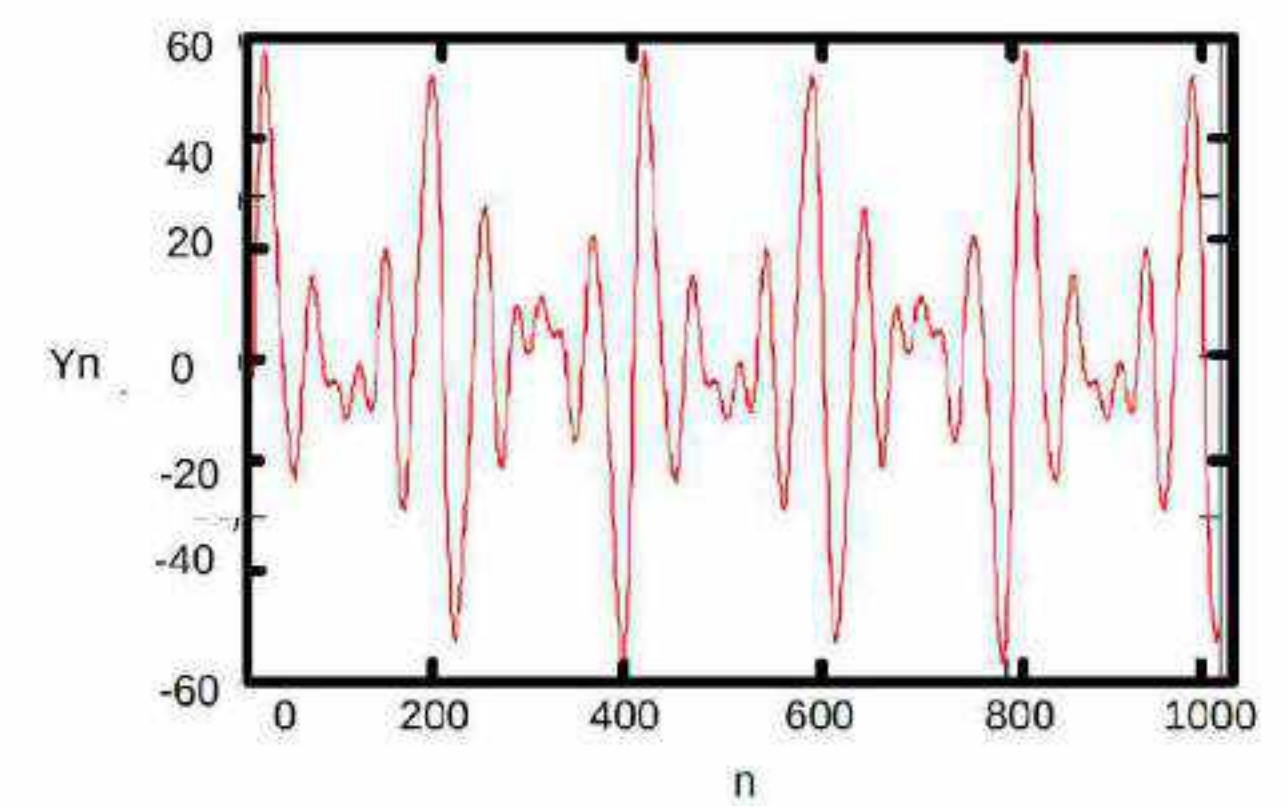
Bj — это коэффициент фильтра с номером j.

Имея обработанные таким образом данные, мы получаем симуляцию блоков работы с сырыми данными.

Пример сгенерированного входного сигнала до фильтрации.



После фильтрации с правильно подобранными коэффициентами получается более «чистая» форма без высокочастотных примесей.



На выходе этого процесса у нас есть массивы готовых цифровых данных, которые можно передавать на следующие этапы работы системы и использовать для выполнения задания к лабораторной работе.

После реализации некоторого количества алгоритмов в виде программного кода, его желательно зафиксировать до следующей лабораторной работы. Для этого лучше использовать репозитории типа Github или Gitlab.

Напомним, что, для работы с репозиториями у нас в системе должна быть установлена программа Git. Для ОС Windows это программное обеспечение нужно устанавливать отдельно.
Для создания репозитория введем команды в консоли

1 git init
2 git add .
3 git commit −m ’Initial commit’
4 git remote −v
5 git remote add
origin ’https://github.com/oleg−creator−arch/trade_coin_bot.git’
6 git push origin main

Созданный проект на сайте Github, может выглядеть следующим образом.

Теперь нам доступны команды:

- Клонирования: git clone https://github.com/repo/repo.git
- Принятия изменений: git pull
- Создания ветви: git checkout –b newbranch
- Переход на существующую ветку: git checkout branchname
- Добавления файлов в индекс: git add .
- Фиксации изменения (коммита): git commit –m “new commit”
- Отправки изменений на сервер Github: git push

И так далее.

Критерии оценивания:

Максимальное количество баллов за практическую работу: 10 баллов.
9-10 баллов– обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

7-8 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

6 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-5 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 3.
Разработка аппаратной части ИС и АС или ее модели.

Цель занятия:
Изучить способы управления версиями программного кода, разработать каркас приложений, научиться работать с цифровыми данными, применять алгоритмы предварительной обработки информации, подготавливая для дальнейшего применения, применять CASE-средства поддержки процесса разработки программного обеспечения.

Средства выполнения практической работы:
Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Установленные средства разработки нативного или кроссплатформенного программного обеспечения для языков JavaScript, Python, Pascal, C++, C# или Java по выбору студента.

В результате проведения занятия студент должен:
Знать:

- способы современной разработки программного обеспечения с применением нативных или кроссплатформенных инструментов;
- современные инструменты поддержки процесса разработки программного обеспечения, относящиеся к классу CASE-средств;
- методы и алгоритмы обработки сигналов различной физической природы.

Уметь:

- применять на практике CASE-средства разработки программного обеспечения, нативные и кроссплатформенные инструменты разработки.

Методика проведения занятия:
Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.
3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.

5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями.

Содержание практического занятия:
В рамках практического занятия подробно рассматриваются и обсуждаются следующие ключевые вопросы

1. Нативные программные средства, инструменты и библиотеки.
2. Кроссплатформенные программные средства, инструменты и библиотеки для программирования аппаратуры или для разработки моделей.
3. Принципы выполнения программного кода на уровне микроконтроллера.
4. Способы компиляции программ и их прошивки.
5. Принципы работы компиляторов для микроконтроллеров.
6. Способы кросскомпиляции приложений.
7. Использование виртуальных машин и технологий виртуализации при разработке программного обеспечения, в том числе и для аппаратных моделей.
8. Технология работы с языком Си.
9. Технология работы с языком Pascal.
10. Технология работы с веб-стеком для разработки программного обеспечения.
11. Технология работы со стеком Java.
12. Технология работы со стеком C#.
13. Типовая структура проекта.
14. Использование систем контроля версии для управления проектами.
15. Методы предварительной обработки сигналов.
16. Алгоритмы цифровой фильтрации.
17. Алгоритмы агрегации данных и их усреднения.

В общем случае, каждому студенту предлагается разработать программную реализацию приема данных от аппаратного источника данных или ее модели с предварительной обработкой полученных данных.

Это общее задание, а частные случаи варьируются относительно варианта, а также желания и возможностей студентов, как и в первой лабораторной работе.

Студенту формируется следующее задание:

1. Доработка задания из первой и второй лабораторной работы, если это необходимо.
2. Доработка структурной схемы программного обеспечения, если это необходимо.

3. Расширение структурной схемы блока приема сигнала от датчика в зависимости от выбранного в Лабораторной работе № 1 варианта источника данных:
   - аппаратный аналоговый датчик,
   - аппаратный цифровой датчик.
   - математическая модель датчика с генератором цифрового сигнала,
   - внешнее API (в сети Интернет).
4. Доработка блока приема данных от датчика.
5. Доработка буфера принятого сигнала от датчика.
6. Доработка блока дискретизации или синхронизации выборки данных в соответствии с заданной частотой выборки.
7. Реализация блока предварительной обработки данных от датчика согласно модели цифровой фильтрации или усреднения (скользящие средние).

Для облачного документооборота по-прежнему предлагается использовать OneDrive или Google Drive, но выбор может быть изменен со стороны студента.

Для составления или редактирования UML-диаграмм по-прежнему предлагается использовать бесплатный инструмент Draw.io, который существует в виде отдельного приложения, в виде онлайн-ресурса и даже в виде расширения для Google Drive.

Для оформления или редактирования структуры базы данных по-прежнему предлагается использовать бесплатный ресурс dbdiagram.io или StarUML.

Для Управления проектами по-прежнему предлагается использовать бесплатный вариант JIRA или его аналоги типа Trello.

В идеальном случае, для получения сигналов и их предварительной обработки предполагается использовать аппаратные комплекты разработчиков типа:

1. Arduino.
2. Raspberry Pi.
3. STM SDK.

Если в Лабораторной работе 1 предлагается выбрать способ получения исходных данных, в Лабораторной работе 2 требуется реализовать блок приема данных из внешнего источника, а в данной Лабораторной работе 3 требуется

использовать или моделировать аппаратный блок для обработки данных и их защиты.

Напомним, что студенты могут выбрать математическую модель сигнала, поэтому приведем здесь способ обработки таких данных. Однако в этом случае им также необходимо изучить и сымитировать программно методы дискретизации и квантования сигналов.

На выходе этого процесса у нас есть массивы готовых цифровых данных, которые можно передавать на следующие этапы работы системы и использовать для выполнения задания к лабораторной работе.

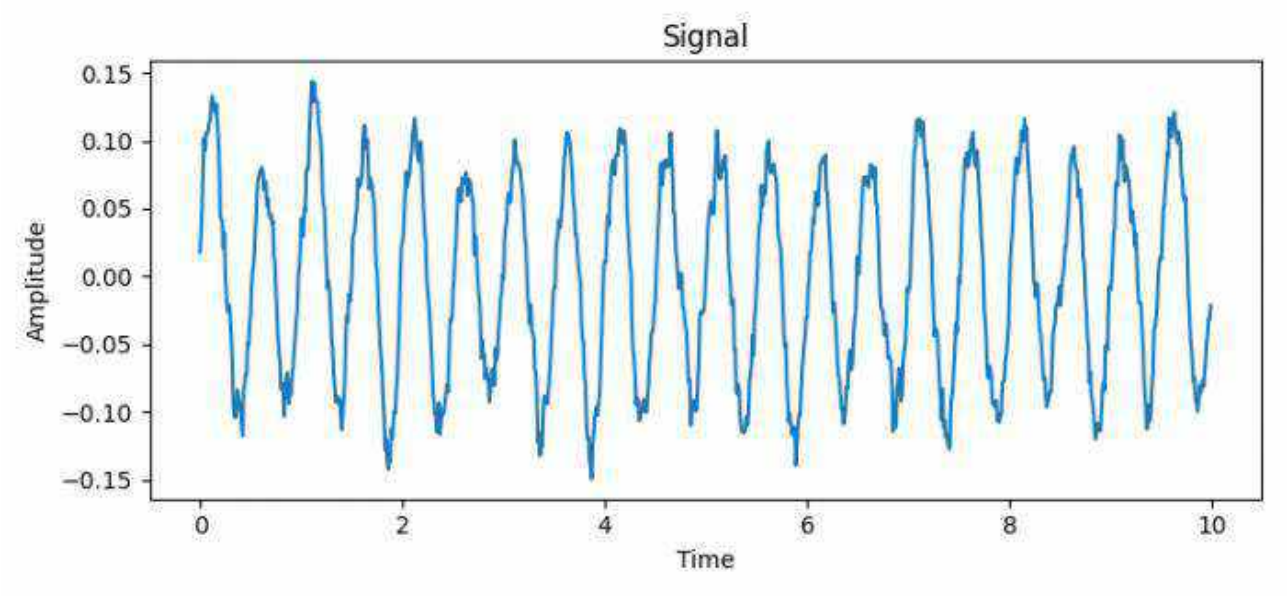
Также на этом этапе работы необходимо выполнить визуализацию полученных данных в виде графика.

Например, при использовании языка Python для разработки можно воспользоваться специальными математическими библиотеками, как показано в примере ниже.

```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True

np.random.seed(0)

dt = 0.01 # sampling interval
Fs = 1 / dt # sampling frequency
t = np.arange(0, 10, dt)

# generate noise:
nse = np.random.randn(len(t))
r = np.exp(-t / 0.05)
cnse = np.convolve(nse, r) * dt
cnse = cnse[:len(t)]
s = 0.1 * np.sin(4 * np.pi * t) + cnse
fig, axs = plt.subplots()
```

```
axs.set_title("Signal")
axs.plot(t, s, color='C0')
axs.set_xlabel("Time")
axs.set_ylabel("Amplitude")
```

В результате мы получим график смоделированного сигнала.



Критерии оценивания:

Максимальное количество баллов за практическую работу: 10 баллов.
9-10 баллов– обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

7-8 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

6 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-5 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

Лабораторная работа № 4.
Обеспечение защиты и хранения данных ИС и АС.

Цель занятия:
Разработать структуру хранилища данных, подключить инструменты хранения данных (базе данных) типа SQL или NoSQL и обеспечить безопасное хранение или передачу простым шифрованием.

Средства выполнения практической работы:
Проведение занятий предполагается в аудитории, оснащенной:

1. Комплектом персональных ЭВМ в компьютерных классах с выходом в глобальную сеть Интернет.
2. Мультимедийной доской для демонстрации материалов.
3. Установленной ОС Windows или ОС Linux.
4. Установленным текстовым редактором MS Office или Libre Office.
5. Установленным редактором графических схем типа Draw.io.
6. Установленные средства разработки нативного или кроссплатформенного программного обеспечения для языков JavaScript, Python, Pascal, C++, C# или Java по выбору студента.
7. Локальной базой данных (сервером или просто драйвером SQLite3).

Результаты освоения материала:
В результате проведения занятия студент должен:
Знать:

- способы современной разработки программного обеспечения с применением нативных или кроссплатформенных инстурментов;
- современные инструменты поддержки процесса разработки программного обеспечения, относящиеся к классу CASE-средств;
- способы проектирования баз данных;
- способы обеспечения защиты данных методами шифрования.

Уметь:

- применять на практике CASE-средства разработки программного обеспечения, нативные и кроссплатформенные инструменты разработки, а также проектирования.

Методика проведения занятия:
Практическое занятие проводится следующим образом:

1. Рассмотрение теоретических вопросов.
2. Разъяснения преподавателя.

3. Диспут или «круглый стол» по тематике занятия.
4. Выполнение заданий преподавателя.
5. Анализ и оценка выполненных работ и степени овладения, обучающихся запланированными компетенциями.

Содержание практического занятия:
В рамках практического занятия подробно рассматриваются и обсуждаются следующие ключевые вопросы

1. Современные SQL базы данных.
2. Современные NoSQL базы данных.
3. Методы проектирования структуры баз данных.
4. CASE-средства для разработки структур баз данных.

В общем случае, каждому студенту предлагается разработать программное решение для трансляции принятых и обработанных данных от источника данных или ее модели после предварительной обработки в базу данных с защитой информации.

Это общее задание, а частные случаи варьируются относительно варианта, а также желания и возможностей студентов, как и в первой лабораторной работе.

Студенту формируется следующее задание:

1. Доработка задания из второй лабораторной работы, если это необходимо.
2. Разработка структурной схемы базы данных при помощи CASE-средств.
3. Создание схемы базы данных при помощи драйвера базы данных.
4. Реализация блока приема данных после обработки.
5. Реализация программной прослойки над движком базы данных.
6. Реализация блока защиты информации.
7. Реализация блока сохранения информации в базу данных.
8. Реализация блока чтения данных из базы данных.

Для облачного документооборота по-прежнему предлагается использовать OneDrive или Google Drive, но выбор может быть изменен со стороны студента.

Для составления или редактирования UML-диаграмм по-прежнему предлагается использовать бесплатный инструмент Draw.io, который существует в виде отдельного приложения, в виде онлайн-ресурса и даже в виде расширения для Google Drive.

Для оформления или редактирования структуры базы данных по прежнему предлагается использовать бесплатный ресурс dbdiagram.io или StarUML.

Для Управления проектами по-прежнему предлагается использовать бесплатный вариант JIRA или его аналоги типа Trello.

В идеальном случае, для сохранения данных после их предварительной обработки предполагается по-прежнему использовать в качестве источников аппаратные комплекты разработчиков типа:

1. Arduino.
2. Raspberry Pi.
3. STM SDK.

Но, по-прежнему, допускается эмулировать процесс получения входных данных одним из следующих способов:

1. Математически.
2. При помощи файлов или базы данных с отсчетами или измерениями.
3. При помощи внешних сетевых API (открытых источников).

В завершении лабораторной работы мы получаем структуру базы данных, таблицы, поля и типы полей. Несколько таблиц необходимо связать друг с другом.

В качестве примера мы можем предложить следующую структуру, сформированную при помощи dbdiagram.io.

При помощи псевдо-языка описания структуры базы данных мы можем реализовать следующую схему:

1. Сущность Пользователь, имеющий поля id, username, email, created, active.

```
Table User {
  id int [pk]
  username varchar [not null]
  email varchar [not null, unique]
  created datetime
   active boolean
}
```

2. Сущность Платформа, имеющая поля id, name, created, owner_id, address_id.

```
Table Platform {
  id int [pk]
  name varchar [not null]
  created datetime
  owner_id int [ref: > User.id]
  address_id int [ref: - Address.id]
}
```

3. Сущность Адрес, где установлена Платформа с полями id, line_1, line_2, city, country, town, latitude, longitude, ip

```
Table Address {
  id int [pk]
  line_1 varchar [not null]
  line_2 varchar
  city varchar
  town varchar
  country varchar
  latitude float4
  longitude float4
  ip ip
}
```

4. Сущность Сенсор, имеющий поля id, name, uuid, created, platform_id

```
Table Sensor {
  id int [pk]
  name varchar [not null]
  uuid uuid
  created datetime
  platform_id int [ref: > Platform.id]
}
```

5. Сущность Измерение, имеющее поля id, value, created, sensor_id

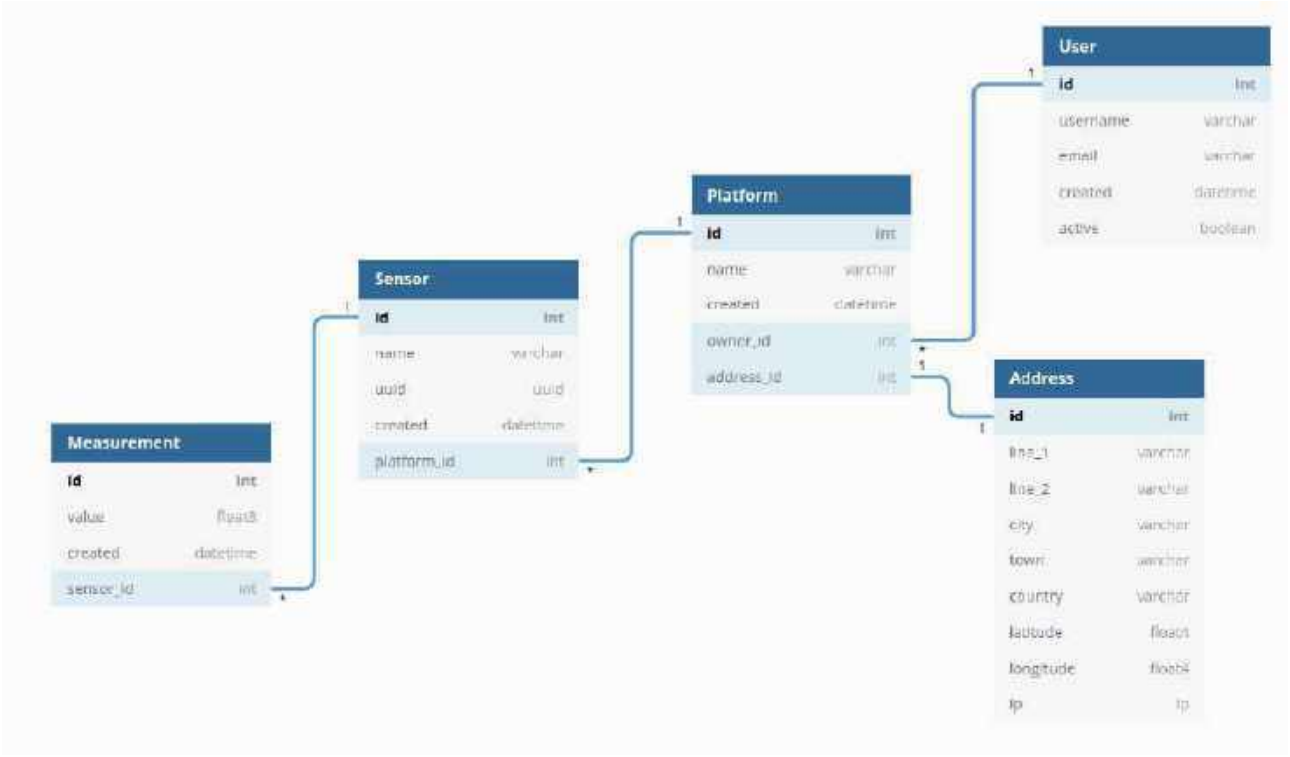
```
Table Measurement {
```

```
  id int [pk]
  value float8
  created datetime
  sensor_id int [ref: > Sensor.id]
}
```

Сущности связаны следующими отношениями:

1. Один пользователь может иметь несколько Платформ, но Платформа принадлежит одному Пользователю (один ко многим).
2. На одной Платформе может быть установлено несколько Сенсоров, но каждый Сенсор может быть только на одной Платформе (один ко многим).
3. Каждая Платформа имеет только один Адрес (один к одному).
4. Каждый Сенсор формирует Измерения (один ко многим).

Схематично, при помощи CASE-средств мы можем сформировать структуру БД следующим образом.



По заданию лабораторной работы данные должны быть защищены при хранении или при передаче. Если мы говорим о работе с внешним API по протоколу HTTPS, то это итак происходит благодаря алгоритмам, заложенным в сам протокол.

Если мы говорим о защите на уровне базы данных, то можно использовать специальные расширения типа pg_crypto для базы данных PostgreSQL (то есть решение на этом уровне зависит от типа базы данных используемой в проекте).

Если нужно защищать данные на программном уровне, то можно воспользоваться стандартным алгоритмом шифрования, входящим в используемый фреймворк или реализовать его самостоятельно, пользуясь примерами и документацией в сети Интернет.

Главное, разобраться во всех этапах создания системы или программно-аппаратных комплексов, в современных правилах формирования и сопровождения проектов.

Итак, в заключении цикла лабораторных работ 1 — 4 мы получили систему, которая выполняет следующие действия:

1. Получает данные от аналогового или цифрового датчика (или от их модели/внешнего API).
2. Проводит дискретизацию аналогового сигнала или семплирование цифровых данных.
3. Выполняет цифровую обработку одним из предложенных алгоритмов, например, скользящими средними или цифровым фильтром, в зависимости от типа данных.
4. Сохраняет информацию в базе данных.
5. Защищает информацию одним из методов шифрования.

Мы можем считать подобную систему полнофункциональным элементом системы Интернета Вещей, состоящим из аппаратной части (или ее модели) и программной части, базы данных и каналов связи.

Критерии оценивания:

Максимальное количество баллов за практическую работу: 10 баллов.
9-10 баллов– обучающийся в течение модуля активно принимает участие в устных опросах, даваемые им ответы верны и позволяют высоко оценить уровень владения материалом;

7-8 баллов – обучающийся принимает участие в устных опросах, даваемые им ответы, как правило, верны и позволяют оценить владение материалом на достаточном уровне;

6 баллов – обучающийся не проявляет инициативы для участия в устных опросах, а даваемые им ответы фрагментарны и верны лишь частично, что не позволяет оценить владение материалом на достаточном уровне;

1-5 баллов – обучающийся пытается участвовать в собеседовании, но дает неверные ответы, показывает отсутствие базовых знаний;

0 баллов – обучающийся не принимает участия в устных опросах, либо даваемые им ответы принципиально неверны.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Профессиональная и академическая коммуникация в области компьютерных наук»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024г.

`Breaking up long sentences

Whether they are Nobel Prize winners, Oxford professors, or first-year university students, all readers prefer sentences that they:

- only need to read once
- don't have to read slowly because the sentence does not require intense concentration
- can process word by word and thus understand the build-up of the author's logic immediately, rather than only being able to reach their interpretation of the whole meaning at the end of the sentence

These goals are much easier to achieve if you write short sentences. The average length of a sentence in English has become shorter and shorter over the centuries. In Shakespeare's time it was about 45 words, 150 years ago, about 29 words, and today's experts recommended between 15 and 18 words. In the world of academic writing, I think you should aim for an upper limit of around 25 words.

The referee of the paper where the following sentence appeared, asked the author to "delete this sentence or rewrite so that it means something sensible". 'Sensible' means something that makes sense. Note: I have changed the key words in this sentence to protect the author, but the structure is identical.

> Even if the occurrence of this particular form of pulmonary tumor occurs on a rare basis, since the behavior of these tumors is extremely difficult to predict and the histological features resembling a discrete cell tumor may lead to misdiagnose a C2 tumor as a C1 tumor, it would be of interest to characterize those lesions and to take them into account in the differential diagnosis of hereditary or congenital tumors.

The referee's criticism was very serious. He/She recommended that the sentence be deleted because in his/her opinion it seemed to make no sense. It made no sense because it was one long sentence containing a lot of very different ideas. The problem is that referees do not usually have the time to decipher your sentences and work out the connections between the ideas contained therein. If they do not understand immediately, then this is likely to aggravate them.

The author then rewrote the sentence as follows.

> This particular form of pulmonary tumor appears to be extremely rare. Its behavior is extremely difficult to predict. Moreover, the histological features, which resemble a discrete cell tumor, may mean that a C2 tumor is misdiagnosed as a C1 tumor. It would thus be interesting to characterize these lesions and to take them into account in the differential diagnosis of hereditary or congenital tumors.

By breaking up one long sentence into four shorter sentences, the author managed to explain his concepts more clearly. His original sentence contained 71 words. The rewritten version contains four sentences of 11, 7, 22 and 24 words, making a total of 64 words, so less words than the original sentence.

His paper contained one other such sentence - the referee's comment in this case was: "Cut this sentence - it is meaningless as it stands". These two sentences, plus a series of other minor changes to the English, were enough for the referee to recommend an initial rejection of the paper. The cost to the author was a delay of three months to publication. In the meantime another author could have published (but fortunately didn't!) a similar paper and thus deprive him of his 'scoop' (i.e. being the first person to report some new information).

Below are two other examples from other papers. Note how the RVs bring out the meaning much more clearly, by splitting the sentence up into different units of thought.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Since several organic pollutants, such as PCBs, can bioaccumulate within the trophic web, *at a level* directly related to environmental levels, and levels within an organism's diet, *for an accurate* risk assessment, all the information on congener levels in the biota and environment were integrated with ...	*It is known that* several organic pollutants, such as PCBs, can bioaccumulate within the trophic web. *This takes place at a level* directly related to environmental levels, and levels within an organism's diet. *Therefore to get an accurate* risk assessment, all the information on congener levels in the biota and the environment were integrated with ...
Thus for a correct evaluation of environmental risk, the analytical effort has to take a holistic approach, *in other words the bio-monitoring* and the chemical measurements have to be integrated, taking into account the diversity and similarities between organisms and between them and their environment, *to have as a complete a vision* as possible of all the possible transport routes, and ...	To assess environmental risks correctly, analyses thus need to take a holistic approach. *Bio-monitoring* and chemical measurements need to be integrated, taking into account any diversity amongst organisms and between organisms and their environment. *This would contribute towards a complete vision* of all the possible transport routes, and ....

Long sentences contain one or more of the following:

1. a link word or phrase (e.g. *and, moreover, in fact, although, due to the fact that*).
2. a list of items, most of which are qualified (i.e. by enumerating their characteristics). This is typical when authors describe a procedure that has many parts or some equipment / software that has many components.
3. one or more semicolon or colon, or a lot of commas. This is typical of an author who does not want to waste time organizing his/her thoughts in a way that will be clearer to the reader.

Link words

Link words and punctuation are used either add to or qualify the preceding part of the sentence, or to introduce a new idea. The resulting sentence in all three cases is often too long to be understood easily on a first reading.

Long sentences are caused by adding on too many parts to the main clause. First we need to decide what constitutes a long and complex sentence.

S1. We did several surveys, which all gave the same result.

S1 is ten words long. It is easy to read even though it has two parts (separated by the comma).

However, if we expand it too much it becomes more difficult to read:

S2. * We did several surveys aimed at investigating whether stress increases in proportion to the number of children a couple has *and* each survey led to the same result, i.e. that there is no correlation, *thus confirming* the hypothesis that stress in the family is generally connected to factors other than size.

S2 is 51 words long. It is still possible to understand on a first reading but it requires more effort on the part of the reader. Because it is so long, the reader cannot be sure which are the most important elements in it. The reader could assimilate and judge the weight of the information if the sentence were divided up into three parts.

S3. We did several surveys aimed at investigating whether stress increases in proportion to the number of children a couple has. Each survey led to the same result, i.e. that there is no correlation. This confirmed the hypothesis that stress in the family is generally connected to factors other than size.

In S3 the reader can easily and immediately understand the information because it is now presented in three shorter blocks. Basically, you should be able to read a sentence in one breath - try reading S2 aloud without stopping to breathe. It is not easy.

In S2 the words in italics (*and, thus confirming*) identify where the sentence could be stopped because they are used to add additional information.

So a good general rule is that if the first part of a sentence is more than 12–15 words long, don't add a second part that is more than 10–12 words.

The rest of this chapter examines how to divide up longer sentences into shorter sentences.

In the OV below, *and* is used in two different ways:

(1) to join two verbs (*speak and write*) and two nouns (*English and Italian*)

(2) to add additional information (*and that this is true .. and to this end*)

In the first case there is no problem, but the second usage makes the sentence too long (65 words). The revised version rearranges the order in which the information is given, and divides the sentence into three parts.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
The aim of this paper is to confirm that how we speak *and* write generally reflects the way we think *and* that this is true not only at a personal but also at a national level, *and* to this end two European languages were analyzed, English *and* Italian, to verify whether the structure of the language is reflected in the lifestyle of the respective nations.	How we speak and write generally reflects the way we think and act. *This* paper aims to prove that this thesis is true not only at a personal but also at a national level. *Two* European languages were analyzed, English and Italian, to verify whether the structure of the language is reflected in the lifestyle of the respective nations.

The OV below contains three ideas that are linked together using *and*, thus creating one long sentence.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
The treatments are very often expensive and technically difficult, *and* their effectiveness very much depends on the chemical and physical characteristics of the substances used for impregnation, *and* on their ability to ...	The treatments are very often expensive and technically difficult. *Their* effectiveness very much depends on the chemical and physical characteristics of the substances used for impregnation. *Also important* is their ability to ...

The RV replaces the first *and* with a full stop - which is generally the simplest way to reduce the length of a sentence. The second occurrence of *and* cannot simply be replaced by a full stop. Instead, the writer uses *also* to alert the reader of additional details and then uses *important* to recall the concept of effectiveness.

Sentences containing multiple uses of *and* are often found in the materials and methods sections of a paper. It is much easier for readers to understand what materials you used and what procedures you followed if you divide your descriptions into short sentences. Each sentence should only cover one or two items or steps – however see Sect. 15.4 for cases where this is not applicable.

S1. *All samples were collected at the same time (9 AM) every day to prevent any effects of possible circadian variation *and* then stored after treatment at 4°C until assay.

S2. All samples were collected at the same time (9 AM) every day to prevent any effects of possible circadian variation. *They* were then stored after treatment at 4°C until assay.

In S1 readers initially think that the *and* clause is going to introduce a second prevention. Readers then have to revise their perception when they realize that *and*

actually introduces the next step. S2 resolves this initial ambiguity by beginning a new sentence to highlight that the author is now talking about a different step. Here are two more examples that illustrate the same point.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Seeds, sterilized for 3 min in NaOCl (1% available chlorine) *and* rinsed with distilled water, were germinated on moist filter paper (Whatman No. 2) in Petri dishes *and* grown in the dark at 23°C.	The seeds were sterilized for 3 min in NaOCl (1% available *chlorine), and* rinsed with distilled water. *They* were then germinated on moist filter paper (Whatman No. 2) in Petri dishes and grown in the dark at 23°C.
At the beginning we performed 2D and 3D forward modeling of a medium where only the lithological discontinuities were taken into account *and* compared the apparent synthetic resistivity *and* phase curves with our experimental data.	At the beginning we performed 2D and 3D forward modeling of a medium where only the lithological discontinuities were taken into account. *We* then compared the apparent synthetic resistivity and phase curves with our experimental data.

as well as is used to add some additional information. It is often used as an alternative to *and* when the sentence might otherwise contain too many *ands* and would thus confuse the reader. If using *as well as* will create a very long sentence, it is best to break the sentence. However you cannot begin the new sentence with *as well as*. Instead you have to repeat some part of the previous sentence, as in the two RVs below

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
This finding could be explained by the specific properties of gold, silver and platinum *as well as* by the conditions in which these metals were found, for example silver was found in ...	(1) This finding could be *explained* by the specific properties of gold, silver and platinum. *Another explanation could be* the conditions ...
	(2) ... silver and platinum. *The conditions* in which these metals were found could *also* be an *explanation*. For example, ...

You cannot always break up a long sentence that contains a link by beginning a new sentence using that link word. This is because not all link words can be used at the beginning of a sentence. For example, when *whereas* is used to compare two findings in one long sentence, it should be replaced with *on the other hand* when the sentence is split into two.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
The levels of cadmium in Site C were comparable to the levels found in Sites A and B in the previous years, *whereas / on the other hand* the levels for copper were much lower in Site C with respect to the values found in the previous sampling campaigns in 2008 and 2010.	The levels of cadmium in Site C were comparable to the levels found in Sites A and B in the previous years. *On the other hand*, the levels for copper were much lower in Site C with respect to the values found in the previous sampling campaigns in 2008 and 2010.

The use of *although* and *however* is the same as with *whereas* and *on the other hand*, respectively.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
The levels of cadmium in Site C were comparable to the levels found in Sites A and B in the previous years, *although / however* this was not the case for the levels found in the south-east part of Site C.	The levels of cadmium in Site C were comparable to the levels found in Sites A and B in the previous years. *However*, this was not the case for the levels found in the south-east part of Site C.

Although can only be used in a two-part sentence, where one part depends on the other. For example:

> *Although this book* was written for non-native speakers, *it can also* be used by native speakers.

In the RV above, *although* would not be possible because there is no dependent clause.

These link words are used to explain the reasons for 'something' that has just been mentioned (S1) or is about to be mentioned (S2). The 'something' to be done in the examples below is to simplify a procedure.

S1. *It was found necessary to make some simplifications to our procedures (essentially we did A, B and C), due to the difficulties in measuring the weight of the various compounds, particularly with regard to the weights of X, Y and Z.

S2. *Owing to the difficulties in measuring the weight of the various compounds, particularly with regard to the weights of X, Y and Z, it was found necessary to make some simplifications to our procedures, essentially by doing A, B and C.

In such cases, it might be clearer for the reader if you split the sentence into three (S3).

S3. We encountered difficulties in measuring the weight of the various compounds, particularly the weights of X, Y and Z. We thus decided to make some simplifications to our procedures. This entailed doing A, B and C.

Words such as *since* and *although* are often used in a subordinate clause at the beginning of a sentence, as in S1 below.

S1. **Since* English is now spoken by 1.1 billion people around the world and is used as a lingua franca in many international business and tourism scenarios between people of different languages and between native English speakers and non-native speakers, *the learning of foreign languages in the United Kingdom has suffered a huge decline*.

The problem with S1 is that readers are forced to carry an idea in their head before they understand how this idea relates to the idea in the main clause (in italics). It would be much easier for readers to understand if S1 was split into two parts and rewritten as in S2.

S1. English is now spoken by 1.1 billion people around the world and is used as a lingua franca in many international business and tourism scenarios between people of different languages and between native English speakers and non-native speakers. The consequence is that the learning of foreign languages in the United Kingdom has suffered a huge decline.

Like *although* (see Sect. 3.8) the link words *since* and *as* require a dependent clause. For example:

Since / As you are a PhD student, *you* probably have to write a lot of papers in English.

This means that *since* and *as* could not be used in the RV below.

ORIGINAL	REVISED
The chemical characterization of organic paint materials in works of art is of great interest in terms of conservation, *because / since / as* the organic components of the paint layer are particularly subject to degradation.	The chemical characterization of organic paint materials in works of art is of great interest in terms of conservation. *This is because / In fact* the organic components of the paint layer are ...

which is used to add information. For example:

S1. English is now the world's international language, *which* is why it is used in scientific papers.

S2. English, *which* has now become the world's international language, is studied by more than a billion people.

S3. English, *[which is] now spoken* by more than a billion people, is the world's international language.

In S1 *which* is used to introduce an additional piece of information (in this case an explanation). In S2 *which* gives some extra information about the subject of the sentence (the English language). In S3, *which* serves the same purpose as in S2, it is in brackets because it could be cut.

In all three cases, the meaning is quick and easy to understand because the sentences are quite short.

Problems arise when sentences are longer, as highlighted in the OV below.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
English is now the world's international language and is studied by more than a billion people in various parts of the world thus giving rise to an industry of English language textbooks and teachers, *which* explains why in so many schools and universities in countries where English is not the mother tongue it is taught as the first foreign language in preference to, *for example*, Spanish or Chinese, *which* are two languages that have more native speakers than English.	English is now the world's international language and is studied by more than a billion people in various parts of the world thus giving rise to an industry of English language textbooks and teachers. *This* explains why in so many schools and universities in countries where English is not the mother tongue it is taught as the first foreign language. *For example*, English is taught in preference to Spanish or Chinese, *which* are two languages that have more native speakers than English.

In the OV the introduction of two new pieces of information using *which* makes the sentence unnecessarily long (79 words). In the RV, the first occurrence of *which* is replaced by *this*, which stands for *this fact*. Using *this* either alone or associated with a noun (e.g. *this fact, this decision, this method*) is a very common and useful way to reduce the length of a sentence.

The OV below contains an example of the use of *which* as in S2.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
English, *which* has now become the world's international language and is studied by more than a billion people in various parts of the world thus giving rise to an industry of English language textbooks and teachers, is generally used in scientific papers.	(1) English is generally used in scientific papers. In fact, English has now become the world's international language and is studied by more than a billion people in various parts of the world. This has given rise to an industry of English language textbooks and teachers. (2) English has now become the world's international language and is studied by more than a billion people in various parts of the world. This has given rise to an industry of English language textbooks and teachers. Today, English is generally used in scientific papers.

In the OV, the subject (*English*) and the main verb (*is*) are separated by 35 words. This means that by the time readers reach the main verb, they may have forgotten what the subject is.

There are two ways to resolve this problem. In the first RV, the author has decided to make scientific papers the key topic, so now this appears at the beginning of the sentence rather than at the end. In the second RV, the author first gives some information about English and then talks about scientific papers. The choice of using the first or the second technique, will depend on the emphasis you want to give to each piece of information.

The OV below contains an example of the usage given in S3. Even in short sentences, this kind of usage is dangerous as you may not know whether you can or cannot omit *which*.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
English, [which is] *now spoken* by more than a billion people from all over the world, the biggest populations being those in China and India, and more recently in some ex British colonies in Africa, is the world's international language.	English is the world's international language. It is *now spoken* by more than a billion people from all over the world. The biggest populations are those in China and India, and more recently in some ex British colonies in Africa.

The OVs below show two other examples where *which* has been omitted. Note how the words *area* and *distinction* are repeated. This repetition is not considered bad style in English scientific writing.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Using the method described by Peters et al. (2010), we assessed the state of pollution of three sites in a coastal area [which was] *characterized* by high levels of agricultural, industrial and tourist activity, as well as occasional volcanic activity (the last major eruption was in 1997).	Using the method described by Peters et al. (2010), we assessed the state of pollution of three sites in a coastal area. *This* area *is characterized* by high levels of agricultural, industrial and tourist activity, as well as occasional volcanic activity (the last major eruption was in 1997).
Using the approach described by Smith and Jones (2011), a *distinction*, [which was] *useful* for analysis purposes, particularly in the final stages of the project, was made between the three types pollution: agriculture, industry and tourism.	Using the approach described by Smith and Jones (2011), a *distinction* was made between the three types of pollution: agriculture, industry and tourism. *This distinction* was useful for analysis purposes, particularly in the final stages of the project.

Another way writers typically link phrases together is to use the - *ing* form a verb. If using the - *ing* form will significantly add to the length of a sentence, you can use another form of the verb and begin a new sentence.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Using automatic translation software (e.g. Google Translate, Babelfish, and Systran) can considerably ease the work of researchers when they need to translate documents *thus saving* them money (for example the fee they might have otherwise had to pay to a professional translator) and *increasing* the amount of time they have to spend in the laboratory rather than at the PC.	Using automatic translation software (e.g. Google Translate, Babelfish, and Systran) can considerably ease the work of researchers when they need to translate documents. *Such software saves* them money, for example the fee they might have otherwise had to pay to a professional translator. It *also increases* the amount of time they have to spend in the laboratory rather than at the PC.

The RV above shows two ways to deal with the - *ing* form. First, you can repeat the subject (*software*) and then change the -ing form into the present tense (*saves, increases* rather than *saving, increasing*), or whatever tense is appropriate.

In the OV below, the - *ing* form is used instead of a relative clause: the author could have written *which indicates*. In such cases, you can break the sentence immediately before the - *ing* form and then start a new sentence with *This*.

Often you need to explain the rationale for adopting a particular procedure or line of research. To do this, writers typically use expressions such as *in order to, with the purpose of, with the aim to, in an attempt to*

This is fine if you can express the rationale in a few words, as in this example:

In order to test our hypothesis, we sampled a random selection of documents.

But if your rationale is longer than about 15 words, you probably need to split the sentence up, as shown below:

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Our readability index is based on a series of factors - length of sentences and paragraphs, use of headings, amount of white space, use of formatting (bold, italics, font size etc.) - in order to provide writers with some metrics for judging how much readers are likely to understand the writers' documents.	*We wanted to provide* writers with some metrics for judging how much readers are likely to understand the writers' documents. *We thus produced a readability index* based on a series of factors - length of sentences and paragraphs, use of headings, amount of white space, and use of formatting (bold, italics, font size etc.).
In order to establish a relationship between document length and level of bureaucracy and to confirm whether documents, such as reports regarding legislative and administrative issues, vary substantially in length from one language to another, *we conducted an analysis of A, B and C.*	(1) *We conducted* an analysis of A, B and C. *The aim of the analysis was to* establish .... (2) *We wanted to establish* a relationship between ... language to another. *To do this*, we conducted ...

The two techniques shown in the RV are

- either say what you did and then why you did it
- or give your rationale and then say what you did

The first is generally more helpful for the reader because it helps to put the rationale in context.

When commas are used in lists, they are fine:

> Many European countries are now part of the European union, these include France, German, Italy, Portugal, Spain, ...

However, when commas are used to separate various clauses within a sentence, readers have to constantly adjust their thinking. Also, the more commas there are in a sentence, the longer the sentence is likely to be.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
As a preliminary study, in an attempt to establish a relationship between document length and level of bureaucracy, we analyzed the length of 50 European Union documents, written in seven of the official languages of the EU, to confirm whether documents, such as reports regarding legislative and administrative issues, vary substantially in length from one language to another, and whether this could be related, in some way, to the length of time typically needed to carry out daily administrative tasks in those countries (e.g. withdrawing money from a bank account, setting up bill payments with utility providers, understanding the clauses of an insurance contract). The results showed that ...	Our aim was to see if there is a direct relationship between the length of documents produced in a country, and the length it takes to do simple bureaucratic tasks in that country. Our hypothesis was: the longer document, the greater the level of bureaucracy. In our preliminary study we analyzed translations from English into seven of the official languages of the European Union. We chose 50 documents, mostly regarding legislative and administrative issues. We then looked at the length of time typically needed to carry out daily administrative tasks in those countries. The tasks we selected were withdrawing money from a bank account, setting up bill payments with utility providers, and understanding the clauses of an insurance contract. The results showed that ...

The OV demonstrates that the excessive use of commas is a sign of lazy writing. The writer simply begins a sentence and keeps adding details to it, without thinking about how the reader will assimilate all these details. It also indicates that the writer is probably not clear in his / her own mind about what he / she wants to say.

Note that the RV:

- uses more words in total, but is considerably easier to follow
- rearranges the various subordinate clauses and puts them into a more logical order and in separate sentences
- divides up the information into paragraphs - the first explains the rationale, the second shows how the investigation was carried out. This makes the connection between ideas much clearer

Commas can also be dangerous if you use them to build up a series of phrases each of which describes the previous one, as in S1.

S1. In particular, the base peak is characteristic of the fragmentation of dehydroabietic acid, the main degradation marker formed by aromatization of abietadienic acids, the major constituents of pine resins.

Initially when reading S1 it seems that the *peak* is a *characteristic* of a series of items separated by commas. Then as we read further we understand that *the main degradation marker* is not in fact a second element in a series of items. Given that *the main degradation marker* comes immediately after *dehydroabietic acid* we assume that this acid must be a *marker*. We then realize that in fact it refers back to *fragmentation*. S1 thus requires much interpretative effort by the reader and is better rewritten as in S2:

S2. The base peak is characteristic of the fragmentation of dehydroabietic acid. *This fragmentation* is the main degradation marker formed by aromatization of abietadienic acids, *which are* the major constituents of pine resins.

S2 divides S1 into two separate sentences and also clarifies the relationships between the various elements.

Semicolons (;) are not commonly used in modern English. If you tend to use a semicolon before introducing an additional idea or additional information, think about using a period (.) instead.

By 1066 English, or Old English as it is known, was firmly *established; it* was a logical language and was also reasonably phonetic. This situation changed dramatically when England was invaded by the Normans in *1066; in* fact, for the next 250 years French became the official language, and when English did come to be written again it was a terrible concoction of Anglo-Saxon, Latin and French.

The author of the above extract used semicolons to show that the two parts of the sentence to some extent depend on each other. Although this usage could be considered correct, today it is considered as unnecessary. Thus the two semicolons could easily be replaced by full stops, with no change of meaning for the reader.

When we read we automatically pause for an instant when we reach a full stop. This is our mental equivalent to pausing and inhaling air when we are speaking. Semicolons don't allow for such a pause and thus make the reading process slightly more tiring. Semicolons also make the sentence look longer, which makes them more tiring on our eyes.

Some writers also use a colon (:) in the same way as a semicolon. Again, if your sentence is going to be very long as a result of using a colon, it is better to replace the colon with a full stop and begin a new sentence.

S1. Old English had two distinct advantages over Modern English: it had a regular spelling system and was phonetic.

S2. Old English, which is the language spoken in most parts of England over 1,000 years, was a relatively pure language (the influence of Latin had not been particularly strong at this point, and the French influence as a result of the Norman Conquest was yet to be felt) and had two distinct advantages over Modern English: it had a regular spelling system and the majority of words were completely phonetic.

S3. Old English was the language spoken in most parts of England over 1,000 years. It was a relatively pure language since the influence of Latin had not been particularly strong at this point, and the French influence as a result of the Norman Conquest was yet to be felt. It had two distinct advantages over Modern English: it had a regular spelling system and the majority of words were completely phonetic.

In S1 the use of the colon (:) is fine, because the whole length of the resulting sentence is less than 20 words. But S2 is already too long even without the subsidiary clause introduced by the colon. S2 would in fact be better divided up into three parts as in S3.

The only time you really need to use semicolons is to divide up short lists to show how each element in the list relates to each other. Note how S2 is clearer than S1 through the helpful use of semicolons.

S1. *The partners in the various projects are A, B and C, P and Q, X and Y and Z.

S2. The partners in the various projects are A, B and C; P and Q; X; and Y and Z.

S2 shows more clearly that there are four groups of partners: (1) A, B, C; (2) P, Q; (3) X; (4) Y, Z.

But if your list is long, as in the OV below, it is better to divide it up into shorter sentences.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Our system is based on four components: it has many data files (the weather, people, places, etc.); it has procedures which it tries to use to combine these files by working out how to respond to certain types or patterns of questions (this entails the user knowing what types of questions it can answer); it has a form to understand the questions posed in a natural language (so the user may need to know English) which it then translates into one of the types of questions it knows how to answer; finally, it has a very powerful display module, which it uses to show the answers, using, graphs, maps, histograms etc.	Our system is based on four components. Firstly, it has many data files, for example the weather, people, and places. Secondly, it has procedures which it tries to use to combine these files by working out how to respond to certain types or patterns of questions and this entails the user knowing what types of questions it can answer. Thirdly, it has a form to understand the questions posed in a natural language, which means the user needs to know English. It then translates the natural language into one of the types of questions it knows how to answer. Finally, it has a very powerful display module, which it uses to show the answers. These answers are shown using graphs, maps, histograms etc.

The RV is longer than the OV but it is much clearer for the reader because it:

- uses six short sentences rather than one long one. The semicolons have been replaced by full stops.
- clearly distinguishes the four components by using *firstly, secondly* etc.
- removes the brackets

Phrases in parentheses can considerably increase the length of a sentence. Parentheses are best used just to give short lists that act as examples. For example:

> Several members of the European Union (e.g. Spain, France and German) have successfully managed to reduce their top tax threshold from 42 to 38%.

In the example above the information in parentheses does not interrupt the logical flow of the sentence and it does not occupy much space.

Parentheses should be avoided when giving explanations or examples that are not lists. For example:

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Using automatic translation software *(e.g. Google Translate, Babelfish, and Systran)* can considerably ease the work of researchers when they need to translate documents thus saving them money *(for example the fee they might have otherwise had to pay to a professional translator)* and increasing the amount of time they have to spend in the laboratory rather than at the PC.	Using automatic translation software *(e.g. Google Translate, Babelfish, and Systran)* can considerably ease the work of researchers when they need to translate documents. Such software saves them money, *for example the fee they might have otherwise had to pay to a professional translator.* It also increases the amount of time they have to spend in the laboratory rather than at the PC.

In the OV the first use of parentheses is fine, but the second interrupts the flow of the sentence and considerably adds to its length.

Structuring paragraphs and sentences

Every paper has a title and the readers know where to find it, i.e. at the top of the first page of the paper. Readers know that the title will be followed by the Abstract and at (or towards) the end of the paper they expect to find the Literature Cited.

Just as readers have certain expectations with regard to the structure of the entire paper, they also have expectations with regard to how a section, paragraph and a sentence should be structured. These expectations are less conscious or explicit than expectations regarding the position of a title and the abstract. However they are based on how readers usually find and receive information in a section, paragraph and sentence.

Each paragraph is like a microcosm of a paper – it has its own title (the topic sentence), the intermediate sentences are like the sections of the paper, and the last sentence is like the conclusions.

A well-structured paragraph in any other part of a section (i.e. not the first paragraph) is thus generally as follows:

1. A topic sentence that tells the reader what the paragraph is about and in some way connects with the previous paragraph.
2. From one to eight sentences in a logical sequence that develop the topic.
3. A concluding sentence, possibly referring back to the first sentence or forward to the next paragraph.

The three elements of this structure are dealt with in detail in the subsections below.
Your aim is to show readers how your paragraph fits in with what came before and what is coming after. You need to organize your information for the reader, rather than the reader trying to organize the information that you have given him / her. Only one specific idea should be covered in each sentence, and only one general idea in each paragraph.

Known information is traditionally placed at the beginning of a sentence or paragraph. Below are the first three sentences from the abstract of a fictitious paper entitled 'Readability and Non-Native English Speakers' intended for a journal dedicated to communication in the world of business.

VERSION 1 Readability formulas calculate how readable a text is by determining the level of difficulty of each individual word and the length of sentences. All types of writers can use these formulas in order to understand how difficult or readable their texts would be for the average reader. However, readability formulas are based purely on what is considered difficult for a native English speaker, and do not take into account problems that may be encountered by non-natives. In this paper ...

The first word, *readability*, is one of the author's key words. It immediately alerts the reader to the topic of the sentence and of the abstract (and paper) as a whole. However, the information contained in it is not new - readability formulas and their indexes are well established in the literature on business communication.

The role of the first two sentences is thus to set the context and gently guide the reader into the paragraph. The third sentence then introduces the new element, i.e. the fact that readability indexes do not take into account non-native speakers. The third sentence thus highlights the problem that the paper intends to tackle.

However, the abstract could have begun like this:

VERSION 2 Current readability formulas are based purely on what is considered difficult for a native English speaker. They fail take into account problems that may be encountered by non-natives. One thousand five hundred PhD students from 10 countries were asked to evaluate the difficulty of five technical texts from their business discipline written by native English speakers. Three key difficulties were found: unfamiliar vocabulary (typically Anglo-Saxon words), unfamiliar cultural references, and the use of humor. The paper also proposes a new approach to assessing the level of readability of texts to account for such difficulties.

In Version 2, the author still begins with his key word, *readability*. But he precedes it with *current*, which signals to the reader that the author will then probably propose an alternative. The author also assumes that his readers will be aware of what a readability formula is, so he feels he doesn't need to mention it. Thus, in the second sentence he immediately underlines a critical problem with current formulas. In the third sentence he then tells his readers what his research was and then what was found.

Version 3, below, contains only new information.

VERSION 3 Unfamiliar vocabulary (typically Anglo-Saxon words), unfamiliar cultural references, and the use of humor: these, according to our survey of 1500 PhD students, are the main difficulties non-native speakers have when reading a business text in English. Our

results highlight the need to adjust current readability formulas in order to take non-native speakers into account. The paper also proposes a new approach to assessing the level of readability of texts to account for such difficulties.

This version is designed to immediately attract the reader's attention. In contrast, the first 50 words of Version 1 contain no new information at all. Version 2 has 40–50% new information or more, depending on whether readers are familiar with the limitations of readability formulas with regard to non-natives.

So, which version should you use?

The best version to use depends on two factors:

1. the section of the paper
2. what you are trying to achieve

Version 1 would only be appropriate in an Abstract if the journal where it is being published does not usually deal with communication and / or readability indexes. In this case the readers need the context to be set for them. It might be more acceptable in an Introduction in a slightly more specialized journal. In an Introduction the aim is not principally to attract attention, if readers are reading your Introduction you can presume that you already have their attention.

So the information contained in Version 1 would be used in an Introduction just to remind the readers of the context. This is a very typical way to begin an Introduction - it is what readers expect and therefore it is generally a good technique.

Version 2 would be appropriate as an Abstract or Introduction in a specialized journal on business communication.

Version 3 would only be appropriate in an Abstract and exclusively in a very specialized journal. It can only be used if you have clear findings, or a clear new methodology, to report. It works very well because it does not force readers to read background information that they are probably already familiar with.

You might also choose Version 3 as an Abstract for a congress. In such cases you are competing for the attention of the referees who will use your Abstract to decide whether to include your contribution at the congress. If your Abstract is accepted, you will then be competing with other authors / presenters in motivating the audience to come and watch you rather than a parallel session.

In many languages Versions 2 and 3 would not be acceptable. In the words of one of my Greek PhD students:

New information in Greek comes at the very end. The rule is that first the author gives extensive background information and only at the end he / she introduces the new concept. This is the generally accepted (and considered correct) way of writing.

This means that when you write in English you may be going against what is considered good style in your own language. But don't let breaking a taboo stop you from expressing yourself in the way that will best highlight your results and thus attract more readers.

S1 and S2 begin with the same subject *English*, which is the main topic of the sentence. They then present the same two pieces of information, but in a different order.

S1. English, which is the international language of communication, is now studied by 1.1 billion people.

S2. †English, which is now studied by 1.1 billion people, is the international language of communication.

In both cases if you removed the 'which' clause (in italics) the sentence would still make sense. But if you removed the final clause it wouldn't. This would seem to indicate that the final clause is where we locate the most important information. Thus the relative position of the various parts of the phrase tells the reader the relative importance of the information contained on those parts.

In S1, the order of the information tells you that the fact that English is *the international language of communication* is old news, but that *1.1 billion people* is new information that the reader probably does not already know. Thus, the order of the information in S2 is a little strange because it puts the new information (*1.1. billion people*) before the old information (*international language*).

Readers tend to focus on the first and last words of a sentence, so avoid placing your most important information in the middle of a long sentence. Readers don't want to make an effort to identify the key points, they want to be told immediately.
Here are some more examples that show how by changing the order of information within a sentence you can achieve a different effect:

S3. English is now studied by 1.1 billion people, though this number is expected to drop with the rise in importance of Chinese.

S4. Although English is now studied by 1.1 billion people, this number is expected to drop with the rise in importance of Chinese.

S5. Although the importance of Chinese is expected to lead to a drop in the numbers of people studying English, 1.1 billion people still study English.

S3–S5 all contain the same information, but the weight that this information is given varies.

In S3 the reader learns some information. This information is then qualified with *though*, which is used to introduce some new information that the author imagines that the reader does not know.

In S4 the reader is immediately alerted to the fact that the information contained at the beginning of the sentence is going to be qualified by new information in the second part. The order of the information in S4 is thus more logical than in S3.

In S5 the writer assumes that the reader already knows the importance of Chinese and instead focuses on the fact that despite the increase in the number of Chinese speakers, English is *still* studied by a lot of people. 'still' is the key word and it is located very close to the end of the sentence.

In S1–S5 there are two parts to each sentence, and the writer gives more emphasis to the second part. Sometimes, you may want to give equal weight to the two parts.

S6. English is the international language of communication. It is now studied by 1.1 billion people.

S7. The importance of Chinese is expected to lead to drop in the numbers of people studying English. Despite this, 1.1 billion people still study English.

In S6 and S7, the writer wants the reader to notice and absorb the two pieces of important information separately. She does this by presenting the information in two distinct sentences. This device should not be used too often because it can lead to a series of very short sentences, which after a while begin to sound like a list.

A key issue when linking up sentences in a paragraph is to decide how to link one sentence to the previous one. The following is an extract from the beginning of a paragraph from a paper on pollution in soil. It fails to make a strong impact because of its lack of logical progression.

(S1) The *soil* is a major source of *pollution*. (S2) Millions of *chemicals* are released into the environment and end up in the soil. (S3) The impact of most of these *chemicals* on human health is still not fully known. (S4). In addition, *in the soil* there are naturally occurring amounts of potentially *toxic substances* whose fate in the terrestrial environment is still *poorly known*.

S1 puts *the soil* as the topic of the sentence. S2 is more specific and talks about the quantity of this pollution - *millions of chemicals*. S3 reports the impact of the chemicals mentioned in S2. But S4 does not continue this logical progression from general to increasingly more specific. Instead, it begins by putting *soil* in the topic position. This breaks the logical progression, because *soil* was the topic of S1. The following sentence would be a good replacement for S4, which would thus continue the logical structure developed in S1–S3.

> S5 There are also naturally occurring amounts of potentially toxic substances *in the soil* whose fate in the terrestrial environment is still poorly known.

The formula is thus:

1. S1: main topic (*soil*) introduces subtopic 1 (*pollution*)
2. S2: subtopic 1 is specified by introducing subtopic 2 (*millions of chemicals*).
3. S3: subtopic 2 is specified introducing subtopic 3 (*impact of these chemicals*).
4. S4: a further / related aspect of subtopic 3 is introduced via subtopic 4 (*impact of toxic substances, i.e. chemicals, is poorly understood*).
5. etc.

Basically each sentence is link in a chain. A full chain is a paragraph. And a series of linked chains makes up a section.

This concept of a chain of logical progression is not common to all languages. Here is what Nobel Prize Winner in Physics, Tony Leggett, notes about Japanese:

> In Japanese it seems that it is often legitimate to state a number of thoughts in such a way that the connection between them, or the meaning of any given one, only becomes clear when one has read the whole paragraph or even the whole paper. This is not so in English; each sentence should be completely intelligible in the light of what has *already* been written. Moreover, the connection between one thought and the next should be completely clear when it is read; for instance, if you deviate from the 'main line' of the thought to explore a side-track, this should be made clear at the point where the sidetrack *starts*, not where it finishes.

ORIGINAL VERSION (OV)	REVISED VERSION (RV)
Memory can be subdivided into various types: long-term memory, which involves retaining information for over a minute, and short-term memory, in which information is remembered for a minute or less, for example, the memory required to perform a simple calculation such as 5×7×3. Another type of short-term memory is also recognized: sensory memory, for example we see a video as a continuous scene rather than a series of still images. Research shows sex differences in episodic (i.e. long term) memory: women tend to remember better verbal situations, whereas men have a better recollection of events relating to visuals and space. Long-term memory can be further subdivided into recent memory, which involves new learning, and remote memory, which involves old information.	Memory is the capacity to store and recall new information. It can be subdivided into two main types: short-term and long-term. Short-term memory involves remembering information for a minute or less, for example, the memory required to perform a simple calculation such as 5×7×3. Another type of short-term memory is sensory memory, for example, we see a video as a continuous scene rather than a series of still images. Long-term memory can be further subdivided into recent memory, which involves new learning, and remote memory, which involves old information. Interestingly, research shows sex differences in remote memory: women tend to remember better verbal situations, whereas men have a better recollection of events relating to visuals and space.

In the OV, the beginning of the first sentence gives the illusion to the reader that the various types of memory will be introduced in a logical order. In reality a rather random selection of information is given, with no clear sequence. This makes it hard for the reader to follow. The RV uses shorter sentences and follows a much more logical series of steps:

(1) definition of memory given

(2) clear indication of the number of types of memories (OV *various* types, RV *two main* types)

(3) short-term memory mentioned first, as later in the paragraph long-term memory will be developed in more detail

(4) additional information about short-term memory (the discussion of short-term memory ends here)

(5) returns to second topic (long-term memory), which is then subdivided into *recent* and *remote*

(6) interesting fact about remote memory

In the RV, each sentence extends the information given in the previous sentence, and the reader can sense the logical progression. The author presents a list of topics at the beginning of a paragraph that he intends to discuss further in the later part of the paragraph. He then deals with the topics in the same order and format as he initially presented them: first short-term memory, then long-term.

Your aim is to provide readers with a step-by-step approach to enable them to understand your reasoning. It must be clear from the beginning of your sentence what this logical progression is. This means that at mid point or end point in a sentence, readers should not have to change their perspective of this logical progression. OVs 1–5 below are all correct English, but they don't help the reader to follow your logical flow.

	ORIGINAL VERSION (OV)	REVISED VERSION (RV)
1	It is important to remark that our components are of a traditional design. *However*, we want to stress that the way the components are assembled is very innovative.	*Although* our components are of a traditional design, the way they are assembled is very innovative.
2	Working in this domain entails modifying the algorithms as *we are dealing* with complex numbers.	*Since we are dealing* with complex numbers, working in this domain also entails modifying the algorithms.
3	Therefore, the rescaled parameters seem to be appropriate for characterizing the properties, *from a statistical point of view*.	Therefore, *from a statistical point of view*, the rescaled parameters seem to be appropriate for characterizing the properties.
4	The number of times this happens when the user is online is generally *very few*.	This *rarely* happens when the user is online.
5	Documentation on this particular matter is almost *completely lacking*.	There is *virtually no documentation* on this particular matter.
6	*Consequently we found this particular type of service not interesting.	Consequently we did not find this particular type of service interesting.

The RVs all provide signals to the reader about what they can expect next.

In OV1 readers initially think that *traditional design* is the key information that the author wants to give them. The author then introduces new information that completely contrasts with the preceding information. In such cases, you need to forewarn your readers of such contrasts by using a linker that introduces a qualification, such as *although*, at the beginning of the phrase (as in RV1).

In RV2 and RV3 the author immediately tells readers the point of view he wants them to assume, whereas in OV2 and OV3 this key information is only given at the end of the sentence. The strategy adopted in RV2 also enables you to present the information in chronological order: (1) what we already know (2) new information.

In the OVs 4–6, readers initially think that something affirmative is being said, but then they have to readjust their thinking when the negation is introduced at the end of the sentence. English tends to express negative ideas with a negation. This helps the reader to understand immediately that something negative is

being said (RV4 and RV5). OV6 is incorrect English because the verb and the negation (*not*) have been separated. Generally *not* is located immediately before the verb.

When you need to describe the various stages in a procedure, methodology, project and so on, it helps to use a numbering system. For example, *first(ly), second(ly), third(ly), finally*. It is also important to continue your numbering system in the same way that you started it, and not to abandon it. Compare these two versions:

ORIGINAL VERSION	REVISED VERSION
Our methodology can be divided into three main parts: first of all the characterization of demographic changes between 2000 and 2010, in order to obtain a scenario for the future with regarding to population shifts. The results from this first part were used as inputs to obtain maps for 2010 to 2015. The resulting maps and input maps regarding climatic and political characteristics were inserted into our model in order to predict future patterns.	Our methodology can be divided into three main stages. *Firstly*, we characterized demographic changes between 2000 and 2010, in order to obtain a future scenario for population shifts. *Secondly*, we used the results from the first part as inputs to obtain maps for 2010 to 2015. *Finally*, the resulting maps along with input maps regarding climatic and political characteristics were inserted into our model in order to predict future patterns.

The OV is a little misleading. The colon in the first sentence gives the reader the impression that the author is going to mention all three stages together within the same sentence. The second two stages are not clearly marked. The RV separates the OV's first sentence into two parts. In the RV, first the author announces that there are three stages. Then she talks about these three stages in three separate sentences, which begin with a number indicator. This also makes the paragraph visually easier to follow.

The only advantage of a long paragraph is for the writer, not for the reader. It enables writers to save time because they avoid having to think about where they could break the paragraph up to aid reader comprehension. But breaking up long paragraphs is extremely important.

Firstly, long blocks of text are visually unappealing for readers, and tiring for their eyes. They fail to meet the basic rule of readability – make things as easy as possible for your reader. Evidence of this can be found in newspapers. If you look at newspapers from 100 years ago, they were basically big blocks of text that took a great deal of effort to read. Today many online newspapers have one sentence per paragraph, with lots of white space between each paragraph.

Secondly, your points and the related logical sequence of these points will be much more clearly identifiable for the reader if they are in a separate paragraph.

Thirdly, you will find that you will write more clearly if you use shorter paragraphs. This is because it will force you to think about what the main point of your paragraph is and how to express this point in the simplest way. If you just have one long paragraph, the tendency is just to have one long flow of frequently disjointed thoughts. This tendency is known in English as 'rambling'.

Fourthly, having shorter paragraphs enables you (and your co-authors) to quickly identify if you need to add extra information, and allows you to do this without having to extend an already long paragraph. Likewise, it enables you to identify paragraphs that could be cut if you find you are short of space.
The maximum length of a paragraph in a well-written research paper is about 15 lines. But most paragraphs should be shorter. If you have already written more than 8–12 lines or 4–6 sentences, then you may need to re-read what you have written and think about where you could start a new paragraph.

When you begin to talk about something that is even only slightly distinct from what you have mentioned in the previous 4–6 sentences, then this is a good opportunity to begin a new paragraph. For example, when you have been talking about how another author has approached the problem of X, and you then want to make a comparison with your own approach. The topic (i.e. X) is the same, but the focus is different. Likewise, if you have been comparing X and Y, and you have spent a few sentences exclusively on X, then when you start on Y you can use a new paragraph.

The table below shows the typical phrases used to connect one sentence to the next in order to create a logical progression of thought. These typical phrases also act as markers to indicate that you could begin a new paragraph.

TYPICAL PHRASES	FUNCTION OF THE PHRASE
In order to do this / To this end / With this mind	To state the purpose of something. For instance, you outline a requirement, and then you begin to say how you could meet this requirement
Then / Following this / Afterwards	To indicate a temporal relationship
For example, / An example of this is / In fact, / Unlike / Nevertheless,	To give an example or supporting/negating evidence. By 'example' I don't mean just a list of items, but a complete example or evidence that supports or negates what you have just been saying and that requires several sentences to explain
In addition / Another way to do / An additional feature of	To add additional points. For instance, if you are focusing just on one thing (e.g. X) and you talk about X's attributes
On the other hand / However / In contrast	To qualify what you have just said: i.e. to indicate an exception or the two sides of an argument
Due to / Since / Although	To give reasons for something
Thus / Therefore / Consequently / Because of this	To indicate a consequence
This means that / This highlights that / These considerations imply that / In conclusion / In sum	To announce and give a mini conclusion about what you have said in the previous sentences
Figure 1 shows / As can be seen in Table 2	To talk about figures, tables etc.
Firstly, secondly, finally	To introduce elements in a list
As far as X is concerned, / In relation to X, In the case of / With regard to / As noted earlier	To introduce a new element; to recall something mentioned earlier
It is worth noting that / Interestingly	To add some additional information or make some comment, not necessarily directly about something you have mentioned before but as an aside.

In all the examples in the table, I am talking about cases where you need at least three sentences (or two quite long ones) to achieve the function desired. For example, when you use *firstly, secondly* etc., you only need to begin a new paragraph if the sentence that begins *firstly* is then followed by another two or more sentences. If you only need one sentence for each item, then you don't need to begin a new paragraph.

There is no minimum length to a paragraph. A paragraph can occasionally be just one sentence. However, a series of paragraphs containing only one or two short sentences would be a little strange.

Where you begin a new paragraph will also depend on which section you are writing. In the review of the literature, you may want to begin a new paragraph when (i) you begin to talk about a different phase in the logical build up of research in your field, or (ii) you start talking about another author. In the Methods, it may help the reader to identify the various components or understand the various steps, if these components or steps are in separate (probably quite short) paragraphs.

Performing lexical and grammatical exercises.

1. Breast cancer among premenopausal women also showed variation **according to / in accordance with / in agreement with** the history of the patients.
2. Dataset characteristics of each problem were measured **according to / depending on / following** Smith et al. [4] and [15].
3. **Depending on / Following / In agreement** with the status of the couple, various solutions are possible:
4. **According to / In accordance with / Following** this method, the Bernoulli and hydrostatic equations can be combined to give…
5. For all three proteins we obtained polypeptide chains with molecular weights **in agreement with / in compliance with / in accordance with** the values from the literature.
6. The focus can be on the insertion or deletion of each video shot, **depending on / in compliance with / in agreement with** user preference.
7. The objective of the present study was to estimate 10 year probabilities of osteoporotic fractures in men and women **according to / in compliance with / in agreement with** age and bone mineral density.
8. Our results were **according to / in compliance with / in agreement with** the literature.
9. They were used **according to / depending on / in agreement with** the manufacturer's operating manual.
10. To date, all the tests have been carried out **according to / in compliance with / in agreement with** EU regulations.

1. In the first five chapters, the book covers **as / how / like** to search literature databases, **as / how / like** to interpret critically and appraise reports…
2. The second set has properties **as / how / like** those of the first set, with the difference that…
3. **As / How / Like** can be seen in the table, the values are considerably lower this time.
4. **As / How / Like** a prototype it worked well, but not in its final version.
5. It behaves **as / how / like** it should do.
6. It behaves **as / how / like** the other one.
7. It can be used **as / how / like** an alternative.

8. We used a piece of wood **as / how / like** a lever.

1. We studied **both / either / neither** English and Spanish. So we don't have any problems translating to and from these two languages.
2. You can study **both / either / neither** English or Spanish, i.e. you only have the option to study one of them.
3. You cannot study **both / either / neither** Russian and Korean, just one of the two.
4. You cannot study **both / either / neither** Russian or Korean, you can only study Chinese.
5. This is true **both for / for both** the students (there are 30 in the class) and the professors.
6. This is true **both for / for both** the students (i.e. Adrian and Anna) and the professors.
7. We had fun **both in / in both** the parks (i.e. Green Park and Hyde Park) we visited and also the museums.
8. We had fun **both in / in both** the parks (we lost count of how many we went to) and the museums.
9. This software will work with **both / either / neither** MAC or Windows.
10. This software will **both / either / neither** work with MAC nor with Windows, only on UNIX systems.

Insert the words below into the spaces.
addresses, aim, aimed at, aims to, continuation, feasibility study, framework, propose, scope, targeted, this end, undertook
1. Our _____ is to provide a short, practical analysis of how this language is used.
2. This article _____ de fi ne the difference between a hazard and a danger.
3. This article is the result of a _____ investigating…
4. This work _____ the problems inherent in…
5. This work is a direct _____ of the work begun by Zappata [2014].
6. To _____ we have tried to…
7. We have _____ funding as being our main priority.
8. We _____ a new code for calculating the number of hours required.
9. We _____ this study to…
10. Within the _____ of these criteria, we propose to…
11. Defining P and Q falls outside the _____ of this article.
12. It is _____ students of engineering.

Match the phrases (1–25) with functions (A–D)
(A) Establishing why your topic (X) is important.
(B) Outlining the past-present history of the study of X (no direct references to the literature).
(C) Outlining the possible future of X.
(D) Indicating the gap in knowledge and possible limitations.
1. A neglected area in the fi eld of analytical chemistry is…
2. Although this approach is interesting, it fails to take into account three critical factors.
3. By 2025, computers will have become redundant.
4. Concerns have arisen which call into question the validity of…
5. Despite this interest, no one to the best of our knowledge has studied…
6. Few researchers have addressed the issue of…
7. GISs have many applications in the field of…
8. However, there has been little discussion on…
9. In the next few years Nigeria is likely to have become…
10. It is not yet known whether these problems will be solved in the near future.
11. It is well known that psychologists tend to…
12. Moreover, other approaches have failed to provide…
13. Most studies have only focused on China to the detriment of India.

14. Psychometric tests are a critical part of the job interview process.
15. Recent developments regarding the future of the Internet have led to…
16. Roses are among the most well-known flowers on the planet.
17. Since 2012 there has been a rapid in the use of nanotechnologies.
18. The first studies in child psychology saw children as…
19. The Indonesian economy has received much attention in the past decade due to…
20. The last two years have witnessed a huge growth in the number of studies on this topic.
21. The main characteristics of bilinguals are: 161
22. The next decade is likely to see a considerable rise in unemployment.
23. There is little or no general agreement on…
24. There is still considerable controversy surrounding…
25. Traditionally, the focus on bilingualism has always been…

In each sentence delete the one word / phrase that is not appropriate / grammatical.
1. This paper **outlines / proposes / describes / discovers / presents** a new approach to…
2. This paper **validates / examines / seeks to address / focuses on / discusses / investigates** how to solve…
3. This paper is **an overview of / a review of / a report on / a preliminary attempt** how bilinguals separate the two languages while talking.
4. The aim of our work is to **further / extend / widen / broaden / amplify** current knowledge of…
5. This paper **takes a new look at / re-examines / revisits / informs / sheds** new light on how politicians use their power,
6. In the literature, 'psychotic' usually **refers / often refers / is usually referred** to a patient who…
7. Vitous [2015] has **provided / put forward / put down / proposed** a new definition of X, in which…
8. In the literature **there lacks of a general definition of X / a general definition of X is lacking / there is no clear definition of X.**
9. In their **seminal / groundbreaking / cutting edge / state-of-the-art** paper of 2001, Peters and Jones… 10. Experiments on X were **conducted / carried on / carried out / performed on** X in 2009 by a group of researchers from…
11. More recent evidence [Obama, 2013] **shows / suggests / investigates / highlights / reveals / proposes** that.
12. He **claims / argues / criticizes / maintains / suggests / points out / underlines** that…
13. Kamos's [23] assumptions seem to be **sensitive / realistic / well-founded / well-grounded / plausible / reasonable / acceptable.**
14. Many experts contend, **however / instead / on the one hand**, that this evidence is not conclusive.
15. This has led authors **as / such as / for example / for instance** Mithran [32], Yasmin [34] and Hai [35] to investigate…

Module 2 Basics of Academic writing

Features of academic writing (academic style, critical writing, referencing, developing paragraphs, writing plans). Research planning. Choosing an appropriate journal to publish a research paper. Analysis in research papers. Choosing a paper as a model. The correct order to write the various sections.

Read as many papers as you can from your chosen journal. This should help you to gain a clearer picture of what the editors of the journal are looking for to enable them to keep their readership levels high. Below are some of the typical things that editors hope to find in manuscripts.

TYPE OF PAPER	Original research, or a systematic review, or a position paper etc. (for more on the various types of paper consult Google Scholar or Wikipedia)
SUBJECT	Hot topic (contemporary issues), original and innovative; or controversial; or classic
AIM	Clarity of purpose, i.e. the research objectives are clear
RESEARCH	Well conducted, methodology clear, ethical, reproducible, no bias, limitations admitted
RESULTS	In line with research objective; entirely new or confirmation of other results already published in the same journal; not too broad as to be meaningless; can be generalized outside your very specific field
LENGTH OF PAPER	Short or long
STYLE	Personal (*we*, *I*), or impersonal (exclusively passive form), or mix (personal and impersonal)

Sometimes journals have themed or special issues on specific topics. These special issues are announced many months in advance of publication. Keep a look out for an issue that covers your specific area - it may be the perfect opportunity for you.

Choose one paper that is close to your topic, that is written by a native English speaker, and that you enjoyed reading. Use this paper as a model into which you can 'paste' your own research.

Notice how your model paper is structured:

- how does the author begin?
- what points does s/he make in each section?
- how does s/he link paragraphs together?
- how does s/he connect the Results with the Discussion?
- how does s/he present the Conclusions?

There is no standard order in which you should write the various sections of your paper. You should choose the order that suits you best. This may involve writing several sections simultaneously.

Many authors start with the Methods, which is often the easiest section to write because this is the part that will usually be clearest in your mind. Beginning with the Methods will also give you the confidence and impetus you need to move on to the other sections of the paper.

In reality, it is best to start with the Abstract as this will help you to focus / orient your ideas on what are the key aspects of your research. In any case, if you are going to present your work at a conference, the organizers will ask you to submit an abstract before you write the related paper - you can still change the Abstract when you have finished writing the actual paper.

You might find it useful to look at the scientific study protocol that you wrote when you outlined the aims of your research at the beginning of your PhD or before you began your current project. Here you should have written out your goals very clearly, and this will help you to write your Abstract.

The hardest part for most authors is the Discussion where you have to interpret your results and compare them with other authors' results. While you are writing the Discussion, you may find it useful to draft the Introduction, as some of the authors you mention will appear both in the Introduction and the Discussion.

A typical order for writing the various sections is thus:

Abstract (very rough draft)
Methods
Results
Discussion
Introduction
Conclusions
Abstract (final version)

It is a good idea to write the Results and Discussion before the Introduction. This is because you will only truly understand the significance of what you have done after you have written these two sections. Laying the background foundations on which you can highlight the significance of your research is a major part of the Introduction.

Academic vocabulary. Vocabulary and academic style. Noun phrases. Nouns referring to ideas and phenomena, ways of thinking, processes and activities. Verbs for structuring academic assignments. Adjectives and typical combinations with nouns in academic texts.

1. The packaging of products **affects / effects / influences** whether we will buy the product or not.
2. Many teenage girls are **affected / conditioned / interested** by photos of skinny models, to the extent that they may become obsessed with losing weight.
3. Whether teachers have previous experience or training will inevitably **affect / condition / influence** the way they teach.
4. The choice of what to study at university is strongly **conditioned / influenced / interested** by the possibilities of a career.
5. We found that the general public was only marginally **affected / conditioned / interested** by the government's campaign to encourage people to eat more healthily.
6. It was found that religion can, under certain circumstances, totally **affect / condition / influence** the way believers behave.
7. The way we define X does not **affect / effect / influence** the way X is perceived.
8. Does the job we do **affect / condition / effect / influence** the chances of us taking drugs?
9. It is believed by some that correction may have a detrimental **effect / influence** a change on a student's confidence and may even **affect / condition / influence** their behavior during lessons.
10. The method chosen was found to **affect / condition / interest** the performance to the extent that choosing the wrong method inevitably gave catastrophic results. This finding **affected / conditioned / interested** the researchers, who then went on to repeat the experiment in Japan and Korea, with very different results.

1. This may be desirable in the long term to **allow / let / mean** a greater degree of control over the…
2. The paper shows the results of an approach that **allows us to extrapolate the data / permits to extrapolate the data / means the data can be extrapolated** more easily than with other methods.
3. Increased connectivity **enables / lets / permits** new ways of conducting business, **allowing / enabling / permitting** companies to trade…
4. We also **allow / permit / let** users the flexibility of editing incoming and sent messages.
5. These governments do not **enable / let / permit** immigrants to have citizenship.
6. The formulation of this new theory **allows to / means we can / permits to** obtain a more general expression of the overall transfer function.
7. This kind of behavior is not **allowed / enabled / permitted**.
8. Her parents **allowed / let / permitted** her do anything she wanted.

1. The data were **analyzed / elaborated / processed** using StAT 2.0.
2. Unfortunately, the authors fail to **analyze / elaborate / process** on the method they used.
3. Medical or cause-of-death information was **analyzed / elaborated / processed** separately.
4. The food had evidently been **analyzed / elaborated / processed** before or during the outbreak of the disease.
5. They have **analyzed / elaborated / processed** an experimental method of investigating emotion.

1. The possibility that this might happen was **foreseen / predicted** by Sterling [2] and also by…
2. On the basis of these intelligence tests, we must consider what new insights we would **expect / predict** to find.
3. We **expect / forecast** that, as a result of this new approach, PTs will become increasingly popular.
4. We did not **anticipate / predict** finding a solution, so we were surprised when…
5. In their first study, they **forecast / foresaw** an increase in the elderly population of 12.6 million between 2013 and 2023.
6. For the moment, urban planners do not **anticipate / forecast** large population increases in the region.
7. No one **expected / foresaw** this happening – it took everyone by surprise.
8. The research findings reveal that the overwhelming majority of firms participating in the study did not **foresee / forecast** the economic crisis.
9. The congress has been **anticipated / brought forward** from July to June.
10. If in the early 1980s anyone had **anticipated / predicted** that within a few years the Internet would have had more impact than the invention of the wheel, they would have been ridiculed.
11. **Forecasting / Foreseeing** the weather in the long-term is a highly frustrating and ultimately unreliable activity.

1. We do not **argue / claim / pretend** to provide a complete solution to this problem.
2. There is no point **arguing / claiming / pretending** that these problems are likely to disappear in the near future, something needs to be done now.
3. A child has no difficulty in **arguing / claiming / pretending** that a banana is a telephone.
4. They **argue / claim / pretend** that we do not need a government, but that we should be self-governing. However, this line of thinking does not…
5. Kasamir refuses to romanticize the freedom fighter as a heroic rebel, **arguing / claiming / pretending** instead that freedom fighters themselves are fulfilling their natural duty to liberate their country from an oppressive regime.
6. The children **argued / claimed / pretended** that they spend many hours a day **arguing / claiming / pretending** with their parents.

1. We then estimated the unemployment rate that would have **arisen / give rise to / raised / risen** by 15% or more if those measures had not been introduced.
2. However, we did not include these samples. In fact, including them would have **arisen / give rise to / raised / risen** an overrepresentation of…
3. In these cases the government should have **arisen / give rise to / raised / risen** taxes rather than…
4. Inflation could have **arisen / give rise to / raised / risen** to 12% if the Central Bank had not intervened.
5. Inflation has **arisen / give rise to / raised / risen**. Social problems have **arisen / give rise to / raised / risen** due to the consequent high levels of unemployment, which has **arisen / give rise to / raised / risen** violence across the country.
6. The proofreaders of the document have **arisen / give rise to / raised / risen** several issues with regard to the use of English. These issues seem to have **arisen / give rise to / raised / risen** from the fact that there are a considerable number of grammatical errors. In fact the number of such complaints about our documents has **arisen / give rise to / raised / risen** dramatically.

1. These fertilizers are designed to **ascertain / check / control / verify** the growth of grass weeds.
2. This allows us to provide a cross check of previous results on how well parents are able to **ascertain / check / control / verify** their children.
3. Formal verification is another way to **ascertain / check / control / verify** the validity of protocols.
4. The case notes of all patients recorded as having this pathology were reviewed to **ascertain / check / control / verify** the diagnosis and to **ascertain / check / control / verify** the nature of death.
5. We thus needed to **ascertain / check / control / verify** whether it was indeed Ca++ that was responsible for this effect.
6. Someone's exact movements can be **ascertained / checked / controlled / verified** if they are carrying a GPS device.
7. This is becoming an increasingly vital problem in situations such as **ascertaining / checking / controlling / verifying** the identity of criminals.
8. All patients were screened by telephone interview to **ascertain / check / control / verify** the possible diagnosis of high blood pressure.
9. Readers are invited to examine the references given with this article to **ascertain / check / control / verify** the fact that our results our truly representative.
10. Thousands of extra police officers were employed to **ascertain / check / control / verify** the crowds.

1. Although language expertise has been **assumed / hypothesized / supposed** to be highest in bilingual children, it has never actually been proved.
2. The problem is that although the students apparently study what they are **assumed / hypothesized / supposed** to study, the examination itself may not actually test what it is **assumed / hypothesized / supposed** to test.
3. In this chapter it is **assumed / hypothesized / supposed** that the reader is familiar with…
4. In the literature it has been **assumed / hypothesized / supposed** that abnormalities in the connections of white matter pathways may be a fundamental cause of…
5. On this basis, it is **assumed / hypothesized / supposed** that students will only want to undertake a Ph.D. if they…
6. Students were informed that they were not **assumed / hypothesized / supposed** to use a dictionary during the exam, however many students nevertheless brought a dictionary with them as they had **assumed / hypothesized / supposed** that the invigilators would not be strict.
7. In conclusion, in this paper it has been **assumed / hypothesized / supposed** that the more money we have, the more we will be unhappy.
8. It has been **assumed / hypothesized / supposed** by Smith et al. that the rate at which we learn knowledge is proportional to the rate at which we…
9. For this purpose let us **assume / hypothesize / suppose** that we have two systems, I and II, which we permit to interact from the time t 0 to t…
10. We had **assumed / hypothesized / supposed** that patients would automatically wish to be treated. In reality…

1. This therapy is complex and involves many steps. At each step, comprehensive quality assurance procedures are required to **assure / ensure / insure** the safe and accurate delivery of a prescribed dose.
2. The cost of **assuring / ensuring / insuring** buildings in those parts of the country subject to earthquake can be up to 75% higher. In addition, to **assure / ensure / insure**, for example, a car, will also cost considerably more.
3. To **assure / ensure / guarantee** the quality of teaching based on this concept of teaching, schools need to **assure / ensure / insure** that the system is open enough to allow for variations in student types.
4. The government should **assure / ensure / guarantee** the protection of all its citizens against such threats.
5. This continuous back up policy **assures / ensures / guarantees** that data will not be lost.
6. We **assure / ensure / guarantee** you that you will receive a reply by the end of this week.
7. I'm just writing to **assure / ensure / insure** you that we are working on the problem.
8. Please **assure / ensure / guarantee** that you are using the latest version.

WRITTEN	SAID	WRITTEN	SAID
CARDINALS AND ORDINALS			
101	a / one hundred and one	58,679	fifty eight thousand six hundred and seventy nine
213	two hundred and thirteen	2,130,362	two million, one hundred and thirty thousand, three hundred and sixty two
1,123	one thousand, one hundred and twenty three		
13th	thirteenth	31st	thirty first
CALENDAR DATES			
10.03.20	the tenth of March two thousand and twenty (GB)	1996	nineteen ninety six
			nineteen hundred and ninety six
GB: day / month / year	or March (the) tenth two thousand and twenty (GB)	1701	seventeen oh one
			seventeen hundred and one.
US: month / day / year	October third two thousand twenty (US)	2010s	twenty tens
FRACTIONS, DECIMALS, PERCENTAGES			
¼	a quarter / one quarter	0.25	(zero) point two five
½	a half / one half	0.056	(zero) point zero five six
¾	three quarters	37.9	thirty seven point nine
10%	ten per cent	100%	one hundred percent

WRITTEN	SAID	WRITTEN	SAID
SQUARES, CUBES ETC.			
$4\ m^2$	four meters squared, four square meters	2^5	two to the power of five
$5\ m^3$	five cubic meters, five meters cubed		
MONEY			
678	six hundred and seventy eight euros	$450,617	four hundred fifty thousand six hundred seventeen dollars
¥1.50	one yen fifty (cents)	$1.90	a dollar ninety
MEASUREMENTS			
1 m 70	one meter seventy	3.5 kg	three point five kilos
3 m × 6 m	three meters by six		
100^0	one hundred degrees	-10^0	minus ten degrees
			ten degrees below zero
PHONE NUMBERS			
0044 161 980 4166	zero zero four four one six one nine eight zero four one double six	ext. 219	extension two one nine
	or oh oh four four etc.		

Module 3 Specific grammar of Academic writing

Learning the active vocabulary of the module.

Theory of translation, punctuation for academic writing.

Performing lexical and grammatical exercises.

Translation of scientific articles from a foreign language into Russian.

Structure of a paper. Grammar used in Introduction, Literature review, Methods, Conclusion, Discussion Sections. Generating titles. Problems of string of nouns in titles. Making titles concise. Genres of academic writing. Learning outcomes. Basic structures. Types of academic writing.

English has a strict order in which words can appear in a sentence. S1 shows an example of this order.

S1. The researchers sent their manuscript to the journal.

This order is rarely altered. It is:

1. subject (*the researchers*)
2. verb (*sent*)
3. direct object (*their manuscript*)
4. indirect object (*the journal*)

The key is to keep the subject, verb, direct object and indirect object as close to each other as possible. This is illustrated in S2, which maintains the exact order of S1.

S2. Last week *the researchers sent their manuscript to the journal* for the second time.

S3. **The researchers* last week *sent* for the second time *to the journal their manuscript.*

S3 is incorrect English. The position of *last week* and *for the second time* is wrong, and the indirect object comes before the direct object.

Word combinations in academic texts. At academic institutions. Vocabulary: applications forms, academic courses, online learning. Facts, evidence, data, numbers, graphs and diagrams, time, cause and effect.

Module 4 Academic style

Learning the active vocabulary of the module.

Specific analysis. Structured research articles abstracts. Tips for writing research article abstracts. Conference abstracts. Writing introductions, overall shape of an introduction. Tips for writing different kinds of introductions (introduction to course paper, book reviews, journal article, book chapters, research reports, proposals).

Performing lexical and grammatical exercises.

1. It was found that X = 2, whereas / on the contrary Kamatchi [2011] found that X = 1.
2. Despite the fact / Although that Li and Mithran [2014] found that X = 2, we found that X = 3.
3. In contrast to / On the contrary earlier findings [Castenas, 2009], we found that x does not equal y.
4. This study has not confirmed previous research on X. Nevertheless / despite, it serves to…
5. Notwithstanding the fact that / Despite these results differ from earlier studies (Cossu, 2001; Triana, 2002), they are consistent with those of…
6. Georgiev is correct to claim that x = y. Nevertheless / however, his calculation only referred to a limited case.
7. The current study does not support previous research in this area. In fact, contrary to / unlike what was previously thought, we found that…
8. Despite / Nevertheless the lack of agreement, we believe our findings compare well with…
9. Although / Despite there was some inconsistency…
10. This does not justify non-invention. However / on the contrary it is another reason for increased intervention.

(1) Since / When writing first began, there was little or no punctuation. Punctuation was introduced many hundreds of years later to help the reader. Punctuation tells us (2) both / when we can pause and helps us to see connections between the elements in the sentence. Readability (3) however / thus has a visual element to it as well. This visual element is (4) also / besides affected by how we read. Today, much reading is done directly from a screen, (5) other than / rather than from a hard copy. (6) Because / Why we generally want information fast, particularly (7) since / when searching on the Internet, we tend to scan. Scanning means not reading each

individual word (8) but / yet jumping forwards three or more words (or sentences) at a time. The distance that we jump (in terms of the number of words or sentences) depends on the value that those words are adding in our search for information. (9) If / Yet they add no value we tend to jump further. (10) If / When we continue to get no value, instead of scanning left to right along a line of text, we scroll from top to bottom. We (11) thus / still read vertically (12) instead of / rather than horizontally until we find what we want.

This has huge implications for you as a writer. (13) If / When you want your reader to read your paper in depth, (14) then / thus you cannot afford to fill your sentences with redundancy. (15) If / When you write a series of very long sentences, you will encourage your reader to scan and scroll. This means that they may never see / read the key information contained within all the redundancy.

Writing a readable text entails being able to understand the nature of communication: thinking about your audience and the impact of how you organize your thoughts and words. (16) If / Unless you write a readable text, you will find personal satisfaction not in how erudite and elegant your phrases sound, (17) but / however in the ease with which you allow your readers to absorb your ideas. Remember that no one will be under any obligation to read your paper. (18) If / When readers don't find it useful, (19) either / or interesting, (20) both / or at least pleasurable, (21) and / however they have the feeling that it was not written with them in mind, they will simply stop reading. Your findings will (22) only / then be lost in oblivion.

1. The aim of this study was to assess the effects of sending children away to school at the age of eight (or earlier) and its impact on their adult life (particularly after the age of 50) and thus to reach some definitive conclusions as to whether boarding schools (i.e. those schools where children study and sleep) actually fulfill the important educational and social roles that they claim to have.
2. People who have attended boarding schools often have no realization of the effect that leaving their parents at a very young age has had on their emotional development because the signs of this effect generally do not become sufficiently apparent until middle age and are often due to a kind of subconscious repression which is why such subjects do not make the connection between their current levels of over-emotiveness and their childhood lack of parental affection.
3. Questionnaires were sent to 5000 ex-boarding school adults with an age ranging between 40 and 60 all of whom had previously given permission to access their medical records and all of whom were or had been married, with the purpose of setting up a database of subjects' responses regarding their school time experiences and their experiences now as adults.
4. A substantial increase in sensitivity to emotional situations characterizes the first stages of adult life leading to a possible uncontrolled release of anger or apparently unexplained feelings of anxiousness that appear to come from nowhere and may last for several days thus making life quite difficult not only for the subjects themselves but also for those living around them.
5. Treatments for these subjects are often very expensive and technically difficult, and their effectiveness very much depends on the willingness of the subject to undergo therapy and on the degree of stress, emotional disturbance and marital discord that they had experienced.

The following extract is from an Introduction. It is one long paragraph and contains three very long sentences, averaging over 80 words each. Divide the paragraph into three shorter paragraphs, and break up each sentence into shorter more manageable sentences.

The aim of this paper is to confirm that how we speak and write generally reflects the way we think and that this is true not only at a personal but also at a national level, and to this end two European languages were analyzed, English and French, to verify whether the structure of the language is reflected in the lifestyle of the respective nations. English is now the world's international language and is studied by more than a billion people in various parts of the world thus giving rise to an industry of English language textbooks and teachers, which explains why

in so many schools and universities in countries where English is not the mother tongue, it is taught as the first foreign language in preference to, for example, Spanish or Chinese, which are two languages that have more native speakers than English. As a preliminary study, in an attempt to establish a relationship between document length and level of bureaucracy, we analyzed the length of 50 European Union documents, written in seven of the official languages of the EU, to con fi rm whether documents, such as reports regarding legislative and administrative issues, vary substantially in length from one language to another, and whether this could be related, in some way, to the length of time typically needed to carry out daily administrative tasks in those countries (e.g. withdrawing money from a bank account, setting up bill payments with utility providers, understanding the clauses of an insurance contract). The results showed that...

Our system is based on four components: it has many data files (the weather, people, places, etc.); it has procedures which it tries to use to combine these files by working out how to respond to certain types or patterns of questions (this entails the user knowing what types of questions it can answer); it has a form to understand the questions posed in a natural language (so the user may need to know English) which it then translates into one of the types of questions it knows how to answer; finally, it has a very powerful display module, which it uses to show the answers, using, graphs, maps, histograms etc.
Using automatic translation software (e.g. Google Translate, Babelfish, and Systran) can considerably ease the work of researchers when they need to translate documents thus saving them money (for example the fee they might have otherwise had to pay to a professional translator) and increasing the amount of time they have to spend in the laboratory rather than at the PC.

Development of academic writing skills in accordance with research areas of master students.Writing CVs.Formal letters and e-mails. Structuring the content of an e-mail. Planning an e-mail. Formal greetings. Regrets and replies. Academic correspondence styles. Statements of purpose, personal statements.

Watch the video
https://www.ted.com/talks/guy_katz_how_to_write_an_email_that_will_always_be_answered and answer the questions.

1. What does Guy compare an e-mail with? Why?
2. How many letters does each person who works in an average office receive a day according to the statistics?
3. What was the worst e-mail he had ever received?
4. What kind of e-mails cannot be replaced by any other means of communication?
5. What are the five ingredients for a great e-mail?
6. How can we add emotionally to a formal e-mail?
7. What does it mean if a teenager finishes a text message with a full stop?
8. What increases the chances to get an answer?
9. What are the two things that e-mails have in common?
10. What should be written under the PS line? Why?

Module 5 Professional communication

Aspects of spoken English and professional communication. Interviewing and advising. Negotiation. Chairing a formal meeting. Telephoning. Finding the voice in the academic community (communicating with advisors and committee members, co-authors, requests, reminders, writing apologies, grant applications, letters, fellowship application).

Preparing presentations in English. Writing and editing the text of the slides. Outline and transitions. Writing a speech. Methodology and results discussion. Conclusions. Questions and answers. Useful phrases. Pronunciation and intonation.

Translation theory: what to avoid. Avoiding Ambiguity and Vagueness. Which/who vs. that. Which/that and who. Uncountable nouns. Pronouns. Referring backwards: the former, the latter. False friends. Terms, definitions, references. definitions. Introduce mathematical clichés.

Module 6 Preparing a research article

Learning the active vocabulary of the module.

Academic vocabulary: talking about ideas. Reporting verbs and nouns. Analysis of results. Points of view. Degrees of certainty. Presenting an argument. Describing research methods. Classifying. Making connections. Comparing and contrasting. Evaluation and emphasis. Summary and conclusion.

Useful phrases in Introduction, Literature review, Methods, Conclusion, Discussion.

Writing a research paper. Literature review

1. What are the seminal works on my topic? Do I need to mention these?
2. What progress has been made since these seminal works?
3. What are the most relevant recent works? What is the best order to mention these works?
4. What are the achievements and limitations of these recent works?
5. What gap do these limitations reveal?
6. How does my work intend to fill this gap?

Persistence has most often been studied in terms of cultural differences. Blinco (1992) found that Japanese elementary school children showed greater task persistence than their American counterparts. School type and gender were not factors in moderating task persistence. This left culture as the remaining variable.

Heine et al. (2001) furthered this idea by testing older American and Japanese subjects on responses after success or failure on task persistence. Japanese subjects were once again found to persist longer (in post-failure conditions), and this was speculated to be because they were more likely to view themselves as the cause of the problem. If they were the cause of the problem, they could also solve the problem themselves; although, this could only be accomplished through work and persistence. Americans were more likely to believe that outside factors were the cause of failure.

These cultural studies hinted that task persistence may be predictable based on attribution style. A later experiment showed that attribution style and perfectionism level can be correlated with final grades in college-level classes (Blankstein & Winkworth, 2004).

1. introduction to topic
2. support from the literature
3. mini summary
4. introduction to next topic. And so on.

style 1 Blinco [1992] found that Japanese elementary school children showed …
style 2 In [5] Blinco found that Japanese elementary school children showed …

style 3 A study of the level of persistence in school children is presented by Blinco [1992].
style 4 A greater level of persistence has been noticed in Japan [5].

	ORIGINAL VERSION (OV)	REVISED VERSION (RV)
1	Long sentences *are known to be* characteristic of poor readability [Ref].	Long sentences *are* a characteristic of poor readability [Ref].
2	*In the literature* the use of long sentences *has also been reported* in languages other than English [Ref].	Long sentences *are* not exclusive to English [Ref].
3	The use of long sentences *has been ascertained* in various regions of Europe during the Roman period [Ref].	Long sentences *were used* during the Roman period in various regions of Europe [Ref].
4	The concept of author-centeredness *has been suggested as playing* a role in the construction of long sentences [Ref].	Author-centeredness *may play* a role in the construction of long sentences [Ref].
5	*Several authors have proposed* that in scientific writing the occurrence of a high abundance of long sentences *is* correlated to … [Ref].	In scientific writing the occurrence of a high abundance of long sentences *may* be correlated to … [Ref].

As far as we know, there are no studies on …
To [the best of] our knowledge, the literature has not discussed …
We believe that this is the first time that principal agent theory has been applied to …

Generally speaking patients' perceptions are seldom considered.
Results often appear to conflict with each other …
So far X has never been applied to Y.
Moreover, no attention has been paid to …
These studies have only dealt with the situation in X, whereas our study focuses on the situation in Y.

	EXAMPLES
1	Wallwork [2012] stated x = y.
	Huang [2013] agrees with this statement, but Xanadu [2014] does not.
2	In [6] Wallwork stated that x = y. Then in [9] he added that x + 1 = y + 1.
3	A proposal for a conference on this topic was put forward by Tang [2014].
3	This is not the first time that such a proposal has been put forward [Himmler, 2012; Goldberg, 2013].
	This is not the first time such a proposal has been put forward [6, 27, 33].
4	This proposal was first put forward in [6].
	In [6] a proposal for a conference on this topic was put forward.

	YES	AMBIGUOUS OR WRONG
1	In [6] Wallwork put forward a proposal for the scientific community to allow personal forms. = another author In [6] we put forward a proposal for the scientific community to allow personal forms. = the author of the current paper	In [6] the author put forward a proposal for the scientific community to allow personal forms.
2	In a previous paper [Gomez, 2], we found that x = y.	In [Gomez, 2], it was found that x = y.
3	In [6] Wallwork stated that all journals should allow the use of personal forms. Two years later he added that the ISO should set some standards regarding the style of bibliographies [9].	In [6] Wallwork stated that all journals should allow the use of personal forms. Two years later he added that the ISO should set some standards for scientific writing [Wallwork, 2014].

SUGGESTED USAGE	EXAMPLES
one author: name + comma + year	Wallwork, 2015
two authors: name1 'and' name2 + year	Wallwork and Southern, 2016
three authors: name1 + comma + name2 'and' name3 + year (Note: writing the names of three authors is quite unusual)	Wallwork, Brogdon and Southern, 2016
three or more authors: name1 + et al.	Wallwork et al., 2016
two or more references: ref1 + semicolon + ref2 + semicolon etc.	Wallwork et al., 2016; Sanchez, 2017; Poplova, Huang and Sun, 2018
several works by same author: name + comma + year1 + comma + year2 etc.	Wallwork, 2012, 2014, 2016

SUGGESTED USAGE	EXAMPLES
when the author is the subject of the verb: name + year in parentheses. Alternatively: name + reference number in parentheses	Wallwork [2012] suggests that …
	Wallwork [6] suggests that …
when the author is not the subject of the verb: both name and year in parentheses	It has been suggested that one plus two is equal to four (Moron, 2011).

	YES	ALSO POSSIBLE
1	Wallwork et al [2016] put forward a proposal for the scientific community to allow personal forms.	Wallwork and co-workers [2016] put forward …
2	Wallwork et al [2016] suggested that …	Wallwork *et al* [2016] suggested that …
		Wallwork *et al.* [2016] suggested that …

	YES	NOT RECOMMENDED (1–4), NO (5)
1	See Figure 1 and Table 2.	See figure 1 and table 2.
2	See Fig. 1a and Figs. 2a and 2b.	See Fig. 2a and "b.
3	Figure 2 below **shows** the initial settings.	The following figure (Figure 2) **gives a schematic overview of** the initial settings.
3	Figure 3 **shows** the architecture.	The **snapshot depicted in** Figure 3 **shows a view of** the architecture.
3	For **details, see** [Kyun, 2013].	For **further details on this topic, the reader is kindly invited to refer to** [Kyun, 2013].
4	Figure 2 below **shows** the initial settings.	The initial settings **are shown** in Figure 2 below.
		In Figure 2 the initial settings **are shown**.
5	**As** can be seen in the figure below …	**As it** can be seen in the figure below …

Figure 1. The main characteristics of the shock absorbers.

	YES	NOT RECOMMENDED (1), WRONG (2–5)
1	As mentioned in **Section 2**, this procedure is ...	As mentioned **above**, this procedure is ...
1	This procedure is extremely complex and is described in **Section 4**.	This procedure is extremely complex and is described **later**.
2	The function **mentioned above** is ...	The function **above mentioned** is ...
	The **above-mentioned** function is ...	
3	This feature is known as an 'automatic rendering and masking agent' **hereafter** ARM agent.	This feature is known as an 'automatic rendering and masking agent' **in the following** ARM agent.
4	The **following versions** can be used:	The versions that can be used are **the following**:
	The versions that can be used are **as follows**:	

Supporting the publication process. Manuscript submission, responding to reviewers and editors, writing acknowledgments.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»

ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Психология управления личностными ресурсами»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024 г.

Оглавление

Занятие 1. Требования к современному профессионалу

Цель: формирование у магистрантов дифференцированных представлений об ИТ-специалисте.

Задачи:

ознакомление с современными и перспективными запросами рынка труда в ИТ-сфере;

развитие представлений об успешном ИТ-специалисте;

анализ взаимосвязи профессиональных и личностных качеств как предпосылок успешной профессиональной деятельности.

Теоретическое введение

Материалы лекции 1.

Опрос работодателей «Универсальные компетенции (soft skills) профессионала». Опрос проведен Институтом компьютерных технологий и информационной безопасности ЮФУ в декабре 2019 г. Цель опроса: выявление наиболее актуальных универсальных компетенций (soft skills) для личностно-профессионального развития студентов. Респонденты: 14 руководителей (заместителей руководителей) предприятий / организаций, основная сфера деятельности – разработка ПО, гео-информационные системы, разработка WEB-приложений, компьютерное зрение и машинное обучение.

Основные результаты опроса приведены в таблице 1.

Все предложенные компетенции оценены респондентами как важные (оценки *2 – «значимо», 3 – «весьма важно», 4 – «сверх важно»*), предложены также другие компетенции: «Уважение к чужому мнению и непредубеждённость к новому», «Common sense»). Всем респондентам приходилось взаимодействовать с выпускниками Института компьютерных технологий и информационной безопасности (ИКТИБ) ЮФУ.

К наиболее значимым для успешной профессиональной деятельности респондентами отнесены следующие soft skills (оценки *между 3 – «весьма важно» и 4 – «сверх важно»*, перечень дан в соответствии с рангами):

1. Способность обучаться и саморазвиваться.
2. Ответственность в работе.
3. Честность, этичность.
4. Самоорганизация и тайм-менеджмент.
5. Способность к эффективной коммуникации (умение выражать свои мысли, слушать, презентировать результаты).
6. Способность работать в команде (общительность, дружелюбие, эмоциональный интеллект).

Наиболее сформированы у выпускников ИКТИБ следующие soft skills (оценки ближе к *3 – «достаточно сформировано»,* перечень дан в соответствии с рангами):

1. Честность, этичность.
2. Способность обучаться и саморазвиваться.

3. Способность работать в команде (общительность, дружелюбие, эмоциональный интеллект).

4. Гибкость и адаптивность.

5. Стрессоустойчивость.

Наибольший разрыв между значимостью компетенций для успешной профессиональной деятельности и сформированностью у выпускников ИКТИБ обнаружен в оценках следующих компетенций:

1. Самоорганизация и тайм-менеджмент.

2. Ответственность в работе.

3. Способность обучаться и саморазвиваться.

4. Способность к эффективной коммуникации (умение выражать свои мысли, слушать, презентировать результаты).

Таблица 1

Результаты опроса

Средние оценки/ ранги soft skills	степень значимости	ранг	степень сформированности	ранг	выше сформировано*	ниже сформировано*	разница между значимостью и сформированностью
Гибкость и адаптивность	2,93	7	2,62	4	4	2	0,31
Критическое мышление	2,93	7	2,23	9	2	8	0,69
Способность к эффективной коммуникации (умение выражать свои мысли, слушать, презентировать результаты)	3,36	5	2,31	8	1	11	1,00
Способность работать в команде (общительность, дружелюбие, эмоциональный интеллект)	3,21	6	2,69	3	2	9	0,54
Самоорганизация и тайм-менеджмент	3,43	4	1,85	10	0	12	1,54
Способность обучаться и	3,86	1	2,77	2	0	11	1,08

саморазвиваться							
Ответственность в работе	3,64	2	2,38	7	1	12	1,23
Честность, этичность	3,57	3	2,85	1	1	10	0,77
Вежливость, деловой этикет	2,79	8	2,23	9	2	7	0,46
Стрессоустой-чивость	2,93	7	2,54	5	4	4	0,31
Целеустремлен-ность, инициативность	2,93	7	2,46	6	2	5	0,38
Креативность, творческие способности	2,57	9	2,54	5	3	6	-0,08
Способность влиять на других, управлять работой команды	2,50	10	1,85	10	2	5	0,54

* Количество респондентов, оценивших сформированность компетенции у выпускников ИКТИБ выше/ниже её важности.

Вопросы для собеседования

1. С помощью каких характеристик можно описать профессионала?
2. Какие из характеристик профессионала представляются вам наиболее важными в вашей профессиональной сфере?
3. Почему в современных условиях все больше требований предъявляется к личности профессионала?
4. Что такое профессиональная компетентность? В чем отличие компетентного профессионала от знающего?
5. Какие soft skills востребованы в ИТ-сфере?
6. Каковы инвариантные требования к личностным качествам профессионала?
7. В чем отличие субъекта профессиональной деятельности от ее исполнителя?

Практикум

Задание 1. Специфика запроса ИТ-компаний: вакансии

1.1. Познакомьтесь с классификацией ИТ-компаний (рис. 1). Приведите примеры известных вам компаний, относящихся к разным классам.

1.2. Познакомьтесь с карьерным навигатором (рис. 2). Охарактеризуйте специфику деятельности различных ИТ-специалистов.

1.3. Выберите интересующий вас класс компаний, проанализируйте рынок труда и выпишите наиболее массовые вакансии.

2.3. Общее обсуждение:

Каково состояние рынка труда в ИТ-сфере?

Какие перспективные запросы рынка труда в ИТ-сфере вы обнаружили?

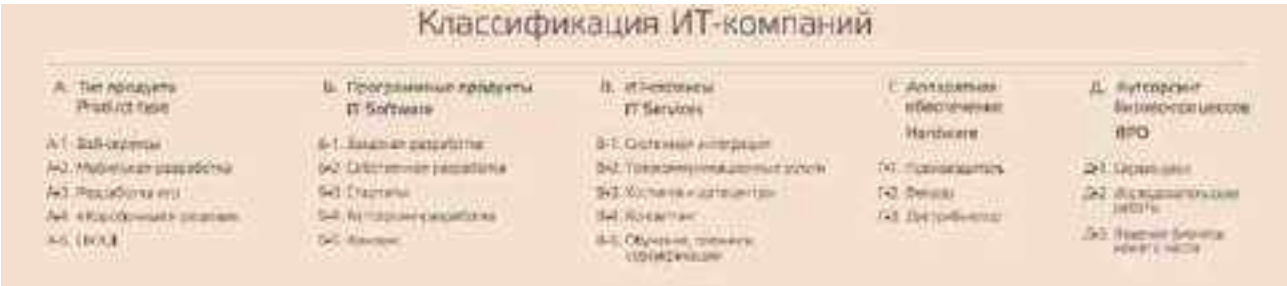


Рис. 1. Классификация ИТ-компаний

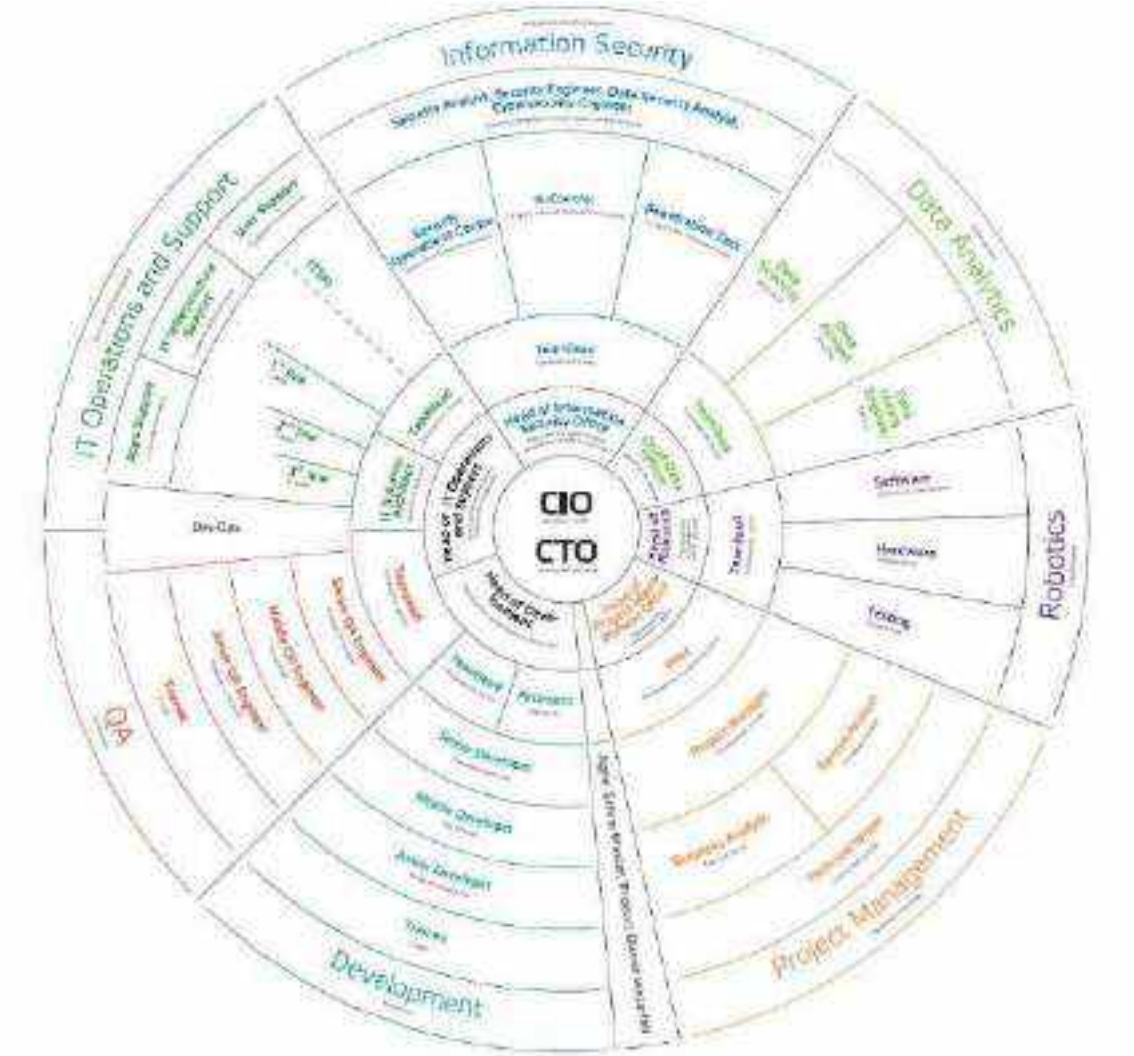


Рис. 2. Карьерный навигатор

Задание 2. Специфика запроса ИТ-компаний: требования к ИТ-специалистам

2.1. Группа делится на команды по 2–3 человека, выбирает конкретную должность и проводит анализ запроса работодателей к соискателям конкретных вакансий (профессиональных компетенций и «мягких навыков»).

2.2. Каждая группа публично представляет результаты анализа.

2.3. Общее обсуждение:

Что необходимо, чтобы быть успешным ИТ-специалистом, востребованным на рынке труда?

Какова взаимосвязь профессиональных и личностных качеств как предпосылок успешного трудоустройства и профессиональной деятельности?

Литература (информационные ресурсы):

https://hh.ru/
https://vkrabota.ru/
https://rabota.ru/
https://career.habr.com/

Занятие 2. Профессиональная этика ИТ-специалиста

Цель: развитие готовности магистрантов использовать этические правила при решении задач создания и внедрения систем искусственного интеллекта.

Задачи:

ознакомление с основными этическими проблемами, связанными с использованием систем искусственного интеллекта (ИИ);

анализ норм профессиональной этики при создании и внедрении систем искусственного интеллекта;

развитие умений применять этические нормы и стандарты в области искусственного интеллекта при создании систем ИИ.

Теоретическое введение

Внедрению ИИ и других цифровых технологий препятствует низкий уровень доверия граждан к алгоритмам и новым технологиям в принципе, а также отсутствие понятных этических рамок в применении ИИ. Основные этические проблемы, связанные с применением систем ИИ:

- ответственность за этичное/неэтичное поведение ИИ, за принятие ошибочных решений, ущерб из-за сбоев и т. п.;
- предубежденность алгоритмов (bias);
- обеспечение и регулирование прозрачности ИИ (объяснительная компонента);
- проблема приватности при применении технологий ИИ;
- надежность технологий ИИ[1].

Вопросы практической реализации этической компоненты ИИ становятся все более и более значимыми, среди них:

- реализация машинной этики;
- формализация этических понятий;
- верификация и валидация этической компоненты;
- стандартизация машинной этики;
- стандартизация этических аспектов ИИ.

Проекты этических стандартов IEEE:

P7001. Transparency of Autonomous Systems (Прозрачность автономных систем).

Стандарт представляет собой руководство для оценки прозрачности в процессе разработки ИИ и предлагает механизмы для повышения прозрачности (например, обязательное защищенное хранение данных датчиков и данных о внутреннем состоянии аналогично регистратору данных полета или «черному ящику»).

P7002. Data Privacy Process (Обеспечение конфиденциальности данных)

Стандарт ориентирован на защиту приватности граждан и в основном касается использования персональных данных граждан рекламными сетями при

[1] Этика и «цифра»: этические проблемы цифровых технологий: Аналитический доклад [Электронный ресурс]. URL: https://ethics.cdto.center

помощи ИАС. Для стандартизации пока выделены несколько групп: участники взаимоотношений «работник — работодатель», дети (несовершеннолетние), студенты.

P7003. Algorithmic Bias Considerations (Учет необъективности алгоритма)

Стандарт вполне может стать одним из первых, принятых в качестве базовых. Он обязывает разработчиков, в первую очередь систем машинного обучения, как самого популярного сейчас направления, ответственно подходить к данным, используемым для обучения, к их разметке, к тестированию и валидации систем.

P7004. Standard for Child and Student Data Governance (Управление данными детей и студентов).

Стандарт должен будет регулировать работу алгоритмов с данными детей и учащихся. Защита прав детей и детства – одна из важнейших задач социальной политики любого государства. С развитием ИИ технологий увеличивается риск того, что машины в результате общения друг с другом будут принимать решения на основе входных данных, непрозрачных для людей, по принципу «черного ящика».

Кодекс этики в сфере искусственного интеллекта (далее – Кодекс) устанавливает общие этические принципы и стандарты поведения, которыми следует руководствоваться участникам отношений в сфере искусственного интеллекта (далее – Акторы ИИ) в своей деятельности, а также механизмы реализации положений Кодекса[2].

Раздел 1. принципы этики и правила поведения

1. Главный приоритет развития технологий ИИ в защите интересов и прав людей и отдельного человека

Человеко-ориентированный и гуманистический подход

При развитии технологий ИИ человек, его права и свободы должны рассматриваться как наивысшая ценность. Разрабатываемые Акторами технологии ИИ должны способствовать или не препятствовать реализации всех потенциальных возможностей человека для достижения гармонии в социальной, экономической, духовной сфере и наивысшего расцвета личности, учитывать ключевые ценности, такие как сохранение и развитие когнитивных способностей человека и его творческого потенциала; сохранение нравственных, духовных и культурных ценностей; содействие культурному и языковому многообразию, самобытности; сохранение традиций и устоев наций, народов, этносов и социальных групп.

Человеко-ориентированный и гуманистический подход является основным этическим принципом и центральным критерием оценки этичного поведения Акторов в сфере ИИ, перечень которых определен в разделе 2 настоящего Кодекса.

Уважение автономии и свободы воли человека

Акторы ИИ должны принимать необходимые меры, направленные на сохранение автономии и свободы воли человека в принятии им решений, права

[2] Кодекс этики в сфере искусственного интеллекта [Электронный ресурс]. URL: https://www.aiethic.ru/code

выбора и в целом сохранения интеллектуальных способностей человека как самостоятельной ценности и системообразующего фактора современной цивилизации. Акторы ИИ должны на этапе создания систем ИИ (СИИ) прогнозировать возможные негативные последствия для развития когнитивных способностей человека и не допускать разработку СИИ, которые целенаправленно вызывают такие последствия.

Соответствие закону

Акторы ИИ должны знать и соблюдать положения законодательства Российской Федерации во всех сферах своей деятельности и на всех этапах создания, внедрения и использования технологий ИИ, в том числе в вопросах юридической ответственности Акторов.

Недискриминация

В целях обеспечения справедливости и недопущения дискриминации Акторы ИИ должны принимать меры для того, чтобы удостовериться, что применяемые ими алгоритмы и наборы данных, методы обработки используемых для машинного обучения данных, при помощи которых осуществляется группирование и/или классификация данных, касающихся отдельных лиц или групп лиц, не влекут их умышленную дискриминацию. Акторам рекомендуется создавать и применять методики и программные решения, выявляющие и препятствующие возникновению дискриминации по признакам расовой, национальной, половой принадлежности, политических взглядов, религиозных убеждений, возраста, социального и экономического статуса или по сведениям о частной жизни (при этом дискриминацией не может признаваться явно задекларированные Актором ИИ правила функционирования или применения СИИ для разных групп пользователей, сегментированных с учётом таких признаков).

Оценка рисков и гуманитарного воздействия

Акторам ИИ рекомендуется проводить оценку потенциальных рисков применения СИИ, включая социальные последствия для человека, общества и государства, гуманитарного воздействия СИИ на права и свободы человека на разных стадиях жизненного цикла, в том числе при формировании и использовании наборов данных; осуществлять долгосрочный мониторинг проявления таких рисков; учитывать сложность поведения СИИ, включая взаимосвязь и взаимозависимость процессов в жизненном цикле СИИ при оценке рисков.

Для критических приложений СИИ в особых случаях приветствуется проведение оценки рисков посредством привлечения нейтральной третьей стороны или уполномоченного официального органа, но без ущерба для работоспособности и информационной безопасности такой СИИ, а также охраны интеллектуальной собственности и коммерческой тайны разработчика.

2. Необходимо осознавать ответственность при создании и использовании ИИ

Риск-ориентированный подход

Уровень внимания к этическим вопросам в области ИИ и характер соответствующих действий Акторов ИИ должен быть пропорционален оценке

уровня рисков, создаваемых конкретными технологиями и СИИ для интересов человека и общества. Оценка уровня рисков учитывает как известные, так и возможные риски, при этом принимается во внимание как уровень вероятности угроз, так и их возможный масштаб в краткосрочной и долгосрочной перспективе. Принятие значимых для общества и государства решений в области применения ИИ должно сопровождаться научно выверенным междисциплинарным прогнозированием социально- экономических последствий и рисков, изучением возможных изменений в ценностно-культурной парадигме развития общества с учетом национальных приоритетов.

Во исполнение настоящего Кодекса рекомендуется разработка и использование методики оценки рисков СИИ.

Ответственное отношение

Акторы ИИ должны ответственно относиться к вопросам влияния СИИ на общество и граждан на каждом этапе жизненного цикла СИИ, включая неприкосновенность частной жизни, этичное, безопасное и ответственное использование персональных данных к характеру, степени и размеру ущерба, который может последовать в результате использования технологий и СИИ, а также при выборе и использовании аппаратных средств и программного обеспечения, задействованных на различных жизненных циклах СИИ.

При этом ответственность Акторов ИИ должна соответствовать характеру, степени и размеру ущерба, который может последовать в результате использования технологий и СИИ, а также учитывать роль Актора в жизненном цикле СИИ и степень возможного и реального влияния конкретного Актора ИИ на причинение ущерба и его размер.

Предосторожность

Когда деятельность Акторов ИИ может привести к морально неприемлемым для человека и общества последствиям, наступление которых соответствующий Актор ИИ может разумно предположить, им должны быть приняты меры, чтобы предотвратить или ограничить наступление таких последствий. Для оценки категории «моральной неприемлемости последствий» и обсуждения возможных мер их предотвращения Акторы ИИ используют положения настоящего Кодекса, в том числе механизмы, указанные в разделе 2 настоящего Кодекса.

Непричинение вреда

Акторы ИИ не должны допускать использование технологий ИИ в целях причинения вреда жизни и (или) здоровью человека, имуществу граждан и юридических лиц, окружающей среде. Любое использование, в том числе проектирование, разработка, тестирование, внедрение, эксплуатация СИИ, способных целенаправленно причинять вред окружающей среде, жизни и (или) здоровью человека, имуществу граждан и юридических лиц, недопустимо.

Идентификация ИИ в общении с человеком

Акторам ИИ рекомендуется осуществлять добросовестное информирование пользователей об их взаимодействии с СИИ, когда это затрагивает вопросы прав человека и критических сфер его жизни, и

обеспечивать возможность прекратить такое взаимодействие по желанию пользователя.

Безопасность работы с данными

Акторы ИИ должны соблюдать законодательство Российской Федерации в области персональных данных и охраняемых законом тайн при использовании СИИ; обеспечивать охрану и защиту персональных данных, обработка которых осуществляется СИИ или Акторами ИИ в целях разработки и совершенствования СИИ; разрабатывать и внедрять инновационные методы борьбы с несанкционированным доступом третьих лиц к персональным данным; использовать качественные и репрезентативные наборы данных, полученные без нарушения закона из надежных источников.

Информационная безопасность

Акторы ИИ должны обеспечивать максимально возможную защиту от несанкционированного вмешательства в работу СИИ третьих лиц; внедрять адекватные технологии информационной безопасности, в том числе применять внутренние механизмы защиты СИИ от несанкционированных вмешательств и информирования пользователей и разработчиков о таких вмешательствах; содействовать информированию пользователей о правилах информационной безопасности при использовании СИИ.

Добровольная сертификация и соответствие положениям Кодекса

Акторы ИИ могут внедрять системы добровольной сертификации соответствия разработанных технологий ИИ нормам, установленным законодательством Российской Федерации и настоящим Кодексом. Акторы ИИ могут создавать системы добровольной сертификации и маркировки СИИ, свидетельствующие о прохождении данными системами процедур добровольной сертификации и подтверждающих стандарты качества.

Контроль рекурсивного самосовершенствования СИИ

Акторам ИИ рекомендуется сотрудничать в выявлении и проверке информации о способах и формах создания так называемых универсальных («сильных») СИИ и предотвращении возможных угроз, которые они несут. Вопрос применения технологий «сильного» ИИ должен находиться под контролем государства.

3. Ответственность за последствия применения СИИ всегда несет человек

Поднадзорность

Акторам ИИ следует обеспечивать комплексный надзор человека за любыми СИИ в объеме и порядке, зависящих от назначения СИИ, в том числе, например, фиксировать существенные решения человека на всех этапах жизненного цикла СИИ или предусматривать регистрационные записи работы СИИ; обеспечивать прозрачность применения СИИ и возможность отмены человеком и (или) предотвращения принятия социально и юридически значимых решений и действий СИИ на любом этапе жизненного цикла СИИ там, где это разумно применимо.

Ответственность

Акторы ИИ не должны допускать передачи полномочий ответственного нравственного выбора СИИ, не должны делегировать ответственность за последствия принятия решений СИИ — за все последствия работы СИИ всегда должен отвечать человек (физическое или юридическое лицо, признаваемое субъектом ответственности в соответствии с действующим законодательством Российской Федерации). Акторам ИИ рекомендуется принимать все меры для определения ответственности конкретных участников жизненного цикла СИИ с учетом их роли и специфики каждого этапа.

4. Технологии ИИ нужно применять по назначению и внедрять там, где это принесёт пользу людям

Применение СИИ в соответствии с предназначением

Акторы ИИ должны использовать СИИ в соответствии с заявленным предназначением, в предписанной предметной области, для решения предусмотренных прикладных задач.

Стимулирование развития ИИ

Акторы ИИ должны поощрять и стимулировать разработку, внедрение и развитие безопасных и этичных решений в сфере технологий ИИ с учетом национальных приоритетов.

5. Интересы развития технологий ИИ выше интересов конкуренции

Корректность сравнений СИИ

Для поддержания добросовестной конкуренции и эффективного сотрудничества разработчиков при сравнении СИИ между собой Акторам ИИ рекомендуется использовать максимально достоверную и сравнимую информацию о возможностях СИИ применительно к задаче, а также обеспечивать единство методики измерений.

Развитие компетенций

Акторам ИИ рекомендуется следовать принятым в профессиональном сообществе практикам, поддерживать должный уровень профессиональной компетенции, необходимый для безопасной и эффективной работы с СИИ, содействовать повышению профессиональной компетенции работников в области ИИ, в том числе в рамках программ и образовательных дисциплин по этике ИИ.

Сотрудничество разработчиков

Акторам ИИ рекомендуется развивать сотрудничество в рамках сообщества Акторов ИИ, прежде всего разработчиков, в том числе путем информирования о выявленных критических уязвимостях с целью предотвращения их массового распространения, а также прилагать усилия для повышения качества и доступности ресурсов в сфере разработки СИИ, в том числе путем:

• повышения доступности данных, в том числе размеченных;

• обеспечения совместимости разрабатываемых СИИ там, где это применимо;

• создания условий для формирования национальной школы развития технологий ИИ, в том числе общедоступных национальных репозиториев

библиотек и моделей сетей, доступных национальных средств разработки, открытых национальных фреймворков и др.;

• обмена информацией о лучших практиках развития технологий ИИ;

• организации или проведения конференций, хакатонов, публичных конкурсов или участия в них, школьных и студенческих олимпиад;

• повышения доступности знаний и поощрения использования открытых баз знаний;

• формирования условий для привлечения инвестиций в развитие технологий ИИ от российских частных инвесторов, бизнес-ангелов, венчурных фондов и фондов прямых инвестиций;

• стимулирования научной, образовательной, просветительской деятельности в сфере ИИ путем участия в проектах и деятельности ведущих научно-исследовательских центров и образовательных организаций России.

6. Важна максимальная прозрачность и правдивость в информировании об уровне развития технологий ИИ, их возможностях и рисках

Достоверность информации о СИИ

Акторам ИИ рекомендуется предоставлять пользователям СИИ достоверную информацию о СИИ, допустимых областях и наиболее эффективных методах применения СИИ, вреде, пользе и существующих ограничениях в их применении.

Повышение осведомлённости об этике применения ИИ

Акторам ИИ рекомендуется проводить мероприятия, направленные на повышение уровня доверия и осведомлённости граждан, являющихся пользователями СИИ, в частности и общества в целом, о разрабатываемых технологиях, особенностях этичного применения СИИ и иных сопутствующих развитию СИИ положениях всеми доступными способами, в том числе путём разработки научных, публицистических материалов, организации научных и общественных конференций, семинаров, а также посредством включения в правила эксплуатации СИИ правил этичного поведения пользователей и (или) эксплуатантов.

Вопросы для собеседования

1. Почему необходимо этическое регулирование создания систем ИИ и взаимодействия человека и искусственного интеллекта?
2. Какие этические проблемы могут возникать при использовании систем искусственного интеллекта?
3. Кто несет ответственность за этичное/неэтичное поведение систем искусственного интеллекта?
4. Каковы основные принципы этики искусственного интеллекта?
5. Какова взаимосвязь правового и этического регулирования в сфере работы с большими данными?
6. Каковы основные положения Кодекса этики использования данных?
7. Какие этические правила следует соблюдать при отборе данных для обучения алгоритмов? Как можно обезличить социальные и персональные данные?

8. Как предотвратить создание алгоритмов и систем принятия решений, дискриминирующих отдельные группы населения?

Практикум

Задание 1. Учет этических принципов при создании, внедрении и использовании технологий ИИ

Группа делится на команды по 3–5 человек. Каждая команда представляет собой комитет по этике, выдвигающий требования и проверяющих их исполнение при создании, внедрении и использовании конкретной системы ИИ (например, система управления беспилотным доставщиком пиццы, рекомендательная система подбора персонала ИТ-компании, система принятия решений о выдаче кредита банком, рекомендательная система выбора образовательной траектории, адаптивная система персонализированного обучения иностранному языку). Задача команды – сформулировать:

1) этические проблемы, которые могут возникнуть при использовании данной системы ИИ;

2) требования к системе ИИ для предотвращения этих проблем;

3) показатели (критерии) для оценки соблюдения требований.

Методические рекомендации

Тип системы ИИ и ее функционал выбирает или предлагает сама команда. После первого этапа обсуждения (10–15 минут) рекомендуется каждой команде публично представить функционал своей системы и перечень этических проблем, которые могут возникнуть при использовании данной системы ИИ. Это позволит совместно со всей группой обсудить и дополнить этот перечень. Итоговый результат – требования к системе ИИ для предотвращения этических проблем и показатели (критерии) для оценки соблюдения требований каждой команде рекомендуется представить в форме таблицы и краткого выступления.

Литература:

Этика и «цифра»: этические проблемы цифровых технологий: Аналитический доклад [Электронный ресурс]. URL: https://ethics.cdto.center

Этика и «цифра»: от проблем к решениям: Аналитический доклад [Электронный ресурс]. URL: https://ethics.cdto.center/2021

Кодекс этики в сфере искусственного интеллекта [Электронный ресурс]. URL: https://www.aiethic.ru/code

Занятие 3. Профессиональное развитие и самореализация (Часть 1. Профессиональная самореализация)

Цель: повышение осмысленности профессионального становления и профессиональной самореализации магистрантов.

Задачи:

развитие представлений о профессиональной самореализации;

определение соответствия индивидуальных ценностей, смыслов, способностей осваиваемой профессии;

помощь магистрантам в осознании собственных возможностей профессиональной самореализации;

развитие навыков самопознания и самоанализа.

Теоретическое введение

Материалы лекции 2.

Термин «самореализация» (self-realisation) впервые был приведен в «Словаре по философии и психологии», изданном в 1902 году. Чаще всего понятие «самореализация» в зарубежных и отечественных исследованиях интерпретируется как «реализация собственного потенциала». Понятие «самореализация личности», обозначающее процесс, в основе которого лежат проблемы роста, развития, самосовершенствования человека, часто употребляют как синоним термина «самоактуализация» (self-actualization), «реализация своих возможностей» (self-fulfillment).

По А. Маслоу, самоактуализация – это полноценное развитие человека, основной мотив ее развития, извечное стремление человека «быть тем, кем он может быть». Это «непрерывная реализация потенциальных возможностей, способностей и талантов как осуществление своей миссии, призвании, судьбы, как более полное познание и, следовательно, принятие своей собственной первоначальной природы, как постоянное стремление к единству, интеграции, или внутренней синергии личности». Потребность в самоактуализации оказывается естественной, необходимой, собственно человеческой потребностью, которая «предоставляет человеческой жизни яркую направленность и глубокий смысл, укрепляет веру в себя и дает силы пережить жизненные невзгоды» (А. Маслоу).

На современном этапе развития психологической науки самоактуализация связывается с процессом развития личности как субъекта жизнедеятельности и формирования жизненной стратегии (К. Абульханова-Славская), интеллектуальной активности (Д. Богоявленская), свободного выбора (В. Петровский), саморегуляции произвольной активности (А. Конопкин, В. Иванников), носителя ответственности, нравственности, веры (Б. Братусь) и смысла (Д. Леонтьев). Самоактуализация рассматривается как процесс, состояние, потребность, результат и свойство личности. Она тесно связана с самопознанием и самосовершенствованием, обеспечивая социальную «результативность» личности.

Самореализация имеет огромное значение для субъективного благополучия человека. В процессе самореализации человек, в соответствии с собственными ценностями и целями, принимает решения, делает выбор, т.е. творит собственную жизнь. По мнению ученых, решающим фактором самореализации, реализации своего потенциала являются не природные задатки человека сами по себе, а сформированные личностные качества как продукт образования и воспитания, обучения (*Коростылева, 2005)*.

Одной из наиболее значимых для самореализации человека является профессиональная сфера. Понятие «профессиональная самореализация» – это более широкое понятие, чем, например, «высокий профессионализм», «успешная карьера». Профессиональная самореализация личности означает и успешность профессионального самоопределения, и ее высокий профессионализм, и карьерный рост. Профессиональная самореализация является критерием профессионализма, отражением системы жизненных смыслов, целей и подразумевает потребность в осознании и реализации своих личностных качеств, рост личностной зрелости, профессионального мастерства. *Профессиональную самореализацию определяют как интегральную динамическую характеристику субъекта труда*, отражающую процесс и результат осуществления им своих сущностных свойств, трансляции своего содержания другим людям и культуре через созидательные и коммуникативные процессы *(Гаврилова, 2015)*.

Профессиональная самореализация личности имеет следующие этапы: профессиональное самоопределение, профессиональное становление в избранной сфере деятельности, профессиональный рост и профессиональная компетентность, которая является и этапом профессиональной самореализации, и фактором ее успешности.

Рассматривая профессиональную самореализацию как результат, можно выделить следующие ее характеристики:

самостоятельность как способность личности к постановке целей, рефлексии, регуляции своей деятельности;

свобода выбора, действий, решений как способность личности к автономности поведения и саморегуляции;

способности к концентрации творческих усилий, креативность, независимость в суждениях в сочетании с ответственностью за результаты собственной деятельности.

Содержательно понятие «профессиональная самореализация» может рассматриваться в единстве трех компонентов: целевом, ресурсном и феноменологическом.

Рис. 3. Структурная модель профессиональной самореализации

Критерием успешности профессиональной самореализации будет совокупная выраженность этих трех компонентов.

Вопросы для собеседования:

1. Почему профессиональное самосознание можно рассматривать как личностный ресурс профессионального развития?
2. Что включает профессиональное самосознание?
3. Каковы факторы формирования профессионального самосознания?
4. Что такое профессиональная «Я-концепция»?
5. Какие функции выполняет самопознание в личностно-профессиональном развитии?
6. Что понимается под профессиональной самореализацией?
7. Каковы условия реализации своего потенциала в профессии?

Практикум

Задание 1. Диагностика профессиональной самореализации

Ответьте на вопросы методики, подсчитайте результаты, проанализируйте характеристики своей профессиональной самореализации.

Методика «Тип и уровень профессиональной самореализации» Е.А. Гавриловой. См. Куповых Ж.Г., Лабынцева И.С., Лызь Н.А., Эксакусто Т.В. Практикум по психологии управления личностными ресурсами: учебно-методическое пособие. – Таганрог: Издательство Южного федерального университета, 2017. С. 57–61. – URL: https://hub.sfedu.ru/repository/material/800839516/

Задание 2. Поиск путей самореализации через профессию

Упражнение «Мои профессиональные творческие успехи»

Инструкция: наверное, каждый из нас уже, так или иначе, выбрал свой путь, свою область деятельности и представляет, каким специалистом он хочет быть. Давайте пофантазируем, каких творческих успехов вы сможете добиться, развиваясь в этом направлении; как в связи с этим изменитесь вы и ваша жизнь, как изменится ваше окружение, семья, друзья. Посмотрите на эту картинку с

разных сторон, побудьте в этих ощущениях. Запомните их. Как только вы почувствуете возможным, открывайте глаза. А теперь нарисуйте свои профессиональные творческие успехи.

На выполнение задания отводится 10–15 минут.

Далее предлагается следующее: «разложите работы по кругу на полу. Походите и посмотрите на работы других. Отметьте те работы, которые вас привлекли. Может быть, чья-то работа или ряд работ оказались очень схожими с вашей в какой-то основной идее. Положите свою работу рядом с той, которая наиболее созвучна с вашей работой».

Задание выполняется по кругу. Если рядом с работой участника уже лежит работа, то он может взять обе эти работы и положить рядом с той, которая для него кажется созвучной. Если образуется какая-то группа и кто-либо из участников хочет к ней присоединиться, то свою работу он все равно кладет рядом с той, которая наиболее подходит к его работе.

В результате упражнения должно получиться несколько групп. После чего группам дается задание. «Каждая группа должна представить свой способ развития и ответить на такие вопросы:

1. Какова главная жизненная или профессиональная цель членов вашей группы?
2. Способы достижения целей – как? С помощью кого или чего?
3. Что происходит с вами и с вашим окружением?
4. Что уже сделано для осуществления цели?
5. Чем вы можете пользоваться из своего уже имеющегося багажа? Что необходимо еще приобрести?
6. Насколько реально достижение этой цели?»

На выполнение групповой работы отводится 10 минут. Далее группа выбирает одного или двух человек, которые будут презентовать творческие успехи и путь развития группы. На презентацию отводится три минуты. По истечении времени все участники могут задавать вопросы или высказать свое мнение.

Методические рекомендации

Рекомендации к выполнению задания 1.

Задание выполняется индивидуально. Может быть вынесено на самостоятельную работу (в рамках подготовки к занятию). Преподаватель оказывает помощь студентам в интерпретации и осознании результатов диагностики типа и уровня профессиональной самореализации. Рекомендуется обсудить следующие вопросы:

Совпадает ли полученный результат с вашими первоначальными представлениями об особенностях своей профессиональной самореализации?

Удовлетворены ли вы полученным результатом?

Если нет, то над чем стоит задуматься, что можно сделать, чтобы профессиональная самореализация стала более продуктивной?

Рекомендации к выполнению задания 2.

Необходимо заранее подготовить чистые листы формата А4, цветные карандаши для работы.

Литература

Гаврилова Е.А. Психодиагностическая методика «Тип и уровень профессиональной самореализации»: разработка, описание и психометрия / Вестник ТвГУ. Серия «Педагогика и психология». 2015. № 3. С. 19–34.

Коростылева Л.А. Психология самореализации личности: затруднения в профессиональной сфере. – СПб.: Речь, 2005. – 222 с.

Эксакусто Т.В. Групповая психокоррекция: тренинги и роли, игры для личностного и профессионального развития / Т.В. Эксакусто, О. Н. Истратова. – Ростов н/Д: Феникс, 2014. – 254 с.

Занятие 4. Профессиональное развитие и самореализация (Часть 2. Креативное мышление)

Цель: осознание магистрантами своего творческого потенциала, повышение готовности к его использованию и развитию.

Задачи:

осознание и преодоление барьеров проявления креативности;

расширение представлений о креативной среде;

формирование навыков и умений управления креативным процессом;

развитие навыков самопознания и самоанализа.

Теоретическое введение

Понятие «креативность» теснейшим образом связано с понятием «творчество». Существует достаточно много определений творчества. Так, например, Я.А. Пономарев рассматривает творчество как взаимодействие, ведущее к развитию. С.Л. Рубинштейн, один из основоположников отечественной психологии, определяет творчество как деятельность, созидающую нечто новое, оригинальное, что потом входит не только в историю развития самого творца, но и в историю развития науки, искусства и т.д. А.М. Матюшкин говорит, что творчество – это выход за пределы уже имеющихся знаний, преодоление, опрокидывание границ.

По поводу сущности творчества также сложились различные мнения. Так, например, психоаналитики считают, что творчество – результат внутриличностных конфликтов. Гуманистическая психология считает, что творчество возникает, когда отсутствуют внутриличностные конфликты. Творческий процесс является реализацией естественного творческого потенциала в случае устранения внутренних барьеров.

Креативность можно определить как способность человека к конструктивному, нестандартному мышлению и поведению, а также к осознанию и развитию своего опыта. Это способность человека отказаться от стереотипных способов мышления или способность обнаруживать новые способы решения проблем. Формами проявления креативности являются:

Быстрота, гибкость, точность, оригинальность мышления.

Богатое воображение.

Чувство юмора.

Приверженность эстетическим ценностям.

Любознательность.

Открытость новому опыту и стремление к его приумножению.

Интеграция конвергентного (логического, последовательного, линейного) и дивергентного (целостного, интуитивного) мышления.

Результатом проявления креативности являются следующие особенности мышления:

Способность генерировать большое количество идей.

Способность к порождению широкого многообразия идей.

Способность генерировать оригинальные идеи.

Способность придавать завершенный вид продуктам мышления.

Существенным условием актуализации этих способностей является самообладание и уверенность в себе.

Креативность необходима человеку в деятельности, общении, повседневной жизни. Она во многом определяет успешность профессиональной деятельности. Креативность помогает не только создавать художественные произведения, но и находить оригинальные решения сложных технических, организационных, научных проблем. Креативность является важным ресурсом любой личности и мощным фактором ее развития, определяющим ее готовность изменяться, отказаться от стереотипов.

Вследствие этого становится очевидной необходимость поиска средств, позволяющих развивать креативность – способность, которой, пусть в разной степени, но *обладает каждый человек.*

Эффективность творческого процесса можно стимулировать благодаря созданию креативной среды, параметрами которой являются: проблемность, неопределенность, принятие, безоценочность.

У креативного процесса есть свои этапы (стадии): подготовка, фрустрация, инкубация, инсайт и разработка.

Подготовка. Этот этап в первую очередь связан с появлением побуждения изменить ту или иную ситуацию, переставшую удовлетворять субъекта творчества. Этап подготовки характеризуется сознательными усилиями по поиску выхода из проблемной ситуации. Субъект логически прорабатывает, анализирует задачу, проблему как в целом, так и отдельные ее элементы, собирает дополнительную информацию.

Фрустрация. Переход на этот этап происходит в тот момент, когда, проанализировав всю имеющуюся в его распоряжении информацию и проверив возникшие варианты решения, индивид все-таки не находит ответ. Иными словами, он оказывается в тупике.

Фрустрация является сигналом для реорганизации деятельности. В этот момент целесообразно осознать, какой барьер препятствует проявлению креативности, в рамках какого стереотипа мы находимся, в какой области нам недостаточно информации и где ее можно получить и т. д. необходимо разложить проблему на фрагменты и собрать дополнительные факты, относящиеся к делу. Чем качественнее будет проведена эта подготовительная работа, тем вероятнее, что можно будет обнаружить те детали и нюансы, которые впоследствии позволят найти комбинации идей, поводящих к решению.

Инкубация. Этап инкубации начинается в тот момент, когда индивид прекращает сознательную работу над проблемой, что связано с логическими операциями левого полушария, и проблема «передается» в правое полушарие. В случае сохранения мотивации, направленной на разрешение проблемы, из правополушарной модели привлекается любая, имеющая отношение к задаче информация. Это творческая работа на бессознательном уровне, практика показывает, что наиболее интересные решения появляются именно во время

паузы в работе над проектом, например, во время отдыха на природе, в машине, в деловом ужине, в душе и т.д. Правильно использовать эффект от процесса инкубации на практике можно, следуя следующим рекомендациям:

перед мозговыми штурмами, переговорами и презентациями заранее ознакомьтесь с проблемой. Не откладывайте подготовку к встрече на последний момент. Поскольку, заранее ознакомившись с задачей, вы уже «поставили вопрос мозгу», пока вы занимаетесь другими делами, инкубационный процесс позволит подсознанию «переварить» и преобразовать информацию;

применяйте технику планирования «на выходные». Она выражается в том, чтобы оставлять творческие размышления над проектами на выходные, когда активная работа мозга переключается на отдых и решение бытовых проблем. В этот инкубационный период, когда сознание, казалось бы, совершенно не думает о работе в нем рождаются интересные творческие идеи. Все новые ощущения или ассоциации любая новая информация становятся своеобразными катализаторами креативного процесса.

Инсайт – это кратковременный, но очень отчетливый этап креативного процесса, момент поступления в сферу сознания решения проблемы. Он характеризуется бурными позитивными эмоциями.

Разработка, или верификация, является завершающим этапом творческого процесса, в ходе которого происходит проверка истинности полученного решения логическими средствами.

«Чтобы повысить свой творческий потенциал, Если Вы банкир, научитесь танцевать чечетку. Если Вы нянечка, займитесь курсом мифологии. Прочитайте книгу на малознакомую Вам тему. Смените газету. Новое соединится со старым в доселе неведомых, удивительных формах». Так пишет Р. Эпштейн, доктор философии, заслуженный директор Кембриджского Центра Исследований поведения.

Вопросы для обсуждения

1. Какое из определений творчества вам ближе и почему?
2. Какими качествами, по-вашему мнению, должен обладать креативный человек?
3. Какие качества характерны антиподу креативного человека?
4. Чем характеризуется креативная среда?
5. Как можно управлять креативным процессом?

Практикум

Задание 1. Определение творческого потенциала

Ответьте на вопросы опросника, подсчитайте результаты, проанализируйте свои творческие способности.

Куповых Ж.Г., Лабынцева И.С., Лызь Н.А., Эксакусто Т.В. Практикум по психологии управления личностными ресурсами: учебно-методическое пособие. – Таганрог: Издательство Южного федерального университета, 2017. С. 46–48. – URL: https://hub.sfedu.ru/repository/material/800839516/

Задание 2. Развитие креативности (тренинг)

Упражнение 1

Цель: создание креативной среды, стимуляция работоспособности.

Инструкция: «Сейчас каждый по очереди скажет своему соседу комплимент, начинающийся на 1 букву его имени. Например, «Сережа, ты смелый!». Время на обдумывание 3 мин».

Вопросы для обсуждения.

Были ли сложности и какие при выполнении данного упражнения?

Чем вызвано перечисление противоречивых характеристик?

Что помешало назвать более разнообразные характеристики?

Упражнение 2

Цель: осознание барьеров креативности.

Участники группы сидят по кругу. У тренера в руках мяч. «Сейчас я начну предстоящую нам работу, брошу кому-то мяч и назову при этом любой предмет. Тот, кому достанется мяч, должен будет назвать три нестандартных способа использования этого предмета. Затем бросит мяч следующему, назвав другой предмет. Будем внимательны и постараемся сделать так, чтобы во время этой работы мяч побывал у каждого». Для более отчетливого проявления барьеров креативности тренер побуждает участников реагировать быстро, т.е. создает внешний барьер: ограничение времени, который дает о себе знать при наличии внутренних барьеров, актуализируя их влияние на человека. В ходе работы, когда кто-нибудь длительное время не может найти очередной вариант нестандартного использования предмета, можно предложить остальным показать, есть ли у них свои варианты, подняв руку или кивнув головой.

Вопросы для обсуждения:

В чем заключались основные трудности, с которыми вы столкнулись при выполнении задания?

Какие состояния возникали и как они изменялись в ходе работы?

Что вам помогало справляться с поставленной задачей?

Упражнение 3

Цель: осознание стереотипов, шаблонов мышления и поведения, которые тормозят развитие, поиск новых решений.

I этап упражнения.

Инструкция: составьте перечень идей, принципов, привычек, стереотипов, которым Вы особенно привержены, невозможность следовать которым вызывает у Вас напряжение.

II этап упражнения**.**

Группа разбивается на команды по 4-5 человек в каждой.

Инструкция: В ходе обсуждения в малых группах, составьте, пожалуйста, общий перечень тех идей, принципов, привычек, которым Вы особенно привержены, невозможность реализации которых вызывает у Вас напряжение. Время выполнения 15 мин.

III этап упражнения.

Каждая команда представляет свой список, фиксируя его на электронной доске, комментируя написанное при необходимости. Время выполнения 7 мин.

IV этап упражнения.

Инструкция: Внимательно прочитайте составленный общий перечень и отметьте в нем, во-первых, две идеи, принципа или привычки, которым Вы на самом деле привержены, но до сих пор это не осознавали, и во-вторых, две – которые у Вас были, но теперь уже не действуют на Вас так сильно, как раньше. Запишите их для себя. Время выполнения 5 мин.

По истечению отведенного времени участникам предлагается по желанию назвать две идеи, принципа или привычки, приверженность которым обнаружена во время работы над общим перечнем, а также те идеи, принципы, привычки, которые уже не оказывают на человека столь сильного влияния. Обсуждение полученных результатов.

Упражнение 4

Цель: развитие гибкости, беглости и точности мышления.

I этап.

Инструкция. напишите вверху листа четыре буквы: Н Г О К. Составьте как можно больше предложений, каждое слово в которых начинается на данные буквы, например: «Николай говорит очень красиво». Время выполнения 3 минуты.

II этап

Каждому участнику предлагается сказать, сколько у него написано предложений, а затем просит каждого прочитать одно из написанных им предложений по выбору самого участника. При ознакомлении с результатами работы участники обнаруживают для себя не использованные ими стилевые, содержательные и другие возможности для составления предложений, что усиливает их мотивацию и позитивно сказывается на результатах последующей работы.

III этап

Инструкция: продолжите составление предложений еще в течение трех минут. Когда отведенное время закончится, каждый участник снова сообщает, сколько ему удалось написать предложений и зачитывает одно из них по своему выбору.

IV этап

Инструкция: разбейтесь на группы по 4-5 человек и напишите рассказ о вашей группе. Количество слов в предложениях, из которых будет состоять этот рассказ, может быть любым, но слова должны начинаться на буквы Н Г О К. При этом знаки препинания могут ставиться в любом месте. Время выполнения пять минут. Когда работа завершена, каждая команда зачитывает свой рассказ.

Вопросы для обсуждения.

Чей рассказ понравился больше и почему?

Какая команда оказалась наиболее креативной?

Какие лингвистические идеи оказались наиболее неожиданными?

Упражнение 5

Цель: осознание идеи неограниченности версионного представления проблемы.

I этап

Инструкция: напишите каждый самостоятельно как можно больше аргументов в пользу бега по утрам. Время выполнения три минуты. Далее каждому предлагается сказать, сколько аргументов у него написано. Эта информация не обсуждается, и тренером не комментируется.

II этап

Инструкция: участникам предлагается объединиться по желанию в пары и поделиться друг с другом сформулированными аргументами. Запомните 1-2 особенно неожиданных и оригинальных аргумента, которые есть у партнера, а у вас нет.

III этап

Инструкция: сейчас у вас будет еще три минуты, в течение которых каждый самостоятельно постарается написать как можно больше аргументов против бега по утрам.

Затем участники обмениваются результатами своей работы так же, как они делали это на предыдущем этапе. Обсуждение: какие из аргументов «за» и «против» бега по утрам особенно понравились вам, показались оригинальными, неожиданными.

Упражнение 6

Цель: снятие эмоционального напряжения, получение личностно-ориентированной обратной связи.

Инструкция: сосредоточьтесь на своем соседе слева. Вспомните все его проявления во время нашей работы, все, что он говорил, делал. Вспомните чувства и отношения, которые возникали у Вас к этому человеку. Для этого у нас будет 2 минуты.

Теперь решите, какое из описаний природы, погоды, времени года, которые Вы встречали в литературе или придуманное Вами, соответствуют Вашим впечатлениям об этом человеке. Когда все будут готовы, каждый, по очереди, скажет своему соседу возникшее у него описание. Время на обдумывание 3мин.

Вопросы для обсуждения.

Понравилась ли Вам прозвучавшая ассоциация?

Была ли она для Вас неожиданной?

Трудно ли было придумать ассоциацию?

Что нового Вы узнали о себе и о других благодаря этому упражнению и всему тренингу?

Методические рекомендации

Для реализации на занятии из приведенного перечня упражнений преподаватель может выбрать упражнения, соответствующие возможностям помещения и интересам учебной группы.

Рекомендации магистрантам к заданию 1:

прежде чем отвечать на вопросы опросника, проанализируйте и дайте свою оценку собственному творческому потенциалу;

пройдите тестирование и подсчитайте результат;

сравните ваши представления о своем творческом потенциале и результаты опросника;

подумайте над тем, как повысить свой творческий потенциал.

Рекомендации магистрантам к упражнению 2.

Изобретая новые способы употребления предметов, не прибегайте к универсальным способам использования большинства предметов: почти любой предмет можно нарисовать, потрогать, понюхать, многие предметы можно подарить.

Рекомендации магистрантам к упражнению 3.

В процессе обсуждения результатов упражнения или при самостоятельном выполнении данного упражнения, ответьте на следующие вопросы:

Какое влияние на самочувствие, поведение, жизнь оказывает приверженность тем или иным привычкам?

Что помогло преодолеть ранее существовавшие стереотипы?

Как это может повлиять на Вашу жизнь?

Литература

Психогимнастика в тренинге/ Под. ред. Хрящевой Н.Ю. – СПб.: Речь; Институт тренинга, 2001. – 256 с.

Рогов Е.И. Настольная книга практического психолога. Книга 2. М., 1999.

Рудестам К. Групповая психотерапия. – М.: Питер, 1998. – 384 с.

Истратова О.Н., Эксакусто Т.В. Справочник по групповой психокоррекции. – Ростов н/Д: Феникс, 2006. – 443 с.

Занятие 5. Профессиональное развитие и самореализация (Часть 3. Проектирование профессионального будущего)

Цель: освоение магистрантами способов проектирования профессионального будущего и планирования карьеры, повышение осмысленности и целенаправленности профессионального развития.

Задачи:

развитие представлений о карьерных ориентациях и других личностных ресурсах для успешной карьеры;

развитие умений рефлексировать собственные цели, ресурсы, возможности, ограничения и проектировать профессиональное будущее;

помощь студентам в осознании жизненных и профессиональных целей, в определении перспектив личностно-профессионального саморазвития;

развитие навыков оценки результатов собственной деятельности, а также умений определять приоритеты деятельности на основе понимания ценностей и целей.

Теоретическое введение

Материалы лекции 2.

Проектирование профессионального будущего включает определение перспектив профессионального развития и планирование карьеры. Если профессиональное развитие предполагает преобразование человеком своего внутреннего мира, становление профессиональных знаний и умений, мотивов, качеств и способностей, то в карьере акцент делается на внешней активности человека, внешней детерминации его жизнедеятельности, продвижении по служебной лестнице.

Несмотря на то, что существуют факты значительного карьерного продвижения без высокого уровня профессионализма или случаи профессионального развития без должностного продвижения, тем не менее успешная карьера всегда связана с развитием.

Индивидуальный карьерный путь зависит как от множества объективных условий жизни и деятельности, в том числе специфичных для конкретных организаций, так и от профессионального потенциала, личностных ресурсов и карьерных ориентаций самого человека.

Карьерные ориентации – это базовые социальные установки, отражающие значимость карьеры для человека, предпочитаемый им тип карьеры, готовность индивида реализовать тот или иной карьерный путь. Согласно типологии Э. Шейна выделяется восемь основных карьерных ориентаций.

1. Профессиональная компетентность. Эта ориентация связана с наличием способностей в определенной области. Человек с такой ориентацией хочет быть мастером своего дела, он бывает особенно счастлив, когда достигает успеха в профессиональной сфере, но быстро теряет интерес к

работе, которая не позволяет развивать свои способности. Одновременно такой человек ищет признания своих талантов, что должно выражаться в статусе, соответствующем его мастерству. Он готов управлять другими в пределах своей компетентности, но управление не представляет для него особого интереса. Поэтому многие из этой категории отвергают работу менеджера. Обычно это самая многочисленная группа в большинстве организаций, обеспечивающая принятие компетентных решений.

2. Менеджмент. В данном случае первостепенное значение имеет ориентация личности на интеграцию усилий других людей, принятие ответственности за конечный результат деятельности группы сотрудников или организации в целом. Такая работа требует навыков межличностного и группового общения, эмоциональной уравновешенности, чтобы нести бремя ответственности и власти. Человек с карьерной ориентацией на менеджмент будет считать, что не достиг вершин своей карьеры, пока не займет должность, на которой сможет управлять различными сторонами деятельности предприятия или организации: финансами, маркетингом, производством продукции, разработками, продажами. С возрастом и опытом работы эта карьерная ориентация проявляется сильнее.

3. Автономия. Первичная забота личности с такой ориентацией – освобождение от организационных правил, предписаний и ограничений. Ярко выражена потребность все делать по-своему: самому решать когда, над чем и сколько работать. Такой человек не хочет подчиняться правилам организации (рабочее место, время, форменная одежда). Конечно, каждый человек в некоторой степени нуждается в автономии, однако, если такая ориентация выражена сильно, то личность готова отказаться от продвижения по службе или от других возможностей ради сохранения своей независимости. Такой человек может работать в организации, которая обеспечивает достаточную степень свободы, но не будет чувствовать серьезных обязательств или преданности организации и будет отвергать любые попытки ограничить его автономию.

4. Стабильность. Эта карьерная ориентация обусловлена потребностью в безопасности и стабильности, желанием, чтобы будущие жизненные события были предсказуемы. Различают два типа стабильности: стабильность места работы и стабильность места жительства. Стабильность места работы подразумевает поиск работы в такой организации, которая обеспечивает определенный срок службы, имеет хорошую репутацию (не увольняет сотрудников), заботится о своих работниках после увольнения и, выглядит более надежной в своей отрасли. Человек с такой ориентацией (его часто называют «человеком организации») ответственность за управление карьерой перекладывает на нанимателя. Он будет совершать какие угодно географические передвижения, если того потребует компания. Человек второго типа, ориентированный на стабильность места жительства, связывает себя с географическим регионом, «пуская корни» в определенном месте, вкладывая сбережения в свой дом, и меняя работу или организацию только тогда, когда это предотвращает его «срывание с места». Люди, ориентированные на стабильность, могут быть талантливыми и занимать высокие должности в

организации, но, предпочитая стабильную работу и жизнь, они откажутся от повышения, если оно грозит риском и временными неудобствами, даже в случае широко открывающихся возможностей роста.

5. Служение. Основными ценностями при данной ориентации являются «работа с людьми», «служение человечеству», «желание сделать мир лучше» и т.д. Человек с такой ориентацией стремится продолжать работать в этом направлении, даже если ему приходится менять место работы. Он не будет работать в организации, которая враждебна его целям и ценностям, и откажется от продвижения или перевода на другую работу, если это не позволит ему реализовать главные ценности жизни. Люди с такой карьерной ориентацией чаще всего работают в области охраны окружающей среды, проверки качества продуктов и товаров и т.п.

6. Вызов. Основные ценности при ориентации данного типа – конкуренция, победа над другими, преодоление препятствий, решение трудных задач. Социальная ситуация чаще всего рассматривается с позиции выигрыша – проигрыша. Процесс борьбы и победа более важны для человека, чем конкретная область деятельности или квалификация. Например, торговый агент может рассматривать каждый контракт с покупателем как игру, которую надо выиграть. Новизна, разнообразие и вызов имеют для людей с такой ориентацией очень большую ценность, и, если все идет слишком просто, им становится скучно.

7. Интеграция стилей жизни. Человек ориентирован на то, чтобы в жизни было все сбалансировано, он не хочет, чтобы доминировала только семья, или только работа, или только саморазвитие. Он стремится к балансу всех сфер.

8. Предпринимательство. Человек с такой карьерной ориентацией стремится преодолевать препятствия, готов к риску. Он не желает работать на других, а хочет иметь свою марку, свое дело, свое финансовое богатство. Причем это не всегда творческий человек, для него главное – создать дело, концепцию или организацию, построить ее так, чтобы это было продолжением его самого, вложить туда душу. Предприниматель будет продолжать свое дело, даже если сначала он будет терпеть неудачи и ему придется серьезно рисковать.

Планирование личной карьеры предполагает самопознание, оценку жизненной ситуации, соотнесение жизненных и профессиональных целей, осознание карьерных ориентаций, постановку карьерных целей, анализ факторов, критических пунктов, ресурсов, разработку частных задач и планов деятельности, способствующих достижению целей карьеры, определение путей личностно-профессионального развития. В процессе постановки карьерных целей и разработки планов карьеры используются технологии самоанализа, позволяющие лучше понять себя, методики постановки карьерных целей и определения средств их достижения. Для оптимизации постановки карьерных целей необходимо прояснение собственных интересов и потребностей, а также возможностей и ресурсов для их реализации. Этому может служить SWOT-анализ, который предполагает выявление сильных, слабых сторон, возможностей и угроз (Strengths, Weaknesses, Opportunities, Threats). Уточнение

собственных достоинств (сильные стороны) позволяет ощутить внутренние ресурсы для дальнейшего продвижения, осознание своих ограничений и слабых сторон помогает прояснить цели развития.

Вопросы для собеседования:

1. Какую роль играет прогнозирование своего будущего в профессиональном саморазвитии личности?
2. Что необходимо учитывать при прогнозировании и проектировании профессионального будущего?
3. Что такое карьера? Как она связана с профессиональным развитием?
4. Какие факторы влияют на индивидуальный карьерный путь?
5. Что такое карьерные ориентации? Как они связаны с жизненными целями и ценностями человека?
6. На каких уровнях и каким образом осуществляется планирование карьеры?
7. Всем ли профессионалам в процессе становления приходится сталкиваться с кризисами, противоречиями, трудностями? Может ли траектория профессионального развития быть бескризисной?
8. Какие жизненные и профессиональные обстоятельства могут стать источником кризиса профессионального развития? Почему эти обстоятельства по-разному влияют на разных людей, приводя или не приводя к кризису?
9. Каковы могут быть негативные последствия кризисов профессионального становления, каковы – позитивные? Приведите примеры из собственного опыта.
10. Почему одни профессионалы успешно преодолевают кризис и развиваются, а другие в аналогичной ситуации «сдаются» и деградируют?
11. Какие противоречия являются движущей силой профессионального развития? Выделите группы противоречий и приведите примеры.
12. Может ли новая информационная среда рассматриваться как фактор личностно-профессионального развития человека? Почему?

Задание 1. Определение карьерных ориентаций

Ответьте на вопросы опросника, подсчитайте результаты, проанализируйте свои карьерные ориентации.

Тест «Якоря карьеры» Э. Шейна см. Куповых Ж.Г., Лабынцева И.С., Лызь Н.А., Эксакусто Т.В. Практикум по психологии управления личностными ресурсами: учебно-методическое пособие. – Таганрог: Издательство Южного федерального университета, 2017. С. 70–72. – URL: https://hub.sfedu.ru/repository/material/800839516/

Задание 2 «Планирование карьеры»

Выполните 13 упражнений, которые разработала Н. Карр-Руфино в рамках многоэтапной технологии планирования карьеры.

Упражнение № 1. Что вам нравится делать?

Цель: Узнать себя лучше, определить свои интересы и способности, приспособить их к своей карьере.

Подготовительный этап. Нарисуйте таблицу:

Занятия	Индивидуальное/ коллективное	Степень близости	Фактор риска	Делал(а) в последний раз	Удовлетворяемая потребность	Использованный опыт, навыки
1	2	3	4	5	6	7
Любимые виды деятельности						
1.						
2.						
…						
Навыки и знания						
1.						
2.						
…						

Часть А. Интересы – любимые виды деятельности.

Шаг 1. В первом столбце напишите 5–10 самых любимых своих занятий. Не пытайтесь ничего писать в других столбцах, пока не заполните первый. Постарайтесь уложиться в 10 мин.

Шаг 2. Проанализируйте все выписанные виды деятельности, заполняя остальные столбцы:

– в столбце 2 для каждого любимого занятия поставьте знак «–», если вы предпочитаете выполнять эту деятельность индивидуально, знак «+», если предпочитаете компанию и знак «/», если все равно;

– в столбце 3 поставьте знак «I» напротив занятия, которое считаете личным и знак «I+», если оно слишком личное;

– в столбце 4 отметьте знаком «R» те виды деятельности, которые связаны с риском (если фактор риска очень высок, поставьте знак «R+»);

– в столбце 5 напишите приблизительную дату, когда вы последний раз занимались этой деятельностью;

– в столбце 6 определите, какая потребность удовлетворяется данной деятельностью (например, потребность достижения – Д, потребность власти – В, потребность принадлежности – П);

– в столбце 7 определите навыки и знания, используемые при выполнении деятельности.

Шаг 3. Проранжируйте (пронумеруйте) все виды деятельности – от самой любимой до приемлемой.

Часть В. Навыки и знания – то, что вы делаете хорошо.

Шаг 1. Дополните столбец 1, записав в первой колонке 5–10 вещей, которые вы умеете делать очень хорошо.

Шаг 2. Заполните для дополненных видов деятельности столбцы 2–7.

Шаг 3. Выстройте виды деятельности в порядке их значения для вас, учитывая свой опыт.

Часть С. Примеры и взгляды.

Шаг 1. Проанализируйте взаимосвязи между разными факторами: индивидуальный/коллективный характер деятельности, интимность, риск, удовлетворение потребностей, навыки/знания. Сопоставьте виды деятельности, которые вы умеете делать и которые вам нравятся. Нравится ли вам заниматься тем, что вы умеете? Посмотрите на колонку с датами: вы развиваете свой потенциал или пренебрегаете им? Почему сложилась такая ситуация? Какова в этом роль семьи, друзей, учителей? Ваша личная? Какой глубокий внутренний источник питает вашу страсть к работе и жизни?

Шаг 2. Какие мысли возникли в результате проведенного анализа? Какие изменения нужно осуществить в отношении собственного облика, способностей и пр.?

Часть D. Барьеры на пути к карьере.

Посмотрите на свои интересы, способности, взгляды. Выявите некоторые общие барьеры для реализации ваших знаний и способностей, которые могут быть основой для вашего успеха. Напишите названия этих барьеров на отдельных листах бумаги. Напишите на каждом листе все используемые знания, навыки, интересы. Посмотрите на выявленные барьеры, компонуйте листы в разном порядке.

Упражнение № 2. Первоначальная формулировка целей.

Цель: начать процесс определения наиболее значимых целей.

Шаг 1. Сформулируйте в одном предложении вашу личную миссию.

Шаг 2. Учитывая то, что цель – это итоговый результат, перечислите 5 главных для себя целей. Не забудьте о целях, имеющих отношение к семье, карьере, личному развитию.

Упражнение № 3. Окончательный список целей.

Цель: конкретизация целей, определение приоритетов.

Шаг 1. Отличите деятельность от целей.

Просмотрите составленный при выполнении упражнения № 2 список целей. Вычеркните их него те, которые являются деятельностью.

Шаг 2. Переформулируйте свои цели, чтобы конкретизировать их. Можете воспользоваться приведенными ниже формами:

Занимать _____(должность) к _____(дата).

Иметь отношения с _____ (описание человека, который нравится или которому доверяете) к _____(дата).

Иметь состояние в размере _____руб. к_____(дата).

Весить _____ (кг) к _____(дата).

Иметь_____ (диплом, сертификат) к _____(дата).

Получать пенсию в размере_____ к _____(дата).

Иметь _____ дней свободного времени в год к _____(дата).

Научиться _____ (определить навыки, знания) к _____(дата).

Путешествовать по _____ в _____(дата) в течение_____ (период времени).

Проводить _____ часов в (день, неделю и т. д.), занимаясь любимым делом вместе с _____.

Другие цели:

Шаг 3. Мобилизуйте свои умственные способности.

Перечислите другие цели, которые не включены в предыдущий список. Импровизируйте, используйте весь творческий потенциал своей личности. Пока отодвиньте практическую и критическую сторону своего Я на второй план. Пусть ваши цели будут фантастичными и вместе с тем простыми.

Шаг 4. Оценивайте и определяйте приоритеты.

После того, как вы смело перечислили все цели, определите, какая из них имеет для вас наибольшее значение. Поставьте рядом с ней № 1. Потом определите наиболее важную из оставшихся – поставьте № 2 и т. д. Вы хотите вычеркнуть некоторые цели? Можно ли какие-то цели переформулировать, сделав более реалистичными? Все ли цели достаточно конкретны?

Упражнение № 4. Четкая формулировка целей.

Цель: помочь четко сформулировать конкретные цели.

«Прогноз 6 месяцев». Представьте, что вам осталось жить 6 месяцев. Закройте глаза и попробуйте четко представить эту ситуацию. Перечислите 5 основных вещей, которые вы хотели бы сделать в последние 6 месяцев своей жизни (при условии, что все приготовления уже сделаны и состояние здоровья хорошее).

«Неожиданное богатство». Представьте, что кто-нибудь дал вам 5 млн. долларов. Закройте глаза и попробуйте четко представить эту ситуацию. Перечислите 5 вещей, которые вы хотите заполучить в следующие 6 месяцев (не забывайте, что это последние месяцы в жизни).

Анализ:

– Что из составленного списка не связано со временем и деньгами? Что вы можете получить без денег, а что – не чувствуя недостатка времени?

– Можно ли эти вещи считать вашими целями?

– Можно ли их достичь, немного изменив нынешнее положение дел?

– Вернитесь к упражнению № 4 и выберите окончательные цели.

Упражнение № 5. Сделайте ваши цели эффективными.

Цель: помощь в повышении эффективности целей.

Шаг 1. Представьте себе конечный результат. Постарайтесь выбрать время, чтобы расслабиться и представить конечный результат.

Шаг 2. Выявите источник каждой цели. Вы уверены, что это именно ваша цель?

Шаг 3. Пройдите тест на определение своего эмоционального уровня энергии.

Задайте себе вопросы:

– Чувствую ли я прилив энергии, когда размышляю над определенным выбором?

– Ощущаю ли недостаток энергии, когда обдумываю этот выбор?

– Какой выбор кажется мне наиболее привлекательным с эмоциональной точки зрения, когда я представляю его себе.

Шаг 4. Превратите старые барьеры в новые возможности.

Определите свои прежние и нынешние мысли, чувства, взгляды, решения и т. п. относительно выбранных целей. Мешают ли они вам в достижении этих целей? Какие новые взгляды и отношения нужно выработать?

Какие действия, шаги могут улучшить ситуацию?

1. Выберите главную цель.

2. Перечислите свои убеждения и взгляды, которые не позволяют достичь результата.

3. Определите и перечислите новые убеждения и взгляды, которые помогут вам на этом пути.

4. Повторите этот процесс в отношении всех остальных целей.

Упражнение № 6. Какая часть жизни для вас важна больше всего?

Цель: помощь в определении приоритетов.

Ответьте на вопросы:

1. Если бы в данный период вам пришлось выбирать между целями в карьере и личной жизни, какой бы выбор вы сделали?

2. Если бы пришлось делать выбор между карьерными целями и целями саморазвития, как бы вы поступили? (Вам может помочь список целей, составленный при выполнении упражнения № 3).

3. Какие цели вы бы предпочли: касающиеся вашей личной жизни или саморазвития?

4. Если бы в этом месяце вам пришлось выбирать одну часть жизни (карьера, саморазвитие, семья/личная жизнь), какую бы вы выбрали?

5. Перечислите все три части жизни в порядке значимости для вас.

6. Проанализируйте ответы на вышеизложенные вопросы и попытайтесь определить приоритеты.

Упражнение № 7. Способы достижения ваших целей.

Цель: помощь в определении видов деятельности, способствующих достижению целей.

Для целей, касающихся карьеры, личной жизни и саморазвития определите виды деятельности, способствующие их достижению.

Упражнения № 8. Берите на себя риск.

Цель: помощь в преодолении страха риска.

Шаг 1. Какие привлекающие вас действия кажутся слишком рискованными и отчаянными? Перечислите их.

Шаг 2. Какие еще действия, по вашей оценке, могут быть слишком смелыми и рискованными?

Шаг 3. Выберите наименее рискованный поступок. Можно ли каким-то образом снизить фактор риска? Напишите как. Сделайте то же самое с оставшимися рискованными поступками. Сопоставьте все рискованные поступки с действиями, которые помогут избежать их.

Шаг 4. Какие цели привлекают вас, но их достижение кажется слишком рискованным?

Повторите еще раз шаг 3 для этих целей.

Упражнение № 9. Самая подходящая профессия для вашей карьеры.

Цель: помощь в выборе профессии.

Часть А. Задавайте вопросы о карьере. У вас должно появиться представление о работе, которая является средством достижения карьерных целей.

Следующие вопросы помогают определить, какая профессия вам нужна:

1. В компании какого типа вы хотели бы работать? Вы можете назвать конкретную фирму?

2. Какой диплом, сертификат, навыки вам понадобятся?

3. Какие специальные знания и опыт необходимы? Какие способности?

4. Какие люди могут рассказать вам больше о работе, о том, что вам нужно знать, как преподнести себя, как завоевать доверие в компании? Познакомьтесь с людьми, которые могут вам помочь.

5. Какая работа нужна вам, чтобы подготовить себя к достижению карьерных целей? Какой опыт (в каких сферах деятельности) вам необходим? Как эти сферы связаны друг с другом?

6. Имея карьерный план, определите, кто может дать объективную оценку его эффективности.

Часть В. Обсуждение с супругом (ой)/ партнером.

Обсудите с супругой (супругом) или своим партнером, близким другом (противоположного пола) ваши и ее (его) карьерные цели и способы их достижения.

Часть С. Взаимная поддержка.

Обсудите ваши карьерные цели с друзьями.

Часть D. Сделайте свои карьерные цели заметными.

Найдите способ сделать так, чтобы ваши цели были всегда у вас на виду:

1. Выберите три главные цели. Запишите цели и даты их достижения на карточке. Поместите карточку на видное место.

2. Нарисуйте на карточке цветными фломастерами яркие символы своих основных целей. Положите карточку на видное место.

Упражнение № 10. План действий на месяц.

Цель: подготовка плана действий, который поможет сосредоточиться на приоритетных целях в текущем месяце.

Шаг 1. Назовите свою главную карьерную цель. Рядом в порядке важности напишите основные действия, даты их совершения в этом месяце. Самый важный поступок запланируйте на сегодня и не вычеркивайте его из списка пока не осуществите. Если вы в течение 7 дней его не осуществили, оцените снова свои цели и действия.

Шаг 2. Сделайте то же самое со второй и третьей карьерными целями.

Шаг 3. Проделайте то же самое со своими целями, касающимися личной жизни и саморазвития.

Упражнение № 11. Однолетний план действий.

Цель: разработать годовой план, позволяющий сосредоточиться на приоритетных целях.

Шаг 1. Перечислите 3 главные карьерные цели, которые вы намерены достичь в течение года. Отметьте даты их достижения.

Шаг 2. То же самое проделайте для целей, касающихся саморазвития.

Шаг 3. Повторите данную процедуру для целей в личной жизни.

Упражнение № 12. Пятилетний план действий.

Цель: разработать пятилетний план.

Шаг 1. Перечислите 3 карьерные цели, которые Вы планируете достичь за пять лет. Укажите даты их реализации.

Шаг 2. То же самое проделайте для целей, касающихся саморазвития.

Шаг 3. Повторите данную процедуру для целей в личной жизни.

Методические рекомендации

Предложенные задания носят индивидуальный характер, выполняются последовательно и самостоятельно. Задание 1 может выдаваться на самостоятельную работу (в рамках подготовки к практическому занятию). На занятии преподаватель организует выполнение упражнений и оказывает индивидуальную помощь студентам в осознании жизненных и профессиональных целей посредством ответов на вопросы студентов или наводящих вопросов. Результаты выполнения заданий доступны только самому магистранту (не сдаются преподавателю). Для контроля выполненной работы допускается запросить у магистрантов результаты упражнений 10, 11 или 12.

Литература

Киселева Е.В. Планирование и развитие карьеры: учебное пособие для студентов высших учебных заведений. – Вологда: Легия, 2010.

Лызь Н.А., Лызь А.Е. Управление личностными ресурсами: образование и профессиональное развитие: учебное пособие. – Таганрог: Изд-во ЮФУ, 2016.

Митина Л.М. Психология личностно профессионального развития субъектов образования. – М.; СПб, 2014.

Практикум по психологии менеджмента и профессиональной деятельности / под ред. Г. С. Никифорова, М. А. Дмитриевой, В. М. Снеткова. – СПб.: Речь, 2001.

Занятие 6. Субъектные качества как ресурсы личностно-профессионального развития (Часть 1. Субъективный контроль и рефлексивность)

Цель: повышение готовности магистрантов к использованию своих ресурсов в учебно-профессиональной деятельности и последующем профессиональном развитии.

Задачи:

удовлетворение познавательных интересов магистрантов в сфере знаний о человеке вообще и о себе, в частности;

развитие представлений об уровне сформированности собственных субъектных качеств;

развитие ценностного отношения к личности;

развитие аналитико-синтетических, исследовательских, практических психологических умений, навыков самопознания и самоанализа.

Вопросы для собеседования:

1. Почему формирование компетентного профессионала не ограничивается профессиональным обучением, а требует развития личности?
2. Как Вы понимаете непрерывность образования? Охарактеризуйте суть тезиса «от одного образования на всю жизнь – к образованию через всю жизнь».
3. По мнению О.В. Долженко, «акт образования станет возможным, если я сам являюсь носителем некой идеи образованности. Тогда для меня выявится сущность образования. Благодаря моей идее обретет смысл реальность, называемая системой образования». Какова ваша идея образованности? Какой смысл имеет для вас образование? Раскройте возможности высшего образования в реализации вашей идеи образованности.
4. Насколько качество получаемого вами образования зависит от вас самих? Являетесь ли вы субъектом учебно-профессиональной деятельности?

Теоретическое введение

Материалы лекции 3.

Главная задача высшего образования – личностно-профессиональное развитие студента, которое осуществляется в условиях учебно-профессиональной деятельности. Выполняя ее, студент не только проявляет свои личностные свойства, способности, качества, но и изменяется как ее субъект. С другой стороны, сформированные субъектные качества проявляются в разных видах деятельности, включая учебную, определяя ее качество и интенсивность. Данное занятие направлено на диагностику уровня сформированности основных субъектных качеств и рефлексию полученных

результатов с целью выработки понимания перспектив дальнейшего саморазвития.

Субъектные качества позволяют личности выполнять функции самоорганизации, саморазвития и управления жизнедеятельностью в целом. К ним относятся активность, саморегуляция, рефлексивность, ответственность, ценностные ориентации. Благодаря этим качествам субъект способен эффективно выполнять разные виды деятельности, решать задачи личностного и профессионального саморазвития.

Субъект не просто адаптируется к требованиям учебной, профессиональной или иной деятельности и следует за складывающимся стечением обстоятельств. Обладая устойчивыми внутренними ресурсами, он способен преобразовывать и эти обстоятельства, и деятельность, и самого себя *(Лызь, Лызь, 2016)*.

Важнейшей характеристикой субъекта является его активность – это своеобразная мера субъектности во взаимодействии человека с миром, внутренний источник детерминации поведения, обеспечивающий целостность и развитие субъекта *(Волочков, 2007)*. Активность субъекта связана и проявляется в постоянном разрешении противоречий между той сложной системой, которую представляет он сам, включая его цели, и объективными (социальными, природными, техническими и др.) системами. Активность вообще может стать субъектной активностью только в случае принятия субъектом ответственности за свою жизнь и свою профессиональную карьеру, в случае восприятия себя как причины происходящего. В широком смысле ответственность есть добровольное принятие и осуществление необходимости, требований жизни. Взять на себя ответственность значит самостоятельно, т. е. рассчитывая на свои силы, ставить и решать различного рода задачи (профессиональные, коммуникативные и т. д.). Момент добровольности превращает личность в субъекта, который сам, по своей воле, самостоятельно, преодолевая трудности, берется за реализацию необходимого.

Наиболее легко ответственность как субъектная личностная характеристика обнаруживается в ситуациях преодоления преград, в трудных для личности условиях. Именно в таких ситуациях актуализируются все сущностные признаки ответственности: готовность к преодолению трудностей, определение собственных ресурсов, возможностей для преодоления, механизмы контроля, осознанное предвосхищение последствий ситуации, следовательно, обнаруживается истинная, подлинная ответственность как способ саморегуляции и самоконтроля *(Дементий)*. Ответственность непосредственно связана с такими фундаментальными личностными качествами, как уверенность, самостоятельность, способность к самоконтролю. Одним из важнейших качеств ответственной личности является обращенность требований к самой себе, а не к окружающим. Эмпирическим аналогом ответственности можно рассматривать интернальный локус контроля, т.е. восприятие себя как причины происходящего.

Процесс становления и развития ответственности неразрывно связан с развитием рефлексии и определяется наличием рефлексивной позиции субъекта жизнедеятельности.

В психологической науке рефлексия – это процесс самопознания субъектом внутренних психических актов и состояний. Рефлексивность раскрывается А.В. Карповым как качество субъекта, с помощью которого он способен как формировать определенные качества своей самости, так и распознать те качества, которые не принадлежат ему как субъекту *(Карпов, 2003)*. Рефлексия деятельности субъекта рассматривается в ее трех формах по «временному» принципу: ситуативная (актуальная), ретроспективная и перспективная рефлексия. Ситуативная рефлексия обеспечивает непосредственный самоконтроль поведения человека в актуальной ситуации, осмысление ее элементов, анализ происходящего. Ретроспективная рефлексия проявляется в склонности к анализу уже выполненной в прошлом деятельности и свершившихся событий. В этом случае предметами рефлексии являются мотивы и причины произошедшего, содержание прошлого поведения, его результаты, допущенные ошибки. Перспективная рефлексия выражается в тщательности обдумывания и планирования человеком своего будущего поведения.

По мнению ученых, рефлексия выступает значимым компонентом самопонимания, результатом которого является объяснение человеком своих чувств, мотивов, умение обнаруживать смысл поступков. Рефлексивные свойства характера связаны с целями жизни и деятельности, ценностными ориентациями, установками, способствуя образованию и стабилизации единства личности, выполняя функцию саморегуляции, которая является одним из важнейших и принципиально необходимых психических механизмов реализации внутренне детерминированной активности субъекта, в том числе и субъекта учебно-профессиональной деятельности.

Практикум

Задание 1. Определение локуса контроля

Задание позволяет выявить убеждения человека относительно того, где локализуются силы, подвергающие влиянию, управлению и контролю то, что происходит в его жизни. Лица, убежденные в том, что главные силы, определяющие их жизнь, находятся внутри них самих (это их усилия, старания и способности), называются интерналами, или людьми с внутренним локусом контроля (ЛК). Лица, убежденные в том, что происходящее с ними зависит от внешних факторов (от других людей, от судьбы или случая), называются экстерналами, или людьми с внешним локусом контроля. Чем выше показатели интернальности, тем более вероятно, что человек чувствует себя хозяином собственной судьбы, отличается уверенностью в себе и обладает более высоким уровнем развития саморегуляции жизнедеятельности. Чем ниже показатели интернальности, т. е. чем ближе человек к полюсу внешнего локуса контроля, или экстернальности, тем менее человек уверен в себе и больше

нуждается в помощи, являясь недостаточно самостоятельным в решении различного рода жизненных задач.

Для определения собственного локуса контроля ответьте на вопросы опросника, подсчитайте результаты, проанализируйте общую интернальность и ее аспекты.

Методика «Локус контроля» (Е.Г. Ксенофонтова). См. https://psytests.org/personal/usk.html *(Можно заменить методикой «Уровень субъективного контроля»)*

Задание 2. Решение кейсов по развитию ответственности сотрудников организации

Кейс 1

Андрей Петров три месяца назад получил должность начальника отдела маркетинга в крупной производственной компании. О компании отзывается, как об авторитарной. Его непосредственный руководитель (управляющий директор) требует много, сам договоренностей не соблюдает, совещания проводит только с целью выговоров и наказаний. Подчиненные в его отделе привыкли делать только то, что говорил им прежний руководитель, к новому руководителю относятся с подозрением, инициативы не проявляют. Когда он написал план работы отдела на год, его раскритиковали и сказали, что он не умеет работать. По личностным характеристикам Андрея Петрова можно охарактеризовать как уверенного, амбициозного и общительного человека, который хочет развиваться и хорошо зарабатывать.

Вопросы к кейсу

За что в данной ситуации несет ответственность Андрей Петров?

За что он точно не несет ответственность?

Какие меры он может предпринять по повышению ответственности персонала?

Кейс 2

У вас в отделе проходит довольно важное совещание, которое ведет руководитель. Вдруг у него звонит мобильник, он слушает и говорит вам: «Меня вызывает генеральный, вы должны без меня принять решение». Руководитель уходит. Через минут пятнадцать – двадцать обсуждения становится ясно, что дискуссия стала хаотичной, зашла в тупик.

Вопросы к кейсу:

Ваша реакция.

Назовите три плюса и три минуса того, что эффективность руководителя оценивается не по его личным результатам, а по результатам деятельности всего его коллектива в целом.

Кейс 3

Из-за одного из разработчиков, который по личным причинным «халтурил» в течение последнего квартала, весь отдел не выполнил план.

Вопросы к кейсу:

Проанализируйте данную ситуацию с позиции виновности и наказания всех участников данной ситуации.

Как вы считаете, почему одни отделы выполняют плановые показатели, у них хорошая слаженная команда, а в других подразделениях планы не выполняются и текучка кадров?

Кейс 4

Ваш подчиненный систематически вас переспрашивает о том, как делать ту или иную работу, но в итоге делает все хорошо.

Вопросы к кейсу:

Чем может быть вызвана такая ситуация (как можно больше вариантов).

Какие могут быть ваши дальнейшие действия.

Задание 3. Определение типа рефлексии

Ответьте на вопросы опросника, подсчитайте результаты, проанализируйте собственную рефлексивность.

Опросник «Дифференциальный тип рефлексии» (Д.А. Леонтьев, Е.М. Лаптева, Е.Н. Осин, А.Ж. Салихова) (Можно заменить опросником рефлексивности Карпова) https://psytests.org/personal/reflexion.html)

Задание 4. Развитие профессиональной рефлексии

Упражнение «Хорошо-плохо»

Цель: осознание своей профессиональной деятельности, необходимости и важности своей работы.

Описание упражнения: участники по очереди говорят, например, «системным программистом быть хорошо, потому что…», приводя аргументы своему высказыванию. Затем, следующий говорит «системным программистом быть плохо, потому что…», также приводя свои аргументы. И т.д., упражнение необходимо закончить на высказывании «системным программистом быть хорошо, потому что…».

Методические рекомендации

Предложенные диагностические задания (1 и 3) носят индивидуальный характер, выполняются самостоятельно (возможно выдача заданий на самостоятельную работу в рамках подготовки к занятию). При необходимости преподаватель оказывает помощь студентам в осмыслении полученных результатов. В группе обсуждаются следующие вопросы:

Являются ли полученные результаты устоявшимися характеристиками личности или возможна работа по совершенствованию субъектных качеств?

Был ли у кого-нибудь позитивный опыт по развитию данных качеств?

Литература

Брушлинский А.В. Психология субъекта и его деятельности / Современная психология: Справочное руководство. – М.: ИНФРА-М, 1999. С. 330 – 346.

Волочков А.А. Активность субъекта бытия: Интегративный подход. – Пермь, 2007. – 376 с.

Дементий Л.И. Ответственность как способ жизни [Электронный ресурс] http://psy.tsu.ru/data/pdf/1_17.pdf

Карпов А.В. Рефлексивность как психическое свойство и методика ее диагностики // Психологический журнал. 2003. №5. С. 45–57.

Ксенофонтова Е.Г. Исследование локализации контроля личности – новая версия методики «уровень субъективного контроля» // Психол. журнал. 1999. Т.20. №2. С.103–114.

Леонтьев Д.А., Осин Е.Н. Рефлексия «хорошая» и «дурная»: от объяснительной модели к дифференциальной диагностике // Психология. Журнал Высшей школы экономики. 2014. Т. 11. № 4. С. 110–135.

Занятие 7. Субъектные качества как ресурсы личностно-профессионального развития (Часть 2. Саморегуляция и жизнестойкость)

Цель: повышение готовности магистрантов к использованию и развитию своих ресурсов в учебно-профессиональной деятельности и последующем профессиональном становлении.

Задачи:

удовлетворение познавательных интересов магистрантов в сфере знаний о человеке вообще и о себе, в частности;

развитие представлений об уровне сформированности собственных субъектных качеств, о способах саморегуляции и самоконтроля;

развитие умений определять приоритеты деятельности на основе понимания ценностей и целей, а также определять пути совершенствования деятельности;

развитие навыков оценки результатов собственной деятельности.

Теоретическое введение

К важным субъектным качествам также относятся саморегуляция и жизнестойкость.

Саморегуляция генетически выступает как специфическая для человека наиболее общая функция его психики, позволяющая человеку реализовать себя как творца, исполнителя и контролера своей деятельности, поступков, жизнедеятельности в целом. Осознанная саморегуляция выступает как системно организованный процесс внутренней психической активности человека по инициации, построению, осуществлению, поддержанию и управлению всеми видами и формами активности, которые направлены на достижение принимаемых субъектом целей *(Моросанова, Аронова, 2007)*.

Саморегуляцию учебной деятельности можно определить как осознанную активность студента по инициации, построению, поддержанию и управлению учебной деятельностью, обеспечивающую достижение принимаемых им учебных целей.

Система саморегуляции имеют свою структуру, компоненты которой выполняют различные функции в осознанном управлении деятельностью. Это такие компоненты как планирование (целеполагание), моделирование условий, программирование действий, оценка результатов. Процесс саморегуляции способствует выработке гармоничного поведения, на его основе развивается способность управлять собой сообразно реализации поставленной цели, направлять свое поведение в соответствии с требованиями жизни и профессиональными или учебными задачами.

Индивидуальные особенности саморегуляции:

1) Индивидуальные особенности планирования целей. Они описывают индивидуальные различия в выдвижении, принятии, удержании целей. Различия в планировании целей связаны с разной активностью выдвижения

целей, адекватностью этого процесса внешним и внутренним субъектным условиям, иерархичностью целей.

2) Особенности моделирования, т.е. анализа внешних и внутренних условий деятельности и выделения комплекса условий, значимых для достижения цели. Модель значимых условий выполняет в психической регуляции деятельности функцию источника информации об условиях, учет которых необходим для определения программы реализации деятельности.

3) Особенности программирования предстоящих исполнительских действий, необходимых для достижения поставленной цели. В функции программирования входит антиципация компонентного состава предстоящих действий, способов, которыми они будут осуществляться, и собственно последовательности осуществления планируемых действий.

4) Особенности контроля, оценивания и коррекции своей активности. Эти регуляторные процессы пронизывают весь процесс саморегуляции, так как на каждой стадии достижения цели происходит контроль актуального состояния системы и результатов действий путем их сличения с прогнозируемыми параметрами, оценка рассогласования и принятие решения о коррекции исполнительских (управляющих) действий или о переходе к следующей стадии реализации деятельности.

Личностно-регуляторные свойства:

адекватность (условий деятельности субъективно принимаемой модели условий, программы, способов контроля, критериев успешности и других блоков регуляции);

осознанность (представлений об условиях и о программе действий, о контролируемых параметрах, о критериях успешности и т.д. в соответствии с их соотносительной значимостью для достижения цели);

гибкость (процесса регуляции, возможность внесения коррекций в функционирование различных регуляторных блоков, когда этого требуют условия деятельности);

надежность и устойчивость (функционирования регуляторных блоков и их структуры в условиях психической напряженности).

Жизнестойкость представляется как своеобразный «стержень» субъекта, его основа, которая обеспечивает устойчивость ценностно-смысловой сферы и способность преодолевать трудные жизненные ситуации, что является одним из важнейших факторов эффективности любой деятельности, в том числе и учебно-профессиональной. Жизнестойкость характеризует меру способности личности выдерживать стрессовую ситуацию, сохраняя внутреннюю сбалансированность и не снижая успешность деятельности. С. Мадди включает в понятие жизнестойкости три сравнительно автономных компонента: вовлеченность, контроль, принятие риска.

Выраженность жизнестойкости в целом и ее компонентов препятствует возникновению внутреннего напряжения в стрессовых ситуациях за счет стойкого совладания со стрессами и восприятия их как менее значимых *(Леонтьев, Рассказова, 2006)*.

Вовлеченность определяется как «убежденность в том, что вовлеченность в происходящее дает максимальный шанс найти нечто стоящее и интересное для личности». Человек с развитым компонентом вовлеченности получает удовольствие от собственной деятельности. В противоположность этому, отсутствие подобной убежденности порождает чувство отвергнутости, ощущение себя «вне» жизни. Если вы чувствуете уверенность в себе и в том, что мир великодушен, вам присуща вовлеченность.

Контроль представляет собой убежденность в том, что борьба позволяет повлиять на результат происходящего, пусть даже это влияние не абсолютно и успех не гарантирован. Противоположность этому – ощущение собственной беспомощности. Человек с сильно развитым компонентом контроля ощущает, что сам выбирает собственную деятельность, свой путь (аналог интернального локуса контроля).

Принятие риска – убежденность человека в том, что все то, что с ним случается, способствует его развитию за счет знаний, извлекаемых из опыта, – неважно, позитивного или негативного. Человек, рассматривающий жизнь как способ приобретения опыта, готов действовать в отсутствие надежных гарантий успеха, на свой страх и риск, считая стремление к простому комфорту и безопасности обедняющим жизнь личности. В основе принятия риска лежит идея развития через активное усвоение знаний из опыта и последующее их использование.

Компоненты жизнестойкости развиваются в детстве и отчасти в подростковом возрасте, хотя их можно развивать и позднее. Их развитие решающим образом зависит от отношений родителей с ребенком. В частности, для увеличения вовлеченности принципиально важно принятие и поддержка, любовь и одобрение со стороны родителей. Для развития компонента контроля важна поддержка инициативы ребенка, его стремления справляться с задачами все возрастающей сложности на грани своих возможностей. Для развития принятия риска важно богатство впечатлений, изменчивость и неоднородность среды.

Задание 1. Определение стиля саморегуляции поведения

Ответьте на вопросы опросника, проанализируйте уровень развития осознанной саморегуляции и ее индивидуальный профиль. Методика «Стиль саморегуляции поведения» В.И. Моросановой. https://psytests.org/personal/sspm.html

Задание 2. Развитие саморегуляции

Упражнение «Борьба мотивов»

Зачастую в нас борются противоречивые импульсы и желания. Этот феномен известен в психологии под названием «борьба мотивов». Иногда нам даже трудно понять, чего мы больше хотим. Например, после окончания магистратуры пойти в аспирантуру или поскорее начать зарабатывать и начать взрослую самостоятельную жизнь; начать работу над творческим проектом или пойти с друзьями на футбол. Магистрантам, как будущим специалистам, необходима способность к саморегуляции и умение принимать ответственные

решения в условиях борьбы мотивов. Данное упражнение поможет научиться согласовывать противоборствующие мотивы.

Цель: анализ и согласование собственных противоречивых мотивов.

Инструкция: предлагается сейчас каждому вспомнить ситуацию, в которой вы испытывали борьбу мотивов. Вы хотели сделать что-то важное, но не делали этого, потому что одновременно вам хотелось чего-то другого. Постарайтесь вспомнить такую ситуацию. Будет даже лучше, если эта ситуация актуальна для вас сейчас: хотите сделать что-то, но при этом вам хочется чего-то другого.

Далее предлагается озвучить ситуации. Затем выбрать ту ситуацию, которая кажется наиболее перспективной для анализа, и предложить группе рассмотреть ее по следующим вопросам:

1. Хочу ли я достичь своей цели?
2. Хочется ли мне сейчас этим заниматься?
3. Что мне хочется делать?
4. Как соединить это с тем, что я решил(а) сделать?

Например: А. решила переосмыслить, переписать и довести до сведения своего персонала новые должностные обязанности. Однако она так устала, что ей хочется одного – спать. В результате она не может ни писать инструкции, ни спать.

1. Хочу ли я достичь своей цели?

В принципе, да. Это моя цель, я сама ее перед собой поставила.

2.Хочется ли мне сейчас этим заниматься?

Сейчас мне этим заниматься не хочется.

3.Что мне хочется делать?

Мне хочется выспаться, а потом уехать в отпуск. Ну, хотя бы выспаться. Для этого мне дня два нужно валяться в постели.

4.Как соединить это с тем, что я решила сделать?

На выходных, лежа в постели в постели, на диктофон записывать свои мысли об инструкции!

Результатом упражнения должно быть ощущение возможности и полезности соединения противоречивых стремлений вместо борьбы с одним из них.

Упражнение «План реализации цели»

Цель: научиться планировать свои действия по достижению конкретной жизненной цели.

Инструкция: Составьте черновик пошагового плана для достижения конкретной цели. Начните с конечного результата, а потом шаг за шагом спланируйте весь путь, включая то, что вы можете сделать по этому плану уже сегодня. При возникновении сложностей, можно начать с того, чтобы решить: что самое первое надо сделать, чтобы ее достичь? Если у вас нет уверенности в том, каким должен быть этот план, спросите себя вновь: что мешает уже сегодня иметь все то, к чему я стремлюсь. Возможно, ответом на этот вопрос будет что-то, над чем уже сейчас можно работать, чтобы изменить ситуацию.

Решив такую промежуточную задачу, вы сможете приблизиться к достижению вашей главной цели.

Обсуждение:

Что нового и полезного вы для себя вынесли из этих упражнений?

Какие выводы вы сделали о себе?

Будете ли вы использовать полученный опыт в повседневной жизни?

Упражнение «Психологическое время»

Цель: развитие внутреннего «чувства времени», осознание различий между людьми, для которых психологическое время течет быстрее либо медленнее реального.

Описание упражнения.

Участников просят сесть, расслабиться и закрыть глаза. Их задача – постараться как можно точнее определить, когда пройдет ровно минута после условного сигнала ведущего. Считать про себя до шестидесяти нельзя, надо просто прислушаться к своей интуиции. Когда, с точки зрения участника, минута истекла, он молча поднимает руку, но глаза не открывает. Ведущий фиксирует, сколько времени реально прошло до этого момента (при большом количестве участников ему понадобится помощь одного – двух ассистентов). Потом участники делятся на две группы: тех, кому показалось, что минута прошла быстрее, чем в реальности (их время меньше 60 секунд), и тех, кому показалось, что она шла дольше. Внутри этих подгрупп обсуждается, каковы особенности людей, психологическое время которых течет по сравнению с реальным «быстрее» или «медленнее». Как правило, те, для кого время течет быстрее, существуют по принципу «и жить торопится, и чувствовать спешит», те же, кому ход времени представляется существенно медленнее, обычно и жить предпочитают размеренно и неспешно. Кроме того, восприятие времени зависит от состояния, в котором находится человек: когда он очень активен, возбужден, то кажется, будто время ускоряет свой ход.

Обсуждение: представители от каждой из подгрупп характеризуют людей, для которых психологическое время в данный момент отличается от реального. Затем остальные участники сравнивают эти точки зрения и высказывают свои соображения насчет причин такого различия.

Задание 3. Способы саморегуляции и самоконтроля

Даны 4 проблемы саморегуляции деятельности:

Расстановка приоритетов

Распределение времени (тайм-менеджмент)

Самодисциплина (преодоление лени, усталости)

Нестабильность среды (изменение обстоятельств).

Подумайте, в чем суть конкретной проблемы планирования и организации деятельности (что именно препятствует эффективной организации деятельности). Предложите способы саморегуляции и самоконтроля в контексте решения конкретной проблемы и рекомендации по повышению эффективности деятельности. Группа делится на 4 команды, каждая команда за 10 минут должна найти решение и представить его группе.

Задание 4. Определение уровня жизнестойкости

Ответьте на вопросы опросника, проанализируйте уровень жизнестойкости и выраженность ее компонентов.

Тест жизнестойкости Мадди (в адаптации Д. А. Леонтьева и Е. И. Рассказовой) https://psytests.org/personal/hardinessB.html

Задание 5. Развитие жизнестойкости путем формирования гибкости в оценке жизненных событий

Цель: формирование умений находить в жизненных событиях как положительных, так и отрицательных сторон; осознание того, что оценка ситуации зависит не столько от жизненных событий как таковых, сколько от того, как мы их воспринимаем.

Описание упражнения.

Предлагается придумать и кратко, одной – двумя фразами описать примеры двух жизненных событий – хорошего и плохого. Потом участники объединяются в команды по 3 – 4 человека, в них обмениваются своими примерами и находят в каждом из этих событий по три положительных момента и по три отрицательных. Это должно касаться и хороших, и плохих событий. Так, например, положительное событие «внезапно разбогатеть» может иметь отрицательным следствием ухудшение отношений с окружающими людьми (т.к. они начнут завидовать), а отрицательное событие «заболеть» – дать возможность прочитать интересную книгу, на которую раньше не хватало времени, отдохнуть от суеты и т. п. После обсуждения (через 10 – 15 мин.) представители от каждой из команд озвучивают примеры событий, а также положительных и отрицательных следствий каждого из них.

Вопросы для обсуждения:

К чему приводит «однобокость» мировосприятия?

Как ее можно преодолеть?

Каким образом приобретение гибкого взгляда на жизненные события способствует повышению жизнестойкости?

Вопросы для собеседования:

1. Какие качества лежат в основе готовности профессионала проявлять инициативу, действовать в нестандартных ситуациях, нести ответственность за принятые решения?
2. Какие факторы влияют на принятие ответственных/безответственных решений в профессиональной деятельности?
3. Какие внешние и собственные внутренние ресурсы вы считаете наиболее значимыми в плане личностно-профессионального развития в образовании?
4. Какие субъектные качества играют наиболее важную роль в управлении своим образованием и профессиональным развитием?

Методические рекомендации

Предложенные диагностические задания (1 и 4) носят индивидуальный характер, выполняются самостоятельно (возможно выдача заданий на самостоятельную работу в рамках подготовки к занятию). При необходимости

преподаватель оказывает помощь студентам в осмыслении полученных результатов. В группе обсуждаются следующие вопросы:

Являются ли полученные результаты устоявшимися характеристиками личности или возможна работа по совершенствованию субъектных качеств?

Был ли у кого-нибудь позитивный опыт по развитию данных качеств?

Литература

Карпинский К.В. Человек как субъект жизни. – Гродно: ГрГУ, 2002.

Леонтьев Д.А, Рассказова Е.И. Тест жизнестойкости. – М.: Смысл, 2006.

Моросанова В.И., Аронова Е.А. Самосознание и саморегуляция поведения. – М.: Изд-во «Институт психологии РАН», 2007. – 213 с.

Осницкий А.К., Бякова Н.В., Истомина С.В. Развитие саморегуляции на разных этапах профессионального становления // Вопросы психологии. 2009. № 1. С. 3–12.

Занятие 8. Образование, самообразование и саморазвитие профессионала (Часть 1. Образование и самообразование)

Цель: повышение готовности магистрантов к построению собственной образовательной траектории.

Задачи:

способствовать пониманию основных приоритетов и механизмов образования, осознанию полученного опыта на предыдущем этапе образования, повышению осмысленности дальнейшего обучения;

развитие умений анализировать собственные возможности и ограничения, ставить задачи и проектировать процесс профессионально-личностного саморазвития;

развитие умений анализировать и выбирать целесообразные формы, методы и средства самообразования.

Теоретическое введение

Материалы лекции 3.

Вопросы для собеседования:

1. Каковы, на ваш взгляд, должны быть ценностные приоритеты образования, чтобы оно могло обеспечивать устойчивое развитие общества?
2. Охарактеризуйте высшее образование, отвечающее запросам современного общества. Что в нем необходимо изменить? Что уже изменилось за время вашего обучения в вузе?
3. Почему для личностного развития студентов необходим значимый и самоуправляемый характер выполняемой ими деятельности?
4. Какие организационные условия обучения в вузе обеспечивают индивидуализированный характер образования и удовлетворение образовательных потребностей студентов?
5. Для чего необходима академическая мобильность студентов? Как она реализуется?
6. Что такое кредитно-модульная система организации образовательного процесса? Какие функции она выполняет?
7. В чем заключается отличие инновационных образовательных технологий от традиционных?
8. Что такое самоуправляемая учебная деятельность? Почему она является условием формирования профессиональной компетентности?

Практикум

Задание 1. Рефлексия опыта

Ответьте письменно на 3 вопроса: Какого опыта, полученного на первом уровне высшего образования, недостаточно для успешной профессиональной самореализации? В чем заключается специфика обучения в магистратуре по

сравнению с первым уровнем высшего образования? Как сделать свое образование индивидуализированным?

Задание 2. Интернет-ресурсы в образовательной траектории.

Подумайте о вашей дальнейшей образовательной траектории. Познакомьтесь с Федеральным порталом «Моё образование». Составьте перечень онлайн-курсов и других ресурсов интернет-пространства, которые могут способствовать вашему профессионально-личностному развитию. Представьте публично идею и основные ресурсы своей будущей образовательной траектории.

Задание 3. Готовность к самообразованию

Какие составляющие входят в готовность человека проектировать и реализовать собственную образовательную траекторию? Групповое составление и обсуждение перечня составляющих готовности. Оценка каждым магистрантом уровня сформированности каждой составляющей и в целом готовности проектировать и реализовать собственную образовательную траекторию.

Методические рекомендации

Задание 1 целесообразно выполнять с использованием интерактивных инструментов (Jamboard Google, Mentimeter и др.) для сбора максимального количества идей. В общем обсуждении выявляются ключевые идеи и обсуждается самонаправленный и непрерывный характер образования, а также взаимосвязь обучения в магистратуре и самообразования с жизненными и профессиональными целями.

При выполнении задания 2 на этапе публичного представления целесообразно предложить поделиться самыми интересными образовательными и развивающими ресурсами интернет-пространства.

Литература

Лызь Н.А., Лызь А.Е. Управление личностными ресурсами: образование и профессиональное развитие: учебное пособие. – Таганрог: Изд-во ЮФУ, 2016.

Федеральный портал «Моё образование» https://online.edu.ru/

Открытые образовательно-развивающие интернет-ресурсы.

Занятие 9. Образование, самообразование и саморазвитие профессионала (Часть 2. Саморазвитие)

Цель: повышение готовности магистрантов к саморазвитию.

Задачи:

формирование рефлексивной жизненной позиции, развитие представлений о собственных жизненных, профессиональных, образовательных целях, возможностях и ресурсах;

развитие умений анализировать собственные возможности и ограничения, ставить задачи и проектировать процесс профессионально-личностного саморазвития;

развитие умений анализировать и выбирать целесообразные формы, методы и средства саморазвития.

развитие навыков самопознания и самоанализа.

Теоретическое введение

Материалы лекции 4.

Вопросы для собеседования

1. Охарактеризуйте собственную образовательную и профессиональную компетентность. В каких составляющих обнаруживается недостаточный уровень? Почему? Как можно развивать свою компетентность?
2. Какие методы и средства вы используете для самообразования, развития мировоззрения, общей и профессиональной культуры? Какие из них наиболее эффективны?
3. Какие функции и задачи профессионального образования позволяют решить информационные и коммуникационные технологии?
4. Каковы возможности и ограничения информационных технологий обучения?
5. Каковы возможности проектного метода обучения? В чем заключаются трудности реализации технологии проектов? Каковы ее ограничения?
6. Какие трудности могут возникнуть при организации групповой деятельности, при руководстве деятельностью коллектива (обучающихся / подчиненных)? Как их можно преодолеть?
7. В чем заключается различие между развитием профессионала и его саморазвитием?
8. Какие виды активности личности могут относиться к саморазвитию? Приведите примеры из собственного опыта.
9. Почему саморазвитие опирается на самопознание и осознание жизненных целей?
10. Почему позитивное самоотношение и вера в себя являются условиями профессионального развития и саморазвития?

11. Какие характеристики личности наиболее важны в инициации саморазвития?
12. Почему с течением времени высшее образование все больше интегрируется с неформальным и информальным?

Практикум

Задание 1. Определение принципов саморазвития

Ниже приведены рекомендации по построению успешной карьеры и счастливой жизни. Познакомьтесь с ними. С чем вы согласны, а с чем вам хотелось бы поспорить? Известны ли вам еще какие-либо принципы развития и советы по этому вопросу?

Ступени лестницы, ведущие к новой карьере (С. Маркус, Дж. Фридленд):

1. Повышайте свою ценность как работника, показывая, каким образом компания могла бы с вашей помощью решить собственные задачи, особенно касающиеся прибылей.

2. Постоянно ищите новые и более выгодные способы повысить свою ценность в глазах работодателя.

3. Не держите себя в «информационном вакууме».

4. Упреждайте, а не реагируйте.

5. Постоянно ищите возможности повысить образование.

6. Ставьте себе высокие цели относительно карьеры и материального положения и детально продумывайте планы, которые позволят вам достичь задуманного.

7. Избегайте отрицания (реакции типа «нет»).

8. Сейчас, на своем рабочем месте, подготовьтесь к выживанию и к переходу на новую работу или на следующую ступень в карьере.

9. Мотивируйте своими целями, а не обидой, страхом или отчаянием.

10. Будьте напористы, стремясь продать себя.

11. Пусть растет ваша мотивация и количество обязательств, которые вы можете принять на себя.

12. Рассматривайте собственные недостатки и слабости в разумной перспективе.

13. Поймите, что в современном мире, чтобы выжить и идти вперед, диктовать, как Вы будете меняться, должна ваша основная работа.

Основные навыки достижений (Н. Карр-Руфино):

1. Определение своего собственного успеха. Нужно рассматривать успех как необходимость, а не случайность в своей жизни. Важно понять, что даже неудачи тоже есть успех, т. к. учат нас на пути к успеху. Нужно сосредоточиться на риске, стремлении, творчестве, решении проблем и особенно на обучении. Формулировка вашего собственного успеха – ответ на вопрос: «Кто вы?».

2. Отождествление себя с успешными людьми. Если вы испытываете негативные чувства (зависть, недовольство и т. п.) по отношению к успешным людям, то эти чувства отделяют вас от успеха. Вы не хотите, чтобы вам

напоминали, что вы не рискуете и ни к чему не стремитесь. Отождествление себя с успешными людьми приближает успех.

3. Преодоление страхов. Важно избавиться от страха перед неудачами, неуспехом и риском. Лучший способ избавиться от страха – определить его причину.

4. Использование своих внутренних ресурсов лидера. Представьте в своем воображении то, что вы хотите создать. Наши желания мотивируют нас и составляют основу наших целей и стремлений.

«Заповеди» счастливой жизни (Джо Витале). Пообещайте себе:

Быть сильным, чтобы ничто не смогло нарушить ваше душевное спокойствие.

Желать здоровья, счастья и процветания каждому человеку, которого вы встречаете на своем пути.

Дать почувствовать каждому из ваших друзей, что в нем есть что-то особенное.

Искать светлую сторону в любом событии, и сделать свой оптимизм реальностью.

Думать только о лучшем, работать только для лучшего и ожидать только лучшего.

Воспринимать успех других с не меньшей радостью, чем свой собственный.

Забыть об ошибках прошлого и сосредоточиться на великих достижениях в будущем.

Уделять собственному совершенствованию так много времени, чтобы его не оставалось на критику окружающих.

Быть слишком свободным, чтобы волноваться, слишком великодушным, чтобы гневаться, слишком сильным, чтобы бояться, и слишком счастливым, чтобы допускать существование проблем.

Хорошо думать о себе и демонстрировать этот факт миру, но не громкими словами, а впечатляющими поступками.

Верить в то, что весь мир на вашей стороне, пока вы верны лучшему в себе.

Задание 2. «Точки роста»

С учетом результатов диагностики на предыдущих практических занятиях составьте список собственных направлений профессионально-личностного саморазвития и «точек роста».

Методические рекомендации

Задание 1 на первом этапе выполняется индивидуально. На втором этапе группа делится на команды по 3–4 человека, каждая из которых составляет свой перечень из пяти универсальных «заповедей» личностно-профессионального саморазвития. На третьем этапе в группе обсуждаются «заповеди» личностно-профессионального развития, которые совпали у большинства команд.

Задание 2 каждый магистрант выполняет письменно для себя (работы не сдаются преподавателю). После выполнения второго задания в группе обсуждаются вопросы:

1. Каковы трудности самопознания, проектирования профессионального будущего, планирования саморазвития?

2. Какая деятельность дает наибольший опыт, способствующий вашему развитию?

3. Как сделать собственное развитие более целенаправленным и эффективным? Какие внешние и внутренние ресурсы вам необходимы для реализации саморазвития?

Литература

Лызь Н.А., Лызь А.Е. Управление личностными ресурсами: образование и профессиональное развитие: учебное пособие. – Таганрог: Изд-во ЮФУ, 2016.

Практикум по психологии менеджмента и профессиональной деятельности / под ред. Г. С. Никифорова, М. А. Дмитриевой, В. М. Снеткова. – СПб.: Речь, 2001.

Маралов В.Г. Основы самопознания и саморазвития. М.: Издательский центр «Академия», 2004.

Занятие 10. Деловое общение: ресурсы, техники, приемы (Часть1. Развитие коммуникативной компетентности, навыков вербальной и невербальной коммуникации)

Цель: развитие коммуникативной компетентности, осознания перцептивных и коммуникативных процессов, навыков вербальной и невербальной коммуникации в деловом общении.

Задачи:

- развитие навыков рефлексии собственной психологической позиции в общении и распознавания психологической позиции партнера; умений эффективного использования той или иной психологической позиции в зависимости от целей коммуникации;
- осознание процессов перцепции и интеракции в коммуникации;
- развитие навыков управления своим невербальным поведением, распознавания эмоций и психологических состояний по невербальным проявлениям партнера.

Теоретическое введение

Коммуникативная компетентность с одной стороны, определяется знаниями об эффективной коммуникации, структуре, видах общения и т.п., с другой – навыками вербальной и невербальной коммуникации. Рассматривая деловое общение важно помнить о его структуре, которая во многом определяет эффективность коммуникативного процесса. Наиболее распространенной в социальной психологии является структура, предложенная Г.М. Андреевой, включающая три взаимосвязанные стороны: *коммуникативную, интерактивную и перцептивную*. Для эффективной коммуникации важно помнить о перцептивных механизмах, которые способствуют восприятию и пониманию партнерами друг друга (интерпретация, идентификация, рефлексия, каузальная атрибуция, аттракция) и ухудшают качество общения (эффект первого впечатления, эффект ореола, стереотипизация, проекция, упрощения и т.п.). Важно также осознавать виды взаимодействия, преобладающие в общении: доминирование, манипуляция, соперничество, партнерство, сотрудничество).

Вербальная коммуникация опирается на ряд правил: высказывание должно лаконичным, достаточно правдивым, релевантным, ясным. Нормы вербальной коммуникации заключаются в следующем.

- персональность адресации;
- спонтанность и непринужденность;
- эмоциональность;
- ситуативность.

Невербальная коммуникация – это коммуникация без использования слов. Инструментом общения является тело человека, обладающее широким диапазоном средств и способов передачи информации. Умение хорошо ориентироваться в невербальных знаках и сигналах необходимо для того,

чтобы управлять собственными невербальными сигналами в общении, и соответственно, формировать у собеседника то восприятие собственной личности, которое необходимо для достижения поставленных целей общения. Умение распознавать невербальные знаки помогает правильно оценивать собеседника, в частности, как личность, как конкурента или как партнера.

К основным невербальным средствам общения относятся:

- мимика, жесты, позы, походка, контакт глаз (оптико-кинесический канал);
- интонация, громкость, темп, тембр голоса и т.п. (акустический канал);
- межличностное пространство (проксемика);
- прикосновения в общении (такесика);
- запахи (ольфакторный канал).

Рассмотрим примеры проявления мимики и жестов в общении.

Мимика.

Одним из самых ярких сигналов, подаваемых в общении, являются эмоции собеседников (даже если общение носит рациональный, деловой характер). Эмоции проявляются прежде всего в мимике, жестах, позе, являясь одним из основных средств коммуникации. Рассмотрим подробнее признаки различных эмоций в мимике человека (табл. 2). Наиболее информативными на лице человека для считывания эмоций являются губы, брови и глаза.

Таблица 2

Невербальные признаки эмоций

Эмоция	**Невербальные признаки**	**Иллюстрация**
Радость	*Мимика* 1) Губы расслаблены 2) Уголки губ приподняты 3) Небольшие морщинки в уголках глаз 4) Мышцы в уголках глаз напряжены 5) Щеки приподняты	
Гнев	*Мимика* 1) Брови опущены, сведены или напряжены 2) Блеск в глазах 3) Глаза широко открыты или наоборот, сужены 4) Рот закрыт, губы сужены или напряжены *Взгляд, поза:* 1) «Неотрывный» взгляд на объект гнева 2) Сжатые кулаки	
Презрение	*Мимика* 1) Приподнятый уголок рта с одной стороны 2) Приподнятая бровь 3) Взгляд на объект «сверху вниз»	

Печаль	*Мимика* 1) Верхние веки слегка опущены 2) Рассредоточенный взгляд 3) Губы расслаблены 4) Уголки губ опущены	
Удивление	1) Брови приподняты 2) Глаза широко раскрыты 3) Рот приоткрыт 4) Мышцы лица и губ в целом расслаблены	
Отвращение	1) «Сморщенное» выражение лица (складки на лице и между бровями) 2) Верхняя губа напряжена и приподнята *Жесты, позы:* 1) Отворот головы, туловища (периодический либо полу отворот)	
Страх	*Мимика* 1) Мышцы лица в целом напряжены 2) Глаза широко открыты или видно напряжение в веках 3) Брови могут быть приподняты и вытянуты 4) Губы могут быть немного вытянуты	

Помимо проявления базовых эмоций, в общении, можно наблюдать целые комплексы действий и эмоциональных состояний, а также качеств личности: уверенность и неуверенность, раздражение и агрессивность, согласие и несогласие, оценка получаемой информации, желание доминировать и пр.

Движения рук и тела передают много сведений о человеке. Во-первых, в них проявляются состояние организма и непосредственные эмоциональные реакции. Это позволяет судить о биологически обусловленном темпераменте человека (сильные или слабые у него реакции, быстрые или замедленные, инертные или подвижные) *(Скаженик, 2006)*.

Во-вторых, позы и движения тела выражают многие черты характера: степень его уверенности в себе, зажатость или раскованность, осторожность или порывистость.

В позе и движениях проявляется и социальный статус человека. Такие выражения, как «идти с высоко поднятой головой», «расправить плечи» или, напротив, «стоять на полусогнутых», представляют собой не только описание позы, но и выражают определенное психологическое состояние человека.

В-третьих, в позе и жестах проявляются культурные нормы, усвоенные человеком. Например, воспитанный в европейских традициях мужчина никогда

не будет разговаривать сидя рядом со стоящей женщиной, независимо от того, как он оценивает ее личные достоинства *(Пиз, 1992)*.

Выделяют следующие основные группы жестов в общении: жесты уверенности, жесты неуверенности и раздражения, жесты, выражающие агрессивность, жесты согласия и несогласия, жесты оценки получаемой информации *(Скаженик, 2006)* (табл. 3).

Таблица 3

Жестикуляция в деловом разговоре

Жесты уверенности	• кисти рук соединены кончиками пальцев, ладони не соприкасаются; • кисти рук сцеплены сзади, подбородок высоко поднят; • во время передачи информации локти не прижаты к туловищу; • руки в карманах, большие пальцы снаружи/большие пальцы в карманах, а кисти снаружи; • одна рука обхватывает другую в области ладони
Жесты неуверенности, раздражения	• прижатые вплотную к бокам локти; • ерзание в кресле; • одной рукой человек поправляет пуговицу или запонку на рукаве другой, браслет часов или манжет; • человек двумя руками держит букет цветов, чашку с чаем, сумочку (женщины); • потирание уха
Жесты, выражающие агрессивность	• тесно сплетенные пальцы рук, особенно если руки находятся на коленях; • поза на стуле «верхом»; • руки в карманах, большие пальцы снаружи: у мужчин – амбициозность, у женщин – агрессивность
Жесты несогласия	• боковой взгляд – жест недоверия (в случае, когда взгляд отводится и возвращается вновь, подобное движение воспринимается партнером как жест несогласия); • прикосновение к носу или легкое потирание его – чаще проявляется при наличии в переговорах или дискуссии контраргументов; • ноги у сидящего направлены к выходу – желание уйти; такое же желание проявляется тогда, когда собеседник снимает очки и демонстративно откладывает их в сторону
Жесты согласия	• кивок головой, заинтересованный взгляд, легкая улыбка
Жесты, относящиеся к оценке получаемой информации	• рука у щеки; • один палец отставлен, остальные под подбородком (при критической оценке сказанного или негативном отношении к партнеру в данный момент); • почесывание подбородка (в конфликтных дискуссиях в сочетании со взглядом искоса связано с обдумыванием следующего хода в диалоге); • почесывание пальцем спинки носа (озабоченность, сомнение); • манипуляции с очками; • рука поглаживает шею – недовольство, отрицание, гнев

Вопросы для самоконтроля

1. Назовите перцептивные механизмы, способствующие восприятию и пониманию людьми друг друга?
2. Какие эффекты могут искажать восприятие человека партнером по общению?

3. Что такое общение и какие виды общения вы знаете?
4. Какие элементы входят в структуру делового общения?
5. Какие правила повседневной и деловой коммуникации вы знаете?
6. Что можно узнать о партнере по голосовым характеристикам его речи?
7. Какое значение для деловой коммуникации имеет пространственная ориентация партнеров?
8. В чем заключаются особенности интерактивной стороны делового общения?
9. Какие выделяют виды ситуаций и типы взаимодействий?
10. Как проявляется толерантность в общении?
11. Какие вы знаете механизмы взаимопонимания в процессе делового общения?
12. Как правильно наблюдать за партнером по общению?
13. Внешность человека: определение, компоненты, значение.

Практикум

Задание 1. Развитие коммуникативной компетентности (эссе «Общение в моей жизни»)

Творческая работа состоит из двух частей. В первой части проводится диагностика особенностей межличностных отношений (см. Методики). Во второй части проводится анализ своего собственного опыта межличностных отношений и общения на основе полученных результатов.

МЕТОДИКИ (Ссылки на тесты для диагностики)

- Опросник ОМО (Шутц): https://psytests.org/interpersonal/omofiro.html
- Опросник Типы МЛО (Т.Лири): https://psytests.org/interpersonal/leary-run.html
- Проективный тест Пальмера: https://psytests.org/funtest/pahlmer.html

При написании работы можно опираться на следующий план:

1) Насколько я общительный человек
2) Насколько широкий круг друзей и знакомых я имею/предпочитаю иметь
3) Какие трудности встречаются у меня в общении чаще всего и почему
4) Что у меня лучше всего получается в процессе общения с другими
5) Какой я в общении, какие стратегии выбираю в общении с другими
6) Насколько я удовлетворен (согласен/не согласен с) полученными результатами
7) Планирую ли я что-то менять в общении и в отношениях с другими

Задание 2. Деловая коммуникация: развитие навыков вербальной коммуникации

Выполните задания в соответствие с правилами, приемами, техниками эффективной деловой коммуникации

1.	Приведите пример ситуации, когда эффект первого впечатления может помешать восприятию и пониманию партнера по общению.
2.	Ваш друг часто занимает у вас незначительные суммы и не отдает до тех пор, пока вы не напомните. Сегодня он опять просит у вас немного денег взаймы, но вам не хочется ему давать. Используя ***правила***

	уверенного поведения и эффективной коммуникации откажите ему в просьбе.
3.	Приведите пример своих стереотипов: гендерных, предметных, этнических, профессиональных, возрастных

Задание 3. Развитие навыков невербальной коммуникации

Посмотрите на картинку и определите «основную» эмоцию. Смотрите в первую очередь на губы, брови и глаза (рис.4).

Рис. 4. Невербальные проявления эмоций.

Методические рекомендации

Ознакомьтесь с теоретическим материалом по теме. Ответьте на вопросы для самоконтроля. При выполнении Задания 1 обращайте внимание на самоанализ и рефлексию, которые будут способствовать полноте, всесторонности раскрытия темы, логичности и последовательности изложения материала; следуя плану эссе, проявляйте творческий подход. Прежде, чем приступать к эссе, внимательно прочитайте интерпретацию, полученную по результатам диагностики.

Показатели для оценки эссе (требования):

- знание фактического материала по проблеме общения;
- степень самостоятельности и обоснованности аргументов и обобщений (полнота, глубина, всесторонность раскрытия темы, логичность и последовательность изложения материала, корректность аргументации, характер и достоверность примеров, иллюстративного материала, широта кругозора автора, наличие знаний интегрированного характера, способность к обобщению);
- культура оформления материалов работы.

При выполнении Заданий 2 и 3 опирайтесь не только на свой житейский опыт, но и теоретический лекционный материал, относитесь к заданиям с

пониманием того, что развитие навыков как вербальной, так и невербальной коммуникации требует тренировки. Если вы будете тренироваться с удовольствием, результат будет вами достигнут и легче, и быстрее.

Литература и интернет-ресурсы

Аллан Пиз. Язык телодвижений. – Нижний Новгород: Издательство «Ай Кью», 1992.

Пономаренко Л.П., Белоусова Р.В. Основы психологии для старшеклассников: пособие для педагога: в 2 ч. Ч.1. – М.: Гуманит. изд. центр «ВЛАДОС», 2001. – С. 187–188.

Скаженик Е.Н. Деловое общение. Учебное пособие. – Таганрог: Изд-во ТРТУ, 2006.

Психологические тесты онлайн: https://psytests.org

Занятие 11. Деловое общение: ресурсы, техники, приемы (Часть 2. Психологический контакт. Просьба и отказ в деловом общении)

Цель: совершенствование коммуникативной культуры и коммуникативной компетентности.

Задачи:

- развитие представлений о психологическом контакте в общении;
- развитие навыков установления контакта с партнером по общению;
- развитие навыков ассертивной просьбы и ассертивного отказа в деловом общении.

Теоретическое введение

Установление психологического контакта – одна из важнейших задач начала беседы. Этапы установления психологического контакта предполагают определенную последовательность (рис. 5).

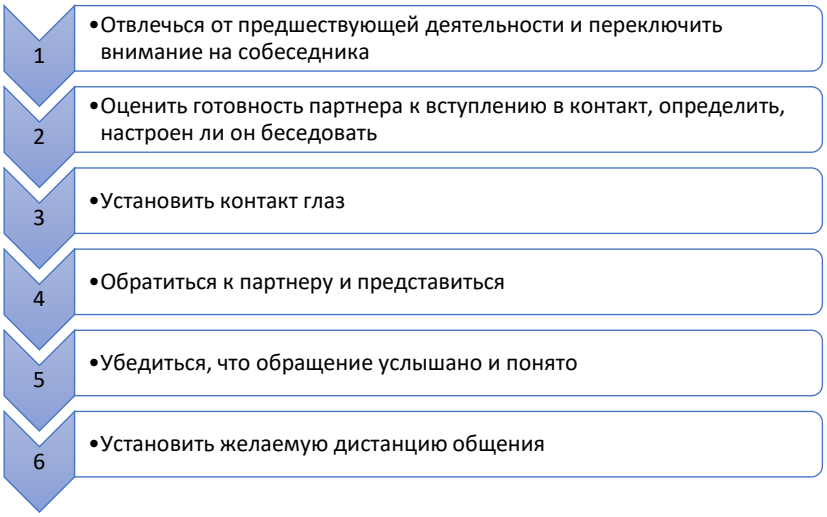


Рис. 5. Этапы установления психологического контакта.

Убедиться в том, что контакт с партнером установлен, можно по следующим признакам: налажен контакт глаз с партнером, речь собеседников и их действия согласованы (гармоничны).

Разрыв психологического контакта – это показатель завершенности деловой беседы. Разрыв контакта должен начинаться задолго до того, как человек завершил речь и всю беседу. Для того, чтобы правильно завершить контакт важно следовать следующим правилам: убыстрение темпа речи, «свертывание» темы, подведение итогов, изменение поведения (посмотреть на часы, убрать бумаги со стола), возвращение к прерванной деятельности.

Показателем эффективности установления или разрыва контакта является отсутствие напряжения в ходе разговора и после его окончания.

Ассертивная просьба

В повседневной жизни просьбу часто путают с предъявлением требований. Просьба – это открытое изложение своей позиции и своих желаний, сделанное в такой форме, что другой человек может согласиться или

отказать в выполнении этих желаний, поскольку это зависит от его позиции и его желания. Требования обосновываются законами, официальными распоряжениями или этическими нормами, правилами морали. Обращаясь с просьбой, можно апеллировать лишь к отношениям с партнером. Поэтому обращение с просьбой надо начинать с установления благоприятных отношений с собеседником. Для эффективного обращения с просьбой необходимо *(Ромек, 2002)*:

6. Перед обращением сформулировать просьбу про себя как можно четче.
7. Обращаться только к тем, кто может реально помочь.
8. Устанавливая контакт, убедиться, что партнер готов выслушать.
9. Изложить просьбу, подчеркивая реальность ее исполнения.
10. Предоставить партнеру свободу действий: можно лишь советовать, но не указывать, что и как нужно делать, выполняя просьбу.
11. Убедиться, что просьба понята правильно.
12. Поблагодарить партнера.
13. При необходимости договориться о следующей встрече.

Если собеседник отказывает в выполнении просьбы, необходимо:

- определить причину отказа (используя техники активного слушания);
- с учетом причин отказа переформулировать просьбу, изменив ее, предложить компромисс.
- попросить совета, кто еще может помочь в выполнении этой просьбы.

Следуя данным правилам, важно уделять внимание своему невербальному поведению: контакту глаз, дистанции общения, позе, интонации. Важно в процессе обращения с просьбой к собеседнику вести себя уверенно, но не высокомерно. Представить, что человек, к которому идет обращение, хочет помочь, но не знает как. Прямо и вежливо сформулировать свою просьбу.

Просьба перестает быть просьбой и становится требованием, когда человек дает собеседнику понять (это легко делается интонацией), что он ожидает (имеет право) определенного ответа; будет чувствовать себя обиженным (расстроенным, рассерженным, покинутым), если просьба не будет удовлетворена.

Ассертивный отказ

Умение твердо и корректно отказывать – важное умение человека, который стремиться к самостоятельности в мышлении и поведении, который является личностью, ответственной за свои поступки. Умение говорить «нет» – это умение обозначать границы своего суверенитета. Человек, не умеющий отказывать становится жертвой манипулятора, теряя свою индивидуальность. Особые проблемы со словом «нет» имеют люди с нарушенной самооценкой. Они не могут и не позволяют себе устанавливать границы своего «я». Существуют правила «обозначения границ» и корректного отказа собеседнику.

14. *Уделить собеседнику (просящему) внимание.* Важно показать, что вы заинтересованы в беседе с партнером, поэтому поблагодарите его за то, что обратились именно к вам, переспросите, если что-либо вам не ясно.

15. *Определить свою позицию* (предпочтения, чувства, понимание ситуации). Важно понять, сможете ли вы сделать то, о чем вас просят, хотите ли вы этого и будете ли делать. Если вы решили отказать, то ведите себя уверенно и конструктивно, извинения здесь не уместны.

16. *Скажите «нет».* Уверенно и однозначно откажите: «Спасибо, нет»; «Мне это не подходит».

В случае, если собеседник настаивает на просьбе, апеллируя к жалости, милосердию или даже к шантажу важно оставаться твердым в своем отказе, т.е. использовать технику «перманентного отказа»:

- благодарность за доверие, предложение;
- аргументация отказа;
- вежливый, но твердый отказ;
- выслушивание и повторение аргументов оппонента;
- снова вежливый и твердый отказ.

Некоторые дополнительные техники отказа:

Пауза: «Я позвоню вам в конце недели»; «Я дам вам знать к середине дня» и т.п.

Сочувствие: «Это действительно серьезная проблема, я понимаю», «Да, плохо дело, но я надеюсь, вы непременно найдете выход».

Минимум извинений: «Простите меня за то, что мои интересы для меня ближе ваших».

Точность в определении границ помощи: «Я готов помочь вам перенести мебель, но только не укладывать вещи»; «Я могу вас подвезти до работы, но только если к четверти девятого вы будете готовы».

Внимание интонации и жестам: стойте или сидите в удобной для вас позе. Смотрите в глаза собеседнику. Говорите уверенно, полным голосом. Ваша поза и ваш голос должны служить дополнительным подтверждением вашей уверенности.

Избегайте чувства вины.

Вопросы для собеседования

1. Развитие каких социально-психологических умений способствует повышению уровня коммуникативной культуры?
2. В каких формах осуществляется деловое общение? Каковы их особенности?
3. Какие методы используются в процессе делового общения для воздействия на партнеров?
4. Назовите функции делового общения.
5. Охарактеризуйте виды делового общения работников в организации.
6. Раскройте сущность принципов делового общения.
7. По каким признакам можно выявить наличие психологического контакта в общении?
8. Что такое ассертивная просьба?
9. Какие правила существуют для обращения к собеседнику с просьбой?
10. Чем просьба отличается от требования?

11.Какие правила и техники ассертивного отказа существуют?

Практикум

Для того, чтобы закрепить навыки установления и разрыва контакта, сформировать навыки ассертивной просьбы и отказа, навыки уверенного поведения в беседе можно использовать специальные техники и упражнения из тренинга ассертивности.

Задание 1. Упражнение «Односторонний контакт»

Цель: овладение навыками вступления в контакт и выявление признаков контакта.

Описание упражнения

Все участники тренинговой группы делятся на пары: один исполняет роль инициатора беседы, другой – его собеседника. Задача инициатора беседы установить контакт и разговорить собеседника, поскольку он не знает ситуации и его необходимо ввести в курс дела и выполнить задание ситуации.

Ситуация 1.

Вас пригласил на беседу преподаватель. Вы не знаете, о чем пойдет речь, но у его кабинета видите другого студента, у которого может быть можно прояснить ситуацию.

Ситуация 2.

Вы пришли на пересдачу экзамена. Обратитесь к студенту из другой группы (так как из вашей группы никого нет) и узнайте, какие вопросы задает в основном преподаватель.

Ситуация 3.

Вы покупаете газету в киоске и завязываете разговор с продавцом о квартирах, сдающихся поблизости.

Ситуация 4.

Вы приехали в другой город. Покупая журналы, узнайте у продавца, какие культурные центры он вам посоветует посетить.

Обсуждение и выводы

Участникам каждой пары при обсуждении задаются вопросы:

- Как Вы поняли, что хотел от вас партнер?
- Испытали ли Вы напряжение?
- Удалось ли вступить в контакт?

В случае необходимости упражнение можно повторить, используя более эффективную стратегию.

Задание 2. Упражнение «Ассертивная просьба»

Цель: изучение ассертивной просьбы о любезности.

Описание упражнения

Упражнение выполняется в рабочей группе.

1. Вспомните ситуацию, в которой вам необходимо было получить что-то для вас важное и вы хотели, чтобы ваш партнер сам догадался и предложил вам это.

2. Представьте, что этот человек сидит на пустом стуле в центре круга. Сформулируйте вашу просьбу и попросите его о любезности.

3. Остальные члены группы делятся с говорящим своими впечатлениями: была ли просьба ассертивной, не присутствовала ли в ней агрессия или робость.

4. В роли «просящего» по очереди должны побывать все участники.

Обсуждение и выводы

- Какой опыт вы приобрели, выполняя это упражнение?
- Что вы узнали о себе?
- Какие закономерности вы увидели и к каким выводам пришли?
- Какие достоинства и ограничения вы обнаружили у этой техники? Какова, на ваш взгляд, область ее применения?

Задание 3. Упражнение «Берегись, хозяин!» (Турнер, 2002)

Цель: отработка навыка ассертивной просьбы, поделиться друг с другом знанием и опытом общения в условиях производственных отношений.

Описание упражнения.

1. Прочитайте текст «Отсутствие доказательств». Запишите любые замечания, которые придут в голову после прочтения текста.

2. Попробуйте разыграть данную ситуацию для отработки навыков ассертивной просьбы.

Текст «Отсутствие доказательств»

Билл был нанят компанией по производству скобяных изделий для выполнения плотницких работ. Он превосходно разбирался в древесине, и покупатели ценили его советы.

Однако Билл оказался совершенно никудышным работником. Он часто опаздывал, устраивал длительные перерывы, работал медленно, делая множество ошибок, и оставлял мусор, убирать который приходилось другим.

Управляющий, Кен, промучился с Биллом два года, пока не нашел, наконец, подходящую замену. После этого он попытался уволить его. Но Билл пожаловался в профсоюз на несправедливость увольнения, поскольку экспертиза, которой его работа была бы признана неудовлетворительной, не проводилась ни разу. С полным сознанием собственной правоты он заявил, что раз не было официальной оценки, то он считал себя вправе рассматривать свою работу как соответствующую принятому стандарту.

В компании существовала система ежегодной письменной оценки качества производства, однако у Джейн, непосредственного начальника Билла, никогда не доходили до этого руки, и последние три года она вообще этим не занималась.

Теперь стоит вопрос: какие аргументы могут представить Кен и Джейн, которые доказали бы представителю профсоюза справедливость увольнения Билла? Могут ли они это сделать? Или они обречены работать с Биллом вечно?

Обсуждение и выводы

- Удалось ли Кену и Джейн правильно сформулировать свою просьбу профсоюзному представителю?
- Какие аргументы они использовали?
- Была ли их стратегия эффективной?

Задание 4. Техника «Ассертивный отказ» (Ромек, 2002)

Цель: изучение личностных границ и техники ассертивного отказа.

Описание упражнения

Упражнение выполняется в рабочей группе.

1. Группа просит участника сделать что-либо из того, что он делать не обязан.

2. Задача первого участника – последовательно в течение трех минут отказывать группе в просьбе.

3. После того, как отведенное время закончится, члены группы делятся с говорящим своими впечатлениями: был ли отказ ассертивным; не присутствовала ли в нем агрессия или робость.

4. Затем «отказывающим» становится следующий участник.

5. В роли «отказывающего» по очереди должны побывать все участники.

Обсуждение и выводы

- Удалось ли каждому участнику отказать в просьбе?
- Эффективными ли были выбираемые им стратегии отказа?
- Что получилось, а что нет в ситуации отказа?

Методические рекомендации

1. Постарайтесь сделать свои выступления аргументированными и убедительными.

2. Соберите обратную связь у членов группы относительно точности выполнения вами тех или иных приемов.

3. Проанализируйте полученную обратную связь и свои собственные впечатления от участия в упражнениях.

4. Полученные знания постарайтесь использовать в повседневной жизни, фиксируйте свои успехи и достижения.

Литература

Истратова О.Н. Эксакусто Т.В. Справочник по групповой психокоррекции. Ростов н/Д, «Феникс», изд. 3-е, 2011, - 443 с.

Ромек В.Г. Тренинг уверенности в межличностных отношениях. СПб.: Речь, 2002.

Турнер Д. Ролевые игры. Практическое руководство. – СПб: Питер, 2002. – 352 с.

Фопель К. Психологические группы. Рабочие материалы для ведущего. – М.: генезис, 2001 - 256 с.

Эксакусто Т.В. Практикум по групповой психокоррекции: тренинги, упражнения, ролевые игры Ростов н/Д, «Феникс», 2007. 235 с.

Занятие 12. Деловое общение: ресурсы, техники, приемы (Часть 3. Активное слушание. Техника формулирования вопросов

Цель: освоение навыков активного слушания в процессе общения, развитие коммуникативной компетентности за счет овладения техникой формулировки вопросов в беседе.

Задачи:

- знакомство с понятиями активного (рефлексивного), нерефлексивного, эмпатического слушания;
- развитие навыков активного слушания в процессе общения с партнером;
- знакомство с видами вопросов; отработка навыка формулирования вопросов в общении.

Теоретическое введение

Виды слушания в общении

В психологии принято выделять три вида слушания *(Атватер, 2001)*: эмпатическое, нерефлексивное и рефлексивное (или активное) слушание. Эмпатического слушание чаще используется в межличностном общении, в то время как в деловом общении его, как правило, не используют. *Эмпатическое общение* предполагает проявление переживаний, сочувствия словам партнера и является очень эмоциональным («Я очень сочувствую вам», «Я искренне рад за вас», «Ах, это прекрасно!», «Это ужасно…» и т.п.). Чаще всего эмпатическое слушание используют психологи в беседе с клиентом, т.к. такой прием помогает человеку справиться с трудностями за счет проявления своих эмоций и чувств. Нерефлексивное слушание также используется в деловом общении достаточно редко (однако, в виде исключения может являться эффективным приемом в деловой коммуникации). *Нерефлексивное слушание* или «умение внимательно молчать» используется для того, чтобы дать высказаться собеседнику, не вмешиваясь в его речь с замечаниями и комментариями. Такие ситуации могут возникать, когда человек испытывает сильные эмоциональные переживания или когда ему необходимо выговориться, проговорить все, что он задумал. Именно поэтому вмешательство в речь собеседника должно быть сведено к минимуму (например: «Да?!», «Продолжайте», «Интересно» и т.п.).

Таблица 4

Приемы активного слушания

Приемы активного слушания	Содержание приема	Примеры фраз
Выяснение	Обращение к говорящему за уточнениями	«Я не совсем понял», «Что Вы имеете в виду?», «Пожалуйста, уточните», «Не могли бы вы пояснить это?», «К сожалению, я не все понял», «Не могли бы вы повторить?», «Может, вы сформулируете по-другому?»
Перефразир	Собственная форму-	«Как я Вас понял...», «Вы думаете, что...», «По

ование	лировка (перефор-мулировка) сооб-щения собеседника для проверки его точности	Вашему мнению...», «Вы считаете, что…»
Отражение чувств	Отражение слушаю-щим (реципиентом) эмоционального состояния говоря-щего (комму-никатора)	«Вероятно, Вы чувствуете...», «Вы несколько расстроены...», «Вы вроде бы огорчены...», «Вы переживаете...», «Представляю, как вам тяжело...»
Резюмирование / обобщение	Подытоживаются основные идеи и чувства говорящего, что помогает связать части сообщения в смысловое целое	«Вашими основными идеями, как я понял, являются...», «Если подытожить сказанное Вами, то...», «Итак, можно сказать…», «Обобщая то, что вы сказали...», «До сих пор мы рассматривали...», «Итак, ваша основная мысль, если я правильно вас понял, сводится...»
Пересказ	Это шаг вперед по сравнению с пояснением того, что мы услышали, т.е. говорящему «возвращается» суть его сообщения с продолжением его мысли, чтобы он смог оценить, правильно ли мы его поняли	«Вы имеете ввиду...», «Насколько я мог вас понять...», «Значит, с вашей точки зрения...», «Итак, вы полагаете...», «Иными словами, вы считаете...»

Рефлексивное (активное) слушание направлено на распознание, расшифровку, декодирование смыла, значения сообщения, поступающего от собеседника. Способствуют этому специальные приемы: выяснение, перефразирование, отражение чувств, резюмирование (табл. 4).

Умение задавать вопросы

Задавать вопросы в процессе общения – значит обеспечивать взаимопонимание с партнером, приобретать сведения, критически мыслить и проявляя собственную позицию, идти навстречу собеседнику. Спрашивать – значит оказывать доверие собеседнику, проявлять заинтересованность к его личности.

Умение задавать вопросы– это особое искусство владения речью и невербальной коммуникацией. С одной стороны, вопрос заданный не вовремя, либо неуместный вопрос, может приводить к возникновению барьеров и трудностей в общении, что становится отправной точкой развития конфликтов. С другой, если не задавать вопросы, пренебрегать ими, это может привести к искаженному восприятию партнера по общению, возникновению домыслов и иллюзий. Особенно важными и необходимыми вопросы становятся в ходе деловой беседы. Ее эффективность во многом обусловлена точностью и корректностью вопросов, которые собеседники задают друг другу. Психологи, исходя из разных оснований, выделяют множество видов вопросов, создают их разные классификации. Одна из известных и широко распространенных

классификаций основана на широте предстоящего ответа. В ней выделяются три группы вопросов *(Айламазьян, 1999)* (табл. 5).

Таблица 5

Виды вопросов (классификация с точки зрения широты предстоящего ответа)

Виды вопросов	Специфика вопросов	Примеры
Закрытые вопросы	Вопросы, на которые ожидается ответ «да» или «нет»; они обращены ко всему объему содержащегося в них смысла. Эти вопросы могут привести к созданию напряженной атмосферы в беседе, поскольку резко сужают информационную составляющую беседы и могут нарушить ход мысли говорящего	«Это все, что Вы хотели сказать?»; «Это трудно для вас?»; «Вы предпочитаете это сделать сами?»
Открытые вопросы	Это вопросы, на которые нельзя ответить «да» или «нет», они требуют развернутого объяснения. Благодаря их использованию собеседник находится в более активном состоянии он имеет возможность, по своему усмотрению, строить содержание ответов	«Каково Ваше мнение по данному вопросу?»; «Почему Вы считаете такой подход неудачным?»; «Что Вы собираетесь делать летом?»
Выясняющие / уточняющие вопросы	Вопросы являются обращением к говорящему за уточнением; они вынуждают собеседника размышлять, обдумывать и комментировать то, что было сказано	«В этом ли состоит проблема, как Вы ее понимаете?»; «Что Вы имеете в виду?»

Существует классификация вопросов, которые выделяют в зависимости от смысла соотносимых с ними ответов *(Айламазьян, 1999)* (табл. 5).

Таблица 6

Виды вопросов (классификация с точки зрения смысла, соотносимых с вопросами ответов)

Виды вопросов	Специфика вопросов	Примеры
Альтернативные вопросы	Вопросы содержат возможный выбор, который предстоит сделать собеседнику; ответ на него будет охватывать лишь часть (большую или меньшую) смысла, содержащегося в вопросе.	«Где Вам удобнее будет встретиться со мной, в офисе или на выставке?», «Куда вам было бы интересно поехать в Египет или в Турцию?»
Избирательные вопросы	Эти вопросы задают некоторый круг «предметов», не называя их конкретно, из которых можно сделать выбор. Этот выбор содержится в ответе на избирательный вопрос	«Чем он болен?» - «Гриппом».
Иксовые вопросы	Вопросы не предполагают «подсказку» ответа. На вопросы такого типа могут последовать любые ответы, которые явным образом не связаны со смысловыми ориентирами, которые содержатся в вопросе	«Что он сказал?»; «Что Вы собираетесь делать летом?»
Прямые вопросы	Вопросы, которые непосредственно касаются исследуемого предмета и формулируются,	«Знаете ли Вы, что…?»; «Что Вы думаете о...?»; «Каково

	как правило, в личной форме	Ваше мнение по поводу...?»

К обеспечивающим продуктивный внешний диалог относятся информационные, зеркальные и эстафетные вопросы (табл. 7).

Таблица 7

Виды вопросов (классификация с точки зрения продуктивности диалога)

Виды вопросов	Специфика вопросов	Примеры
Информационный вопрос	Это вопрос, который построен так, что он вызывает содержательный ответ (мысль, суждение, изложение фактов, позиции и пр.)	«Какие меры вы приняли?»
Зеркальный вопрос	Вопрос состоит в повторении с вопросительной интонацией части утверждения, произнесенного собеседником, чтобы заставить его увидеть свое утверждение, с другой стороны	«Вы действительно думаете, что сделали все, что могли в этой ситуации?»
Эстафетные вопросы	Эти вопросы придают диалогу динамику; они опережают, развивают высказывания партнера, не перебивая, а помогая ему	«Вы хотите сказать, что…»

Вопросы для самоконтроля

1. Какие виды слушания используются в общении?
2. Какой из видов слушания не используется в деловом общении?
3. Что такое активное (рефлексивное) слушание?
4. Какие приемы активного слушания можно использовать в общении и с какой целью?
5. Для чего необходимы вопросы в ходе общения?
6. Какие виды вопросов вы знаете?
7. Какие виды вопросов вы чаще всего используете в своем общении?

Практикум

Задание 1. Анализ умения слушать партнера по общению

Ответьте на вопросы данного опросника, подсчитайте результаты, проанализируйте свое умение слушать партнера по общению.

Опросник «Слушаете ли вы собеседника?» (И. Атватер)

Инструкция: Вашему вниманию предлагается методика на определение степени выраженности умения слушать другого человека. Перед Вами 16 вопросов на каждый из которых Вы должны ответить утверждением «да» или «нет». Следует помнить, что нет правильных или неправильных ответов, т. к. люди различны и каждый высказывает свое мнение. Главное, старайтесь отвечать честно, не пытайтесь произвести благоприятное впечатление, ответы должны соответствовать действительности. Свободно и искренно выражайте свое мнение. В этом случае Вы сможете лучше узнать себя.

Текст опросника

1. Ждете ли Вы терпеливо, пока другой закончит говорить и даст Вам возможность высказаться?

2. Спешите ли Вы принять решение до того, как поймете сущность проблемы?

3. Слушаете ли Вы лишь то, что Вам нравится?

4. Мешают ли Вам слушать собеседника Ваши эмоции?

5. Отвлекаетесь ли Вы, когда собеседник излагает свои мысли?

6. Запоминаете ли Вы вместо основных моментов беседы какие-либо несущественные?

7. Мешают ли Вам слушать предубеждения?

8. Прекращаете ли Вы слушать собеседника, когда появляются трудности в его понимании?

9. Занимаете ли Вы зачастую негативную позицию к говорящему?

10. Всегда ли Вы слушаете собеседника?

11. Ставите ли Вы себя на место говорящего, чтобы понять, что заставило его говорить именно так?

12. Принимаете ли Вы во внимание тот факт, что у Вас с собеседником могут быть разные предметы обсуждения?

13. Допускаете ли, что у Вас и у Вашего собеседника может быть разное понимание смысла употребляемых слов?

14. Пытаетесь ли Вы выяснить тот факт, чем вызван спор: разными точками зрения, постановкой вопроса и т.п.?

15. Избегаете ли Вы взгляда собеседника в разговоре?

16. Возникает ли у Вас непреодолимое желание прервать собеседника и вставить свое слово за него или в пику ему, опередить его в выводах?

Обработка результатов

Подсчитайте количество ответов «нет» на вопросы: 2, 3, 4, 5, 6, 7, 8, 9, 15, 16.

Подсчитайте количество ответов «да» на вопросы 1, 10, 11, 12, 13, 14.

Сложите количество совпадений с ответами «да» и количество совпадений с ответами «нет». Сумма соответствует вашему умению слушать собеседника.

Интерпретация

6 баллов и ниже – свидетельствуют о низкой степени выраженности умения слушать других, о направленности в ходе общения на себя (т. е. удовлетворение своих притязаний вне зависимости от интересов партнера). Снижена чувствительность в оценке текущей ситуации – когда молчать и слушать, а когда говорить. Необходимо обучение навыкам эффективного слушания.

От 7 до 10 баллов – средняя степень выраженности умения слушать собеседника. Данное умение скорее проявляется ситуативно и зависит от личной значимости (заинтересованности) получаемой информации. Требуется совершенствование навыков и приемов активного слушания.

10 баллов и выше свидетельствуют о явно выраженном умении слушать других вне зависимости от личной значимости получаемой информации. Такой человек является хорошим собеседником, может работать в команде.

Задание 2 «Виды вопросов»

Составьте по два примера к каждому из следующих видов вопросов, встречающихся в деловом общении:

- информационные вопросы (используются для сбора сведений);
- контрольные вопросы (необходимы для контроля за ходом деловой коммуникации);
- ориентационные вопросы (используются, чтобы знать придерживается ли партнер идей, высказанных ранее);
- подтверждающие вопросы (необходимы, чтобы добиться взаимопонимания);
- ознакомительные вопросы (используются для ознакомления с мнением собеседника);
- однополюсные вопросы (повторение вопроса собеседника, в знак того, что понятно, о чем идет речь и для того, чтобы выиграть время на обдумывание ответа);
- встречные вопросы (необходимы для сужения темы разговора);
- направляющие вопросы (в случае отклонения от темы направляют беседу в нужное русло);
- альтернативные вопросы (предоставляют возможность выбора);
- провокационные вопросы (используются, чтобы установить правильно ли партнер понимает ситуацию);
- вступительные вопросы (необходимы для формирования у партнера заинтересованности в разговоре);
- заключающие вопросы (необходимы для подведения итогов разговора);
- закрытые вопросы (наводящие вопросы, на которые можно коротко ответить);
- открытые вопросы (выявляют ключевые моменты беседы).

Методические рекомендации

Рекомендации к заданию 1:

- подберите литературу по проблеме слушания в общении, ориентируясь на издания, раскрывающие сущность активного (рефлексивного) слушания;
- прежде, чем отвечать на вопросы опросника проанализируйте свое умение слушать собеседника;
- пройдите тестирование и подсчитайте результат;
- проанализируйте ваши представления о своем умении слушать и результаты опросника;
- напишите краткое резюме, включающее ваши советы самому себе по развитию навыков эффективного слушания.

Рекомендации к заданию 2:

- подберите литературу по рассматриваемой проблеме, ориентируясь на издания, раскрывающие прикладные аспекты делового общения и взаимодействия;
- проанализируйте собранный материал, составьте общее впечатление о видах вопросов, которые могут использоваться в деловой беседе для ее эффективности;
- самостоятельно сформулируйте примеры к каждому из видов вопросов (2-3 примера);
- запишите предлагаемые примеры вопросов последовательно и в соответствие с их классификацией.
- в заключении работы рекомендуется сделать краткий вывод, отразив собственное мнение по эффективности видов вопросов в деловой беседе (в каких случаях и при каких обстоятельствах те или иные вопросы будут способствовать эффективной деловой беседе);
- оформите список используемой литературы.

Критерии оценки задания 2:

- соответствие утверждений видам вопросов;
- полный список примеров по всем видам вопросов, встречающихся в деловой беседе;
- уровень творчества, оригинальность приводимых примеров;
- степень самостоятельности вклада студента (учитывается опора на свой опыт в деловом общении);
- аргументированность выводов.

Литература

Айламазьян А.М. Метод беседы в психологии. – М.: Смысл, 1999. – 222 с.

Атватер И. «Я вас слушаю» //Психология внимания. Хрестоматия /ред. Ю.Б. Гиппенрейтер, В.Я. Романов. М.: ЧеРо, 2001.

Куницына В.Н., Казаринова Н.В., Погольша В.М. Межличностное общение СПб: Питер, 2001. – 544 с.

Шейнов В.П. Как управлять другими. Как управлять собой. М: АСТ, 2009. – 544 с.

Занятие 13. Коммуникация в деловом общении (Часть1. Публичное выступление)

Цель: освоение способов и приемов эффективного публичного выступления.

Задачи:

- рассмотрение структуры и принципов построения публичного выступления;
- изучение приемов эффективного публичного выступления;
- обсуждение типичных ошибок выступающего;
- развитие навыка лифтовой речи.

Теоретическое введение

В настоящее время ораторское умение высоко ценится в профессиональной среде. От эффективного устного доклада, презентации своих идей и предложений, грамотных ответов на вопросы зависит авторитет сотрудника, его статус. Умение говорить и умение удерживать внимание аудитории в течение определенного времени– это настоящее искусство. Формирование навыков публичного выступления требует упорного труда, тренировки и самоанализа. Выступающий должен стимулировать интерес людей к обсуждаемой проблеме, использовать способы воздействия на установки слушателей и их поведение.

Существует ряд способов построения речи в публичном выступлении. В частности, выделяют индуктивный способ (от частного к общему), дедуктивный (от общего к частному), концентрический (расположение материала вокруг главной проблемы), исторический (изложение материала в хронологической последовательности) и др.

Структура речи (всей или отдельного модуля) должна включать три основных элемента: введение, основная часть, заключение.

Вступление

1. Привлечение внимания.
2. Установление контакта с аудиторией.
3. Фиксация внимания.
4. Анонс выступления.

Изложение основного содержания

5. Первый проблемный модуль.

5.1. Описание существующего положения.

5.2. Описание того, каким это положение должно быть.

5.3. Предлагаемый метод решения.

6. Переход к следующему проблемному модулю.
7. Второй проблемный модуль и т.д.

Модульный способ построения речи

- ✓ *Каждый модуль посвящен какому-то отдельному вопросу*
- ✓ *Внутри себя каждый модуль структурирован, в нем есть:*

- ✓ *мини-вступление,*
- ✓ *основная часть с коротким резюме*
- ✓ *фраза для перехода к следующему модулю.*

Заключение

8. Логическое обобщение.

9. Выводы. Краткий обзор сказанного.

10. Призыв к действию.

11.Эмоционально-мемориальная информация.

Существуют также принципы построения хорошо воспринимаемого сообщения:

– от внимания к интересу,

– от интереса к основным положениям,

– от основных положений к возражениям и вопросам,

– ответы, выводы, резюмирование.

Ошибки, которые необходимо избегать при выступлении:

Ошибка 1: *Несоответствие содержания речи и поведения оратора*. Когда содержание слов расходится с тоном речи, осанкой и языком тела, публика это замечает. Если вы начнете говорить «Здравствуйте, как я рад вас всех видеть....» дрожащим неуверенным голосом – будьте уверены, у слушателей моментально появится недоверие и к сказанному вами, и к самому говорящему. Поэтому слова должны совпадать с теми эмоциями, которые испытывает оратор. Доверяя искренности эмоций, люди легче воспринимают информацию, у них возникает установка на доверие к докладу в целом.

Ошибка 2: *Оправдания и извинения*. Даже если вы волнуетесь и не уверены в себе, не стоит оправдываться перед аудиторией. Потому что подготовка к выступлению – это ваша ответственность, и аудитории не нравится, когда своими оправданиями вы пытаетесь эту ответственность с себя снять или разделить с присутствующими. Постарайтесь с самого начала концентрироваться не на вашей неуверенности, а на аудитории. Обращайтесь к мыслям и чувствам сидящих перед вами людей. Даже если вы не уверены в себе, в вашу пользу будут играть две закономерности: а) гораздо больше слушателей, чем вы думаете, не обращают внимания на волнение оратора или же снисходительно относятся к нему; б) ваше волнение снижается тем скорее, чем больше внимания вы уделяете другим людям, а не собственным ощущениям.

Что касается извинений, лучше изначально избегать того, за что нужно будет просить прощения, то есть готовиться и продумывать выступление как можно тщательнее.

Ошибка 3: *Отсутствие контроля глаз, бровей, мимики*. Брови – главный элемент мимики, они не только указывают на эмоции, но и управляют ими. Чрезмерно высоко поднятые брови – признак неуверенности и некомпетентности. Также, прямой открытый взгляд и широко раскрытые от страха глаза разделяет всего несколько миллиметров. Лучше всего для публичного выступления подходят смеющиеся глаза и прямые брови.

Ошибка 4: *Использование «не тех» слов.* Отрицательные частицы воспринимаются сознанием позднее, чем основное слово, а часто вообще не воспринимаются. Поэтому используйте только те слова, которые подкрепляют желаемую цель. Если вы хотите создать позитивный настрой, тогда вместо слов «это не плохо», скажите «это хорошо», вместо «не принесет убытков» – «принесет выгоду» – и опишите, какую именно выгоду. Ведь она может выражаться не только в виде незамедлительной денежной прибыли – но, и в виде опыта, или информации, которая поможет в дальнейшем получить финансовую выгоду, и т.п.

Ошибка 5: *Не будьте занудой*. Добавьте в свою серьезную речь улыбку, разбавьте шутками, расскажите забавную историю. Публика ответит вам благосклонностью и вниманием. Можно посмеяться и над собой, если вы допустили какую-то оплошность – слушатели воспримут это как признак вашей уверенности в себе и чувства собственного достоинства.

Ошибка 6: *Всезнайство*. Такие ораторы всегда считают себя намного умнее аудитории, к которой обращаются. В ответ в аудитории всегда появляются скептики, а порой и провокаторы. Поэтому, подключайте слушателей с новой информацией к докладу, умейте оценить их знания. Этим вы продемонстрируете уважение к участникам и внесете оживление в собственное выступление.

Ошибка 7: *Суетливость*. По тому, как движется докладчик, легко понять, насколько он уверен в себе. Испытывая страх перед публикой, начинающий оратор может беспокойно ходить, проделывать суетливые манипуляции с предметами. В итоге публика начинает следить за его перемещениями и перестает следить за темой. Найдите удобное место и займите позицию, с которой вы можете установить зрительный контакт со всей аудиторией. Но не стоит и «замирать» на одном месте. Перемещайтесь, но перемещайтесь осознанно, контролируя пространство.

Ошибка 8: *Монотонность и скучность.* Чтобы ваша речь была выразительной и интересной, выделяйте голосом ключевые моменты публичного выступления: повышайте высоту звука в конце вопроса; изменяйте темп речи в зависимости от ее содержания. Искусный оратор держит публику «в тонусе», постоянно варьирует громкость и силу своего голоса, придавая ему живости. Когда хочет вызвать напряженность и интерес, он произносит слова чуть тише и медленнее. Говоря громче, он выделяет главное в своем публичном выступлении.

Ошибка 9: *Отсутствие пауз.* Как правило, неопытные ораторы спешат заполнить паузы в речи словами-паразитами: «Эээээ... Значит так... Ну, что еще сказать...». Когда нечего сказать – лучше помолчите, пока придут нужные слова. Иногда оратору необходимо время, чтобы подумать, сверится со своими записями, или же просто попить воды. А публике нужны паузы, чтобы осмыслить сказанное вами. Паузу можно использовать для установления визуального контакта, чтобы проконтролировать, правильно ли вас поняли; для усиления напряжения и драматизма; для возбуждения любопытства («...а что он скажет дальше?») и для многого другого.

Ошибка 10: *Старайтесь не перегружать слушателей фактами*. Используйте примеры из живых разговоров, как будто вы просто общаетесь с собеседником. Ни в коем случае не злоупотребляйте профессиональным жаргоном.

Разновидностью публичного выступления является *лифтовая речь*. Лифтовая речь (Elevator pitch) – это заготовленная и тщательно спланированная мини-презентация, которая четко, ясно, и главное, быстро, рассказывает о вас/вашей компании или продукте. Именно лифтовая речь позволяет быстро и лаконично представить себя, рассказать о вашей компании, обратиться к потенциальному инвестору и т.п.

Правила подготовки лифтовой речи предполагают следующее.

• Краткость: 100-250 слов или 30-60 секунд времени – оптимальный вариант для лифтовой презентации.

• Понятность: меньше аббревиатур и сленговых выражений. Ваша идея должна быть понятна даже бабушке и пятилетней племяннице.

• Актуальность: ваш проект/идея решает определенную проблему визави

• Доверие: собеседник понимает, что перед ним достаточно компетентный человек, который действительно способен решить его проблему

• Концептуальность и последовательность: ваша речь хорошо структурирована и не избыточествует ненужными деталями

• Конкретность: больше конкретики и реалистичности

• Адаптированность: ваша речь учитывает интересы конкретного слушателя

• Диалогичность/ Вовлечение: монолог – для театра; в нашем случае стоит задача вывести собеседника на конструктивный диалог

Структура речи должна включать:

• имя, род занятий

• навыки и опыт

• польза и выгода

• что вас интересует

• призыв к действию (т.е. что сделать дальше: отправить резюме, прийти на встречу, подготовить план и т.п.)

Примеры лифтовой речи

Пример 1. Вы хотите найти работу

В этом примере вы рассказываете об основных своих преимуществах и навыках, подкрепляете их примером и прямо спрашиваете, есть ли вам место в компании:

«Здравствуйте, меня зовут Светлана. Я эколог и ищу работу, которая поможет мне использовать мои исследовательские и аналитические навыки. Последние несколько лет я укрепляла их, сотрудничая с местными экологическими организациями и властями по вопросам сохранения и поддержки качества водных ресурсов. Также в будущем я хотела бы развивать образовательные программы по информированию населения об охране водных ресурсов.

Я знаю, что ваша организация занимается проектами контроля качества воды. Скажите, могу я попробовать стать частью вашей организации и усилить ее своими знаниями?»

Пример 2. Вы хотите достучаться к своему начальнику с новой идеей

Ваш начальник все никак не может найти время, чтобы выслушать вас? Вам понадобится максимум 30 секунд, чтобы донести проблему и пути ее решения. Это можно сделать, даже просто провожая руководителя по коридору:

«У нас в организации большинство сотрудников не имеют доступа к принтерам для распечатки документов, хотя по работе им это необходимо. Нужно дать всем возможность оперативно распечатывать документы, чтобы не терять рабочее время для лишних коммуникаций.

Предлагаю поставить один принтер для использования всеми сотрудниками в кабинете секретаря. Можно приобрести новый принтер, как это сделали в компании N решая похожую проблему, либо использовать для этой цели принтер из отдела закупок, так как он мало задействован в работе отдела. Любые эти варианты дешевле третьего — приобрести принтеры для каждого сотрудника».

Вопросы для самоконтроля

1. Что такое деловая риторика?
2. Назовите правила проведения публичного выступления и ответов на вопросы аудитории.
3. В чем состоят главные отличия устного выступления от письменного доклада?
4. Назовите виды дискуссии. В чем состоит отличие дискуссии от спора?
5. В предлагаемой схеме структура речи должна включать три основных элемента. Назовите и опишите эти элементы.
6. Перечислите основные приемы эффективного выступления.
7. Как вы считаете, какие ошибки являются наиболее критическими для публичного выступления? Почему?
8. Что такое лифтовая речь? Какие правила подготовки ее вы знаете?

Практикум

Задание 1. Анализ публичного выступления

Проанализируйте представленный ниже отрывок публичного выступления и выделите ошибки, которые допустил оратор. Отредактируйте текст так, чтобы он звучал интересно и привлекал внимание аудитории.

«Здравствуйте….ммм….друзья…..Я бы хотел бы рассказать сегодня немножечко о том, что такое стресс, как он протекает, какие характеристики он имеет, о методах борьбы со стрессом и том, что делать, чтобы он не возникал…. ммм….да…..вроде бы все. *(Далее – очень быстро, почти без пауз).* Каждому из нас приходилось испытывать неприятные ощущения непонятного характера…эээ… когда не выходит сосредоточиться, собрать свои мысли, когда паника возникает на ровном месте. Все эти ощущения – первые признаки

стресса… Я извиняюсь, я не очень опытный докладчик, и не уверен, что у меня получится сегодня хорошо перед вами выступить…..(пауза).

Ммм….Как же быть в такой ситуации? Ага… было бы неплохо научиться управлять стрессом. Вообще, я очень хорошо разбираюсь в этой проблеме. Я посвятил ее изучению около года, и написал уже несколько статей в популярные журналы. Поэтому, о чем бы вы меня не спросили сегодня – я смогу дать вам достойный ответ».

Задание 2. Публичное выступление

Подготовьте выступление, рассчитанное на 4-6 мин, придерживаясь модульного способа построения речи и структуры публичного выступления. Выступите перед одногруппниками.

Примерные темы:

- Честь, достоинство и деловая репутация
- Профессиональная этика ИТ-специалиста
- Должен ли искусственный интеллект соблюдать этические нормы?
- Как быстро справиться со стрессом
- Как научиться рисковать
- Как сделать свою жизнь интересной
- Как создать профессиональную команду
- Как найти работу своей мечты
- Как стать настоящим руководителем
- Как преодолеть прокрастинацию

Критерии оценки: соблюдение временного регламента, соблюдение правил модульного способа построения речи, структуры публичного выступления, содержательность выступления, демонстрация коммуникативных навыков, аргументированность, убедительность выступления, свободное владение материалом по теме выступления, культура выступления, ясность и четкость.

Методические рекомендации

Ознакомьтесь с теоретическим материалом по теме «Публичное выступление: подготовка и реализация». Ответьте на вопросы для самоконтроля. Выполняя *задание 1* психологического практикума, внимательно прочитайте текст и выпишите обнаруженные ошибки на отдельный лист.

Выполняя *задание 2*, придерживайтесь следующей общей структуры выступления. Введение должно занимать 10-20% всей речи. В основной части речи выдвигаются основные тезисы (эта часть занимает – 60-65%). Выделите главную идею вашей речи. Выделите подзаголовки, разделив вашу идею на несколько составных частей. Определите ключевые слова, которые вы повторите несколько раз, чтобы присутствующие лучше запомнили, о чем вы им рассказываете. В заключении обычно формулируются выводы, которые следуют из главной цели и основной идеи выступления, также обычно повторяются основная идея и возвращаются к тем моментам основной части, которые вызвали воодушевление и интерес слушателей. Заключение составляет 20-30% времени.

Литература

Иванова С.Ф. Специфика публичной речи. – М.: 1978.

Скаженик Е.Н. Деловое общение. Учебное пособие. Таганрог: Изд-во ТРТУ, 2006 [свободный доступ: http://www.aup.ru/books/m161/7_1.htm].

Эксакусто Т.В. Основы психологии делового общения: Учебное пособие. – Таганрог: Изд-во ЮФУ, 2015. – 162 с. [свободный доступ: http://pibg.tti.sfedu.ru/?cat=15].

Занятие 14. Коммуникация в деловом общении (Часть 2. Деловая переписка)

Цель: развитие навыков эффективной деловой коммуникации в процессе служебной переписки.

Задачи:

- развитие представлений о деловой переписке, деловом письме, о правилах их реализации;
- изучение структуры делового письма;
- развитие навыка эффективной служебной переписки;
- развитие навыков написания делового письма.

Теоретическое введение

Деятельность делового человека невозможно представить без работы с документами. Подсчитано, что на составление служебных документов и работу с ними у некоторых категорий работников аппарата управления тратится от 30 до 70 % рабочего времени.

Деловые (служебные) письма представляют собой официальную корреспонденцию и применяются для решения многочисленных оперативных вопросов, возникающих в управленческой и коммерческой деятельности. Деловое письмо – всегда официальное сообщение.

При этом служебная переписка является важной частью делового этикета, «общением в миниатюре». Она способствует установлению прочных связей с клиентами и партнерами, улучшению взаимосвязи различных служб, а также увеличению оборота предприятия.

Деловые письма можно классифицировать **по аспектам**:

• письмо-напоминание – факт напоминания;

• письмо гарантийное – выражение гарантии, документ, обеспечивающий исполнение изложенных в нем обязательств. В нем адресату обычно гарантируется оплата или предоставление чего-либо (места работы, проведения исследований и т. п.). Эти письма имеют повышенную правовую функцию, поэтому изложение текста должно быть предельно четким и ясным;

• письмо-подтверждение – указание на достигнутую степень согласия, свершившийся факт;

• письмо-ответ – по своему содержанию носит зависимый характер от инициативных писем, так как тема его текста уже задана и остается изложить характер решения поставленного в инициативном письме вопроса: принятие или отказ от предложения, выполнение просьбы;

• информационное письмо – информирование о намечаемых или уже проведенных мероприятиях;

• письмо-приглашение – письменное приглашение адресату принять участие в каком-либо проводимом мероприятии. Они могут адресоваться как конкретным лицам, так и учреждениям. В них раскрывается характер проводимого мероприятия, указываются сроки проведения и условия участия в нем.

• инициативное письмо – это письмо, требующее ответа; бо̲льшая категория таких писем выражает просьбу (предложение, запрос) к адресату в решении каких-либо вопросов;

• сопроводительное письмо – письменный текст, который информирует адресата о направлении документов, прилагаемых к письму;

• письмо-предупреждение – предупреждение о возможных ответных шагах и т. д.

Одно и то же письмо может содержать гарантию, просьбу и напоминание, т. е. быть *многоаспектным.*

По признаку адресата деловые письма делятся на *обычные* и *циркулярные*. Циркулярное письмо направляется из одного источника в несколько адресов.

По структурным признакам деловые письма делятся на *регламентированные* (стандартные) и *нерегламентированные*. Регламентированное письмо решает типичные вопросы регулярных экономико-правовых ситуаций и реализуется в виде стандартных синтаксических конструкций. Нерегламентированное деловое письмо представляет собой авторский текст, реализующийся в виде формально-логического повествования или этикетного текста.

Качество делового текста складывается из четырех составляющих: мысли, внятности, грамотности и корректности *(Шеламова, 2007)*.

Хорошо структурированное деловое письмо позволяет сэкономить время на его написание и прочтение, получить вразумительный, четкий ответ. Стандартное деловое письмо, на бумажном носителе, имеет следующую **структуру**:

1. *Реквизиты отправителя* включают в себя информацию об отправителе. Чаще всего эта информация располагается на бланке организации-отправителя.

2. *Реквизиты получателя* – указывается должность адресата, название компании, Ф.И.О. и адрес.

3. *Дата написания и отправления* письма. Позволяет различать письма между собой. Часто этот реквизит включен в фирменный бланк.

4. *Заголовок* – кратко о теме письма. Эта часть позволяет определить тематику письма и степень его важности для адресата. Важно правильно подобрать заголовок. При этом он должен быть строгий и лаконичный.

5. *Обращение.* Располагается посередине, и пишется с большой буквы. Обращение должно содержать указание должности, фамилии или имени и отчества. Традиционным считается обращение «Уважаемый».

6. *Преамбула или вводная часть.* В этой части поясняют причины и цели написания письма именно данному адресату. Часто в этой части ссылаются на даты, документы и факты, напоминающие адресату необходимую информацию. Информация, изложенная в преамбуле, должна подвести читателя к адекватному восприятию основной части письма. По объему эта часть варьируется: может быть краткой (одно предложение) и развернутой (несколько предложений и даже абзацев).

7. *Основная часть* текста письма варьируется в зависимости от его конкретного типа, так как в ней формулируется главная цель письма, излагаются аргументы и факты.

8. *Заключение* чаще всего представляет собой формулы вежливости.

9. *Подпись.* Письмо завершается подписью (должность + ФИО) адресанта, которую предваряет стандартная вежливая форма «С уважением». Также возможны варианты: «Искренне Ваш», «С надеждой на продуктивное сотрудничество», «С благодарностью за сотрудничество» и т.п.

10. *Постскриптум*. Это необязательный вспомогательный элемент. Служит для того, чтобы сообщить адресату о важном событии, которое произошло уже после написания письма, или передать ему информацию, которая имеет косвенное отношение к теме письма.

11. *Приложения* являются необязательным дополнением к основному тексту письма и поэтому оформляются на отдельных листах – каждое приложение на своем листе. Какие-либо правила их написания отсутствуют.

При составлении делового письма желательно руководствоваться следующими **правилами**:

- исполнитель должен отчетливо представлять себе сообщение, которое хочет передать, и точно знать, как это выразить в понятной, сжатой и доступной форме;
- письмо должно быть простым, логичным, конкретным и не допускать двусмысленностей. Лаконичные письма, написанные односложными словами, характеризуют пишущих как хороших собеседников, владеющих искусством общения. Фразы должны легко читаться, нежелательно использование большого количества причастных и деепричастных оборотов;
- письмо должно составляться только по одному вопросу, при этом его текст надо разбить на абзацы, в каждом из которых затрагивается лишь один аспект данного вопроса;
- письмо должно быть убедительным и достаточно аргументированным;
- письмо должно быть написано в нейтральном тоне, нежелательно употребление метафор и эмоционально-экспрессивных фраз;
- объем делового письма не должен превышать двух страниц машинописного текста;
- с точки зрения грамматики деловое письмо должно быть безупречным, так как орфографические, синтаксические и стилистические ошибки производят плохое впечатление и действуют на адресата раздражающе;
- деловое письмо должно быть корректным, написано вежливым тоном.

Специалист в области делового письма Р. Теппер полагает, что правильно составленные деловые письма строятся по одной схеме. Начальные строки привлекают **внимание,** следующие за ними одно или два предложения пробуждают **интерес** читателя, затем в двух абзацах высказывается **просьба,** а последняя часть заставляет читателя **действовать.**

Пример делового письма, составленного по данной схеме:

Внимание: «Уважаемая (ый) _____

Сообщаем информацию, которая Вас, возможно заинтересует»

Интерес: «Мы (Я) предлагаем(ю) Вам новые формы сотрудничества, которые могут существенно повысить эффективность выполняемой совместно с Вами деятельности...»

Просьба: «Для введения в работу новых форм сотрудничества нам требуется помощь людей, готовых на первом этапе вложить свой человеческий капитал в разработку ряда механизмов...»

Действие: «Мы призываем Вас присоединиться к нашей инициативе, затрагивающей интересы тысяч людей...»

Помните, что просьбу нужно формулировать так, чтобы выбор вариантов у адресата был ограничен, поскольку, чем меньше вариантов, тем больше вероятность успеха. Употребление стандартизированных словесных оборотов не только позволяет исключить ненужный эмоциональный тон письма, но и является выражением деловой вежливости *(Шеламова, 2007)*.

Интернет-переписка, наиболее часто используемая деловыми людьми на сегодняшний день, имеет свои особенности. При деловой переписке по электронной почте необходимо соблюдать следующие правила виртуального этикета:

- регулярно проверять содержимое своего почтового ящика;
- безотлагательно отвечать на каждое письмо, адресованное непосредственно вам; отсутствие ответа равносильно тому, как если бы вы проигнорировали приветствие, отказались бы пожать протянутую руку или повернулись бы спиной к своему собеседнику;
- соблюдать лаконичность, иногда вполне достаточно нескольких слов;
- в электронном послании всегда надо указывать его основную тему; необходимо учесть, что письма, не снабженные четко сформулированной темой, могут быть проигнорированы;
- высылая письмо незнакомому адресату, важно пользоваться обычным текстовым кодом, иначе не исключено, что он просто не сможет его получить;
- в конце каждого электронного послания обязательно следует указать свое имя, фамилию, должность и место работы, ваш номер электронной почты, а также телефон и обычный почтовый адрес, эти сведения не должны превышать четырех строк;
- нельзя перегружать электронное послание дополнительными материалами (фотографиями, рисунками и т.д.);
- большой объем дополнений высылайте только с согласия или по просьбе адресата. Уточните, имеет ли его почтовый ящик ограничения на принимаемую информацию;
- отвечая на письмо, не следует повторять всю корреспонденцию целиком, достаточно воспроизвести только те ее фрагменты, на которые вы хотите сослаться (такое цитирование вовсе не обязательно, но является определенным жестом вежливости по отношению к вашему виртуальному корреспонденту, который мог и забыть суть своего послания или отдельных аргументов в дискуссии);

- каждое электронное послание одного корреспондента другому является частным, поэтому каждым пользователем должно соблюдаться правило на тайну переписки, (т.е. прежде чем передать его содержание другим лицам, необходимо получить разрешение автора) (*Рамендик, 2003*).

Вопросы для самоконтроля

1. Что такое деловое письмо?
2. Из чего складывается качество делового текста?
3. Какими правилами желательно руководствоваться при составлении делового письма?
4. Что означает схема «внимание – интерес – просьба – действие»?
5. Чем отличается обычное письмо от циркулярного?
6. Каковы основные правила деловой переписки по электронной почте?

Практикум

Задание 1. Анализ деловых писем

Прочтите представленные ниже примеры деловых писем. Проанализируйте их на предмет классификации по следующим признакам: по аспектам, по признаку адресата, по структурным признакам.

Деловые письма для анализа *(Профессиональные шаблоны…, 2017):*

Письмо № 1

Уважаемые участники!

Просим вас ознакомиться с условиями участия в выставке «Город моими глазами» и подтвердить ваше согласие на участие в течение 5-ти дней.

Выставка состоится 09 февраля 2012 года с 9.00 до 19.00 на Гоголевском бульваре в здании «Фотоцентра» Союза журналистов России.

Минимальный размер изображения по короткой стороне – 20 см (напр. отпечаток 20х30), максимальный по длинной стороне – 45 см (напр. отпечаток 30х45).

Фотографии будут размещаться на стендах, на ватмане.

Каждый участник выставки может разместить до 10 работ. Просьба сообщать о количестве фотографий заранее.

Определиться в количестве и формате фотографий и выслать нам на электронный адрес exhibition@mail.ru для предварительного ознакомления их мини-изображения (preview 800х600 точек) с названием и фамилией автора вам необходимо до 01 февраля.

Готовые и утвержденные работы можно будет принести 07 февраля.

С уважением, оргкомитет

Приложение. Сведения о фотовыставке (жюри, партнеры).

Письмо № 2

Уважаемый Иван Иванович,

Спасибо за ваше недавнее резюме. Мы ценим ваш интерес к работе в нашей компании. У вас превосходная квалификация, но, к сожалению, у нас нет никаких доступных вакансий на данный момент.

Мы будем держать вас в курсе и, если в будущем появятся свободные вакансии, мы обязательно сообщим.

С уважением,

директор HR службы компании «Делопись.ру»,

Петр Петров

Письмо №3

Уважаемый Иван Иванович,

Спасибо за ваш интерес к компании «Делопись.ру». Мы получили ваше заявление и находимся в процессе рассмотрения вашего резюме. Если мы решим пригласить вас на интервью, то свяжемся с вами в пределах 15 дней.

С уважением,

старший специалист по работе с персоналом компании «Делопись.ру»,

Петр Петров

Письмо №4

Уважаемый Даниил Олегович,

С большим интересом прочитал Ваше объявление на сайте delopis.ru по поводу кандидатуры веб-дизайнера. Моя квалификация соответствует Вашим требованиям:

– более 3-х опыта в веб-дизайне

– 7 лет опыта в графическом дизайне

– опыт в использовании JavaScript, HTML, Abode Photoshop

– высшее образование

Дополнительно я обладаю превосходными организационными способностями и навыками в деловых переговорах. Эти качества должны позволить мне сделать ценный вклад в «Делопись.ру», если появится такая возможность.

Прилагаю резюме для Вашего обзора. Надеюсь встретиться с Вами, чтобы подробнее обсудить, чем я могу быть полезен Вашей компании.

Благодарю за Ваше потраченное время на рассмотрение моей кандидатуры.

С уважением,

Олег Мамонтов

Прилагаемый документ: Резюме

Задание 2. Подготовка деловых писем

Напишите самостоятельно два деловых письма на любую тематику, сопроводив их аналогичным анализом по представленным выше классификациям.

Методические рекомендации

Ознакомьтесь с теоретическим материалом по теме «Деловая переписка: правила, этика отношений». Ответьте на вопросы для самоконтроля.

При выполнении *задания 1* следуйте приведенному примеру:

Пример: анализ письма №2.

По аспектам это письмо-подтверждение и письмо-ответ; по признаку адресата это обычное письмо, по структурным признакам – регламентированное письмо.

При выполнении *задания 2* ознакомьтесь с подробным описанием структуры делового письма и правилами его составления. Важно соблюдать принципы и правила написания делового письма.

Литература

Рамендик Д.М. Деловое общение М.: Академия, 2003. – 219 с.

Профессиональные шаблоны деловых писем [Электронный ресурс] http://www.delopis.ru/

Шеламова Г.М. Деловая культура и психология общения. –М.: Издательский центр «Академия», 2007. –160 с.

Занятие 15. Взаимодействие в малой группе и управление коллективом (Часть 1. Взаимодействие в малой группе и преодоление конфликтов)

Цель: изучение особенностей взаимодействия в малой группе, анализ и преодоление конфликтов.

Задачи:

- знакомство с видами взаимодействия;
- изучение специфики делового взаимодействия в группе;
- знакомство с особенностями конфликта: признаки, этапы;
- анализ возможностей преодоления конфликтных ситуаций.

Теоретическое введение

Каждый член группы по-своему воспринимает группу, систему взаимоотношений в ней, свое положение в группе (которое довольно часто отличается от реального). Именно поэтому с точки зрения развития группы, формирования команды важно учитывать механизмы восприятия и понимания людьми друг друга: интерпретацию, идентификацию, рефлексию, аттракцию и т.п., а также типы взаимодействия, преобладающие у членов группы. Существуют различные виды взаимодействия (рис.6)

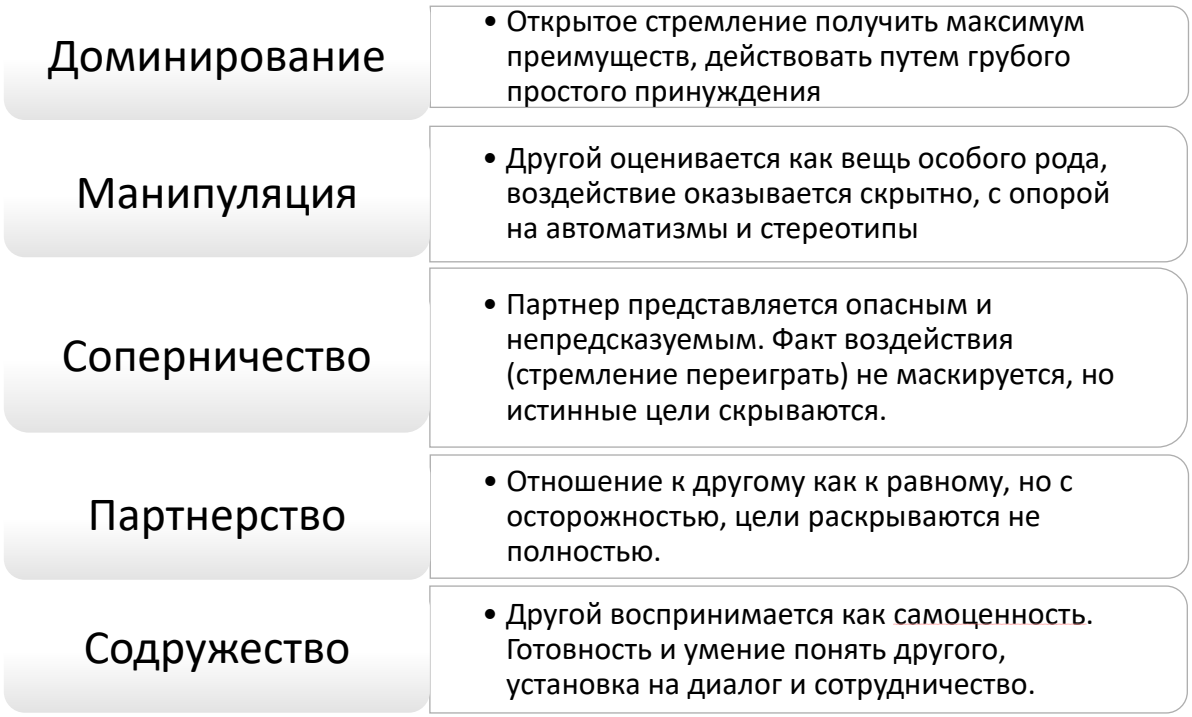


Рис. 6. Виды взаимодействия

Очевидно, что чем чаще в группе преобладает доминирование, манипуляция, соперничество, тем с большей степенью вероятности в группе будет возникать конфликтная ситуация.

Конфликт – это всегда столкновение интересов, целей или мотивов, при котором ни одна из сторон не желает отступать. Соответственно,

необходимыми и достаточными условиями возникновения (наступления) конфликта являются наличие у субъектов социального взаимодействия противоположно направленных мотивов или суждений, а также состояние противоборства между ними.

Причинами возникновения конфликтов могут быть различия целей, психологическая несовместимость партнеров, недостатки в организации переговоров, неудовлетворительные коммуникации, некомпетентность, неполномочность и многое другое. Причины конфликта – это явления, события, факты, ситуации, которые предшествуют конфликту и, при определенных условиях деятельности субъектов социального взаимодействия, вызывают его.

Среди огромного множества причин конфликтов прежде всего выделим так называемые *общие* причины, которые проявляются так или иначе практически во всех возникающих конфликтах. К ним можно отнести следующие причины.

Социально-политические и экономические причины связаны с социально-политической и экономической ситуацией в стране.

Социально-демографические причины отражают различия в установках и мотивах людей, обусловленные их полом, возрастом, принадлежностью к этническим группам и др.

Социально-психологические причины отражают социально-психологические явления в социальных группах: взаимоотношения, лидерство, групповые мотивы, коллективные мнения, настроения и т. д.

Индивидуально-психологические причины отражают индивидуальные психологические особенности личности (способности, темперамент, характер, мотивы и т. п.).

Вторую группу причин в нашей классификации назовем *частными.* Эти причины непосредственно связаны с конкретным видом конфликта. Здесь мы назовем лишь некоторые из них:

- неудовлетворенность условиями деятельности;
- нарушение служебной этики;
- нарушение трудового законодательства;
- ограниченность ресурсов;
- различия в целях, ценностях, средствах достижения целей;
- неудовлетворительные коммуникации.

Причины конфликтов обнаруживают себя в конкретных конфликтных ситуациях, устранение которых является необходимым условием разрешения конфликтов. Конфликтная ситуация – это накопившиеся противоречия, связанные с деятельностью субъектов социального взаимодействия и создающие почву для реального противоборства между ними.

Основные этапы конфликта

1. *Возникновение и развитие конфликтной ситуации.* Конфликтная ситуация создается одним или несколькими субъектами социального взаимодействия и является предпосылкой конфликта.

2. *Осознание конфликтной ситуации хотя бы одним из участников социального взаимодействия и эмоциональное переживание им этого*

факта. Следствиями и внешними проявлениями подобного осознания и связанных с ним эмоциональных переживаний могут быть: изменение настроения, критические и недоброжелательные высказывания в адрес своего потенциального противника, ограничение контактов с ним и т. д.

3. *Начало открытого конфликтного взаимодействия.* Этот этап выражается в том, что один из участников социального взаимодействия, осознавший конфликтную ситуацию, переходит к активным действиям (в форме демарша, заявления, предупреждения и т. п.), направленным на нанесение ущерба «противнику». Другой участник при этом сознает, что данные действия направлены против него, и, в свою очередь, предпринимает активные ответные действия против инициатора конфликта.

4. *Развитие открытого конфликта.* На этом этапе участники конфликта открыто заявляют о своих позициях и выдвигают требования. Вместе с тем они могут не осознавать собственных интересов и не понимать сути и предмета конфликта.

5. *Разрешение конфликта.* В зависимости от содержания, разрешение конфликта может быть достигнуто двумя методами (средствами): *педагогическими* (беседа, убеждение, просьба, разъяснение и т. п.) и *административными* (перевод на другую работу, увольнение, решения комиссий, приказ руководителя, решение суда и т. п.).

Рекомендации по разрешению конфликта

1. Признайте, что у вас есть проблема. Вы эффективно объясните ей свою проблему, если начнете со следующих слов: «Вы не могли бы мне помочь в одном деле...».

2. Опишите потенциальный конфликт с точки зрения поведения, которое вы наблюдаете, возможных последствий и ваших ощущений. Очень важно объяснить все эти три шага [поведение - последствия - чувства] другому человеку... «Когда я наблюдаю…[поведение], я вынужден отвлекаться от своих непосредственных обязанностей [последствия], и я испытываю досаду [чувства]».

3. Старайтесь не позволять другому человеку менять тему разговора.

4. Предложите разумное решение на основе общих ценностей.

5. Продумайте, что сказать, прежде чем вы столкнетесь с другим человеком, чтобы выразить проблему коротко и ясно.

Вопросы для самоконтроля

1. Какие признаки являются важнейшими в описании малых групп?
2. Назовите основные функции малой группы.
3. В чем состоит функция психологической поддержки? Как вы это понимаете?
4. Какие виды взаимодействия могут проявляться во внутригрупповых отношениях?
5. Какие виды взаимодействия потенциально могут приводить к появлению конфликта в группе?
6. Дайте определение конфликта?

7. Какие причины возникновения конфликта являются общими, а какие – частными?

8. Назовите основные этапы развития конфликта.

Практикум

Задание 1. Деловая игра «Маклер»

Цель деловой игры «Маклер» (Козлов, Козлова, 1999): анализ эффективных и неэффективных стратегий и тактических средств различных групп и отдельных участников в ходе коммуникативного взаимодействия.

В игре участвуют 4-5 групп (от 2 до 5 игроков в группе). Лидеры получают пакеты игровых билетов с названиями (номерами) групп (5-7 билетов каждой группы в пакете). Всего необходимо около 30 билетов.

Игровая задача: в случайном взаимообмене карточками с другими игроками группа должна собирать свои билеты и сдавать их посреднику у стенда результатов блоками по 5 штук, за которые она получает по 1 баллу за каждый блок. Новые билеты она набирает здесь же на столе раздаче, куда посредник складывает их в случайном порядке. Регламентация отношений за этим столом осуществляется игроками самостоятельно в процессе всей игры.

Игра завершается по сигналу ведущего, который может делать «рефлексивные паузы», тем самым обостряя ситуацию помощью аутсайдерам, которые получают возможность в этих паузах понять причины своих ошибок и оптимизировать дальнейшие действия.

Подведение итогов и анализ игры.

После подведения игровых итогов группам-участникам предлагается время на самостоятельный анализ по вопросам, соответствующим результатам каждой группы: почему проиграли? почему не проиграли и не выиграли? за счет чего одержали победу? При этом важно проанализировать какие виды взаимодействия они чаще использовали: была ли это манипуляция, или давление на другого, попытка выстроить партнерские отношения и т.п.

В общем хаотичном потоке поиска игроками своих билетов и экспресс-переговоров с последующим взаимообменом достаточно четко отслеживаются эффективные и неэффективные стратегии и тактические средства взаимодействия различных групп и отдельных участников.

Первая группа, условно называемая «неудачные игроки» характеризуется пассивностью и индифферентностью к целям и средствам предложенной «работы». Многих из них «хаотичность» маклерской работы выводит в состояние своеобразного «коммуникативного ступора». Напряженность и не структурированность этой деятельности не для них.

Вторая группа – «более-менее активные игроки», но их вклад в результативность группы отрицательный. Некоторые, набрав большое количество «чужих» билетов, долгое время держат их, увязнув в переговорах с такими же неуступчивыми. Ощущение такое, что им «жалко» расставаться с тем, что у них в руках даже на секунды обмена. Другие увлекаются иррациональной идеей обмена «именно с этим игроком» (или группой), или именно в данной комбинаторике. Частный вариант возможных действий

приобретает в сознании характер общей или обязательной стратегии. Обычно таких игроков большинство.

Третья категория – наиболее активные и точные в индивидуальной тактике и/или построении общей стратегии действий своей группы. Именно они и приносят наибольшее количество баллов, иногда больше, чем их партнеры вместе взятые.

Эти группы представляют три возможных уровня развития «оперативности социального мышления»: застревающий, ситуативный и системный. Каждый из уровней развития определяется рядом факторов.

Первый фактор – мобильность психических процессов в ситуациях экстремального социального взаимодействия. Противоположный эффект – застревание, «паузы» внимания, непродуктивно сконцентрированные на отдельных объектах или процессах, когда человек захвачен переживанием частных, несущественных деталей происходящего. Кроме потери времени он так же теряет возможность контроля за общим характером событий, «возврат» в которые требует уже времени на анализ новой ситуации. Если индивиду удается только в некоторых ситуациях достигать своих целей, можно говорить о «ситуативном» уровне развития оперативности социального мышления.

Второй фактор – скорость преодоления барьеров общения, под которыми понимается тот или иной способ распознавания человеком страха «отношений»: застенчивости, стеснительности, робости и т.п. Способ преодоления барьеров общения подразумевает совершение двух действий: снятие своих собственных возможных барьеров и эффективная помощь партнеру в аналогичном действии.

Третий фактор – особая зона игры – стол раздачи использованных билетов – это своеобразная модель социальной «кормушки». Действия игроков именно в этой зоне предопределяют игровой результат. «Социальная кормушка» – это совокупность социальных коммуникаций, доступ к которым обеспечивает преимущество в процессах перераспределения благ. В случае если о месте ее расположения и характере необходимых в ней действий известно большинству, то взаимодействие «игроков» приобретает достаточно жесткий характер до разворачивания манипулятивных и агрессивных тенденций, что будет продолжаться до тех пор, пока участники не договорятся о правилах поведения.

Четвертый фактор – особое качество лидерского поведения, – способность усиливать интеллектуальную продуктивность в экстремальных ситуациях. Внешняя хаотичность поля игры в действительности скрывает в себе логику социального движения, которая предопределяет его результат. Способность «читать» происходящее дает возможность участнику процесса предпринимать усилия по коррекции будущего. Чтение осуществляется по определенным знакам-действиям игроков, освоение этого словаря знаков-действий – собственно и является объектом социального научения.

Задание 2. «Паспорт проблемы»

Цели:

- отработать умение продуктивно действовать и принимать эффективные решения в кризисной ситуации;

- провести упражнение на анализ конфликта;

- проработать стратегию решения конфликта, разобраться в преимуществах и недостатках различных стратегий, проверить, какая из стратегий поведения в конфликте характерна для каждого из участников, оценить ее эффективность;

- потренироваться в определении четких целей деятельности, в выработке индивидуальной и совместной стратегии и тактики успеха.

Это задание касается различных способов восприятия конфликта и поведения в кризисной ситуации.

Данное задание индивидуальное, а не групповое.

Мы просим членов группы сосредоточиться каждому на той актуальной конфликтной ситуации, которую сами они представляют на нынешний момент как проблему, требующую решения. Чтобы разобраться в межличностном конфликте (именно этот тип конфликта будет материалом для работы) необходимо четко представить себе, с чем мы имеем дело. Составив паспорт проблемы, мы сможем разработать пути ее решения. Кроме того, ответы на поставленные вопросы помогут определить, какой вид направленности преобладает в нас.

Итак, каждый из участников получает бланк, на заполнение его дается 20-30 минут. В конце упражнения группа может собраться вместе и озвучить проблемы эмоционального, интеллектуального или иного свойства, которые проявились в ходе работы над заданием.

Бланк упражнения "Паспорт проблемы"

1. Сформулируйте и запишите, в чем, на ваш взгляд, состоит суть конфликта. _____

2. Разложите конфликт на составляющие:

– что происходит (процесс, действие, поведение сторон) _____

– к чему это приводит (чьи и какие потребности нарушены) _____

– чувства по этому поводу (ваша эмоциональная реакция на угрозу потребностям и на развитие конфликта)_____

3. Определите, что для вас более важно:

а) защита собственных потребностей, принципов, ощущение личного комфорта или

б) сохранение хороших отношений со второй стороной.

4. Какой стиль поведения в конфликтной ситуации вы предпочитаете?

Уход

Характеристика - нежелание сотрудничать, застенчивость.

Цель - воздержаться от конфликта.

Точка зрения - я не хочу говорить об этом.

Индивидуальное чувство удовлетворенности - проигрыш/проигрыш: ни одна из сторон не получает удовлетворения.

Удовлетворенность отношениями отсутствует, конфликт не разрешен.

Какое влияние оказывает на отношения - препятствует гармоничным отношениям: приводит к нагнетанию ситуации и обвинениям.

Когда нужно применять - возможен только временный выход из ситуации или когда вопрос не имеет большого значения.

Приспособление

Характеристика - сотрудничество, застенчивость.

Цель - не расстраивать человека.

Точка зрения - не так важно, чтобы все было по-моему. Важнее сохранить мир.

Индивидуальное чувство удовлетворенности - проигрыш/выигрыш: противоположная сторона получает удовлетворение.

Удовлетворенность отношениями отсутствует: ни одна из сторон не довольна процессом.

Какое влияние оказывает на отношения — причиняет вред отношениям, потому что один человек получает преимущество.

Когда нужно применять - для получения социального уважения или когда вопрос не имеет большого значения.

Принуждение

Характеристика — нежелание сотрудничать, самоуверенность.

Цель - настоять на своем.

Точка зрения - я добьюсь своего, независимо от того, что мне придется сделать.

Индивидуальное чувство удовлетворенности - выигрыш/проигрыш: одна из сторон (принуждающий) получает удовлетворение.

Удовлетворенность отношениями отсутствует: физическое и психологическое страдание проигравшего.

Какое влияние оказывает на отношения - причиняет вред отношениям, потому что один человек оказывается запуган.

Когда нужно применять — в неотложных ситуациях; когда это главный вопрос благополучия одного или многих людей: если кто-то вас использует в своих интересах.

Компромисс

Характеристика - частичное желание сотрудничать.

Цель - получить частичное удовлетворение.

Точка зрения - мои требования будут частично удовлетворены, и я в определенной мере учту интересы другого человека.

Индивидуальное чувство удовлетворенности — проигрыш/проигрыш или выигрыш/выигрыш: ни одна из сторон не получает полного удовлетворения.

Удовлетворенность отношениями — нейтральная или положительная: обе стороны получили по крайней мере частичное удовлетворение.

Какое влияние оказывает на отношения - может помочь или причинить боль, так как удовлетворение достигнуто в результате компромисса.

Когда нужно применять - вопрос не очень важен, когда времени не хватает или когда другие способы решения не эффективны.

Сотрудничество

Характеристика - взаимодействие, настойчивость.

Цель - совместно разрешить проблему.

Точка зрения - давай обсудим и найдем общее решение для обоих.

Индивидуальное чувство удовлетворенности - выигрыш/выигрыш: обе стороны удовлетворены процессом.

Удовлетворенность отношениями — позитивная: отношения укрепляются, потому что партнеры приобретают взаимные выгоды.

Какое влияние оказывает на отношения - укрепляет отношения, потому что выслушаны обе стороны.

Когда нужно применять - всегда.

5. Если, используя выбранный мною стиль поведения, проблема будет разрешена, конфликт снят, то:

Я _____

Другая сторона_____

(Сформулируйте, какие конкретные результаты ожидаются, какова будет эмоциональная доминанта в ощущениях обеих сторон по завершению конфликта, как прогнозируются отношения между сторонами.)

6. Я смогу сказать, что проблема, конфликт разрешены, когда (если) _____

Методические рекомендации

Ознакомьтесь с теоретическим материалом по теме «Взаимодействие в группе и преодоление конфликтов». Ответьте на вопросы для самоконтроля. Проанализируйте результат *задания 1* (деловой игры): что получилось, что не получилось; какие виды взаимодействия вы чаще используете в повседневной жизни. Удалось ли наладить психологический контакт и эффективное взаимодействие? Что могло привести/привело к возникновению напряженной ситуации/конфликта?

Создайте «паспорт проблемы» *(задание 2)*. Проанализируйте что можно сделать в сложившейся ситуации. Готовы ли вы предпринять шаги по преодолению этого конфликта?

Литература

Бакирова Г. Х. Тренинг управления персоналом. – СПб.: Речь, 2006.

Левин К. Разрешение социальных конфликтов. – СПб.: Речь, 2000. – 408 с.

Скотт Д.Г. Способы разрешения конфликтов: пер. с англ. – Киев: Изд-во общества Верзилин, ЛТД, 1991. – 206 с.

Занятие 16. Взаимодействие в малой группе и управление коллективом (Часть 2. Формирование команды)

Цель: изучение принципов формирования команды, основ формирования сплоченности группы.

Задачи:

- развитие представлений о психологии малой группы и команды;
- совершенствование знаний о сплоченности в малой группе, особенностях формирования команды;
- развитие навыков формирования команды и командного взаимодействия.

Теоретическое введение

Команда – это объединение людей, между которыми четко разделены функционально-ролевые обязанности, рабочие действия, ответственность за конкретные результаты. Команду, как правило, характеризует: ответственность, взаимозависимость, равноправие, результат (Тенянко, Шинкарь, 2012). Эффективная команда всегда нацелена на результат, инициативна, проявляет креативность в решении групповых задач, имеет высокую производительность, характеризуется активностью и заинтересованностью при обсуждении возникающих проблем, ориентирована на лучший вариант решения задач (Управление персоналом.., 2002). Команда в сравнении с деятельностью отдельного человека или обычной малой группы:

способна решать достаточно сложные профессиональные задачи и проблемы;

стремится к согласованности и консенсусу для повышения эффективности деятельности;

преодолевает ситуации неопределенности и множественности вариантов решения поставленных задач;

владеет широким диапазоном социально-психологических и профессиональных компетенций;

имеет высокую самоотдачу;

способствует реализации индивидуальных целей через достижение групповых;

способна генерировать разносторонние взгляды при решении проблем для разработки перспективной стратегии.

Команда должна быть сплоченной, иметь благоприятный социально-психологический климат, а члены команды должны быть совместимы, доверять друг другу (испытывать чувство безопасности).

Групповая сплоченность – процесс формирования особого типа связей в группе, которые позволяют внешне заданную структуру превратить в психологическую общность людей, в сложный психологический организм, живущий по своим собственным законам. Понятие «сплоченность» используется для обозначения таких социально-психологических характеристик малой группы, как степень психологической общности, единства

членов группы, теснота и устойчивость межличностных взаимоотношений и взаимодействия, степень эмоциональной привлекательности группы для ее членов. Сплоченность тесно связана с такой переменной, как совместимость. Совместимость – это возможность членов группы эффективно взаимодействовать друг с другом для обеспечения выполнения группой ее функций и профессиональной деятельности. Совместимость предполагает наличие бесконфликтного общения и согласованных действий членов группы в условиях совместной деятельности (Практикум по социальной психологии, 2008). В группе совместимость складывается из иерархии ее уровней: физиологический уровень; психофизиологический уровень; психологический уровень; социально-психологический уровень. Психологический и социально-психологический уровни совместимости являются наиболее значимыми для организации групповой деятельности. С этой позиции групповая совместимость выступает как согласованность функционально-ролевых ожиданий − представлений членов группы о том, что должен делать каждый при реализации общегрупповой цели.

Вопросы для самоконтроля

1. Какая структура отражает своеобразие индивидуального состава группы?
2. Какую систему связей в малой группе отражает коммуникативная структура?
3. Что представляет собой функционально-ролевая структура малой группы?
4. Какие вы знаете теории сплоченности?
5. Как называется процесс, при котором индивид прикладывает ряд усилий в отношении группы, чтобы максимизировать ее вклад в удовлетворение личных нужд?
6. Дайте определение понятия «команда».
7. Какие признаки команды вы можете назвать?
8. В чем преимущества профессиональной команды в сравнении с обычной малой группой?
9. Назовите основные принципы формирования команды.
10. Что такое групповая сплоченность и совместимость?
11. Какие выделяют уровни совместимости?

Практикум

Задание 1. Изучение особенностей малой группы

Проведите диагностику малой группы, членом которой вы являетесь, напишите заключение по результатам диагностики.

Методика определения индекса групповой сплоченности (Сишор)

Цель: изучение уровня групповой сплоченности как показателя интеграции группы, ее психологического единства.

Методика состоит из 5 вопросов с несколькими вариантами ответов на каждый. Членам группы предлагается ответить на вопросы, выбрав тот вариант

ответа, который наиболее точно отражает мнение человека о своем членстве в группе. Ответы кодируются в баллах согласно приведенным в скобках значениям. В ходе опроса баллы указывать не нужно.

Текст опросника

1-й вопрос. Как бы вы оценили свою принадлежность к группе?

1. Чувствую себя ее членом, частью коллектива.
2. Участвую в большинстве видов деятельности.
3. Участвую в одних видах деятельности и не участвую в других.
4. Не чувствую, что являюсь членом группы.
5. Живу и существую отдельно от нее.
6. Не знаю, затрудняюсь ответить.

2-й вопрос. Перешли бы вы в другую группу, если бы представилась такая возможность (без изменения прочих условий)?

1. Да, очень хотел бы перейти.
2. Скорее перешел бы, чем остался.
3. Не вижу никакой разницы.
4. Скорее всего остался бы в своей группе.
5. Очень хотел бы остаться в своей группе.
6. Не знаю, трудно сказать.

3-й вопрос. Каковы взаимоотношения между членами вашей группы?

1. Лучше, чем в большинстве коллективов.
2. Примерно такие же, как и в большинстве коллективов.
3. Хуже, чем в большинстве классов.
4. Не знаю, трудно сказать.

4-й вопрос. Каковы у вас взаимоотношения с руководством?

1. Лучше, чем в большинстве коллективов.
2. Примерно такие же, как и в большинстве коллективов.
3. Хуже, чем в большинстве коллективов.
4. Не знаю.

5-й вопрос. Каково отношение к делу (учебе и т. п.) в вашем коллективе?

1. Лучше, чем в большинстве коллективов.
2. Примерно такие же, как и в большинстве коллективов.
3. Хуже, чем в большинстве коллективов.
4. Не знаю.

Обработка результатов

После того, как все члены группы ответят на вопросы, необходимо посчитать количество баллов, набранных каждым человеком. Для этого необходимо воспользоваться ключом, где номер выбранного ответа на соответствующий вопрос, оценивается определенным количеством баллов. Например, если человек на первый вопрос выбрал первый вариант ответа, ему засчитывается 1 балл. Максимальная сумма, которую может набрать человек – 19 баллов, минимальная –5 баллов. Затем суммируются результаты всех членов группы и вычисляется среднее арифметическое (т.е. полученная сумма делится на количество членов группы, принявших участие в исследовании).

Ключ

№ выбранного ответа	1-й вопрос баллы	2-й вопрос баллы	3-й вопрос баллы	4-й вопрос баллы	5-й вопрос баллы
1	1	1	3	3	3
2	2	2	2	2	2
3	3	3	1	1	1
4	4	4	1	1	1
5	5	5	-	-	-
6	1	1	-	-	-

Интерпретация результатов

15,1 баллов и выше – высокий уровень групповой сплоченности;

11,6 – 15 балла – групповая сплоченность выше среднего;

7– 11,5 – средняя групповая сплоченность;

5 – 6,9 – групповая сплоченность ниже среднего;

4 и ниже – низкая групповая сплоченность.

Задание 2. Тренинг формирования команды и командного взаимодействия

«Групповое рисование»

Цель. Создание атмосферы взаимодействия и взаимопонимания, усиление сплоченности.

Участники делятся на две подгруппы (строятся по росту, цвету волос, цвету глаз, одежды, возрасту, номеру дома и т.д., затем рассчитываются на первый-второй). Каждая подгруппа получает лист ватмана, набор карандашей, красок и начинает групповой рисунок. Причем через каждые три минуты члены команды по кругу меняются местами, и каждый продолжает рисовать рисунок предыдущего участника. Работа идет в течение 15 минут.

Примечания: в ходе упражнения не даются точные указания, лучше дать следующую инструкцию: «Вот лист. На нем одновременно работают все члены группы. Рисуете, что хотите и чем хотите. По моей команде все меняетесь местами. Начали работу».

«Включение позитивной мотивации»

Цель: отработка навыков проявления эмоций, способствующих процессу профессиональной адаптации.

Тренер предлагает участникам занять удобное положение, закрыть глаза и расслабиться; сосредоточиться на своих ощущениях. Далее тренер задает вопрос для всех: «Что делает вашу жизнь интересной (радостной, творческой и т.п.)? Что является «спусковым крючком» для возникновения интереса (радости, творчества и т.п.) в вашей жизни?»

Через некоторое время (5–7 минут), даваемое для индивидуальной визуализации заданной темы и выхода из состояния расслабленности, тренер предлагает участникам обсудить в группе результаты. Затем полученные сведения можно использовать для активизации, мотивации членов команды.

«Психогимнастика с мячом»

Цель: отработка навыка принятия группового решения о стратегии и тактике выполнения поставленной задачи. Способствовать сплочению группы и углублению процессов самораскрытия.

Инструкция: «Передайте в кругу, в любом порядке, кроме соседа справа и соседа слева мяч, но так, чтобы мяч побывал у каждого члена команды 1 раз».

Усложнение: сделать то же самое, но на время; предложить ускорить результат; выполнить любым другим способом на время. Ведущий предлагает всем участникам команды после завершения упражнения сесть в круг и выразить свое состояние на момент начала работы и ее окончания. В процессе сбора обратной связи стоит обсудить следующие вопросы:

– выработка командной стратегии выполнения задания;

– понимание идеи упражнения;

– понимание других участников;

– принятие решений;

– изменения в поведении;

– изменение на эмоциональном уровне и активности участия каждого.

«Три степени доверия»

Цель: диагностика уровня доверия, развитие доверия в команде.

Группа делится на две равные части. Одна половина участников становится в круг, закрывает глаза и берет друг друга за руки. Участники из второй подгруппы располагаются у них за спинами. Первая степень доверия: те, кто находятся внутри, падают наружу. Те, кто снаружи, удерживают их на каком-то расстоянии. Вторая степень доверия: то же самое, но не держась за руки. Третья степень доверия: люди во внутреннем круге с закрытыми глазами поворачиваются на 180 градусов и падают лицом наружу. При каждой пробе люди в наружном круге меняются местами.

Вопросы обратной связи:

Легко ли было выполнять данное упражнение?

Какие возникали трудности?

На каком уровне доверия было легче/сложнее всего?

Упражнение «Фломастеры»

Цель: развитие навыков решения групповой задачи.

Материалы: фломастеры – по количеству человек в группе.

Комната делится прямой линией на две игровые зоны – «зона обсуждения» и «зона молчания». В «зоне молчания» на полу находятся фломастеры.

Задание: поднять одновременно с пола все предметы, при этом:

– человек может касаться только одного предмета,

– в воздухе одновременно может находиться только один предмет.

Правила:

1. У группы на выполнение задания есть 20 минут.

2. Когда группа готова выполнять задание, она сигнализирует ведущему. Ведущий дает команду: «Время». Группа переходит в «зону молчания» и начинает выполнять задание.

3. При нарушении задания ведущий говорит об этом, группа переходит в «зону обсуждения», ведущий приводит в исходное положение материалы (соединенные фломастеры разъединяются).

4. Участники группы могут разговаривать, только когда все они

находятся в «зоне обсуждения». За нарушение этого правила – штраф 1 минута.

5. Когда задание выполнено, ведущий говорит об этом группе.

Примечание для ведущего: два и более фломастера, устойчиво соединенных вместе, рассматриваются как один предмет. Участники должны догадаться об этом самостоятельно.

Упражнение «Веревка»

Цель: развитие навыков командного взаимодействия.

Материалы: веревка длиной 2 метра.

Исходное положение участников: участники стоят в кругу, держась за руки. На плечо одному из них ведущий вешает смотанную кольцом веревку.

Задание: необходимо, чтобы веревка обошла весь круг, при этом:

– участникам нельзя разговаривать друг с другом;

– участникам нельзя расцеплять руки;

– участникам нельзя передавать веревку с помощью рук;

– при падении или разматывании веревки упражнение выполняется сначала;

– когда задание выполнено, ведущий говорит об этом группе.

Вариант выполнения задания, при котором участники сами обойдут круг, не передавая веревки друг другу, считается правильным. Участники должны догадаться об этом возможном варианте самостоятельно.

Методические рекомендации

К заданию 1. После проведения методики, сделайте заключение о степени сплоченности вашей группы. Опишите композиционный состав группы: количество человек, их социальный статус, роли, которые они выполняют в группе. Проанализируйте часто встречающиеся ответы членов группы (попробуйте понять их причины).

К заданию 2. Выполните тренинговые упражнения, ответьте на вопросы обратной связи после каждого упражнения, проанализируйте свою группу с точки зрения признаков команди и командного взаимодействия.

Литература

Тенянко М.Ю., Шинкарь Ю.Ю. Формирование команды в организации //Социально-экономические и финансовые механизмы обеспечения инновационного развития экономики: тез. докл. III Междунар. науч.- практ. конф. – Минск: ГИУСТ БГУ, 2012. – 214 с.

Управление персоналом: учебник для вузов / под ред. Т.Ю. Базарова, Б.Л. Еремина. – 2-е изд., перераб. и доп. – М.: ЮНИТИ, 2002. – 560 с.

Фетискин Н.П., Козлов В.В., Мануйлов Г.М. Социально-психологическая диагностика развития личности и малых групп. М: Психотерапия, 2009. – 544 с.

Хасанова Г. Б., Исхакова Р. Р. Психология управления трудовым коллективом: учебное пособие / Казань: Издательство КНИТУ, 2012. – 260 с.

Занятие 17. Принятие групповых решений (Часть 1. Групповые задачи и их моделирование)

Цель: изучение основ постановки групповой задачи и методов ее решения.

Задачи:

- изучить различные типологии групповых задач;
- рассмотреть влияние характеристик задачи на групповой процесс;
- развить умение анализировать групповые задачи;
- обсудить социально-психологические феномены и распределение ролей сопровождающие решение групповой задачи.

Теоретическое введение

Понятие и виды групповых задач

Задача рассматривается как источник и объект процесса принятия группового решения. Групповое действие, как правило, качественно и количественно превосходит действие так называемого среднего индивида. Но для успеха группы в ней должен обязательно находиться человек, превосходящий своих партнеров в изобретательности и компетентности и, что весьма существенно, обладающий способностью к лидерству.

В самом элементарном выражении задачи можно классифицировать по двум категориям: простые и сложные. Сложность задачи определяется тем, насколько быстро и легко удается специалисту освоить критические операции, необходимые для ее выполнения. Таким образом, чтобы определить сложность задачи, надо раздробить задачу на отдельные мелкие фрагменты и оценить количество и сложность отдельных действий, требующихся для ее выполнения.

К факторам, определяющим сложность задачи относят: неопределенность событий и объектов окружающего мира; новизна событий и объектов окружающего мира; неожиданность событий и объектов окружающего мира; неполнота информации; неясность информации; неточность информации; дефицит времени; степень неопределенности, уникальность подзадач (Стрелков, 1998).

Выделяют также конъюнктивные (требующие кооперации усилий и координации действий) и дизъюнктивные (не требующие кооперации усилий и координации действий задачи).

По классификации Д. Хакмена и Ч. Морриса выделяют следующие виды задач: продукционные, дискуссионные, проблемные.

Измерения групповой задачи (М. Шоу).

- трудность (величина усилия, требуемого для выполнения задачи);
- множественность решения (сложное измерение, включающее набор возможных приемлемых решений, альтернативы выполнения задачи, степень верификации приемлемых решений);
- внутренний интерес к задаче (степень, в которой задача сама по себе представляет интерес для членов группы, побуждая их активность);

• требования кооперации (степень интеграции действий членов группы для выполнения задачи);

• интеллектуально-манипулятивные требования (диапазон требований к решению задачи от чисто умственного до чисто двигательного характера);

• популяционное знакомство (та степень, в которой члены группы уже сталкивались с подобной задачей в жизни).

Влияние характеристик задачи на групповой процесс

Зависимость группового поведения от тех или иных типов задачи выявлялась в сериях лабораторных экспериментов (Д. Хакмен, Ч. Моррис). Было обнаружено, что решение проблемных задач характеризовалось сильной ориентацией на действие, продукционных задач – значительным уровнем оригинальности выдвигаемых идей, а дискуссионных задач – высокой степенью включенности испытуемых в обсуждаемый вопрос.

Такая характеристика, как трудность задачи, также влияет на групповое функционирование. По мере роста трудности задачи увеличивается время ее решения и ухудшение группового продукта. Наряду с этим, рядом исследований установлено, что трудность задачи может усиливать мотивацию членов группы и вести к большей самоорганизации группы путем развития в ней процесса лидерства, влияя тем самым на рост эффективности групповой деятельности.

Важной промежуточной переменной, связывающей трудность задачи с успешностью ее выполнения, является обратная связь в процессе группового функционирования. Позитивная обратная связь, т.е. вызывающая удовлетворенность членов группы, способствует продуктивности решения групповой задачи вне зависимости от степени ее трудности.

Решение групповой задачи активизирует проявление разнообразных социально-психологических феноменов:

феномен нормализации групп;

феномен «сдвига риска»;

феномен качества решений;

феномен «группового духа»;

феномен «поляризации»;

феномен «огруппления» мышления (эффект подражания); и др.

Вопросы для самоконтроля

1. Какие групповые задачи выделяются по критерию сложности? Каковы их отличия?
2. Каковы особенности конъюнктивных и дизъюнктивных задач?
3. Что является промежуточной переменной, связывающей трудность задачи с успешностью ее выполнения?
4. Какой вид измерения групповой структуры дает представление о субординированности позиций индивидов в системе официальных отношений в малой группе:
5. Какой феномен рассматривается как преимущественно социальный по

своей природе, регламентировано функционирующий в системе формальных (официальных, служебных) отношений людей с целью упорядочения, организации этих отношений, управления ими для решения групповых задач?

6. Что относят к основным критериям (измерениям) групповой задачи?

Практикум

Задание 1. Анализ групповых задач

Ознакомьтесь с представленными ниже проектными групповыми задачами. Классифицируйте групповые задачи по разным основаниям. Какие феномены группового принятия решений наиболее вероятны в этих ситуациях? Ответ обоснуйте.

1. Задача: Разработать патент на тему «Инновационные солнечные батареи».

Исполнители: группа из 5 аспирантов технического вуза, не имеющих опыта в написании патентов, работающих в разных направлениях (разработчики и программисты).

2. Задача: Написание учебника для студентов по использованию пакета прикладных программ для решения задач технических вычислений.

Исполнители: группа преподавателей технического вуза во главе с профессором, имеющие опыт в написании учебно-методических пособий, учебников.

3. Задача: Разработка проекта внедрения информационной системы на предприятии (сеть включает в себя 35 компьютеров).

Исполнители: группа системных инженеров, состоящая из 5 человек с опытом работы более 3 лет.

4. Задача: Написание проектного задания по психологии в ограниченное время.

Исполнители: группа студентов-психологов 5 курса, состоящая из 5 человек.

5. Задача: подготовка доклада по тематике «Компьютерные технологии» для презентации на научной конференции.

Исполнители: группа студентов 4 курса технического вуза в составе 3 человек, часть из них имеют подобный опыт.

6. Задача: обследование пациента больницы и установление правильного диагноза.

Исполнители: группа интернов медицинского вуза в составе 3 человек под наблюдением и руководством более опытного врача, недавно поступивших в интернатуру, не имеющие опыта в решении подобных задач, но имеющих общее представление о проведении диагностики в случаях с подобной симптоматикой.

7. Задача: обследование пациента больницы и установление правильного диагноза.

Исполнители: группа ординаторов, имеющих опыт в решении подобного рода задач.

Задание 2. Разработка и решение групповой задачи

Часть 1. Придумайте и классифицируйте научно-технические/ исследовательские/ проектные групповые задачи. Какие методы принятия группового решения могут быть использованы при решении этих задач. Ответ обоснуйте.

Работа выполняется в подгруппах студентов на практическом занятии.

Пример:

1. Задача: Разработка проекта внедрения нового оборудования на производстве.

Исполнители: группа специалистов ПКО в количестве 5 человек, имеющих опыт решения подобных задач.

Для данной группы специалистов поставленная задача будет:

– простая

– конъюнктивная

– дискуссионная.

2. Задача: Разработка проекта внедрения нового оборудования на производстве.

Исполнители: группа студентов технического ВУЗа, не имеющие опыта решения подобных задач, но имеющие представления о производстве.

Для данной группы поставленная задача будет:

- сложная

- конъюнктивная

- проблемная.

Часть 2. Выполняется после того как студенты придумают задачу, которая в дальнейшем «отрабатывается» (решается) с помощью организационно-деятельностной игры «Круги Нейпера».

«Круги Нейпера»

1. Тема (проблема, ситуация)

«Создание проекта трудовым коллективом» (на основе придуманной задачи)

2. Концепция игры.

1. Создается 2 группы (внешний и внутренний круг).
2. Проводится 5 актов, по 5 минут на каждую группу.
3. После 3-го акта обратная связь в парах.
4. Индивидуальная работа: ответы на вопросы анкеты (5 минут).
5. Корреляция данных.
6. Обсуждение в смешанных мини-группах (20 минут).
7. Презентация продукта.
8. «Диверсионный анализ»: анализ продукта, контроль результатов.

Результатом игры должно стать решение 3-х задач.

1. Создать документ для руководителя (инструкции, алгоритма действий и т.д.) по организации групповой работы (совещание, дискуссия, обсуждение).
2. Стать максимально эффективной командой.
3. Создать исследовательский проект/решить поставленную задачу.

Этапы игры

I. Подготовительный этап

1. Постановка целей и задач:

а) для руководителя;

б) для сотрудников;

в) сшивка целей совещания со стратегическими целями компании.

2. Выработка критериев результата.

3. Формирование состава участников, разработка программы.

4. Определение места, времени, регламента.

5. Предоставление информации участниками:

а) информирование о сроках, регламенте, формате совещания.

б) постановка задачи сотрудникам по информационной подготовке.

II. Содержательная часть

1. Создание рабочей атмосферы:

а) приветствие, позитивный настрой, мотивация на работу;

б) выяснение степени готовности к работе.

2. Оглашение повестки дня (цели, регламент и формат работы).

3. Организация обсуждения:

а) анализ реальности (предоставление слова каждому участнику);

б) поиск возможностей (мозговой штурм, ранжирование возможностей).

4. Принятие решения, составление плана действий.

5. Определение сроков, распределение ответственности, предоставление полномочий.

6. Разработка системы и формы отчетности.

III. Заключительный этап

1. Подведение итогов:

а) участниками (обратная связь);

б) руководителем.

2. Снятие напряжения:

а) поблагодарить за работу;

б) выражение поддержки.

3. Планирование сроков и темы следующего совещания.

3. Роли

- Руководитель проекта.
- Участники проекта: распределение по социальным ролям (критик, фасилитатор, организатор, исполнитель и т.п.) происходит в процессе создания проекта.

4. Ожидаемые результаты

- Умение ставить цели и разрабатывать стратегию.
- Умение ставить задачи и разъяснять цели подчиненным (уменьшать сопротивление).
- Умение выявлять личные цели членов команды и встраивать в общегрупповые цели.
- Навыки взаимодействия с другими.
- Построение системы мотивации (нематериальной) и поддержки.

- Умение организовать членов группы так, чтобы каждый понимал свою роль в организации и при этом действовал и работал самостоятельно.
- Умение организовать работу так, чтобы члены группы проявляли инициативу.
- Умение контролировать результаты деятельности.
- Избежать манипуляций в руководстве.
- Делегирование полномочий.
- Выработка собственного эффективного стиля руководства.
- Способы разрешения конфликтов в организации.
- Управление временем.

Групповое обсуждение результатов игры проводится с опорой на следующие вопросы: «Что способствовало эффективной работе в группе?», «Что мешало эффективной работе в группе?».

Критерии оценки:

Правильное составление и эффективное решение групповой задачи; умение использовать правила аргументации, убеждения, ассертивного спора; использование приемов организации групповой дискуссии; способность анализировать ситуацию, делать выводы; проявление лидерских качеств; способность организовать работу малой группы (распределить функционально-ролевые обязанности) в ходе решения задачи, контролировать результаты; проявление навыков активного слушания; проявление коммуникативных навыков, навыков установления и поддержания контактов; демонстрация навыков решения реальных проблем/задач; нацеленность на сотрудничество; демонстрация профессионального мышления; активность; способность работать в команде

Методические рекомендации

К заданию 1. Ознакомьтесь с теоретическим материалом по теме «Моделирование групповой задачи и ее решение» и с подробным описанием феноменов, сопутствующих принятию группового решения, в учебном пособии Т.В. Эксакусто «Основы психологии малых групп и управления коллективом», параграфы 2.3-2.4. Ответьте на вопросы для самоконтроля. Выполните задание психологического практикума, опираясь на предложенные примеры.

Примеры

Задача: Разработка проекта внедрения нового оборудования на производстве.

Исполнители: группа специалистов ПКО в количестве 5 человек, имеющих опыт решения подобных задач.

Для данной группы специалистов поставленная задача будет: простая, конъюнктивная, дискуссионная.

Возможные феномены группового принятия решений:

– социальная фасилитация (работа в группе повышает эффективность работы отдельного индивида)

– огруппление мышления (для экономии времени могут приниматься не самые удачные решения, а те, которые поддерживаются лидером или несколькими членами группы).

К заданию 2. При составлении групповых задач для разных видов групп учитывайте следующую информацию:

- разновидность группы (формальная/неформальная, слаборазвитая/высокоразвитая, временная/постоянная и т.п.);
- возможные процессы и групповые состояния (сплоченность, наличие лидеров, групповую активность, групповую мотивацию и т.п.);
- групповые феномены (групповой синергии, групповой лени, эффект маятника, эффект волны и т.п.);
- сложность задачи;
- наличие опыта решения подобных групповых задач группой;
- время, необходимое для ее решения (существует ли лимит времени или нет);
- готовность группы к решению задачи и т.п.

Литература

Стрелков Ю.К. Инженерная и профессиональная психология. М.: Изд-во МГУ, 1998.

Эксакусто Т.В. Основы психологии малых групп и управления коллективом: Учебное пособие. – Таганрог: Изд-во ЮФУ, 2016.

Занятие 18. Принятие групповых решений (Часть 2. Методы принятия групповых решений)

Цель: изучение и анализ методов принятия группового решения, оценка функционально-ролевой позиции в ходе принятия решения.

Задачи:

- изучение особенностей групповой дискуссии, мозгового штурма, синектики как методов принятия группового решения;
- развитие навыка решения групповых задач с использованием различных методов принятия группового решения;
- анализ эффективности своей ролевой позиции в ходе принятия группового решения, своей коммуникативной и социальной компетентности.

Теоретическое введение

Эффективность принятия группового решения определяется не только проявляющимися феноменами, сопровождающими этот процесс, но и, прежде всего, методами, которые применяются для решения групповой задачи. К основным методам принятия группового решения относятся: групповая дискуссия, синектика, мозговой штурм, фокус-группа и др.

Групповая дискуссия. В ней должны принимать участие все члены группы. Для того чтобы дискуссия была достаточно эффективной, необходимо с самого начала четко определить ее цель. Она может быть различной – от простого обмена мнениями по какому-либо вопросу до принятия и реализации сложного решения. Метод групповой дискуссии особенно важен при обсуждении вопросов, по которым не существует единого мнения и не может быть единственно правильной точки зрения. Смысл коллективного обсуждения таких вопросов состоит не в том, чтобы прийти к их однозначному решению, а в том, чтобы уяснить суть вопроса и его возможные решения, оценить и взвесить их. Главное же в дискуссиях то, что члены группы учатся логически рассуждать, излагать и отстаивать свое мнение, убеждать и слушать других, т.е. обучаются эффективному личному и деловому взаимодействию. Предметом групповой дискуссии, проводимой в организации, могут быть вопросы планирования, оценка новых идей, анализ рынков сбыта, разработка частных стратегий, обсуждение текущих событий и подготовка к ним, технологии функционального взаимодействия между подразделениями или отдельными специалистами, трудовые споры и конфликты между сотрудниками, новые методы работы и многое другое.

Синектика – это метод сознательного применения подсознательного психологического механизма в малой группе, с целью постановки и решения проблем. Особенность этого метода – это попытка отойти от стандартного подхода в решении групповых задач. Синектика определяет процесс принятия группового решения как умственную активность в ситуации постановки и решения творческой задачи, где результатом является творческое ее решение. Соответственно, процесс принятия решения в синектической группе носит

иррациональный характер, а сами решения – рациональный. Главный итог синектики – найти правильное и конкретное решение поставленной проблемы (задачи).

Мозговой штурм – представляет собой метод активизации творческого мышления в группе. Для этого используются приемы снижения критичности и самокритичности у людей, работающих над решением творческой задачи. Для этого используют как прямую инструкцию, дающую установку на свободное от оценок и самооценок выражение творческих идей, так и создание специальных условий работы в группе: запрет на критику предложений по решению проблемы, поощрение комбинирования и развития идей друг друга и т.п. Цель этого метода – выработка нескольких возможных вариантов решения поставленной проблемы (задачи).

Фокус-группа – это групповое интервью, целью которого является получение качественных данных по исследуемой проблеме. Как правило, все участники фокус-группы имеют отношение к обсуждаемой ситуации, событию, виду деятельности. В процессе работы внимание участников фокус-группы акцентируется на их субъективном опыте. Отвечая на вопросы ведущего, они демонстрируют свое понимание, определение и объяснение обсуждаемой темы или проблемы.

Изучая мнения участников фокус-группы по определенному предмету, ведущий действует по строгому плану и стремится получить сумму максимально подробных индивидуальных точек зрения. Обсуждение в фокус-группах ведется свободно и недирективно, с высокой степенью вовлеченности участников, а сам ведущий фокус-группы выступает в качестве ее модератора. Важно отметить, что фокус-группы используются как метод исследования, мониторинга, изучения и прогноза развития ситуации, но не как технология окончательного принятия группового решения.

Вопросы для собеседования

1. Какие методы принятия решения вы знаете?
2. В чем суть синектики как метода принятия группового решения?
3. В чем состоит специфика фокус-групп?
4. Какие методы принятия группового решения вы знаете?
5. Какие критерии измерения групповой научно-технической задачи существуют?

Практикум

Задание 1. Групповая дискуссия: перспективы замысла и проблемного пространства *(по кн. Роберта Дилтса «Управление креативным процессом в группе»)*

Одним из достоинств данного метода является его «универсальность», т.е. возможность решать различные производственные задачи. Данная техника показывает, какое влияние оказывают на способность к поиску потенциального пространства решений различные способы представления и определения

«проблемного пространства» плана или замысла. Выполнять его лучше в группе из четырех человек.

Данная методика осуществляется в три стадии: 1) мечтателя; 2) реалиста и 3) критика.

1) На стадии Мечтателя один из участников группы «исследователь» должен описать план или замысел другим членам группы. Содержание замысла или плана ничем не ограничивается. Это может быть любой реальный или смоделированный деловой проект.

Слушая выступающего, члены группы должны следить за тем, чтобы их восприятие соответствовало перспективе (представлению) Мечтателя. Например, вместо того чтобы критически оценивать или судить о замысле и возможности его выполнения, каждый участник группы на данном этапе должен постараться *визуализировать* каждое отдельное действие в данном повествовании. Когда члены группы будут пытаться увидеть «масштабную картину», им следует слушать с поднятой головой и обращенным вверх взглядом, в симметричной расслабленной позе.

Когда Мечтатель закончит описание «проблемного пространства», членам группы надлежит определить, был ли получен ответ на следующие вопросы:

- *Что* вы намереваетесь делать? (В противоположность тому, что вы намереваетесь прекратить делать, избежать или бросить.)
- *Почему* вы намереваетесь это делать? *Какова цель этого?*
- *Какова* будет компенсация затраченных усилий?
- *Каким* образом вы узнаете, что вы ее получили? *Когда вы ожидаете ее получить?*
- *Куда* вы стремитесь прийти в будущем, воплотив этот замысел?
- *Кем* или подобным кому вы хотите стать в результате выполнения данного замысла?

Этап Мечтателя фокусируется на представлении и расширении восприятия проблемного пространства конкретного плана или замысла. Эти вопросы могут помочь исследователю и остальным членам группы расширить, обогатить и прояснить их мысленную картину проблемного пространства данного замысла или плана.

2) На следующем этапе упражнения каждый участник группы (включая исследователя) должен сделать «раскадровку» плана или замысла. Данное «первое приближение» должно быть очень общим и включать весь план или замысел целиком. «Раскадровка» может быть выполнена в виде любой схемы или чернового наброска. Лучше всего сделать символическое или метафорическое изображение плана или замысла. Например, кто-то один может изобразить некий пейзаж, другой – нарисовать какую-то группу символов (прямоугольников, кружков и звездочек) и соединить их линиями и стрелками.

Каждый должен самостоятельно сделать собственную репрезентативную карту, не заглядывая в рисунки других. Таким образом, каждый рисует свою собственную картину данного проблемного пространства, включая исследователя, который создает сводное изображение на основе всех четырех

образных карт проблемного пространства. Затем участники группы обмениваются рисунками и обсуждают допущения и критерии, стоящие за каждым рисунком и толкованием. Сопоставление различных карт и допущений проблемного пространства обогащает представления о данном пространстве. Участники должны дать объяснение своим рисункам, не выдвигая при этом никаких предложений или решений, не навязывая исследователю способ выражения замысла или плана, они всего лишь показывают и объясняют свою «раскадровку». Далее члены группы могут рассмотреть следующие вопросы этапа Реалиста.

- *Каким* именно образом будет осуществлен данный замысел?
- *Каким* образом вы узнаете, что цель достигнута? Как опробуете критерии исполнения?
- *Кто* будет это делать? (Назначьте ответственных лиц и заручитесь согласием тех, кто будет приводить данный план в исполнение).
- *Когда* будет осуществляться каждый этап? Когда будет выполнена общая задача?
- *Где* будет выполняться каждый этап?
- *Почему* необходим каждый конкретный шаг?

Во время обсуждения члены группы должны следить за тем, чтобы не отклоняться от восприятия Реалиста. В процессе обсуждения всем членам группы следует сидеть прямо или слегка подавшись вперед, глядя прямо перед собой, выпрямив или слегка наклонив вперед голову. В когнитивном плане им следует действовать так, «как если бы» мечта была осуществима и размышления касаются способов реализации данного замысла или плана. Следует установить приоритетность действий и определить, какие шаги нужно предпринять в краткосрочном плане. Каждому участнику группы также потребуется занять другую позицию и взглянуть на план с нескольких разных точек зрения (с точки зрения других участников).

3) На стадии Критика все отдельные «раскадровки» должны быть синтезированы в одну общую. Как правило, это осуществляет руководитель группы (исследователь), который должен установить обратную связь с группой и посмотреть, как его собственная карта проблемного пространства была дополнена каждым из членов группы. Исследователь вновь формулирует данный план или замысел и осуществляет «следующее приближение» путем создания новой или комбинированной «раскадровки».

После этого группа должна рассмотреть данную раскадровку «повторным взглядом». Группа должна отойти «достаточно далеко», чтобы этот повторный взгляд был эффективным. При этом группа может либо сама перейти в другое место, либо удалить раскадровку на некоторое расстояние от себя. Затем члены группы приступают к рассмотрению следующих вопросов этапа Критика:

- *Отвечает ли данный план критериям и целям, лежавшим в его основе?*
- *Почему* кто-то мог бы возражать против этого нового замысла?
- *На ком* отразится этот новый замысел, кто может ему содействовать или воспрепятствовать, какие у них для этого причины и что им это сулит?
- *Что* положительного заключает в себе нынешний метод ведения дел?

•*Каким* образом можно сохранить все положительное при осуществлении данного плана или замысла?

•*Когда* и где вы не хотели бы осуществлять данный план или замысел?

При «повторном взгляде» членам группы следует строго придерживаться восприятия Критика. Их цель состоит в том, чтобы, рассмотрев предмет с различных точек зрения и отыскав слабые места, подобных проблем избежать. При этом весьма полезно сидеть в небрежной позе, с опущенной, слегка склоненной набок головой и взглядом, также обращенным вниз.

Для того чтобы быть конструктивными критиками, участники группы прежде всего должны определить, какие критерии были соблюдены и, по мере возможности, сформулировать свои «критические замечания» в форме вопросов.

После того как все вопросы собраны воедино, группа может назначить нового «исследователя» (Мечтателя) или же может продолжить циклическое повторение всех фаз, постепенно дорабатывая данный план.

Итак, ***весь процесс сводится к следующим этапам***:

1. Исследователь излагает (в течение 5 минут) план или замысел. Члены группы используют стратегию Мечтателя.
2. Члены группы изучают вопросы Мечтателя, чтобы яснее и полнее воспринять «проблемное пространство» плана или замысла.
3. Каждый, включая исследователя, рисует простую «раскадровку» или визуальную карту проблемного пространства (это должно занимать не более 5 минут).
4. Участники группы сравнивают рисунки, поясняют их и обсуждают лежащие в их основе критерии и допущения. Обсуждение рисунка каждого участника также не должно продолжаться более 5 минут.
5. После этого члены группы переходят к вопросам Реалиста, используя его стратегию, чтобы определить дальнейшие конкретные шаги и действия.
6. Все отдельные «раскадровки» синтезируются «исследователем» в одну общую, после чего следуют вопросы Критика с использованием достаточно «удаленной» стратегии.
7. Группа продолжает циклическое движение через все фазы, постепенно дорабатывая этот план.

Задание 2. Синектика: «Время на перекуры»

Прочитайте предложенную задачу, используя метод мозгового штурма предложите варианты решения этой проблемы, выбрав самые оптимальные.

На одном из японских заводов возникла следующая проблема: из-за частых перекуров токарей производительность труда в цехе была не слишком высокой. Поставить у каждого станка контролера - невозможно. Да и видеокамеру над каждым станком не повесишь. Во время "мозгового штурма", в котором принимали участие менеджеры, было найдено простое и остроумное решение, учитывающее человеческую психологию. Какое?

Задание 3. Триады ролей, распределяемых на различных этапах обсуждения

На основе выполненных заданий 1 и 2 оцените, какие ролевые позиции проявлялись в вашей группе в ходе принятия группового решения. Выпишите все ролевые позиции, которые отчетливо проявились на разных этапах решения групповых задач.

1. Восприятие предмета дискуссии.

Роль: **Интегратор.** *Действия:* Воспринимает предмет дискуссии в целом и создает о нем обобщенное представление. *Средства:* Обобщенная информация, общенаучные понятия и категории, установка на решение проблемы в целом.

Роль: **Аналитик.** *Действия:* Воспринимает специальный аспект предмета дискуссии, создает детальное представление об одной его стороне. *Средства:* Специальные знания, установка на решение **об**щей задачи.

Роль: **Системный аналитик.** *Действия:* Те же, что у Интегратора и Аналитика, но используемые в оптимальном соотношении. *Средства:* Те же, что у Интегратора и Аналитика, но используемые в оптимальном соотношении.

2. Анализ предмета дискуссии.

Роль: **Футуролог.** *Действия:* Дополняет имеющуюся информацию прогнозами**,** вводит предмет дискуссии. В сценарии будущего, анализирует последствия и значение для будущего. *Средства:* Интуиция, воображение, методы прогнозирова**ния.**

Роль: **Историк.** *Действия:* Дополняет имеющуюся информацию ретроспективным анализом, выводит следствия о предмете дискуссии из прошлого, определяет происхождение, преемственность, историческое значение предмета дискуссии. *Средства:* Знание истории и развития предмета дискуссии.

Роль: **Синхронист.** *Действия:* На основе имеющейся информации создает вневременное представление о предмете дискуссии, описывает его общую структуру. *Средства:* Функциональный анализ, обобщенное представление о предмете дискуссии, каким он есть, был и будет.

3. Оценка предмета дискуссии.

Роль: **Оптимист.** *Действия:* Преувеличенно хвалит и наиболее высоко оценивает предмет дискуссий, склонен к рискованным поспешным действиям и выводам. *Средства:* Установка преувеличивать все положительное, отсутствие осторожности, преуменьшение риска и опасности.

Роль: **Пессимист.** *Действия:* Осторожен в оценках предмета дискуссии, не склонен рисковать и спешить с окончательными выводами, оценивает новые идеи наиболее низко. *Средства:* Установка преувеличивать все отрицательное, склонность к чрезвычайной осторожности.

Роль: **Реалист.** *Действия:* Реалистическая оценка всех составляющих предмета дискуссии, взвешивание риска. *Средства:* Установки, позволяющие не впадать в край оценки, владение способами, позволяющими компенсировать возможные отрицательные откровения.

4. Принятие решения.

Роль: **Критик.** *Действия:* Суммирует недостатки предмета дискуссии, формулирует решение «против» новой идеи, выдвигает обвинение. *Средства:* Вывод из анализа недостатков, их причин следствий.

Ролы **Защитник.** *Действия:* Суммирует достоинства и формулирует решение «за» новую идею, отклоняет обвинения. *Средства:* Вывод из анализа достоинств, их причин следствий.

Роль: **Судья.** *Действия:* Суммирует достоинства и недостатки, по возможности компенсирует недостатки достоинствами и вынос решение. *Средства:* Способы совмещения структур достоинств недостатков.

Анализ работы в группе

1. Проанализируйте, какие роли были представлены в вашей группе в процессе принятия решения?
2. Может ли один и тот же человек выполнять разные роли в процессе принятия группового решения? Докажите Ваш ответ, опираясь на полученный в вашей группе опыт.
3. Всегда ли плохи амбициозность, конфликтность и повышенная эмоциональность в процессе принятия решения? Почему?
4. По каким признакам можно распознать неформального лидера группы?
5. Что необходимо для успешного управления группой?

Методические рекомендации

Методические указания и подробные инструкции к заданию 1 даны в тексте задания.

Методические указания к заданию 2. Выполняя задание важно строго следовать этапам мозгового штурма: генерация идей, анализ идей, оценка идей. На первом этапе участники «мозгового штурма» должны предложить как можно больше вариантов решений за определенный, достаточно краткий период времени (15-20 мин.). После этапа по выработке предложений начинается дискуссия для объединения и развития предложенных идей. Далее выдвинутые идеи оцениваются группой с точки зрения двух критериев: оригинальности (++, +, 0) и возможности успешной реализации (РР – реально реализовать; ТР – трудно реализовать; НР – невозможно реализовать). Побеждают идеи, набравшие «++» и реально реализуемые на практике (РР).

Методические рекомендации к заданию 3. Задание выполняется в тех подгруппах, в которых проводилось обсуждение и решение групповых задач (задания 1 и 2). Ответы на вопросы выписываются на отдельный лист и сдаются преподавателю для оценки.

Литература

1. *Истратова О.Н., Эксакусто Т.В.* Справочник психолога-консультанта организации. – Ростов-на-Дону: Феникс, 2010. – 640 с.

2. *Канеман Д.* Думай медленно... Решай быстро / Д. Канеман — «АСТ», 2011. – 362 с.

3. *Козьяков Р.В.* Психология управления: учебное пособие [Электронный ресурс]. – М.: Директ-Медиа, 2014. – 201 с.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»

ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Разработка и эксплуатация интеллектуальных автоматизированных систем»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024 г.

Сокращения

АС- автоматизированная система

АСУ- автоматизированная система управления

БД- база данных

СУБД- система управления базой данных

СРС. Самостоятельная работа студента

ИМ- имитационная модель, имитационное моделирование

ГЗУ- групповая замерная установка

ТЗ- техническое задание

ТУ- технические условия

SCADA - Supervisory Control And Data Acquisition – диспетчерское управление и сбор данных – программный пакет, предназначенный для разработки или обеспечения работы систем сбора, обработки, отображе-ния и архивирования.

РС – рабочая станция

Цели и задачи освоения дисциплины

Дисциплина является вводной и обобщенной для специализации подготовки.

Цели освоения дисциплины:

• дать студентам комплексное представление об интеллектуальных автоматизированных системах (ИАС), об их проектировании и особенностям эксплуатации.

Задачи освоения дисциплины:

• дать представление об интеллектуальных автоматизированных системах в различных предметных областях;

• ознакомить с основными этапами жизненного цикла ИАС: исследование объекта автоматизации, проектирование, пусконаладочные работы, авторское сопровождение.

Подготовка к лабораторным занятиям. Основная цель проведения лабораторных занятий – формирование у студентов практических навыков. Подготовку к каждому лабораторному занятию студент должен начать с ознакомления с методическими рекомендациями по выполнению лабораторной работы. Затем студент должен проанализировать предложенный ему вариант задания лабораторной работы и составить для себя план его выполнения. Непосредственно на занятии самостоятельно или в режиме мастер-класса студент выполняет задание лабораторной работы. По итогам его выполнения студент составляет отчет о выполненной лабораторной работе. Требования по оформлению отчета изложены в методических рекомендациях по выполнению лабораторных работ по дисциплине.

Защита лабораторной работы начинается с предъявления преподавателю результата выполнения работы и отчета, в случае удовлетворительного качества предъявленного материала, завершается собеседованием по теме работы.

Методические рекомендации по выполнению лабораторных работ

Лабораторные работы проводятся в аудитории, оборудованной в соответствии с п.п. 7.1 и 7.2. рабочей программы дисциплины (РПД). Основанием для допуска к лабораторной работе являются:

- знания теоретического материала и методических указаний, которые должен продемонстрировать студент в начале занятия.
- отсутствие задолженностей по предыдущим лабораторным работам, если таковые проводились.

При домашней подготовке к выполнению каждой лабораторной работы магистрант знакомится с теоретическим материалом по тематике работы и готовит макет отчета о выполнении, последовательность процедур, которые он должен выполнить на ПЭВМ.

Каждый студент выполняет задания в соответствии с индивидуальным вариантом, который определяется по номеру студента в списке группы. По итогам выполнения каждого лабораторного задания студент завершает оформление отчета с использованием текстового редактора MS Word.

По каждой из лабораторных работ отчет оформляется отдельно.

Содержание отчета:

- Титульный лист.
- Техническое задание (в соответствии с вариантом).

Пошаговая последовательность этапов выполнения заданий, включая результаты выполнения заданий в виде screen-shot. Описываются практические результаты работы (при необходимости по каждому из вопросов излагается краткая теория, последовательность процедур при их реализации на ПЭВМ, приводится подтверждение полученных умений и навыков).

Снятые screen-shot помещаются в текстовый файл отчета как рисунки. Под каждым рисунком должны быть автоматически вставлены номер и название средствами Word.

4. Выводы о проделанной работе.

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы преподавателя по теме занятия.

В процессе подготовки и выполнения лабораторных работ студент руководствуется учебной и методической литературой, указанной в п. 6 РПД.

Критерии оценки для лабораторных работ:

4-5 баллов выставляется магистранту, если он своевременно выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями и в процессе защиты продемонстрировал полноту теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на дополнительные вопросы, связанные не только с процессом выполнения лабораторной работы, но и с пониманием совершенных действий и решенных задач.

2-3 балла выставляется магистранту, если он выполнил все задачи, предусмотренные в лабораторной работе, подготовил отчет в соответствии с требованиями преподавателя и в процессе защиты продемонстрировал наличие достаточных теоретических знаний в объеме содержания учебной дисциплины, относящейся к лабораторной работе. Сумел ответить на вопросы, связанные с процессом выполнения лабораторной работы.

1 балл выставляется студенту, если он более чем на половину выполнил поставленные в лабораторной работе задачи, способен ответить на вопросы, касающиеся теоретической составляющей в объеме содержания учебной дисциплины, относящейся к лабораторной работе.

0 баллов выставляется студенту, при невыполнении требований, предусмотренных в случае удовлетворительной оценки

Основные понятия об АСУ

Автоматизированная система управления (сокращённо АСУ) — комплекс аппаратных и программных средств, а также персонала, предназначенный для управления различными процессами в рамках технологического процесса, производства, предприятия.

АСУ применяются в различных отраслях промышленности, энергетике, транспорте и т. п. Термин «автоматизированная», в отличие от термина «автоматическая», подчёркивает сохранение за человеком-оператором некоторых функций, либо наиболее общего, целеполагающего характера, либо не поддающихся автоматизации.

АСУ с Системой поддержки принятия решений (СППР), являются основным инструментом повышения обоснованности управленческих решений.

Важнейшая задача АСУ — повышение эффективности управления объектом на основе роста производительности труда и совершенствования методов планирования процесса управления.

Различают автоматизированные системы управления объектами (технологическими процессами — АСУТП, предприятием — АСУП, отраслью — ОАСУ) и функциональные автоматизированные системы, например, проектирование плановых расчётов, материально-технического снабжения и т. д.

Потребность постоянно повышать производительность и эффективность труда работников, выпускать больше качественной продукции и т.п. послужили основанием к созданию автоматизированных систем. Автоматизация прочно входит в повседневную жизнь людей. Первоначально с этой целью создавались различные автоматические устройства, способные избавлять человека от выполнения различных рутинных и опасных видов работ, например, роботы-автоматы и др.

Под “автоматической” понимают любую саморегулирующуюся систему. Принцип её работы заключается в сравнении некоторых выходных характеристик с установленным эталоном. Отклонение выходного значения от

эталона вызывает включение элементов обратной связи для корректировки полученного значения. Созданные таким образом устройства позволили облегчить, ускорить, а порой и удешевить выполнение определенных видов работ.

Автоматизация может быть вызвана двумя обстоятельствами: наличием реальной необходимости и возможностью практической реализации.

Автоматизированная система - это система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая автоматизированную технологию выполнения установленных функций. Такая система представляет комплекс технических, программных, других средств и персонала, предназначенный для автоматизации различных процессов. В первую очередь подобные системы создавались в промышленности и были ориентированы на совершенствование методов управления производственными процессами.

Автоматизированная система управления **(АСУ)** -это совокупность экономических и математических методов, технических средств организационных комплексов, обеспечивающих рациональное управление сложным объектом (процессом) в соответствии с заданной целью.

Основное назначение АСУ – получение высокой эффективности разработки, внедрения и эксплуатации различных по назначению производственных систем

Автоматизация базируется на широком использовании средств вычислительной техники (СВТ) и необходимого для них ПО. В качестве технических средств АСУ получили использование многомашинные, многопроцессорные комплексы, образующие с помощью ЭВМ и информационных сетей распределенные системы обработки информации. При реализации АСУ обычно применяются автоматизированные рабочие места и участки. Решаемые в АСУ задачи делят на задачи, требующие немедленного ответа и допускающие определённую его задержку по времени выполнения. В основном выделяют следующие режимы работы АСУ:

- параллельной обработки, квантования временем для пакетной обработки,
- оперативной обработки, реального времени и телеобработки информации и данных.

В режиме *квантования временем* каждой прикладной программе выделяется квант времени, по окончании которого управление передаётся следующей программе. Увеличение скорости ответа системы пользователю достигается путём *оперативной* (онлайновой, непосредственной) *обработки данных*. При сочетании многопрограммного режима работы ЭВМ с квантованием времени и режимом непосредственного доступа образуется *режим разделения времени*.

Режим реального времени предназначен для задач, требующих немедленного ответа. Он характеризуется дистанционной обработкой информации (*телеобработкой*).

Автоматизация позволяет существенно сократить время создания новых образцов техники, продуктов и т.д., а также обслуживания пользователей, значительно повысить уровень их обслуживания, преобразует и видоизменяет отдельные технологические процессы, а порой – все основные традиционно используемые технологии.

АСУ – гибкие интегрированные системы с элементами искусственного интеллекта. Они ориентированы на реализацию безбумажного, безлюдного управления объектом с подстройкой к изменяющимся внешним условиям и ресурсам. Реализация подобных задач строится на применении ЭВМ, объединённых информационной сетью или сетями с другими ЭВМ.

Основные принципы автоматизации информационных процессов

Автоматизированная информационная система (Automated information system, AIS) - это совокупность программных

и аппаратных средств, предназначенных для хранения и (или) управления данными и информацией, а также для производства вычислений.

Основная цель АИС – хранение, обеспечение эффективного поиска и передачи информации по соответствующим запросам для наиболее полного удовлетворения информационных запросов большого числа пользователей.

К *основным принципам автоматизации информационных процессов* относят: окупаемость, надежность, гибкость, безопасность, дружественность, соответствие стандартам.

Окупаемость означает затрату меньших средств, на получение эффективной, надёжной, производительной системы, возможностью быстрого решения поставленных задач.

- *Надежность* достигается использованием надёжных программных и технических средств, использования современных технологий.
- *Гибкость* означает легкую адаптацию системы к изменению требований к ней, к вводимым новым функциям. Это обычно достигается созданием модульной системы.
- *Безопасность* означает обеспечение сохранности информации, регламентация работы с системой, использование специального оборудования и шифров.
- *Дружественность* заключается в том, что система должна быть простой, удобной для освоения и использования (меню, подсказки, система исправления ошибок и др.).

Выделяются четыре типа АИС:

1. Охватывающий один процесс (операцию) в одной организации.
2. Объединяющий несколько процессов в одной организации.
3. Обеспечивающий функционирование одного процесса в масштабе нескольких взаимодействующих организаций.

Реализующий работу нескольких процессов или систем в масштабе нескольких организаций.

АИС можно представить как комплекс автоматизированных информационных технологий, составляющих информационную систему, предназначенную для информационного обслуживания потребителей.

Основное назначение автоматизированных информационных систем не просто собрать и сохранить электронные информационные ресурсы, но и обеспечить к ним доступ пользователей. Одной из важнейших особенностей АИС является организация поиска данных в их информационных массивах (базах данных). Поэтому АИС практически являются автоматизированными информационно-поисковыми системами (АИПС)

Автоматизированная информационно-поисковая система - программный продукт, предназначенный для реализации процессов ввода, обработки, хранения, поиска, представления данных т.п.

- АИПС бывают фактографическими и документальными.
- *Фактографические АИПС* обычно используют табличные реляционные БД с фиксированной структурой данных (записей).
- *Документальные АИПС* отличаются неопределённостью или переменной структурой данных (документов). Для их разработки обычно применяются оболочки АИС.

Автоматизация информационных процессов

Целью автоматизации информационных процессов является повышение производительности и эффективности труда работников, улучшение качества информационной продукции и услуг, повышение сервиса и оперативности обслуживания пользователей.

Способами обеспечения автоматизированных информационных систем и их технологий являются программное, техническое, лингвистическое, организационное и правовое обеспечение, используемые или создаваемые при

проектировании информационных систем и обеспечивающие их эксплуатацию.

Программное обеспечение представляет инструментальную среду программистов, прикладные программы для соответствующих ЭВМ. Это языки программирования, операционные системы, сетевое программное обеспечение, редакторы (текстовые, связей, табличные и др.), библиотеки программ, трансляторы, утилиты и др. Главными среди них являются программные комплексы АИС – системы управления базами данных (СУБД). Их оболочки – это автоматизированные информационно-поисковые системы (АИПС) широкого применения.

Техническое обеспечение АИС включает средства ввода, обработки, хранения, поиска и передачи/приёма информации. Ввод, обработка и хранение данных – стандартные составляющие ЭВМ. Поиск информации осуществляется на основе использования специального ПО. Средства передачи информации представляют собой сетевое и телекоммуникационное оборудование ЭВМ, системы и средства связи.

К *лингвистическому обеспечению* обычно относят:

- типы, форматы, структура информации (данных, записей, документов);
- языковые средства описания (ЯОД, словари данных) и манипулирования данными (ЯМД);
- классификаторы, кодификаторы, словари, тезаурусы и т.п.

В состав *организационного обеспечения* АИС входят структурные подразделения организации, осуществляющие управление технологическими процессами и поддержку работоспособности системы, а также документация для обеспечения эксплуатации и развития системы.

Правовое обеспечение АИС – это совокупность правовых норм, регламентирующих правоотношения при создании и функционировании АИС. Правовое обеспечение включает нормативные документы, регламентирующие деятельность АИС.

АСУ – это, как правило, система «человек-машина», призванная обеспечивать автоматизированный сбор и обработку информации, необходимой для оптимизации процесса управления. В отличие от автоматических систем, где человек полностью исключён из контура управления, АСУ предполагает активное участие человека в контуре управления, который обеспечивает необходимую гибкость и адаптивность АСУ.

Существенными признаками АСУ является наличие больших потоков информации, сложной информационной структуры, достаточно сложных алгоритмов переработки информации. Общими свойствами и отличительными особенностями АСУ как сложных систем являются следующие:

- наличие большого числа взаимосвязанных и взаимодействующих элементов, причём изменение в характере функционирования какого-либо из элементов отражается на характере функционирования другого и всей системы в целом;
- система и входящие в неё разнообразные элементы в подавляющем большинстве являются многофункциональными;
- взаимодействие элементов в системе может происходить по каналам обмена информацией, энергией, материала и др.;
- наличие у всей системы общей цели, общего назначения, определяющего единство сложности и организованности, несмотря на всё разнообразие входящих в неё элементов;
- переменность структуры (связей и состава системы), обеспечивающий многорежимный характер функционирования;
- взаимодействие элементов в системе и с внешней средой в большинстве случаев носит стохастический характер;
- автоматизация имеет высокую степень, в частности широкое применение средств автоматики и вычислительной техники для

гибкого управления и механизации умственного и ручного труда человека, работающего в системе;

- управление в подавляющем большинстве систем носит иерархический характер, предусматривающий сочетание централизованного управления или контроля с автономностью её частей.

Классификация АСУ

- В зависимости от роли человека в процессе управления, форм связи и функционирования звена «человек-машина», оператором и ЭВМ, между ЭВМ и средствами контроля и управления все системы можно разделить на два класса:

информационные системы,

управляющие системы

Информационные системы, обеспечивающие сбор и выдачу в удобном виде информацию о ходе технологического или производственного процесса. В результате соответствующих расчётов определяют, какие управляющие воздействия следует произвести, чтобы управляемый процесс протекал наилучшим образом. Основная роль принадлежит человеку, а машина играет вспомогательную роль, выдавая для него необходимую информацию.

Управляющие системы, которые обеспечивают наряду со сбором информации выдачу непосредственно команд исполнителям или исполнительным механизмам. Управляющие системы работают обычно в реальном масштабе времени, т.е. в темпе технологических или производственных операций. В управляющих системах важнейшая роль принадлежит машине, а человек контролирует и решает наиболее сложные вопросы, которые по тем или иным причинам не могут решить вычислительные средства системы.

Автоматизированные системы управления обладают множеством достоинств. Однако при их внедрении не стоит забывать и про недостатки. Чтобы АСУ принесли максимум плюсов и минимум минусов, необходимо:

- Перед тем, как осуществлять проект внедрения нужно максимально формализовать его цели;
- Никогда не стоит жертвовать стадией предпроектного анализа. Необходимо привлекать профессиональных консультантов для обследования предприятия и постановки задач менеджмента. Затраты непременно окупятся. Но стараться иметь дело при этом с солидными компаниями, так как, к сожалению, кроме консультантов, существуют еще и псевдо-консультанты;
- Нужно старательно подходить к выбору программного обеспечения для построения КИС, так как ошибки дорого обходятся; стараться посмотреть как можно больше систем, и посмотреть их "живьем", а не по маркетинговым материалам разработчиков. Не стоит пытаться разрабатывать систему силами своих программистов. Готовые системы разрабатываются специализированными коллективами на протяжении многих лет и имеют реальную себестоимость гораздо выше продажной цены – известный парадокс характерный для программных и интеллектуальных продуктов;
- Необходимо установить высокий приоритет процессу внедрения системы, среди остальных организационных и коммерческих процессов, наделить высокими полномочиями руководителя проекта;
- Нужно создать среди всех сотрудников предприятия атмосферу неотвратимости внедрения и стараться организационными мерами повысить темп освоения новых технологий;
- Необходимо помнить, что внедрение системы как ремонт – его невозможно закончить, можно лишь прекратить. Так что внедрение, по сути, никогда не закончится, система должна все время совершенствоваться в процессе своей промышленной эксплуатациями вместе с прогрессом информационных технологий и методологий управления деятельностью вашего предприятия.

Выполнение лабораторных работ

Лабораторные работы связаны между собой, последовательное выполнение работ позволит обучаемыми создавать автоматизированную систему в соответствии с требованиями задания. Для освоения рабочей программы достаточно выполнить Лабораторные работы №№1-4. Оценки за эти работы масштабируются с соответствие с баллами, предоставленными в учебной карте дисциплины. За выполнение работ №№5,6 предоставляются бонусные баллы (максимум 10).

Лабораторная работа №1

РАЗРАБОТКА БД АС

Цель работы:

В соответствии с обследованием объекта автоматизации, выполненного на практическом занятии № 1 и в рамках темы «Информационное обеспечение АС» требуется разработать структуру БД АС и реализовать её в предоставленной СУБД. БД должна обеспечивать хранение данных в соответствии с перечнями входных и выходных сигналов (документ предоставляется) и технологическими процессами сбора и обработки данных АС. Подготовка к выполнению работы, а также отладка БД и подготовка отчет выполняется во время СРС. Выбор СУБД осуществляется в соответствии с квалификацией обучаемых.

Общие сведения:

Под ***автоматизированной системой*** (АС) будем понимать механизм, реализующий информационную технологию.

Этот механизм содержит две составляющих: людей, занимающихся эксплуатацией и обслуживанием АС и информационную систему (ИС) как программный-информационный-технический комплекс. То есть ИС это совокупность базы данных, СУБД, приложений, реализующих задачи пользователей и соответствующих технических средств (компьютеры, сетевое оборудование, периферия и т.п.).

Традиционно в последние годы сложилось так, что вопросы проектирования ИС достаточно подробно излагаются во многих учебниках и учебных курсах. Но ИС, хотя и очень важная, но только часть АС. Правда, термин АС часто заменяют термином корпоративная ИС, киберфирма и т.п. Проектирование АС – это отдельное научно-техническое направление, включающее в себя проектирование ИС, анализ и реинжиниринг организаций, проектирование Human relation, т.е. всего, что связано с человеческим фактором.

В соответствии с ЖЦ инженерного изделия различают следующие виды АС:

– **АСНИ** – автоматизированная система научных исследований (Основная цель: моделирование и проведение экспериментов. Решаемые задачи и инструментарий: математическая статистика, планирование эксперимента, методы оптимизации, имитационное моделирование);

– **САПР** – система автоматизированного проектирования (Основная цель: автоматизация процессов расчетов и проектирования. Решаемые задачи: изготовление конструкторской документации, смет, заказных спецификаций, оптимизация проектных решений, снижение сроков проектирования);

– **АСТПП** - автоматизированная система технологической подготовки производства (Основная цель: подготовить конкретное предприятие с его конкретными материальными и человеческими ресурсами к выпуску того или иного изделия или переходу на новую технологию. Решаемые задачи: составление маршрутных и технологических карт, расчет и оптимизация загрузки людей и оборудования; расчеты потребностей и планирование запасов и т.п);

– **АСУТП** - автоматизированная система управления технологическими процессами (Основная цель: управление изготовлением готовой продукции в основном для непрерывных

производств, например, производства аммиачной селитры. Решаемые задачи: задачи автоматического управления и регулирования);

– **ГПС** – гибкие производственные системы (набор производственных модулей, станков с числовым программным управлением, промышленных роботов, из которых можно создать технологическую систему). (Основная цель: автоматизация дискретного производства, например производство автомобилей. Решаемыезадачи: механическая, термическая идр обработка, перемещение изделия и компонентов между производственными модулями, складирование и т.п.);

– **АСУП** - автоматизированная система управления предприятием (Основная цель: решает задачи организации управления и экономики. Основные задачи: бух учет, планирование, кадры, снабжение, сбыт и т. п.).

Зарубежный аналог АСУП – это общеуправленческие системы (MI-management information system и EIS - executive information system):

Кроме того, можно классифицировать АСУП по:

- ***по отраслям производства***, например, банковские учетные и управленческие системы, управление дискретным промышленным производством, системы профилактической и режимной деятельности органов МВД и др.,
- ***по видам деятельности***, например, управление работой склада, система маркетинговых исследований, аналитическая система для работы на фондовом рынке и др.,
- ***по применяемым методам*** обработки информации, например, электронный архив, корпоративная система управления процессом выполнения офисных работ, система статистических расчетов и др.

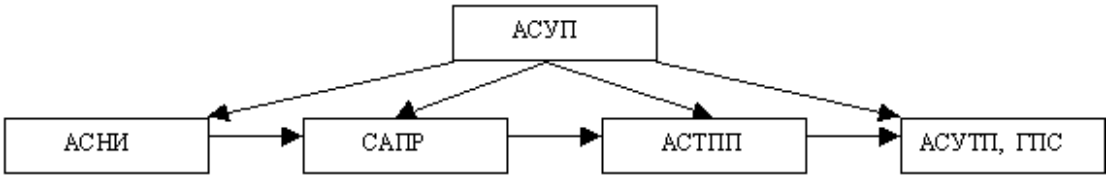


Рис. 1 «Общая схема связи АС»

В настоящее время синонимами АСУП являются термины: информационная система (ИС), автоматизированная ИС (АИС), корпоративная ИС (КИС), система обработки данных (СОД), автоматизированная СОД (АСОД) и др.

Приведенные выше АС являются в основном системами оперативной обработки данных, т.е. предназначены для решения задач постоянно возникающих в процессе инженерной и экономической деятельности и алгоритмы которых сравнительно легко поддаются формализации. Например, расчет запасов материала на складе или начисления зарплаты. Такие системы оперативной обработки всегда являются составной частью, информационным фундаментом любых более сложных СОД, решающих менее формализованные задачи, относящиеся, как правило, к разряду аналитических, нечетких, интеллектуальных.

Порядок выполнения работы:

1) Обоснование выбора СУБД

Выбор был сделан в пользу СУБД PostgreSQL.

PostgreSQL - это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

PostgreSQL обладает следующими преимуществами над другими СУБД:

— Быстродействие и широкие функциональные возможности механизма сервера СУБД;

— Наличие средств удаленного доступа;

— Эффективный инструмент для аналитики больших объемов информации;

— Стабильность работы системы;

— Отличные показатели безопасности, масштабируемости и надежности.

Фундаментальная характеристика объектно-реляционной базы данных — это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает Постгрес невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

2) Структура файлов

Датчики для разработки базы данных:

- Датчик динамического уровня жидкости;
- Датчик усилия в полированном истоке;
- Датчик положения балансира;
- Датчик состояния насоса;
- Датчик отсутствия напряжения на воде СУ;
- Датчик давления устьевого;

- Датчик перегруза;
- Датчик недогруза;
- Датчик деблокировки;
- Датчик активность мощности;
- Механизм включения насоса;
- Механизм выключения насоса.

На рисунке 2 представлена структура файлов с наименованием столбцов базы данных и их типом.

Name	Data type
id	serial
Датчик динамического уровня жидкости	real
Датчик усилия в полированном истоке	real
Датчик положения балансира	real
Датчик состояния насоса	boolean
Датчик отсутствия напряжения на вводе СУ	real
Датчик давления устьевого	real
Датчик перегруза	boolean
Датчик недогруза	boolean
Датчик деблокировки	boolean
Датчик активность мощности	boolean
Механизм включения насоса	boolean
Механизм выключения насоса	boolean

Рис. 2 «Структура Базы Данных»

Содержание отчета:

Отчет о работе должен содержать титульный лист, наименование и цель работы, краткое описание выбранной СУБД, снимки экрана созданной структуры базы данных, выводы о проделанной работе.

Защита отчета:

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы.

Контрольные вопросы:

1. В чем заключается постановка задачи автоматизации?
2. Дайте определение объекту автоматизации, системе управления, автоматизации управления.
3. Виды и типы АС. Классификация АС.
4. Архитектура и типовые задачи АС.
5. История зарождения и развития, поколения АС.
6. Опишите примерную методику обследования объекта автоматизации.

Лабораторная работа 2.

Разработка пользовательского интерфейса ИАС.

Цель работы:

Разработка пользовательского интерфейса ИАС осуществляется в виде последовательности экранных форм с указанием вида, размера и местоположения элементов предоставления данных и элементов управления. Среда разработки выбирается в соответствии с квалификацией обучаемых.

Общие сведения:

В наше время при разработке программного продукта очень важна скорость обработки информации, доступность, хранение и передача. Однако, не малую роль играет создание пользовательского интерфейса, так как именно он определяет, насколько легко и удобно будет пользоваться программой. Пользовательский интерфейс представляет собой совокупность используемых в программе средств ввода данных, способов отображения информации и элементов управления. Эффективность работы программного продукта определяется не только его функциональными возможностями, но и доступностью этих возможностей. Поэтому создание интуитивно-понятного дружественного интерфейса является важной задачей при разработке программного продукта.

Этапы выполнения работы:

В ходе данной работы мы рассмотрим создание средствами СУБД Access пользовательского интерфейса. Данные можно вводить и прямо в таблицы, но для конечных пользователей удобнее осуществлять ввод в отдельных окнах, предоставляющих дополнительные элементы управления для навигации и работы с записями. В Access такие окна называются формами, они служат для организации пользователю удобного графического интерфейса для работы с данными. При работе с формами есть несколько режимов:

- режим Формы;
- режим Конструктора;

• режим Макета.

В режиме Формы пользователь работает с данными; производятся добавление новых записей, просмотр, удаление или редактирование записей таблицы или запроса, являющегося источником данных для формы. В режиме Конструктора осуществляется модификация структуры формы

(изменение внешнего вида, добавление и удаление элементов управления и т.д.). Режим Макета позволяет увидеть данные и в то же время редактировать структуру формы.

Макет формы имеет следующие разделы:

• заголовок формы;

• область данных;

• примечание формы.

Область данных определяет основную часть формы. Этот раздел может содержать элементы управления, отображающие данные из источника данных (таблицы или запроса), а также неизменяемые данные (например, надписи и линии). При печати формы область данных будет повторяться для каждой записи таблицы, тогда как заголовок и примечание – только один раз.

Размещаемые на форме элементы управления могут быть разных типов.

Связанные элементы управления связаны с полями базовой таблицы, которая является источником данных для формы. Если источником данных является запрос, то присоединенные элементы управления могут связываться с полями в разных таблицах. При изменении значения в элементе управления обновляется и значение поля соответствующей записи в таблице.

Свободные элементы управления не связаны с таблицами, в частности они служат для оформления (линии, надписи и т.д.).

Вычисляемые элементы управления – это элементы, значения которых рассчитываются на основе значений других элементов. В качестве источника данных для этих элементов используются выражения и функции.

Ниже перечислены некоторые элементы управления, которые могут быть размещены на форме (полный перечень можно увидеть, работая в режиме Конструктора).

Надпись (Label) содержит небольшой текст, как правило, пользователем не изменяемый. Поле (Text Box) служит для отображения, ввода и изменения данных. Флажок (Check box), Переключатель (Option button), Выключатель (Toggle button) – элементы, определяющие значения логического типа. В Access, в зависимости от настроек поля, это может быть "Истина/ Ложь", "Да/Нсг" или "Вкл./Выкл.".

Элементы Список (List Box) и Поле со списком (Combo Box) позволяют выбрать значения из определенного списка (это может быть просто заданный набор возможных значений или, например, данные одного из столбцов таблицы). Поле со списком также позволяет ввести в поле новое значение.

Кроме "простых" форм, Access позволяет создавать "сложные" формы, включающие данные, собранные из нескольких связанных таблиц. При этом подчиненной формой называется форма, которая встраивается в другую. Форма, содержащая подчиненную форму, называется главной.

В папке с файлами к лабораторной работе находятся две базы Access – уже знакомая но прошлому занятию база lib.accdb, в которую добавлены две формы, и база lib_1 .accdb, отличающаяся от первой только отсутствием форм.

Начнем с создания формы с информацией об изданиях. Создадим ее с помощью мастера и потом изменим в редакторе. Перейдем на закладку Создание → Мастер форм. В первом окне "мастера" надо выбрать таблицу (или запрос), данные из которой будут отображаться в форме (таблица

Book), и поля из этой таблицы. Надо отметить, что идентификатор книги (типа Счетчик) показывать пользователю на форме не надо (рис. П.2.1): изменяется идентификатор автоматически и задать его значение вручную пользователь не сможет, что может привести к путанице.

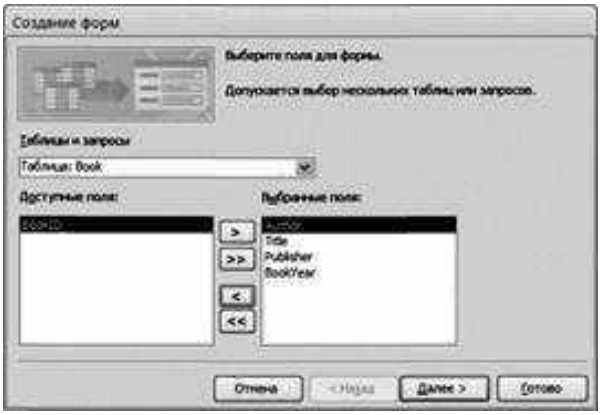


Рис. 3 Выбор полей таблицы

В представленном примере поля располагаются в один столбец, форма названа "Информация об изданиях". В последнем окне мастера выберите опцию "Изменить макет формы", после чего откроется конструктор, в котором нужно изменить размер полей, текст надписей, добавить элементы оформления. В примере в примечание формы добавлена надпись "F1 – справка", а заголовок – дата и время.

Задание. В базе данных lib_1.acc.db в соответствии с приведенным выше описанием и примером создайте новую форму. Не забудьте в заголовке формы поставить дату и время, а в примечании – надпись про справку. Введите через форму какие-нибудь данные. Посмотрите, как будет выглядеть форма при печати (Файл → Печать → Предварительный просмотр).

Задание. Самостоятельно создайте форму для редактирования информации о статусах книг.

Теперь рассмотрим создание составных форм. Такие формы очень 5'добны при работе со связанными таблицами – в одной форме можно представить данные из нескольких таблиц (рис.4).

Проще всего создать подобную форму с помощью мастера. В первом окне мастера укажите, что на форме надо представить информацию об авторе, названии, издательстве, годе издания, взятую из таблицы Book, и информацию об идентификаторе и статусе книги из таблицы Book_in_lib. Мастер предложит создать подчиненные формы. Дальнейшая работа мастера не

отличается от предыдущего случая. Обратите внимание, что мастер создал две формы – основную и подчиненную.

После окончания работы мастера можно отредактировать форму в режиме конструктора, например поменять подписи с подставляемых по умолчанию названий столбцов на какие-то более подробные пояснения.

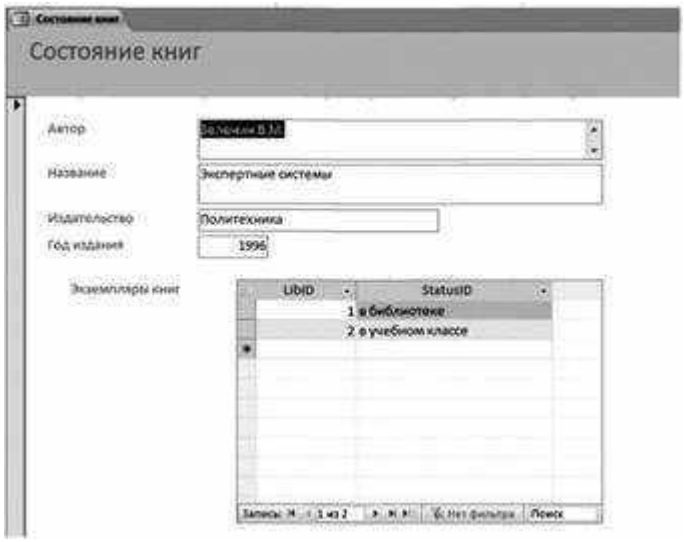


Рис. 4 Составная форма

Задание. Создайте составную форму для отображения информации об издании и текущем состоянии экземпляров книг в библиотеке.

Нередко для удобства работы пользователя нужно разместить на форме какие-нибудь дополнительные элементы, например кнопки. Внизу каждой формы расположены кнопки, позволяющие передвигаться по записям (следующая, предыдущая, первая, последняя, новая запись). Однако для того чтобы удалить запись, приходится использовать соответствующую кнопку с основной напели инструментов, что не очень удобно.

Добавим кнопку удаления на форму "Информация об изданиях". Для этого в режиме редактирования надо воспользоваться панелью элементов. На панели элементов надо выбрать элемент "кнопка" и мышкой нарисовать его на форме. После этого появится диалоговое окно Создание кнопок, в котором надо указать, что создаваемая кнопка относится к категории "Обработка записей" и требуемое действие – "Удалить запись". Далее для созданной

кнопки можно выбрать подходящую пиктограмму или надпись, а также ее имя.

Контрольные вопросы:

1. Что представляет собой пользовательский интерфейс?
2. Перечислите режимы, предназначенные для работы с формами .
3. Какие разделы имеет макет формы?
4. Что представляют собой сложные формы?
5. Принцип работы Мастера форм.

Лабораторная работа №3

РАЗРАБОТКА ГЕНЕРАТОРА ДАННЫХ ДЛЯ ТЕСТИРОВАНИЯ АС

Цель работы:

В соответствии с Таблицей входных сигналов (документ предоставляется) и частотой опроса датчиков или поступления данных в АС разрабатывается имитационная модель потоков данных (генератор данных). Модель в дальнейшем будет использована для тестирования компонентов АС и в лабораторной работе № 3. Генератор должен обеспечивать формирование и передачу в АС данных в соответствии с типом сигналов (логический, цифровой, аналоговый, частотный) и с предоставленными законами их распределения (равномерный, нормальный…). Для моделирования используют генераторы случайных величин различных законов распределения. Необходимо обеспечить запись с сохранение данных в БД, разработанной в лабораторной работе №1.

Общие сведения:

Для успешной реализации проекта необходимо устанавливать реальные этапы с чётко обозначенными началом и окончанием. Разработка детального

плана работ связана с описанием процессов и их последовательности, выполняемых на каждом этапе, необходимых для этого специалистов, средств и ресурсов. Такой подход в большей степени позволяет избежать упущений и ошибок. Он необходим работникам, реализующим внедрение проекта автоматизации, а также оказывает положительное воздействие на лиц, его финансирующих.

Эффективное поэтапное осуществление проектных работ связано с необходимостью разработки графика их выполнения, включающего ресурсы и сроки (этапы) их проведения (см. предыдущие графики и рисунки). Ресурсы включают необходимые персонал, технические и программные средства, финансирование и инфраструктуру. При этом финансирование лучше осуществлять отдельно по каждому виду работ (приобретение средств и ПО, установка, обучение, отдельные этапы проектирования и др.).

Этапы проектирования АС

Для автоматизации различных видов деятельности (управление, проектирование, исследование и т.п.), включая их сочетания, используют положения **ГОСТ 34.601-90**. Он предусматривает следующие стадии и этапы проектирования (таблица 1).

Таблица 1 –Этапы проектирования АС

Стадии	**Этапы**
1. Формирование требований к АС	1.1. Обследование объекта и обоснование необходимости создания АС 1.2. Формирование требований пользователя к АС 1.3. Оформление отчёта о выполненной работе и заявки на разработку АС

2. Разработка концепции АС	2.1. Изучение объекта 2.2. Проведение необходимых научно-исследовательских работ 2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющей пользователя 2.4. Оформление отчёта о выполненной работе
3. Техническое задание	3.1. Разработка и утверждение технического задания на создание АС
4. Эскизный проект	4.1. Разработка предварительных проектных решений по системе и её частям; 4.2. Разработка документации на АС и её части
6. Рабочая документация	6.1. Разработка рабочей документации на систему и её части 6.2. Разработка или адаптация программ
7. Ввод в действие	7.1. Подготовка объекта автоматизации к вводу АС в действие 7.2. Подготовка персонала 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)

	7.4. Строительно-монтажные работы 7.5. Пуско-наладочные работы 7.6. Проведение предварительных испытаний 7.7. Проведение опытной эксплуатации 7.8. Проведение приёмочных испытаний
8. Сопровождение АС	8.1. Выполнение работ в соответствии с гарантийными обязательствами 8.2. Послегарантийное обслуживание

В стандарте указывается также, что:

- Стадии и этапы, выполняемые организациями, участниками работ по созданию АС, устанавливаются в договорах и Техническом задании на основе настоящего стандарта.
- Допускается исключать стадию “Эскизный проект” и отдельные этапы работ на всех стадиях, объединять “Технический проект” и “Рабочая документация” в одну стадию “Техно-рабочий проект”. В зависимости от специфики создаваемых АС и условий их создания допускается выполнять отдельные этапы работ до завершения предшествующих стадий, параллельное во времени выполнение этапов работ, включение новых этапов работ.

Технический проект (preliminary design) содержит принципиальные электрические схемы и конструкторскую документацию объекта разработки и составные его части, перечень выбранных готовых средств программного и технического обеспечения (в том числе типов ЭВМ, операционной системы, прикладных программ и т.д.), алгоритмы решения задач для разработки новых средств программного обеспечения).

Рабочий проект (detailed design) – заключительный этап проектирования, в общем случае предусматривающий уточнение и детализацию результатов предыдущих этапов, создание и испытания

опытного образца объекта автоматизации, разработку и отработку программных продуктов, технологической и эксплуатационной документации.

В современной практике проектирования автоматизированных информационных систем (например, АИПС, АСНТИ, АСУ и др.) он может являться начальным этапом их внедрения в работу организации или службы Заказчика проекта, или головной в ряде других автоматизируемых организаций, служб и т.д.

Порядок выполнения работы:

Таблица 2–Модель потоков данных.

Наименование величины	Диапазон измерений	Вероятность/ Тип генератора	Max аварийное значение	Min аварийное значение	Max предаварийное значение	Min предаварийное значение
Динамический уровень жидкости	0-10	РР	9	1	8	2
Усилия в полированном истоке	0-100	НР	90	10	80	20
Положения балансира	0;1	0,5				
Состояния насоса	0;1	0,5				
Отсутствие напряжения на вводе СУ	50-100	РР	95	55	85	65
Давления устьевого	0-4	НР	3.85	0.15	3.7	0.3
Перегруз	0;1	0,3				
Недогруза	0;1	0,2				
Сигнал деблокировки	0;1	0,4				
Активность мощности	0;1	0,2				

Включение насоса	0;1	0,25				
Выключение насоса	0;1	0,75				

Функция генерации равномерного распределения:

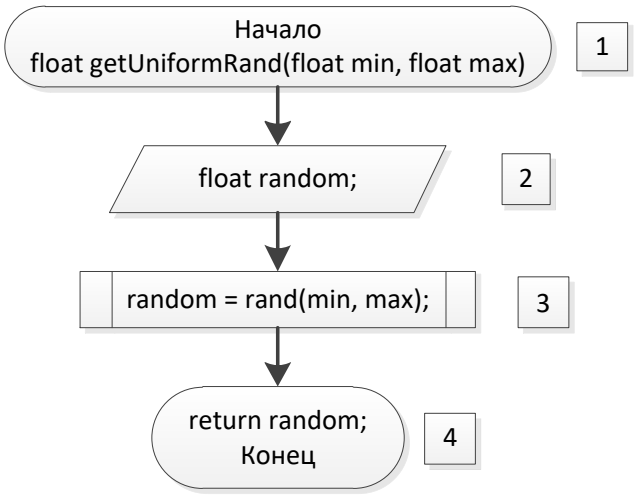


Рис. 5 «Функция getUniformRand»

1. Объявление функции, указано что она получает на вход и что возвращает.

2. Объявление переменной random.

3. Генерация ξ от минимального до максимального и запись в переменную random.

4. Возвращает переменную random из функции, конец функции.

Функция генерации нормального распределения:

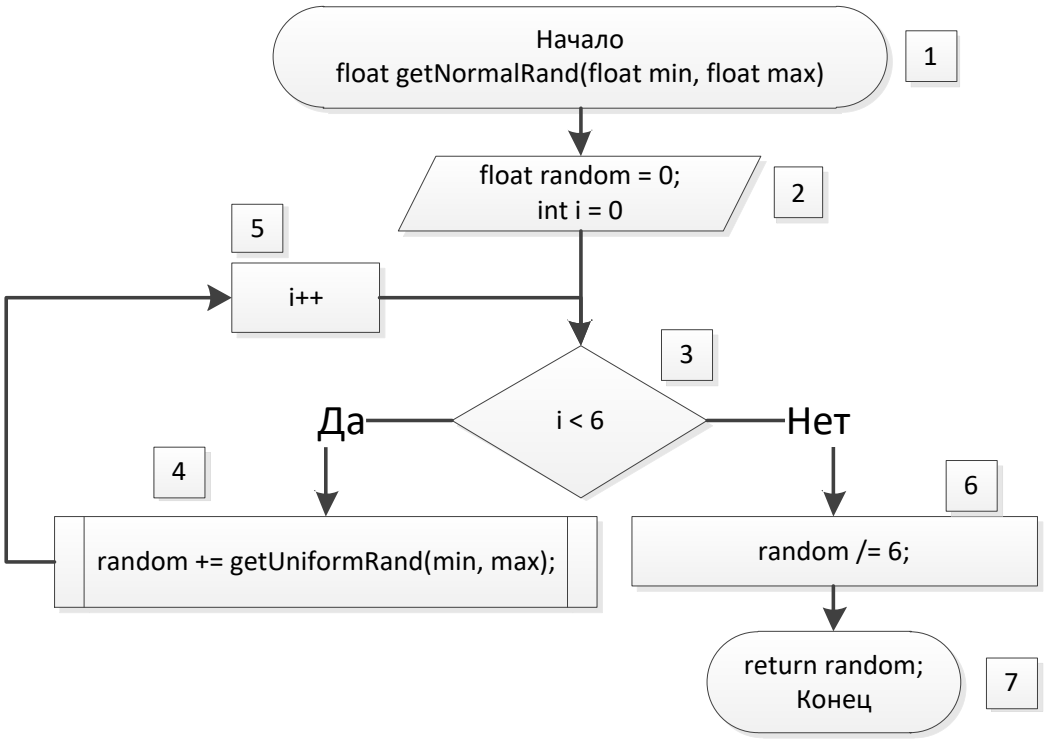


Рис. 6 «Функция getNormalRand»

1. Объявление функции, указано что она получает на вход и что возвращает.

2. Объявление переменной random и счетчика цикла i.

3. Условие цикла от i=0 до 6.

4. Прибавляем к переменной random возвращаемое значение функции getUniformRand (min, max).

5. Увеличиваем значение счетчика цикла на 1.

6. Делим random на 6 и получаем математическое ожидание.

7. Возвращает переменную random из функции, конец функции.

Функция генерации с двумя исходами:

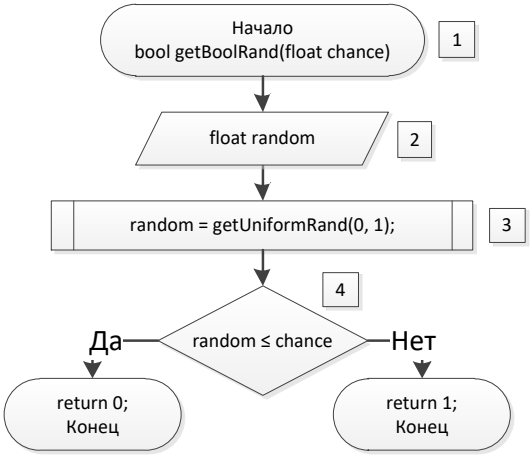


Рис. 7 «Функция getBoolRand»

1. Объявление функции, указано что она получает на вход и что возвращает.

2. Объявление переменной random.

3. Присваиваем переменной random возвращаемое значение функции getUniformRand от 0 до 1.

4. Условие, если переменной random меньше либо равно переменной chance, то возвращаем 0, иначе 1.

Содержание отчета:

Отчет о работе должен содержать титульный лист, наименование и цель работы, имитационную модель потоков данных в виде таблицы, алгоритмы функций генерации значений с двумя исходами и значений по равномерному и нормальному законам распределения, выводы о проделанной работе.

Защита отчета:

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы.

Контрольные вопросы:

1. Этапы проектирования АС.
2. В чем состоят пуско-наладочные работы АС.
3. Процесс приём и сдача в эксплуатацию АС.
4. Авторское сопровождение АС.
5. Эксплуатация АС и обслуживание компонент и систем.
6. Процессы подготовки решений и оптимизация, как ключевые вопросы автоматизированного управления.

Лабораторная работа №4
РАЗРАБОТКА ПОДСИСТЕМ КОНТРОЛЯ УСТАВНЫХ ЗНАЧЕНИЙ АС

Цель работы:

Предусматривается два варианта выполнения работы. В первом необходимо контролировать достижения данными поступающими из генератора данных (лабораторная работа № 1), контролируемых значений, во втором анализируются данные из БД (лабораторная работа № 2). Подсистема контроля должна обеспечивать подачу сигналов оператору АС при поступлении в пакетах данных значений превышающих уставные аварийные и предаварийные значения (таблица значений предоставляется).

Общие сведения:

Жизненный цикл процесса создания АСУ согласно ГОСТ 34 (ГОСТ 34.601-90) включает следующие стадии:

- Формирование требований к АС;
- Разработка концепции АС;
- Техническое задание;
- Эскизный проект;
- Технический проект;
- Рабочая документация;
- Ввод в действие;

– Сопровождение АС.

Формирование требований к АС

На начальном этапе создания АС согласно требованиям ГОСТ 34 необходимо проведение обследования объекта автоматизации. В рамках обследования происходит сбор и анализ данных об организации, производственной структуре и функционировании объекта автоматизации. Источником для получения данных сведений могут послужить устав и регламенты организации, а также общегосударственные законы, постановления и другие нормативно-правовые акты.

Техническое задание (ТЗ)

Ключевая роль при создании АС отводится именно разработке и согласованию технического задания, так как он должен определять требования и порядок разработки, развития и модернизации системы. В соответствии с данным документом должны будут проводиться работы по испытанию и приемке системы в эксплуатацию. Техническое задание может быть разработано как на систему в целом так и на ее части.

Стандартом для разработки данного документа является ГОСТ 34.602-89, регламентирующий содержание разделов и стиль изложения в ТЗ.

Итак, согласно ГОСТ 34 техническое задание должно включать следующие разделы:

– Общие сведения;
– Назначение и цели создания (развития) системы;
– Характеристика объектов автоматизации;
– Требования к системе;
– Состав и содержание работ по созданию системы;

– Порядок контроля и приемки системы;

– Требования к составу и содержанию работ по подготовке объекта; автоматизации к вводу системы в действие;

– Требования к документированию;

– Источники разработки;

– Эскизный и технический проект.

Рабочая документация

Данный этап подразумевает разработку рабочей документации на АС или ее части. Данный пакет документов также согласовывается с заказчиком в индивидуальном порядке и фиксируется в ТЗ. Зачастую пакет рабочей документации ограничивается следующими документами:

– Руководство пользователя (администратора);

– Инструкция по эксплуатации КТС;

– Общее описание системы (в случае присутствия документа «Пояснительная записка к техническому (эскизному) проекту» данный документ нецелесообразен, так как большинство разделов дублируются);

– Программа и методика испытаний.

Ввод в действие

Стадия ввода в действие АС согласно ГОСТ 34 включает подготовку комплекса технических средств, проведение пусконаладочных работ и обучение персонала.

Перед вводом АС в эксплуатацию производятся предварительные испытания, по результатам которых формируется «Протокол испытаний». Протокол фиксирует все замечания к системе, порядок и сроки их устранения, и подтверждает ее готовность к вводу в опытную эксплуатацию.

После полной передачи системы обе стороны подписывают «Акт выполненных работ».

Сопровождение АС

Этап сопровождения АС подразумевает выполнение работ по гарантийному и послегарантийному обслуживанию системы.

Порядок выполнения работы:

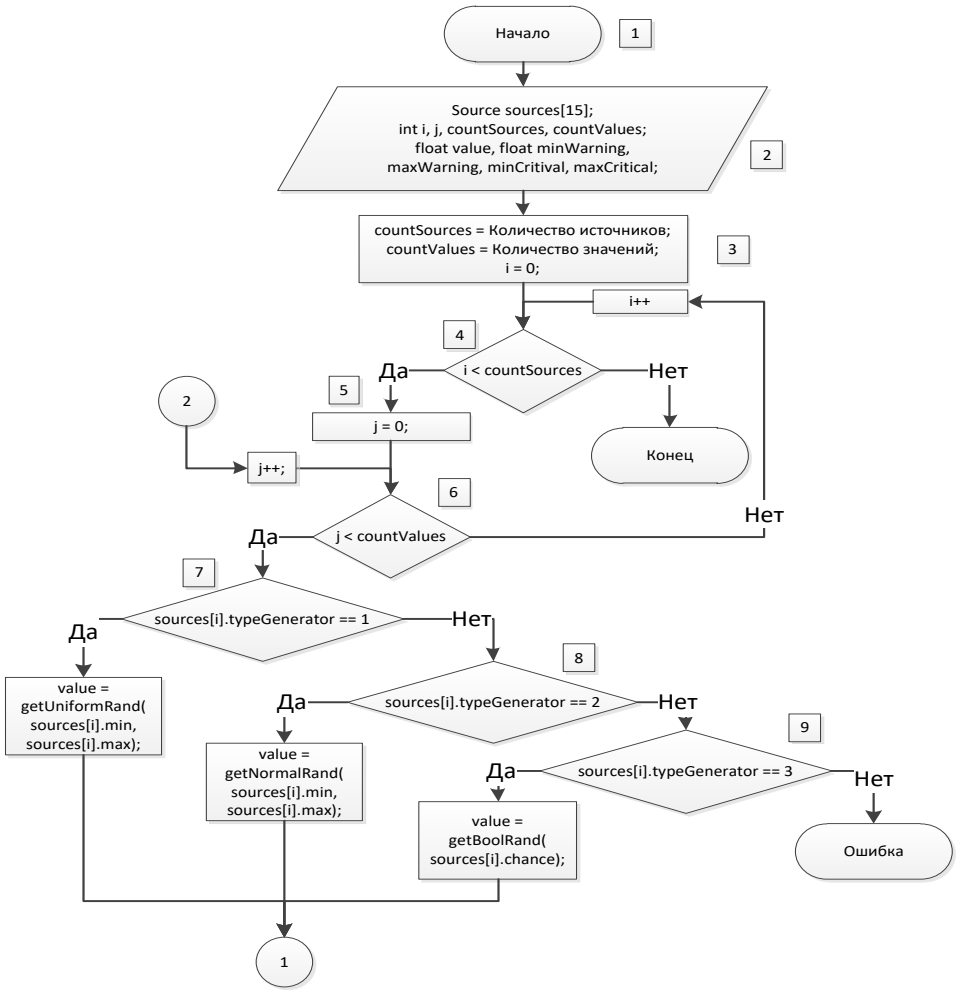


Рис. 8 «Общий алгоритм первая часть»

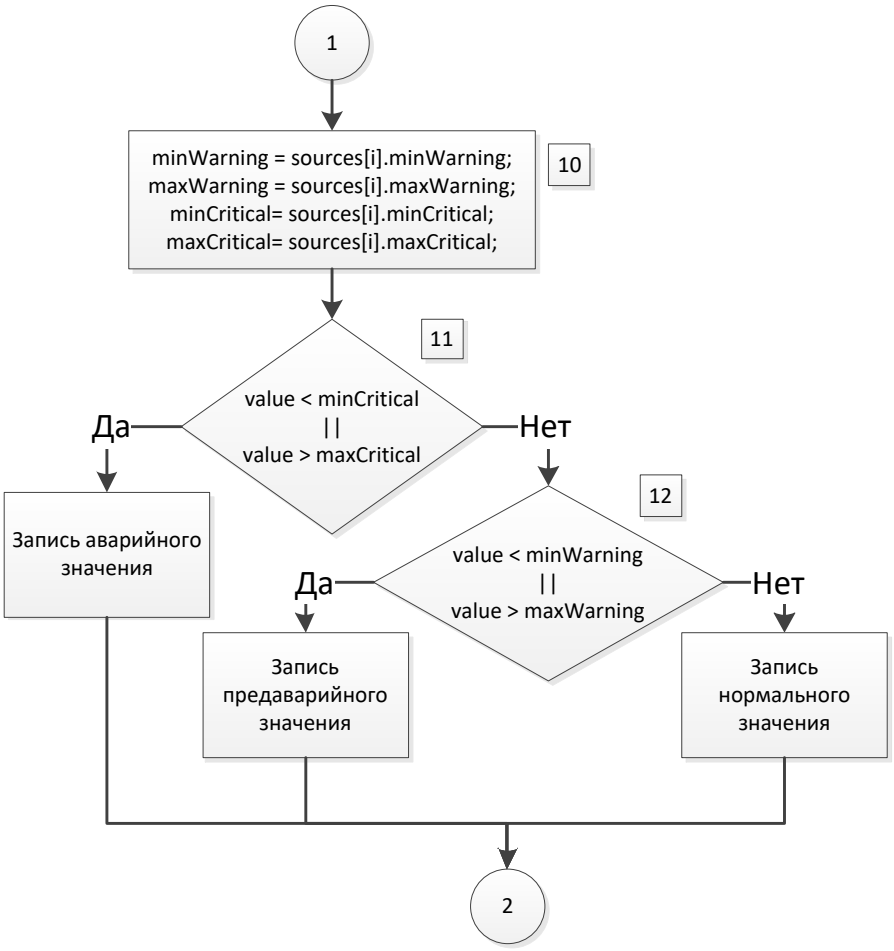


Рис. 9 «Общий алгоритм вторая часть»

1. Начало программы.

2. Объявление необходимых переменных, объект Source включает в себя такие переменные как typeGenerator (тип генератора), minWarning, maxWarning (минимальное и максимальное предаварийное значение), minCritival, maxCritical (минимальное и максимальное аварийное значение), chance (вероятность).

3. Переменной countSource присваиваем количество источников, переменной countValues присваиваем количество генерируемых значений, переменную счетчика цикла i обнуляем.

4. Цикл пока i<countSource, если условие ложно, то конец программы, если истинно, то тело цикла.

5. Обнуляем j – переменная счетчика цикла.

6. Цикл пока j<countValues, если условие ложно, то увеличиваем значения счетчика цикла i, если истинно, то проверки на тип генератора.

7. Если тип i генератора равен 1, то это генератор равномерного распределения и в переменную value записывается значение функции getUniformRand, если нет, то следующая проверка.

8. Если тип i генератора равен 2, то это генератор нормального распределения и в переменную value записывается значение функции getNormalRand, если нет, то следующая проверка.

9. Если тип i генератора равен 3, то это генератор с двумя исходами и в переменную value записывается значение функции getBoolRand, если нет, то такой генератор отсутствует, это ошибка, тогда конец программы.

10. Для упрощения и читаемости из i источника сохраняются в переменные предаварийные и аварийные значения.

11. Если value меньше минимального аварийного и больше максимального аварийного, то запись аварийного значения, иначе следующая проверка.

12. Если value меньше минимального предаварийного и больше максимального предаварийного, то запись предаварийного значения, иначе запись обычного значения.

В данном алгоритме запись означает запись в базу данных и в таблицу с конкретным цветом ячейки, если значение аварийное, выделяется красным, если предаварийное, выделяется оранжевым.

Содержание отчета:

Отчет о работе должен содержать титульный лист, наименование и цель работы, алгоритм генерации значения от типа генератора и контроль сгенерированного значения, выводы о проделанной работе.

Защита отчета:

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы.

Контрольные вопросы:

1. Регламентирующие документы по разработке и функционированию АС.
2. Специфические стандарты предприятий на АС.
3. Отраслевые стандарты АС.
4. Государственные стандарты АС.
5. Международные стандарты АС.
6. Проектная документация на АС.
7. Эксплуатационная документация на АС.

Лабораторная работа № 5

РАЗРАБОТКА ПОДСИСТЕМ ПОСТРОЕНИЯ И ОТОБРАЖЕНИЯ ТЕКУЩИХ И ИСТОРИЧЕСКИХ ТРЕНДОВ

Цель работы:

Необходимо разработать подсистему АС для отображения в экранных и печатных формах графиков трендов (изменения значений) выбранных величин с указанием уровня уставных значений (аварийных, предаварийных и выбранных контролируемых). Для текущих трендов выбирается отображаемая величина из поступающих от генератора данных, а также контролируемые значения, масштабы отображаемых величин. Для исторических трендов, необходимо обеспечить доступ к значениям выбранной величины в заданном временном диапазоне.

Общие сведения:

Совокупность видов обеспечения составляет комплекс средств автоматизации некоторой деятельности. Ни один из видов обеспечения не может быть опущен.

Различают следующие виды обеспечения АС:

- математическое;
- лингвистическое;
- информационное;
- программное;
- техническое;
- организационное.

В теории проектирования АС, но достаточно редко в практическом плане, определяют еще и **методическое обеспечение**, которое включает в себя общие принципы создания АС, математическое и лингвистическое обеспечение.

Математическое обеспечение - это *методы* и *алгоритмы* расчетов и принятия решений, которые используются в информационной технологии и АС. Например, метод узловых потенциалов или алгоритм выбора авиационного двигателя из имеющихся на рынке.

Лингвистическое обеспечение - это совокупность языков для реализации и эксплуатации АС. Таким образом, требуются языки программирования для реализации (например С++) и профессионально-ориентированные языки (ПОЯ) для эксплуатации АС (например язык менеджера, язык бухгалтера и т.п.). Как правило, формального описания ПОЯ не делают; он определяется и реализуется через интерфейс пользователя, т.е. входные и выходные формы, отчеты, документы, сообщения и команды.

Информационное обеспечение - это ***совокупность информации***, необходимой для нормального функционирования АС, представленная ***в заданной форме***. Таким образом, информационное обеспечение - это с одной стороны собственно информация, независимо от формы и способа ее

представления, а с другой стороны - это все что связано с СУБД и базой данных, т.е. структура, целостность, администрирование и т.д.

Программное обеспечение - это совокупность соответствующих систем программирования и прикладных программ, необходимых для работы АС.

Техническое обеспечение - это совокупность средств вычислительной техники, средств связи, расходных материалов и специального оборудования, необходимого для эксплуатации АС.

Организационное обеспечение - это совокупность должностных инструкций, распоряжений, приказов и т.п., регламентирующих права, обязанности и ответственность персонала, занятого эксплуатацией и обслуживанием АС. Например, "Должностная инструкция администратора базы данных" или "Положение о безопасности и секретности при работе с данными" и т.п. Организационное обеспечение не может противоречить уставу организации и существующему законодательству и, как правило, содержит ссылки на них.

Порядок выполнения работы:

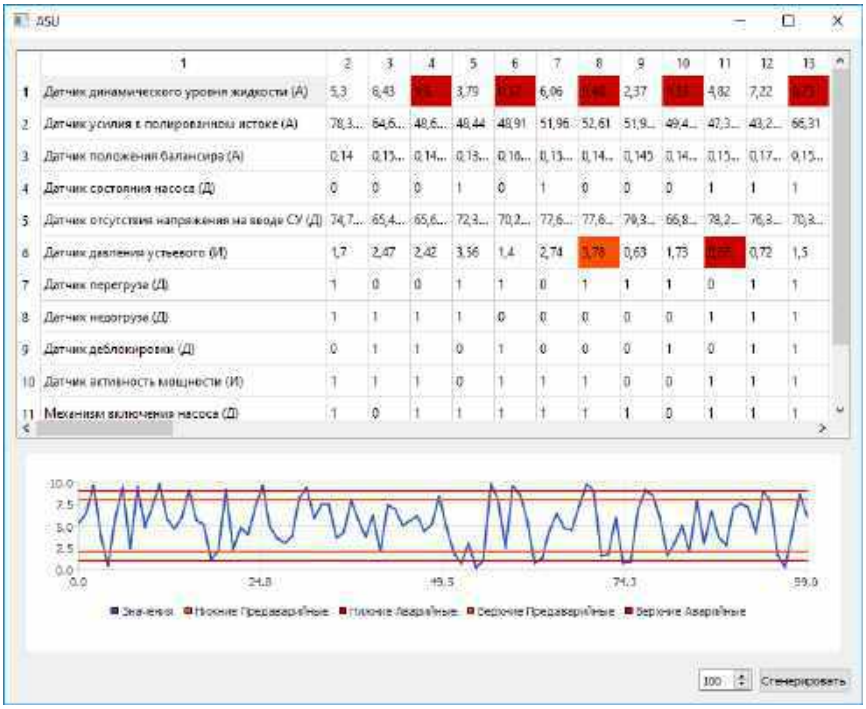


Рис. 10 «Интерфейс программы»

Интерфейс (рис.10) разработан на языке C++, в среде разработке QT, грубо его можно разделить на три части:

1. Таблица с данными датчиков. В ней сделано пользовательское меню, работает по следующему принципу: при нажатии на ячейки с названием источников, я нахожу значение соответствующего источника и вывожу его значения на график.

2. График с данными датчиков. Готовая библиотека для отображения графиков, выводит синим цветом значения источника, оранжевым предаварийные значения, красным аварийные. По оси x номер измерения от 1 до последнего, по оси y от минимального значения до максимального.

3. Количество значений и кнопка. 1) При нажатии на кнопку очищаются все сгенерированные значения. 2) Из spinbox берется количество значений и для каждого источника генерируется указанное количество значений.

Содержание отчета:

Отчет о работе должен содержать титульный лист, наименование и цель работы, программа генерации значений источников, краткое описание работы программы, выводы о проделанной работе.

Защита отчета:

Защита отчета о выполнении лабораторной работы сопровождается демонстрацией полученных результатов, теоретических знаний и ответов на дополнительные вопросы.

Контрольные вопросы:

1. Виды обеспечения АС.
2. Техническое обеспечение АС.
3. Информационное обеспечение АС.
4. Программное обеспечение АС.
5. Метрологическое обеспечение АС.
6. Лингвистическое обеспечение АС.
7. Организационное обеспечение АС.
8. Автоматизация проектирования АС.
9. Системы автоматизированного проектирования АС.
10. Методики проектирования.
11. Надежность компонентов и систем АС.
12. Автоматизированные системы сбора и обработки данных.
13. Автоматизированные системы управления организационного уровня (АСУ П).
14. Автоматизированные системы управления технологического уровня (АСУ ТП).

Лабораторная работа 6* Разработка компонентов систем поддержки принятия решений.

Разрабатываются отдельные компоненты СППР в соответствие с заданием, представленным в учебном пособии Системы искусственного интеллекта, часть 1.

Основная литература

1. Гергель В. П. Технологии построения и использования кластерных систем / В.П. Гергель - Москва: Интернет-Университет Информационных Технологий, 2009. - 470 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=233768

2. Методологические основы построения защищенных автоматизированных систем: учебное пособие / А.В.Душкин - Воронеж: Воронежская государственная лесотехническая академия, 2013. - 258 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=255851

3. Романенко А. В. Основы программирования для автоматизированных систем проектирования и управления инновациями: учебное пособие для студентов, обучающихся по направлению подготовки бакалавров "Инноватика" / А.В. Романенко; А.И. Попов - Тамбов, 2014. - 96 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=277966

4. Автоматизированные системы управления промышленными производствами / Е.П. Догадина - М.|Берлин: Директ-Медиа, 2017. - 344 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=454164

5. Сергеев Н. Е. Основы автоматизированных систем управления: учебное пособие / Н.Е. Сергеев - Ростов-на-Дону|Таганрог: Южный федеральный университет, 2019. - 129 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=598607

Дополнительная литература

1. Проскуряков А. Ю. Алгоритмы автоматизированных систем экологического мониторинга промышленных производств / А.Ю. Проскуряков; А.А. Белов; Ю.А. Кропотов - М.|Берлин: Директ-Медиа, 2015. - 121 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=429423

2. Комплексное обеспечение информационной безопасности автоматизированных систем - Ставрополь: СКФУ, 2016. -242 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=458012

3. ВолковаТ. В. Проектирование компонентов автоматизированных систем в примерах: учебное пособие / Т.В. Волкова;Е.Н. Чернопрудова - Оренбург: Оренбургский государственный университет, 2017. - 178 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=481817

4. Трофимов В. Б. Интеллектуальные автоматизированные системы управления технологическими объектами: учебно-практическое пособие / В.Б. Трофимов; С.М. Кулаков - Москва|Вологда: Инфра-Инженерия, 2017. - 233 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=466931

Список литературы

1. Адрющенко В.А. Теория систем автоматического управления. -Л.: ЛГУ, 1990. -251 с.

2. Автоматизация процессов принятия решений в системах управления /В.С.Симанков, Ю.К.Лушников, В.А.Морозов и др.: Аналитический обзор, 1970-1985 гг., № 4087. -М.: ЦНИИТЭИ, 1986. - 42 с.

3. Автоматизированное проектирование автономных комплексов, использующих возобновляемые источники энергии на основе интеллектуальной подсистемы / В.С. Симанков, Т.Т.Зангиев, П.Е.Сельманов и др. //Современные проблемы нетрадиционной энергетики: Сб. тез. докл. Международной научно-технической конференции, 1994. -С.19-21.

4. ГОСТ 27.301-9 Надежность в технике. Расчет надежности. Основные положения. М.: Издательство стандартов, 1997. - 15 с.

5. ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения. М.: Издательство стандартов, 1987. - 17 с.

6. ГОСТ 34.601-90. Автоматизированные системы управления. Основные положения. Термины и определения. -М.: Изд-во стандартов, 1974. -5 с.

7. Дорри М. Х., Рощин А. А. Инструментальные средства "Экспресс-Радиус" для автоматизации динамических расчетов систем управления. Приборы и системы управления, #3, 1996 г.

8. Кальфа В., Овчинников В.В. Основы автоматизации управления производственными процессами. -М.: Советское радио, 1980. -410 с.

9. Научные основы организации управления и построения АСУ /Под. ред В.П. Бройдо, В.С. Крылова. Изд. 2-е, перераб. и доп. -М.: Высшая школа, 1990. -320 с.

10. Симанков В.С., Луценко Е.В. Синтез адаптивных АСУ сложными системами с применением моделей распознавания образов Краснодар,1998. Деп. в ВИНИТИ 18.09.98, №2839-В98. Ж-л: Автоматизация и современные технологии.1999, №1.

11. Смилянский Г. Л. Какая АСУ эффективна? (Руководителю об автоматизированных системах управления). -М.: Экономика, 1988. - 303 с.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Системная интеграция и корпоративные информационные системы»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024 г.

Введение

Информационная инфраструктура предприятия претерпевает постоянные изменения. Этот процесс может происходить как целенаправленно, в рамках определенной ИТ-стратегии, так и стихийно, под влиянием насущных потребностей бизнеса. Результатом является крайне неоднородный ИТ-ландшафт, содержащий приложения и программные компоненты от разных производителей, которые реализованы на разных платформах и зачастую дублируют отдельные функции. Ситуацию усугубляют процессы слияния и поглощения компаний, приводящие к наследованию новых информационных систем и приложений.

В современных условиях возрастает зависимость бизнеса от информационных технологий, причем для его успешного развития важен не столько набор приложений, автоматизирующих отдельные функции или бизнес-процессы, сколько интеграция и взаимосогласованность информационных систем и приложений. Следует выделить три аспекта, которые обусловливают исключительную актуальность проблемы интеграции.

Во-первых, повсеместное использование информационных технологий приводит к лавинообразному росту объемов цифровой информации. По оценкам компании IDC, ежегодно объем данных увеличивается более чем на 60%. Эта информация распределена по многочисленным гетерогенным приложениям, системам, настольным базам данных и мобильным устройствам. Для эффективного использования распределенных данных должны быть решены проблемы сбора, синхронизации и использования релевантной информации в масштабах предприятия. Речь, прежде всего, идет об обеспечении качества данных, управлении данными и своевременной доставке информации потребителю.

Во-вторых, необходимо поддерживать сквозные бизнес-процессы, охватывающие различные подразделения компании и внешних контрагентов. На сегодняшний день очевидны необходимость интеграции приложений для

поддержания таких стратегических направлений бизнеса, как электронная коммерция, управление цепями поставок, управление взаимоотношениями с клиентами (CRM, Customer Relationship Management), а также необходимость обмена информацией и сервисами между информационными системами в пределах предприятия.

В-третьих, требования бизнеса всегда опережают возможности информационных технологий, поэтому чрезвычайно важно уметь использовать функционал унаследованных систем для поддержки преобразований бизнеса и сохранять инвестиции в информационные технологии.

Интеграционные проекты достаточно дорогостоящи. По подсчетам компании Gartner Group, 80% ИТ-бюджетов компаний составляют расходы на сопровождение систем, из них 35% – затраты на интеграцию приложений, 60% стоимости внедрения корпоративной информационной системы составляют расходы на интеграцию.

Таким образом, интеграция разнородных приложений и систем становится все более и более значимой проблемой развития ИТ-инфраструктуры. Условие успешного развития любой компании на современном этапе – создание ИТ-инфраструктуры, в которой интегрированы все ее вычислительные, информационные и коммуникационные ресурсы.

Задача выбора интеграционной стратегии, соответствующей потребностям бизнеса, нетривиальна несмотря на то, что современный уровень развития информационных технологий позволяет успешно преодолевать технологические трудности связывания приложений.

1. Общие вопросы интеграции корпоративных информационных систем

1.1. Эволюция подходов к интеграции информационных систем

Концепция интеграции информационных систем далеко не нова. Необходимость интеграции стала очевидной, как только на предприятиях появилось более одной информационной системы и локальная сеть.

В процессе развития информационных технологий подходы к решению задачи интеграции менялись. В 1970–80-е гг. корпоративные приложения выполняли достаточно простые функции, сводящиеся к автоматизации отдельных операций и решению относительно несложных, легко формализуемых задач. Постепенное наращивание функциональности привело к возникновению модульной архитектуры систем. В 1990-е гг. считалось, что единственно правильным направлением комплексной автоматизации является создание универсальной информационной системы, охватывающей все области деятельности предприятия. Такая система должна быть основана на единой программно-аппаратной платформе и общей базе данных, а ее отдельные функциональные подсистемы (модули) – взаимосвязаны на основе единого технологического процесса обработки информации.

Классическим примером модульных информационных систем являются системы класса ERP (Enterprise Resource Planning), включающие модули для управления производством, планированием, договорами, материально-техническим снабжением, финансами, кадрами, сбытом, запасами и т.д. Все модули ERP-системы «интегрированы» в единую комплексную информационную систему компанией-разработчиком. Внедрение ERP-системы, как считалось, решает проблему взаимосвязи приложений и снимает необходимость вкладывать значительные средства в интеграцию. В финансовой отрасли ставка делалась на автоматизацию традиционных задач банковской деятельности (ведение бухгалтерского учета, получение обязательной отчетности, автоматизированное расчетно-кассовое обслуживание клиентов, кредитно-депозитная деятельность) и построение

автоматизированной банковской системы (АБС).

АБС также имеют модульный принцип построения, позволяющий легко конфигурировать системы под конкретное предприятие с возможностью последующего наращивания функционала.

Тем не менее подход, направленный на формирование масштабной универсальной системы от одного производителя, как показала практика, не снимает проблему интеграции приложений в силу следующих обстоятельств:

- в условиях развития бизнеса высока потребность в индивидуальных информационных системах. Тиражные решения не всегда позволяют персонифицировать приложения за счет возможностей настройки, к тому же они могут оказаться слишком дорогими, поэтому предприятия используют собственные приложения или приложения других вендоров, реализующие необходимую им функциональность;
- в компаниях всегда остается несколько «устаревших» приложений, замена которых модулями комплексной системы может занять много времени. На время «переходного» периода такие приложения должны быть интегрированы;
- слияния и поглощения компаний являются источником интеграционных проблем: часто в компаниях используются ERP-системы от различных поставщиков, в банках – АБС от разных производителей;
- существует проблема взаимодействия с внешними контрагентами компании;
- необходимость автоматизации новых направлений деятельности (электронные карты, электронные торговые площадки, мобильная связь, облачные сервисы и др.).

Одновременно с развитием универсальных информационных систем возник рынок специализированного программного обеспечения, конкурирующего с отдельными модулями универсальных систем. Конкурентное преимущество такого программного обеспечения – специализация, основанная на глубоком знании отдельных процессов и

технологий. Постепенно специализированные программы стали неотъемлемой частью ИТ-инфраструктуры большинства предприятий наряду с многочисленными вспомогательными системами, являющимися зачастую продуктами собственной разработки. Возникло множество связей между центральной и вспомогательной системами, а также прямых связей между вспомогательными системами (рис. 1.1).

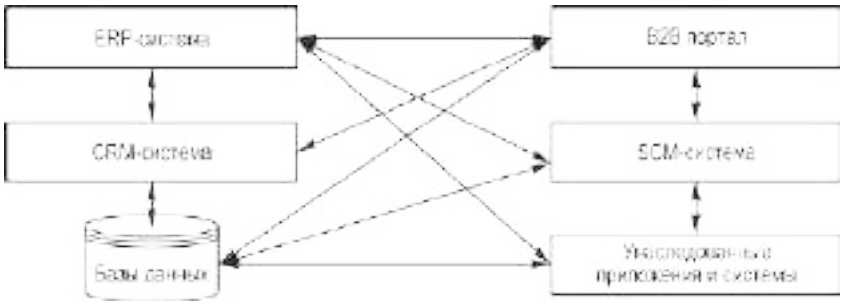


Рис. 1.1. Неупорядоченная ИТ-инфраструктура

На сегодняшний день интегрированные системы постепенно теряют лидерство в ИТ-архитектуре. Этот процесс можно проследить на примере крупных и средних банков, которые отдали предпочтение архитектуре, в которой АБС играет только роль учетной системы, а все основные бизнес-задачи решаются специализированными приложениями, интегрированными между собой.

Принято противопоставлять два крайних подхода к развитию ИТ-инфраструктуры предприятия – это мультии моновендорная стратегии. И та и другая стратегия имеет свои достоинства и недостатки.

Преимуществом моновендорной стратегии (внедряются продукты одного вендора) принято считать готовую методологию внедрения и проработанные вопросы интеграции и взаимодействия компонентов. Однако платой за «простоту» интеграции может быть множество ненужных функций или дублирование уже имеющейся функциональности, а также излишняя «зависимость» от одного поставщика, ограничивающая свободу выбора программных решений. К тому же продукты одного вендора зачастую создаются разными командами с использованием различных технологий,

иногда продукты наследуются крупным вендором вместе с приобретаемой компанией. Для таких продуктов вопросы их интеграции не решены самим вендором.

Мультивендорная стратегия основана на подходе «best-of-bread» (лучшего в своем классе) и требует значительно бóльших затрат на встраивание инородных приложений в сложившуюся архитектуру. Интеграция между информационными системами строится на основе так называемой трехуровневой модели, поскольку каждое биз-

нес-приложение можно представить тремя логическими слоями:

1) уровнем представления (уровень пользователя, решающего задачи ввода/вывода информации);

2) уровнем бизнес-логики (обработка данных, поддержка логики бизнес-процессов);

3) уровнем данных (хранение и администрирование данных).

Связь между приложениями может быть установлена на каждом из этих уровней и требует более или менее «инвазивного» вмешательства в тот или иной слой, что усложняет как сам процесс интеграции, так и совместное функционирование систем.

Более универсальный подход к интеграции приложений основан на использовании программного обеспечения класса middleware.

Системы middleware первого поколения предоставляли дополнительный уровень – уровень передачи сообщения. Под сообщением понимался структурированный набор параметров, описывающих определенный тип транзакции. Система middleware распознавала формат сообщения, исходящего от одного бизнесприложения, и трансформировала его в формат принимающего бизнес-приложения. Набор параметров разных типов транзакций и правила переформатирования хранились в базе данных системы middleware.

Принцип работы систем middleware первого поколения иллюстрирует рис. 1.2.

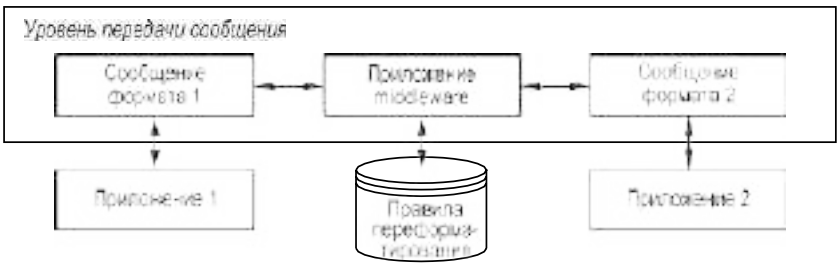


Рис. 1.2. Принцип работы систем middleware первого поколения

Такая «одномерная» модель взаимодействия приложений со временем устарела, и на смену ей пришла технология, основанная на архитектуре ESB (Enterprise Service Bus), которая обеспечивает управляемое взаимодействие между всеми приложениями, подключенными к общей шине предприятия.

Современные промышленные системы класса middleware – это сложное программное обеспечение, способное обрабатывать сообщения на базе универсальных форматов и обеспечивать многоканальную передачу сообщений между всеми бизнес-приложениями.

Основными компонентами таких систем являются:

- интеграционный брокер, играющий роль сервисной шины (выполняет функции переформатирования данных, маршрутизации сообщений, управления транзакциями, мониторинга и контроля взаимодействия приложений);
- набор адаптеров, которые позволяют различным приложениям подключаться к брокеру.

Принцип работы современных систем middleware иллюстрирует рис. 1.3.

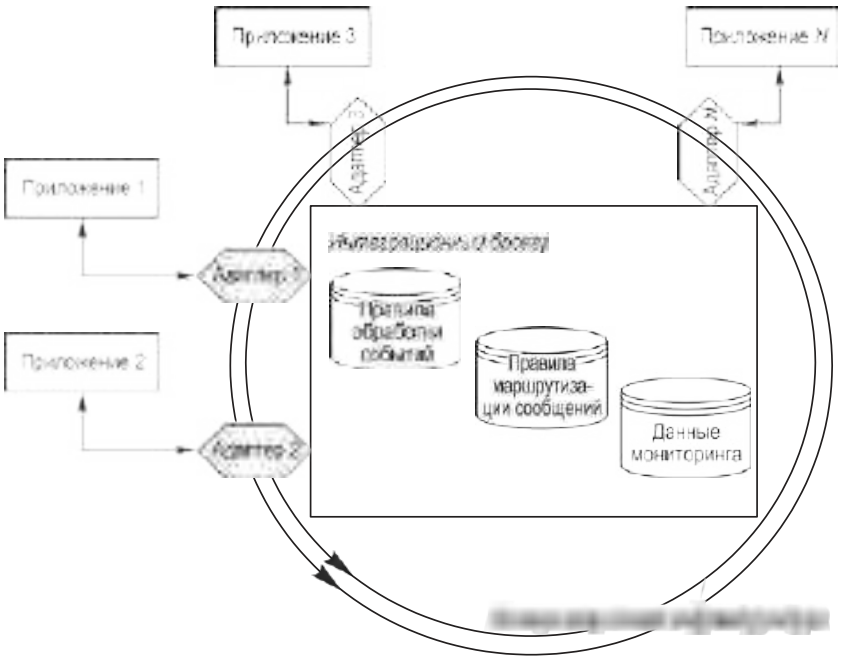


Рис. 1.3. Принцип работы современных систем middleware

1.2. Методология открытых систем и проблема интеграции

В процессе создания интеграционных решений неизбежно возникает проблема стыковки различных программных компонентов. Данная проблема проявляется при расширении систем, включении новых подсистем или новых версий компонентов, тиражировании и повторном использовании прикладных программ, организации обмена данными между приложениями.

Минимизировать затраты на интеграцию и развитие информационных систем позволяет использование методологии открытых систем, которая подразумевает выделение в системе интерфейсной части, обеспечивающей связь с другими системами или подсистемами. Для объединения систем достаточно располагать сведениями только об интерфейсных частях сопрягаемых объектов, выполненных в соответствии с определенными стандартами.

Стандарты, обеспечивающие открытость программного обеспечения, разрабатываются и поддерживаются рядом международных организаций, в частности:

- Совместным техническим комитетом ИСО и МЭК (ISO/IEC/ JTC 1) – ИСО/МЭК/СТК 1 «Информационные технологии»;

▪ Европейской рабочей группой по открытым системам (EWOS);

▪ Национальным институтом стандартов и технологий США (NIST). Методология открытых систем обеспечивает экономию инвестиций при построении информационных систем на различных аппаратных и программных платформах за счет совместимости прикладного

программного обеспечения и сохранения данных.

Открытые системы – это системы, в которых реализован «исчерпывающий и согласованный набор международных стандартов информационных технологий и профилей функциональных стандартов, которые специфицируют интерфейсы, сервисы и поддерживаемые форматы данных, чтобы обеспечить интероперабельность и мобильность приложений, данных и персонала».

Основными нефункциональными требованиями к открытым системам являются:

▪ расширяемость (масштабируемость) – возможность добавлять новые или модернизировать уже имеющиеся функции без изменения остальных функциональных частей информационной системы, а также сохранять работоспособность системы при изменении значений параметров, определяющих технические и ресурсные характеристики системы и/или среды;

▪ мобильность (переносимость) – возможность переносить программы и данные при модернизации или замене аппаратных платформ информационной системы, а также использовать опыт людей, пользующихся данной информационной технологией, без их переучивания при изменении информационной системы;

▪ интероперабельность – обеспечение способности к взаимодействию данной информационной системы с другими системами.

Методологию открытых систем определяют следующие документы:

▪ Технический отчет ISO/IEC TR 10000 Framework and taxonomy of International Standardized Profiles (Основы и таксономия международных

стандартизованных профилей);

- Эталонная модель окружения (среды) открытых систем (RM OSE) – ISO/IEC DTR 14252, Portable Operating System Interface for Computer Environments – POSIX (IEEE, P1003.0, Draft Guide to the POSIX Open System Environment);
- Эталонная модель взаимосвязи открытых систем (RM OSI) – ISO 7498: 1996, Information processing systems – Open Systems Interconnection – Basic Reference Model (ITU-T Rec. X.200).

Сущность методологии открытых систем состоит в том, что при их построении стыковка должна обеспечиваться использованием стандартных интерфейсов между всеми компонентами систем.

Выделяют две группы стандартов:

1) стандарты прикладных программных интерфейсов (Application Program Interface (API) Standards), которые определяют взаимодействие прикладного программного обеспечения с компьютерной системой (прикладной платформой) и предназначены в основном для обеспечения переносимости приложений;

2) стандарты внешнего окружения (External Environment Interface (EEI) Standards), которые определяет взаимодействие информационной системы с ее внешним окружением и предназначены для решения проблем интероперабельности систем, повторного использования программного обеспечения и переносимости данных.

Спецификации интерфейсов взаимодействия – это строгие описания всех необходимых функций, служб и форматов определенного интерфейса. Совокупность таких описаний составляет референтную модель открытых систем.

Интерфейсы взаимодействия компонентов систем иллюстрирует рис. 1.4.

Сегодня методология открытых систем поддерживается всеми компаниями – разработчиками программного обеспечения, производителями

средств вычислительной техники и средств телекоммуникаций.

Практически все современные информационные системы имеют набор хорошо документированных

API для доступа к данным или

функциям системы из различных сред программирования.

Рис. 1.4. Интерфейсы взаимодействия компонентов систем

Интеграция приложений – это стратегический подход к объединению информационных систем, который обеспечивает возможность обмена информацией и поддержания распределенных бизнес-процессов. Интеграция информационных систем дает предприятию такие несомненные конкурентные преимущества, как:

- ведение бизнеса в режиме реального времени с использованием событийно-управляемых сценариев;
- владение достоверной, полной и своевременно полученной информацией.

Задача интеграции – обеспечить эффективный, надежный и безопасный обмен данными между различными программными продуктами, изначально не предназначенными для совместной работы.

Как правило, требования бизнеса эволюционируют быстрее, чем способы их поддержки информационными технологиями.

Основными движущими силами интеграции являются:

- электронный бизнес – интеграция унаследованных информационных систем, поддерживающих ключевую функциональность, с Web-приложениями (Web-сервисами и порталами) с целью получения доступа к

бизнес-функциям через Интернет;

- управление цепями поставок – интеграция разрозненных систем управления заказами, MRP-систем, систем календарного планирования, систем транспортного менеджмента с целью прямого обмена информацией между покупателями и поставщиками в режиме реального времени;
- управление взаимоотношениями с клиентами – получение единого консолидированного представления о клиенте путем объединения данных о нем, распределенных между несколькими изолированными приложениями (интеграция клиентских баз данных, call-центров, интернет-сервисов);
- внедрение ERP – интеграция модулей ERP-систем, поддерживающих базовую функциональность, со специализированным программным обеспечением, используемым организацией;
- электронное правительство – интеграция унаследованных backend систем с front-end Web-приложениями, организация обмена данными между правительственными учреждениями;
- самообслуживание клиентов – возможность клиентов самостоятельно выполнять действия, традиционно являющиеся функцией обслуживающего персонала, требует интеграции пользовательских приложений с back-end-системами;
- Business Intellegence – сбор данных из различных приложений и источников в хранилище данных с целью их обработки и анализа;
- управление знаниями – обеспечение доступа в режиме реального времени к корпоративному контенту, распределенному между многочисленными источниками, с целью управления знаниями в масштабах предприятия;
- облачные технологии – интеграция существующих бизнес-приложений с облачными приложениями и сервисами;
- аутсорсинг бизнес-процессов – интеграция с информационными системами партнеров.

Перечислим основные бизнес-выгоды, которые предприятие может

получить в случае успешной реализации интеграционного проекта:

- улучшение качества поддержки и обслуживания клиентов;
- автоматизация бизнес-процессов;
- уменьшение производственного цикла;
- сокращение количества ошибок обработки данных;
- прозрачность процессов;
- уменьшение стоимости транзакций;
- оптимизация логистических процессов;
- более тесное взаимодействие с бизнес-партнерами;
- быстрое внедрение новых бизнес-сервисов;
- сохранение инвестиций в информационные технологии.

1.3. Типы интеграционных решений: горизонтальная и вертикальная интеграция

Интеграционные решения можно классифицировать разными способами. Например, в зависимости от принадлежности объединяемых приложений выделяют:

- интеграцию корпоративных приложений в пределах предприятия (Application-to-Application Integration – A2A) – автоматический событийно-управляемый обмен информацией между приложениями и системами, действующими на предприятии или в организации;
- интеграцию приложений между предприятиями (Business-toBusiness Application Integration – B2B) – автоматический событийно-управляемый обмен информацией между приложениями или системами нескольких взаимодействующих предприятий или организаций.

Как известно, информационные системы обеспечивают поддержку одного из трех уровней управления: операционного, тактического и стратегического.

В данном контексте существует два варианта построения интеграционного решения:

- горизонтальная интеграция – интеграция информационных систем или

приложений, относящихся к одному уровню,

▪ вертикальная интеграция – интеграция приложений и систем, находящихся на различных уровнях информационной пирамиды. Типичным примером горизонтальной интеграции является автоматизация управления цепями поставок (различные приложения или компоненты обеспечивают полный цикл логистических операций).

На рис. 1.5 показан распределенный бизнес-процесс, поддерживаемый несколькими приложениями.

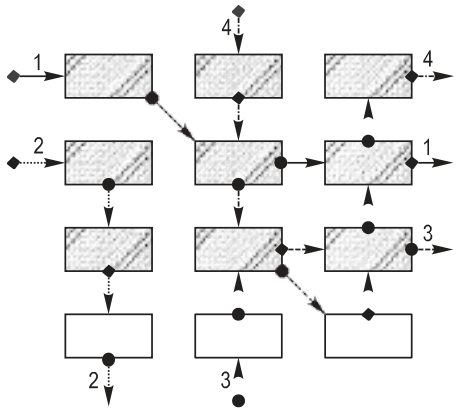


Рис. 1.5. Пример горизонтальной интеграции

Прямоугольниками обозначены приложения, выполняющие отдельные функции, штриховкой – приложения, находящиеся в организации. Процесс 1 является внутренним процессом организации. Процесс 2 начинается в организации и завершается за ее пределами. Процесс 3 начинается за пределами организации, но заканчивается в организации. Процесс 4 выходит за пределы организации и возвращается в нее перед завершением.

Наиболее часто встречающийся пример вертикальной интеграции – сбор данных операционных систем в единое корпоративное хранилище данных с целью их последующего использования для анализа, управления и получения консолидированной отчетности. Рассмотрим интеграционное решение, типичное для многофилиальной территориально распределенной организации (рис. 1.6).

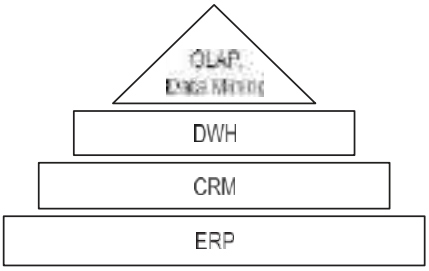


Рис. 1.6. Пример вертикальной

В хранилище данных консолидируется оперативная бизнес-интеграции информация из различных автоматизированных модулей, установленных в офисах и филиалах предприятия, включая территориально удаленные подразделения. Данные поступают в хранилище из разнообразных транзакционных систем, то есть из систем, ориентированных на управление отдельными операциями (например, систем класса CRM или ERP), разрозненных баз данных, электронных таблиц и других источников. Информация, накопленная в хранилище данных, используется приложениями, обеспечивающими технологии бизнес-планирования, управленческого учета и стратегического прогнозирования. Анализ данных выполняется с использованием различных BI-инструментов (OLAP, Data mining).

1.4. Проблемы интеграции

Интеграция приложений – сложная задача. Можно выделить несколько групп проблем, с которыми сталкиваются разработчики интеграционных решений:

1) технические проблемы:

- риск задержки и потери информации, вызванный ненадежностью сетей передачи данных (обеспечивая связь между отдельными частями интеграционного решения, необходимо передавать информацию между устройствами по телефонным линиям, локальным сетям, через маршрутизаторы, общедоступные сети и каналы спутниковой связи);
- недостаточная скорость передачи данных (объективно время доставки данных через компьютерную сеть всегда больше времени вызова локального

метода, поэтому интеграционное решение всегда работает медленнее локального приложения);

- необходимость учитывать все различия между приложениями (разные языки программирования, на которых написаны приложения, разные платформы, разный формат данных);
- ограниченный контроль над интегрируемыми приложениями, представляющими собой, например, унаследованные коммерческие приложения с закрытой структурой баз данных или плохо документированные системы, разработанные ИТ-департаментом;
- необходимость поддерживать интеграционное решение, то есть адаптировать его к изменениям в интегрируемых приложениях. Зачастую изменения в одной системе влекут за собой непредсказуемые изменения в других;

2) методологические проблемы:

- отсутствие корректного формата или семантического слоя для слияния двух и более несопоставимых наборов данных. Устранение семантических различий между приложениями – одна из наиболее сложных и неформализуемых задач интеграции. Действительно, одна и та же сущность (например, «Клиент») может иметь несколько различных семантик, ограничений и допущений в каждой системе. Эта проблема в англоязычной литературе получила название семантического диссонанса;
- необходимость наличия методологии для документирования таких технических аспектов интеграции, как определение записей, структур данных, интерфейсов и потоков данных в масштабах всей организации;
- необходимость определения правил и методов согласования данных во всех интеграционных проектах (правил устранения семантического диссонанса);

3) организационные проблемы:

- недостаток доверия к корректности информации;
- интеграция приложений в большинстве случаев влечет за собой

пересмотр корпоративной политики компании. Объединение информационных систем (например, CRM-системы и системы бухгалтерского учета) требует изменений в порядке взаимодействия между использующими эти системы отделами (отделом маркетинга и бухгалтерией);

- при интеграции ключевых бизнес-функций компании ее деятельность становится зависимой от функционирования интеграционного решения, сбой в работе которого может принести компании существенные убытки (ошибочные платежи, потерянные заказы и т.д.).

Для обоснованного выбора способа интеграции необходимо обладать информацией о характеристиках интегрируемых приложений, а также уметь оценивать влияние способа интеграции на общую архитектуру интеграционного решения.

Следует понимать, что универсального подхода к интеграции приложений не существует, как не существует и единственного варианта интеграционного решения, но всегда можно найти оптимальный способ интеграции в рамках конкретного интеграционного сценария. Ниже перечислены основные критерии, влияющие на выбор способа интеграции приложений:

1) степень связывания приложений. Зависимости между интегрируемыми приложениями должны быть сведены к минимуму (реализация принципа слабой связи между приложениями). Сильное связывание предполагает наличие большого числа допущений между приложениями. Изменение функциональности или сбой в работе одного приложения может привести к нарушению допущений и потере работоспособности интеграционного решения. Результатом сильного связывания являются неустойчивые, плохо масштабируемые и трудно поддерживаемые решения. В идеале интерфейсы объединяемых приложений должны обеспечивать необходимую функциональность, допуская возможность изменения внутренней реализации;

2) простота поддержки интеграционного решения. Следует

стремиться к минимизации изменений, вносимых в объединяемые приложения, и объема кода, необходимого для интеграции;

3) объем данных. Выбор способа интеграции зависит от объема передаваемых данных. Следует учитывать, что несвоевременная доставка или потеря данных приведет к рассинхронизации приложений;

4) стоимость решения. Отдельные интеграционные решения требуют применения специального программного обеспечения (класса middleware), что существенно увеличивает стоимость проекта интеграции. Вместе с тем более «бюджетная» разработка проекта интеграции «с нуля» намного более рискованна и может свести усилия разработчиков к «изобретению велосипеда». В каждом проекте инструмент интеграции необходимо выбирать исходя из масштабов и сроков проекта, задач интеграции, бизнес-требований и наличия квалифицированных кадров.

2. Технологии и стандарты интеграции

2.1. Интеграция на основе промежуточного слоя (middleware)

Технически задача интеграция приложений решается с использованием тех же технологий, что и связывание компонентов распределенных информационных систем.

Взаимодействие удаленных систем описывается в рамках модели взаимодействия открытых систем OSI/ISO, разработанной Международной организацией по стандартам (International Standards Organization, ISO) с целью стандартизации способов связи между хостами от разных производителей.

Модель OSI/ISO разделяет процесс взаимодействия двух сторон на семь взаимосвязанных уровней и определяет функции каждого из них. Каждый уровень поддерживает интерфейсы с вышеи нижележащими уровнями (табл. 2.1).

В простейшем случае взаимодействие между двумя удаленными приложениями можно обеспечить с использованием протокола TCP/IP. Интерфейс к транспортному уровню обеспечивает операционная система,

предоставляя механизм, основанный на сокетах.

Таблица 2.1

Уровни модели OSI/ISO и их характеристики

№	Название	Функции	Пример протокола	Реализация
1	Физический	Передача битов по физическим линиям связи. Определяет механические, электрические и оптические характеристики кабелей и разъемов физического интерфейса сети	Спецификация 10Base-T	Канал связи
2	Канальный	Обеспечивает взаимодействие между двумя компьютерами. Функции – проверка доступности среды передачи, реализация механизмов обнаружения и коррекции ошибок. В протоколах канального уровня, используемых в локальных сетях, заложены определенная структура связей между компьютерами и способы их адресации	Ethernet, Token Ring, FDDI, 100VG-AnyLAN	Операционная система

3	Сетевой	Реализует взаимодействие узлов сети. Этот уровень служит для образования единой транспортной системы, объединяющей несколько сетей с различными принципами передачи информации между конечными узлами	Протокол межсетевого взаимодействия IP стека TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell	
4	Транспортный	Обеспечивает возможность передачи более длинных сообщений между хостами	Протоколы TCP и UDP стека TCP/IP и протокол SPX стека Novell	
5	Сеансовый	Устанавливает и поддерживает соединение между компонентами распределенной системы. Использует механизм соединений транспортного уровня		Промежуточная среда
6	Представительский	Устраняет различия в представлении данных. Этот уровень гарантирует то, что информация, передаваемая прикладным	Secure Socket Layer (SSL)	

		уровнем, будет понятна прикладному уровню в другой системе. При необходимости уровень преобразовывает данные в некоторый общий формат представления или выполняет обратное преобразование		
7	Прикладной	Обеспечивает функционирование приложения. Набор протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам	Протокол электронной почты	Компонент информационной системы

Сокеты обеспечивают примитивы низкого уровня для непосредственного обмена потоком байт между двумя процессами и могут быть использованы для организации простейшего взаимодействия удаленных приложений.

Приведем пример. Пусть клиенту банка необходимо обеспечить возможность перевода денег на свой счет из другого банка, используя Web-приложение (задача интеграции Web-приложения с финансовой системой). Простейшее решение может быть получено с использованием протокола TCP/IP.

Для определенности представим, что необходимо передать только имя клиента и сумму денежного перевода. Приведенный ниже пример кода (C#)

демонстрирует вариант реализации конечной точки Web-приложения, в котором создается сокет с адресом my.bank.com и по сети пересылается сумма и имя клиента.

```
String hostName = "my.bank.com";
//Задание DNS-имени удаленного компьютера
int port = 80;
IPHostEntry hostInfo = Dns.GetHostByName (hostName);
//Преобразование имени удаленного компьютера в IP-адрес
IPAddress address = hostInfo.AddressList [0];
IPEndPoint endpoint = new IPEndPoint (address, port);
Socket socket = new Socket (address.AddressFamily, SocketType.
Stream, ProtocolType.Tcp);
socket.Connect (endpoint);
byte [] amount = BitConverter.GetBytes (1000);
//Преобразование данных в поток байтов
byte [] name = Encoding.ASCII.GetBytes ("Joe");
int bytesSent = socket.Send (amount);
bytesSent += socket.Send (name);
socket.Close ();
```

На первый взгляд простой метод межплатформенной интеграции имеет ряд ограничений:

- связываемые системы должны иметь одинаковое внутреннее представление целых чисел (32 или 64 бит) и одинаковый порядок следования байтов (прямой или обратный). В противном случае передаваемые данные будут неверно интерпретированы приложением-получателем;
- компьютер должен иметь неизменяемый адрес. Перемещение приложения получателя на другой компьютер потребует изменения программного кода;
- интегрируемые приложения должны быть доступны в одно и то же время, поскольку протокол TCP/IP является протоколом с установкой соединения. Если одно из взаимодействующих приложений в момент установки соединения недоступно, то соединение не будет установлено;
- использование соглашения о формате данных, предаваемых между интегрируемыми приложениями. Любое изменение в формате потребует изменений в программном коде как на стороне отправителя, так и на стороне получателя.

Таким образом, использование механизма сокетов приводит к получению плохо масштабируемого, неустойчивого, трудно

поддерживаемого интеграционного решения, обеспечивающего «жесткую» связь между приложениями.

В середине 1990-х гг. появилось понятие промежуточной среды как некоторой дополнительной, не зависящей от операционной системы и приложений «прослойки», реализующей сервисы высокого уровня для инкапсуляции удаленного взаимодействия между программными компонентами. Промежуточная среда (middleware – связующее программное обеспечение) выполняет функции сеансового и представительского уровней модели OSI/ISO. Наличие промежуточной среды позволяет создать открытые, масштабируемые и отказоустойчивые распределенные системы.

Промежуточная среда предоставляет:

- единый, независимый от операционной системы механизм использования одними программными компонентами сервисов других компонентов;
- безопасность – аутентификация и авторизация всех пользователей сервисов компонента и защита передаваемой между компонентами информации от искажения и чтения третьей стороной;
- целостность данных – управление транзакциями, распределенными между удаленными компонентами системы;
- балансировку нагрузки на серверы с программными компонентами;
- обнаружение удаленных компонентов.

В зависимости от модели взаимодействия компонентов можно выделить следующие типы промежуточных сред:

- ориентированные на посылку сообщений – поддерживают связь между компонентами распределенной системы посредством обмена сообщениями (Microsoft Message Queuing, IBM MQSeries, Sun Java System Message Queue);
- объектно-ориентированные – построены на модели удаленного обращения к методу (J2EE, .NET, CORBA);
- транзакционно-ориентированные – ориентированы на поддержку

транзакций в системах распределенных баз данных (мониторы транзакций, все современные промышленные СУБД, например MS SQL SERVER).

Одно интеграционное решение может использовать несколько видов промежуточных сред.

Взаимодействие информационных систем может быть описано в рамках модели «клиент-сервер». В данном контексте под клиентом будем понимать приложение, которое отправляет запрос на обработку. Приложение, отвечающее на запрос, будем называть сервером. В большинстве случаев одно и то же приложение в зависимости от ситуации может выступать в роли как клиента, так и сервера.

Взаимодействие между приложениями может быть синхронным или асинхронным. Синхронным называется такое взаимодействие, при котором приложение, выступающее в роли клиента, отослав запрос, блокируется и может продолжать работу только после получения ответа от приложения, выступающего в роли сервера. Иногда такой вид взаимодействия называют блокирующим.

Схема синхронного взаимодействия представлена на рис. 2.1.

В случае асинхронного взаимодействия приложение-клиент после отправки запроса приложению-серверу может продолжать работу, даже если ответ на запрос еще не пришел. Иногда такой вид взаимодействия называют неблокирующим. Схема асинхронного взаимодействия представлена на рис. 2.2.

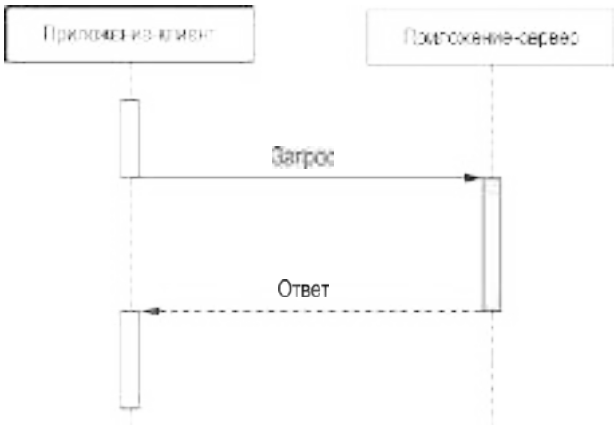


Рис. 2.1. Синхронное взаимодействие

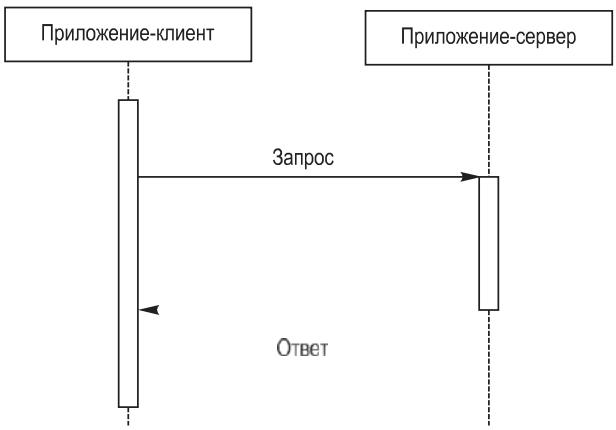


Рис. 2.2. Асинхронное взаимодействие

Технически синхронное взаимодействие проще реализовать, однако оно приводит к неоправданным затратам времени на ожидание ответа. Работа приложения, играющего роль клиента, приостанавливается, хотя оно могло бы выполнять другие действия, не зависящие от ожидаемого ответа.

Асинхронное взаимодействие позволяет достичь более высокой производительности систем, поскольку время между отправкой запроса и получением ответа на него может быть использовано

для выполнения других задач приложением-клиентом. Еще одним преимуществом асинхронного взаимодействия является меньшая зависимость приложения-клиента от приложения-сервера и возможность продолжать работу, даже если машина, на которой находится сервер, стала недоступной.

Сложности реализации асинхронного взаимодействия обусловлены следующими обстоятельствами:

- возможность использования нескольких потоков исполнения, увеличивающая производительность решения, одновременно затрудняет его отладку;
- приложение-клиент должно быть готово принять ответ в любой момент, даже в процессе выполнения другой задачи;
- поскольку асинхронные процессы могут выполняться в любом порядке, приложение-клиент должно уметь обрабатывать результат запроса с учетом времени получения.

2.2. Интеграция уровне вызова процедур

В данную группу входят стандарты, основанные на концепции удаленного вызова метода или процедуры. К ним относят стандарты COM (DCOM, COM+), RMI, CORBA.

В основу перечисленных стандартов положен механизм виртуализации. Идея заключается в создании виртуального компонента, являющегося прокси-объектом (локальным представителем) удаленного метода или процедуры. Приложение-клиент обращается к прокси-объекту, который и передает сообщение к удаленному объекту.

В качестве примера рассмотрим технологии удаленного вызова процедуры (remote procedure call, RPC) и удаленного вызова метода (remote method invocation, RMI).

Удаленный вызов процедуры впервые был реализован в начале 1980-х гг. компанией Sun Microsystems и явился прообразом объектно-ориентированного промежуточного слоя применительно к структурной парадигме программирования. Суть технологии RPC заключается в том, что вызов функции на удаленном компьютере осуществляется с помощью промежуточного программного обеспечения так же, как и на локальном.

Для того чтобы организовать удаленный вызов процедуры необходимо:

- описать интерфейс процедуры, которая будет использоваться для удаленного вызова при помощи языка определения интерфейсов (Interface Definition Language, IDL);
- скомпилировать определение процедуры (при этом будут созданы описание этой процедуры на том языке программирования, на котором будут разрабатываться клиент и сервер, и два дополнительных компонента – клиентский и серверный переходники). Клиентский переходник представляет собой процедуру-заглушку (stub), которая будет вызываться клиентским приложением. Клиентский переходник размещается на той же машине, где находится компонент-клиент, а серверный переходник – на той же машине, где находится компонент-сервер.

При обработке вызова процедуры выполняются следующие действия:

- клиентский переходник выполняет привязку к серверу (определяет физическое местонахождение (адрес) сервера);
- клиентский переходник выполняет маршалинг (упаковывает вызов процедуры и ее аргументы в сообщение в некотором стандартном для данной системы формате);
- клиентский переходник выполняет сериализацию сообщения (преобразует полученное сообщение в поток байтов) и отсылает с помощью какого-либо протокола (транспортного или более высокого уровня) на машину, на которой помещен серверный компонент;
- серверный переходник принимает сообщение, содержащее параметры вызова процедуры, распаковывает эти параметры при помощи десериализации и демаршалинга, вызывает локально соответствующую функцию серверного компонента, получает ее результат, упаковывает его и посылает по сети на клиентскую машину;
- после получения ответа от сервера выполняется процедура десериализации.

Сущность технологии удаленного вызова процедуры иллюстрирует рис. 2.3.

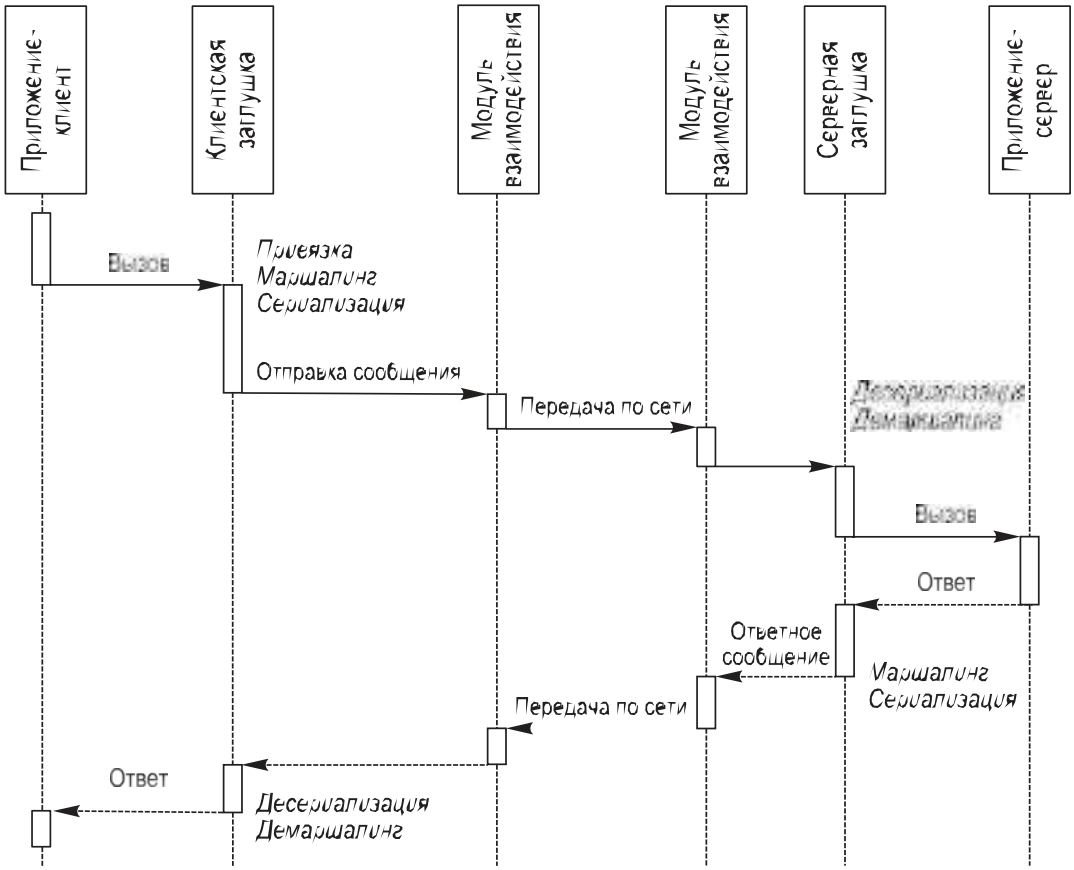


Рис. 2.3. Удаленный вызов процедуры

Удаленный вызов метода является модификацией метода удаленного вызова процедуры для объектной парадигмы программирования. В момент вызова удаленного метода на стороне клиента создается клиентская заглушка, называемая посредником (proxy). Клиентская заглушка в своем интерфейсе имеет все методы объекта-сервера, выполняет маршалинг и сериализацию параметров вызываемых методов и передает их по сети. Поскольку один объект сервера может предоставлять несколько методов, информация о том, какой именно метод вызывается, также упаковывается вместе с аргументами вызова. Промежуточная среда на стороне сервера десериализует параметры и передает их заглушке, называемой каркасом, или скелетоном (skeleton), на стороне сервера.

В табл. 2.2 приводится краткая характеристика стандартов объектно-ориентированного взаимодействия.

Таблица 2.2

Стандарты объектно-ориентированного взаимодействия

Стандарт	Назначение	Достоинства	Ограничения
Com	Средство взаимодействия приложений, функционирующих на одном компьютере	Высокий уровень абстракции Простота использования Очень высокая производительность	COM-приложения способны функционировать только под управлением Windows (двоичная структура объектов компонентной модели Microsoft) Распределенные решения плохо масштабируются (жесткая связь между клиентом и сервером) Невысокая устойчивость к сбоям COM-систем (жесткая привязка сервера к клиенту, двухуровневая архитектура)
Com+	Промежуточная среда для создания распределенных систем, действующих в локальной сети	Поддержка синхронного и асинхронного взаимодействий программных компонентов Динамическая балансировка нагрузки Поддержка логической целостности данных за счет работы с координатором	Использование ограничено взаимодействием компонентов внутри локальной или виртуальной частной сети, построенной на базе Microsoft Windows, в пределах одного предприятия

		распределенных транзакций Возможность ограничения доступа к компоненту в разрезе ролей, связанных с учетными записями пользователей	
DCOM	Промежуточная среда для создания распределенных систем, действующих в локальной сети и сети Интернет	Независимость от языка программирования Простота Поддержка распределенных транзакций	Использование ограничено платформами Windows
CORBA	Набор спецификаций для промежуточного программного обеспечения объектного типа. Создавалась консорциумом OMG как универсальная методология создания распределенных систем в гетерогенных средах	Кроссплатформенность Высокий уровень устойчивости к внутренним сбоям за счет большей изоляции клиентов и серверов (трехуровневая архитектура) Передача данных между компонентами происходит при помощи протокола UDP (обеспечивает экономию системных ресурсов) Набор сервисов (управление транзакциями, безопасностью, жизненным циклом объектов, событиями и т.д.)	Сложность технологии

RMI	Архитектура (Remote Method Invocation, то есть вызов удаленного метода), которая интегрирована с JDK1.1 и реализует распределенную модель вычислений	Самый простой и быстрый способ создания распределенных систем Распределенное автоматическое управление объектами	Поддержка одного языка программирования Java Собственный протокол взаимодействия Трудность интегрирования с существующими приложениями

2.3. Интеграция по данным

Расширяемый язык разметки XML (eXtensible Markup Language) приобрел известность в конце 1990-х гг., когда он начал широко использоваться для переноса данных между различными информационными системами и описания бизнес-транзакций.

XML – это язык разметки документов, предназначенный для хранения структурированных данных, обмена информацией между программами, а также для создания на его основе специализированных производных языков.

В начале 2000-х гг., когда на первый план вышли проблемы интеграции разнородных приложений, появилась концепция Webслужб и основанная на ней парадигма сервис-ориентированной архитектуры, начали развиваться технологии управления бизнеспроцессами (BPM). Крупнейшие поставщики программного обеспечения активно разрабатывали внутрикорпоративные стандарты описания, реализации и исполнения бизнес-процессов для своих программных систем. Появились многочисленные спецификации языков описания бизнес-процессов (JPDL, XLang, WSFL, WSCL, BPSS, WSCI), опубликованные конкурирующими вендорами. На сегодняшний день общепризнанными стандартами процессного управления являются основанные на XML языки: WSDL (Web Service Definition Language), WS-BPEL (Web Services Business Prosecc Execution Language) и WS-CDL (Web

Services Choreography Discription Language).

Целесообразность использования языка XML и основанных на нем технологий для решения интеграционных задач обусловлена следующими преимуществами:

1) XML является самоописывающим языком. XML-документ содержит элементы и атрибуты, использование которых подчинено строгим правилам, однако имена элементов и атрибутов имеют содержательный характер. Такой документ может быть легко прочитан прикладной программой (синтаксическим анализатором) и доступен для прочтения человеку. Имеется возможность задавать имена в соответствии с принятыми в конкретной прикладной области стандартами, а не на основе жесткого синтаксиса;

2) XML – самодокументируемый формат, предназначенный не только для описания данных, но и для определения метаданных, в том числе для описания дополнительных правил, ограничивающих данные в соответствии с информационной моделью предметной области. Например, XSD-схема позволяет задать следующие типы ограничений:

- базовые и производные типы данных (строки, даты, логический и др.);
- расширенные типы данных (произвольные пользовательские типы данных);
- фасеты (дополнительные ограничения, такие как длина, дробные числа и т.д.);
- границы значений (наибольшее и наименьшее значения);
- перечисление (набор допустимых значений для атрибута);
- условия кардинальности (вхождение повторяющегося элемента данных);
- шаблоны;

3) XML – универсальный язык, подходящий для описания практически любых типов документов. В формате XML могут быть описаны основные структуры данных, такие как записи, списки и деревья;

4) XML – расширяемый язык. Любую XML-структуру можно разрабатывать в расчете на оперативное сокращение или расширение. Расширение возможно как за счет включения в структуру новых элементов и/или атрибутов, так и путем использования механизмов пространств имен;

5) XML обеспечивает межплатформенное взаимодействие. Поскольку XML – это текстовый формат, требования для организации взаимодействия между отправляющей и принимающей платформами минимальны и сводятся к двум моментам:

- возможность читать и создавать текстовые файлы в кодировке Юникод (UTF-8 или ASCII);
- наличие синтаксического анализатора для обработки XML-документов;

6) безусловным преимуществом является наличие международных стандартов (поддерживаются консорциумом W3C – www.w3.org).

Язык имеет строго определенные синтаксис и требования к анализу, что позволяет ему оставаться простым, эффективным и непротиворечивым. Актуальными спецификациями являются XML 1.0, XML Information Set, Namespaces in XML, XML Schema. В настоящее время XML широко используется для хранения и обработки документов и поддерживается всеми ведущими производителями программного обеспечения;

7) XML ориентирован на повторное использование, что позволяет уменьшить стоимость разработки интеграционного решения, снизить эксплуатационные затраты, а также способствует продвижению корпоративных и отраслевых стандартов. Например, XML-схемы могут быть спроектированы как модульные внешние компоненты. Нестандартные элементы и типы данных целесообразно определять в отдельных XML-схемах, а затем многократно использовать посредством их включения в документы или с помощью ссылки на них.

XML описывает определенный класс объектов, называемых XMLдокументами. Документ представляется в виде дерева элементов,

каждый из которых может иметь набор атрибутов, а также содержать другие элементы или текст.

XML-документ описывается в терминах логической и физической структуры. Приведем краткие сведения об элементах логической и физической структуры XML-документов.

Логическая структура XML-документа

Логическая структура состоит из следующих элементов:

- объявление;
- определения типа документа;
- элементы;
- комментарии;
- ссылки;
- инструкции по обработке документа.

В табл. 2.3 представлены основные требования спецификации XML 1.0, предъявляемые к синтаксису XML-документов.

Приведем пример XML-документа.

```
<?xml version="1.0" encoding="windows-1251"?>
<!--Пример XML-документа -->
    <Контрагенты>
        <Контрагент Код="Ю023">
        <Наименование>Рога и копыта</Наименование>
        <ИНН>1232345678</ИНН>
        <КПП>775003657</КПП>
        <Адрес Индекс="118200" Город="Москва" Улица="Широкая"
        Дом="100">
            </Адрес>
        <КонтактноеЛицо ФИО="Иванов Иван Иванович" >
            <Телефон>
                <СлужебныйТелефон>74952113477</СлужебныйТелефон>
                <МобильныйТелефон>79056784523</МобильныйТелефон>
            </Телефон>
        </КонтактноеЛицо>
    </Контрагент>
</Контрагенты>
```

Таблица 2.3 Синтаксис XML

Элемент логической структуры	Описание	Пример
Объявление	Размещается в начале документа. Ограничивается тегами <?xml и?>. Включает атрибуты: 1) version – номер версии спецификации XML, обязательный атрибут; 2) encoding – кодировка символов документа (по умолчанию encoding="UTF-8"), необязательный атрибут. Если имена тегов задаются на русском языке, необходимо установить encoding="windows-1251"; 3) standalone – указание на наличие внешних описаний структуры документа, по умолчанию standalone="no", необязательный атрибут. Атрибуты должны следовать в указанном выше порядке. Если атрибуты не определены, то им	<?xml version="1.0" encoding="UTF-8"?>

	присваива- ется значение по умолчанию	
Определения типа документа	DTD (Document Type Declaration) заключается между символами <! DOCTYPE…> и может занимать несколько строк. В этой части объявляются теги, использованные в документе, или приводится ссылка на файл, в котором записаны такие объявления. Секция DTD должна располагаться перед корневым элементом	Пример см. в п. «Языки описания структуры»
Элементы	Элементы являются основными составляющими XML-документа; бóльшая часть данных в XMLдокументах содержится в элементах. Элемент представляется в XML-документе с помощью открывающего (<>) и закрывающего (</>) тэгов. Открывающий тэг записывается в формате <ИмяЭлемента>, а закрывающий тэг – в формате	<Книга> </Книга> <Книга isbn="978-59775-0778-3"> </Книга>

	</ИмяЭлемента>. Имя элемента не может содержать пробелов. Содержимым элемента могут быть символьные данные (текст), другие элементы (известные как дочерние элементы), а также оно может отсутствовать (пустой элемент). XML-документ должен содержать обязательный корневой элемент. Элемент может содержать любое число атрибутов, содержащих дополнительную информацию о данных, которые представляет элемент. Атрибуты указываются в виде пар «название-значение» в открывающем тэге элемента. Значения атрибутов заключаются в кавычки. Названия атрибутов уникальны в рамках одного элемента (в одном элементе не может быть двух атрибутов с	

	одинаковым именем)	
Комментарии	Ограничиваются тегами <! и !>. Используются для документирования. Могут располагаться в любом месте документа	<!-…текст комментария… -->
Ссылки	Ограничиваются символами «&» и «;». Используются для подстановки вместо них символов (ссылки на символы) или различных данных (ссылки на сущности), описанных в определении DTD. Ссылки на символы позволяют вставить в текст документа некоторый символ, который, например, отсутствует в раскладке клавиатуры либо может быть неправильно истолкован анализатором.	&# код_символа_в_ Unicode &#xШестнадцатерич- ный_код_символа имя_сущности
	Ссылки на сущности позволяют включать любые строковые константы в содержание элементов или значения атрибутов. Ссылки на сущности указывают программе-анализатору подставить вместо них строку символов, заранее заданную в определении типа документа.	

	Для включения в XML-документ символьных данных, которые не следует обрабатывать, используется секция <! [CDATA [содержание секции]] >	
Инструкции по обработке	Ограничиваются тегами <? и ?>. Предназначены для передачи информации приложению, работающему с XML-документом. За начальным вопросительным знаком записывается имя программного модуля, которому предназначена инструкция. Далее через пробел записывается инструкция, передаваемая программному модулю	<?xml version="1.0" encoding="windows1251"?> Эта инструкция предназначена программе, обрабатывающей документ XML. Инструкция передает ей номер версии и кодировку, в которой записан документ

Физическая структура XML-документа

Физическая структура XML-документа описывает его как набор сущностей. Документ должен содержать как минимум одну сущность – корневую сущность документа. Сущности могут включаться в XML-документ также с помощью XML-ссылок.

XML-ссылка – это ссылка на внешний объект, содержимое которого размещается в указанном месте документа. Ссылка на сущность работает как подстановка и обеспечивает модульность XML-документа, которая, как будет показано ниже, позволяет объединять данные из разных источников в единую структуру и легко собирать документы, а также их схемы из пригодных для

повторного использования блоков.

Различные приложения могут использовать сущности, имеющие одинаковые имена и содержащие различные данные. Для предотвращения конфликтов имен в XML используются пространства имен, которые представляют собой коллекции имен. В каждой коллекции имен все имена уникальны. Каждая коллекция должна иметь уникальный идентификатор (URI-адрес). Каждое XML-имя характеризуется идентификатором пространства имен и локальным именем в пределах своего пространства имен. Таким образом, появляется возможность определить элементы, имеющие одинаковые имена, но связанные с различными URI.

Рассмотрим правила использования пространств имен на конкретном примере. Пусть в одном документе необходимо объединить данные о клиенте компании, поступающие из разных источников. Из CRM-системы поступает информация о персональных данных клиента, из системы учета заказов – данные о заказе.

CRM

```
<Client>
    <Name>Mary</Name>
    <Phone>+7.602.555.9999</Phone>
    <Fax>+7.602.555.9999</Fax>
</Client>
```

Система учета заказов

```
<Client>
    <OrderNumber>2223</OderNumber>
    <OrderData>10.10.2012</OrderData>
</Client>
```

Пространство имен объявляется с помощью зарезервированного имени xmlns. Ниже приводится пример объявления пространств имен в XML-документе.

```
<Clients xmlns:ClientInfo="http://www.mycompany.com/ClientInformation"
     xmlns:ClientOrderData="http://www.mycompany.com/ClientOrderData">
```

ClientInfo и ClientOrderData являются префиксами пространств имен и представляют сокращенные наименования идентификаторов.

После объявления пространств имен их префиксы могут использоваться в документе для определения принадлежности каждого элемента к конкретному пространству имен.

Для рассмотренного примера XML-документ, содержащий данные из двух пространств имен, будет выглядеть следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<Clients xmlns:ClientInfo = "http://www.mycompany.com/
ClientInformation"
         xmlns:ClientOderData="http://www.mycompany.com/
         ClientOrderData" >
    <ClientInfo:Client>
         <ClientInfo:Name>Mary</ClientInfo: Name>
         <ClientInfo:Phone>+7.602.555.9999</ClientInfo:Phone>
         <ClientInfo:Fax>+7.602.555.9999</ClientInfo:Fax>
    </ClientInfo:Client>
    <ClientOrderData:Client>
         <ClientOderData:OrderNumber>2223</ClientOderData:OrderNumber>
         <ClientOderData:OrderData>10.10.2012</ClientOderData:OrderData>
    </ClientOderData:Client>
</Clients>
```

Имя элемента или атрибута с префиксом называется уточненным именем (qualified name или QName) и используется анализаторами XML для извлечения элементов, принадлежащих соответствующим пространствам имен в пределах глобального XML-пространства имен http://www.w3.org/XML/1998/namespace.

Если пространство имен объявлено без префикса, то оно является пространством имен по умолчанию для тех элементов XML-документа, которые не используют префикс. Каждое пространство имен имеет свою область действия в рамках XML-документа. Объявление пространства имен применяется к элементу, содержащему определение, а также ко всем его дочерним элементам, если оно не переопределяется другим пространством

имен в определении элемента. Имена атрибутов также можно уточнять, используя префикс объявленного пространства имен. Для атрибутов нельзя использовать пространства имен по умолчанию. Если для атрибута не указан префикс, то он не принадлежит ни к какому пространству имен. Атрибуты элементов для связывания с пространствами имен всегда необходимо уточнять префиксами.

Приведем пример использования пространства имен http://www.mycompany.com/ClientInformation как пространства имен по умолчанию:

```
<?xml version="1.0" encoding="utf-8"?>
<Clients xmlns="http://www.mycompany.com/ClientInformation"
         xmlns:ClientOrderData="http://www.mycompany.com/
         ClientOrderData">
    <Client>
        <Name>Mary</Name>
        <Phone>+7.602.555.9999</Phone>
        <Fax>+7.602.555.9999</Fax>
    </Client>
    <ClientOrderData:Client>
        <OrderNumber>2223</OrderNumber>
        <OrderData>10.10.2012</OrderData>
    </ClientOrderData:Client>
</Clients>
```

Каждый XML-документ несет информацию о данных и их структуре (описание метаданных).

XML-документы могут быть двух типов:

1) документы, созданные с учетом логических и структурных правил;

2) документы, не использующие никаких правил, кроме синтаксических правил оформления XML-документов.

Проверку документов первого типа на соответствие заданным правилам осуществляет XML-процессор. Проверка документов второго типа выполняется разработчиком.

При создании документа первого типа описание его структуры может быть выполнено с использованием таких языков, как Document Type Definitions (DTD), XML Schema, RELAX NG, XML Data-Reduced и др.. Наибольшее распространение получили языки DTD и XML Schema.

Далее анализируются сильные и слабые стороны наиболее распространенных языков описания структуры и приводится краткое

изложение их основ. Поскольку данное учебное пособие посвящено проблемам интеграции информационных систем, при рассмотрении языков описания структуры основное внимание будет уделено вопросам модульности и повторного использования схем.

Язык Document Type Definitions (DTD)

Язык Document Type Definitions (DTD) не базируется на XML. Этот язык имеет ряд ограничений в описании метаданных: не является расширяемым, не поддерживает строгое типизирование данных, ограниченно поддерживает пространства имен. Описание структуры на языке DTD постепенно вытесняется технологией XML Schema, однако до настоящего времени продолжает использоваться (иногда совместно с XML Schema).

Приведем краткое изложение правил использования основных конструкций DTD.

Описание структуры на языке DTD может быть включено в XMLдокумент (внутреннее подмножество) или размещено в отдельном файле, также возможен вариант смешанного описания. Во всех случаях для определения DTD необходимо использовать объявление

<! DOCTYPE…>. В табл. 2.4 приведены правила включения DTD-описания в XML-документ.

Таблица 2.4 Правила включения DTD-описания в XML-документ

Вариант описания структуры	Правила записи
Внутренне описание DTD	<! DOCTYPE ROOT [<!– ОБЪЯВЛЕНИЯ DTD –>] > где ROOT – имя корневого элемента; <!– ОБЪЯВЛЕНИЯ DTD –> – описание структуры на языке DTD

Внешнее описание DTD	<! DOCTYPE ROOT SYSTEM "SYSTEM_ID"> где ROOT – имя корневого элемента; SYSTEM_ID – место расположения файла внешнего DTD. Например, <! DOCTYPE Clients SYSTEM "Clients.DTD"> <! DOCTYPE ROOT PUBLIC "PUBLIC_ID" "SYSTEM_ID"> где SYSTEM_ID и PUBLIC_ID – место размещения файла внешнего DTD; PUBLIC_ID не зависит от места размещения XML-файла (например, место в локальной сети); "SYSTEM_ID" будет использовано только в том случае, если нет доступа к "PUBLIC_ID". Например, <! DOCTYPE Clients PUBLIC"-//W3C//DTD Strict/RU" "Client.dtd">

Язык DTD позволяет описать требования к элементам и атрибутам документа. При описании элементов указываются:

- модель содержимого, описывающая также наличие дочерних элементов;
- ограничения на количество повторений элемента в документе.

Для описания элемента используется конструкция следующего вида:

```
! ELEMENT Имя_элемента Модель_содержимого>.
```

Правила записи модели содержимого представлены в табл. 2.5.

Таблица 2.5 Правила записи модели содержимого

Модель содержимого	Описание	Пример
ANY	Элемент может содержать любые дочерние элементы или текст	<! ELEMENT Client ANY>
EMPTY	Элемент не может содержать дочерние элементы или текст, может иметь атрибуты	<! ELEMENT OrderID EMPTY>
(#PCDATA)	Элемент может содержать только текст	<! ELEMENT Phone (#PCDATA) >
(NAME1, NAME2)	Элемент содержит указанные дочерние элементы в указанном порядке, не может содержать текст	<! ELEMENT Client (Name, Phone) >
(NAME1|NAME2)	Элемент содержит один из указанных взаимоисключающих элементов, не может содержать текст	<! ELEMENT Client (Fax |Phone) >
Смешанная модель	Элемент может содержать текст и дочерние элементы	<! ELEMENT Client (#PCDATA, Name, Phone) >

Ограничения на количество дочерних элементов задается следующим образом (табл. 2.6).

Таблица 2.6 Ограничения на количество дочерних элементов

Оператор количества элементов	Описание	Пример
Нет	Допустимо использовать один экземпляр элемента	<! ELEMENT Client (INN) >
*	Элемент может повторяться ноль и более раз	<! ELEMENT Clients (Client*) >
+	Элемент может повторяться один и более	<! ELEMENT Client (Phone+) >

	раз	
?	Элемент может повторяться ноль или один раз	<! ELEMENT Client (Address?) >

Атрибуты элементов объявляются для каждого элемента, если это необходимо, с помощью объявления ATTLIST.

При описании атрибутов указываются:

- тип атрибута;
- ограничения на употребление атрибута.

Для описания атрибутов элемента используется конструкция следующего вида:

```
<! ATTLIST Имя_элемента Имя_атрибута1 Тип_атрибута
          (Значение_по_умолчанию | Ключевое_слово)
          Имя_атрибута2 Тип_атрибута (Значение_по_умолчанию |
          Ключевое_слово) >
```

Правила описания допустимых типов атрибутов и ключевых слов приведены в табл. 2.7.

Таблица 2.7

Правила описания допустимых типов атрибутов и ключевых слов

Тип атрибута	Описание
CDATA	Строка символов
ID	Уникальное в рамках документа значение (аналог первичного ключа в базе данных), элемент не может иметь больше одного атрибута типа ID
Тип атрибута	Описание
IDREF	Ссылка на элемент, обладающий атрибутом ID с тем же самым значением, что и значение заданного атрибута IDREF. Используется для создания связей и перекрестных ссылок в документе. Аналог отношения «один-к-одному» в реляционной базе данных
IDREFS	Последовательность ссылок IDREF, разделенных пробелами. Позволяет смоделировать отношение «один-ко-многим»

ENTITY	Определяет имя внутренней или внешней сущности, предназначенной для повторного использования. В том числе используется для определения имени примитива, игнорируемого анализатором. Позволяет ссылаться на данные, структура которых нарушает разметку по правилам XML (в частности, использовать в XML-документах ссылки на двоичные файлы)
ENTITIES	Перечень значений ENTITY, разделенных пробелами
NMTOKEN	Имя, содержащее только символы, применяемые в именах (строка, состоящая из букв, цифр и символов «.», «-», «_», «:»). Может содержать имена других элементов или атрибутов
NMTOKENS	Перечень значений NMTOKEN, разделенных пробелами

Ограничения на использование атрибутов задаются с помощью следующих ключевых слов (табл. 2.8).

Таблица 2.8

Ограничения на использование атрибутов

Ключевое слово	Описание	Пример
#REQUIRED	Обязательный атрибут	ClientID ID #REQUIRED
#IMPLIED	Необязательный атрибут	Address Region #IMPLIED
«Значение по умолчанию»	Если атрибут отсутствует, то анализатор принимает в качестве его значения значение, заданное по умолчанию. Если атрибут имеется, то он может принимать любое значение	
#FIXED «Значение»	Атрибут является необязательным, но если значение указано, то оно задается по умолчанию	Клиент Тип #FIXED «ФЛ»

В качестве примера рассмотрим XML-документ и соответствующее ему DTD-описание.

```
<Clients>
    <Client ClientID="0001" ClientType="ФЛ" Class="VIP"
    System="CRM">
        <Name>Mary</Name>
        <Phone>
            <Home>+7.677.444.1111</Home>
            <Mobile>+7.555.444.3333</Mobile>
            <Mobile>+7.666.333.2222</Mobile>
        </Phone>
        <mail>Mary@qq.com</mail>
        <Country>Russia</Country>
    </Client>
```

```
<! ELEMENT Clients (Client*) >
<! ELEMENT Client (Name,Phone,Mail,Country) >
<! ELEMENT Name (#PCDATA) >
<! ELEMENT Phone (Home?,Mobile+) >
<! ELEMENT Mail (#PCDATA) >
<! ELEMENT Country (#PCDATA) >
<! ELEMENT Home (#PCDATA) >
<! ELEMENT Mobile (#PCDATA) >
<! ATTLIST Client
    ClientID ID #REQUIRED
    ClientType CDATA #REQUIRED
    Class CDATA #IMPLIED
    System NMTOKEN #FIXED "CRM">
```

Для описания логического компонента, многократно используемого разными XML-документами, применяются примитивы, которые задаются путем указания типа атрибута ENTITY. Примитивы могут быть внутренними (логический компонент повторно используется в одном документе) и внешними (повторно используется логический элемент из внешнего файла).

В зависимости от содержимого примитивы можно разделить на анализируемые примитивы (ссылаются на правильно сформированное содержимое XML) и примитивы, игнорируемые анализатором (ссылаются на не-XML-данные). Поскольку XML-анализатор не способен обрабатывать данные в двоичных и других не-XML-форматах, каждому примитиву, игнорируемому анализатором, должна соответствовать нотация. Нотации применяются для того, чтобы связать формат внешних данных, используемых в XML-документе, с внешней программой-обработчиком. Анализатор отошлет не-XMLданные на обработку указанной программе.

XML-документ, использующий нотации, играет роль унифицирующего

документа, объединяющего разнородные данные.

Форматы описания логических компонентов XML приведены в табл. 2.9.

Таблица 2.9

Форматы описания логических компонентов XML

Тип логического компонента	Формат описания (пример)
Внутренний примитив	<! ENTITY Имя_примитива "Значение"> Пример Строка DTD <! ENTITY Система "1С:Предприятие"> В DTD используется примитив Система, имеющий значение 1С:Предприятие. Строка XML <Клиент ИсточникДанных="&Система;"> В XML-документе атрибуту Источник данных тега Клиент будет присвоено значение 1С:Предприятие
Внешний примитив, обрабатываемый анализатором	<! ENTITY Имя_примитива SYSTEM "SYS_ID"> или <! ENTITY Имя_примитива PUBLIC "PUBL_ID" "SYS_ID">, где "SYS_ID" – ссылка на реально существующий внешний файл (URL); "PUBL_ID" – идентификатор ресурса (URI), не обязательно реально существующий.
	Пример Строка DTD <! ENTITY Описание SYSTEM "C:/Discription.xml">
	В DTD используется внешний примитив Описание, который определяет внешний файл C:/Discription.xml, содержащий описание клиента.
	Строка XML <Клиент>&Описание;</Клиент>
	В тег Клиент будет вставлено содержимое файла C:/Discription.xml
Тип логического компонента	Формат описания (пример)

Внешний примитив, игнорируемый анализатором	<! ENTITY Имя_примитива SYSTEM "SYSTEM_ID" NDATA Имя нотации>, где "SYSTEM_ID" – ссылка на реально существующий внешний файл (URL); Имя_нотации – имя нотации, содержащей информацию о программе-обработчике не-XML-данных.
	Для описания нотации используется следующий формат: <! NOTATION Тип_данных SYSTEM "SYSTEM_ID">, где Тип_данных – имя формата данных; SYSTEM_ID – внешняя программа-обработчик.
	Пример Фрагмент DTD <! ELEMENT Book EMPTY> <! ATTLIST Book image ENTITY #REQUIRED> <! NOTATION gif SYSTEM "gif-viewer.exe"> <! NOTATION jpg SYSTEM "jpg-viewer.exe"> <! ENTITY news SYSTEM "images/news.gif" NDATA gif> <! ENTITY products SYSTEM "images/products.jpg" NDATA jpg> <! ENTITY support SYSTEM "images/support.gif" NDATA gif>
	Фрагмент XML <Book image="news" /> <Book image="products" /> < Book image="support" />
	В XML-документ будут вставлены соответствующие графические файлы

Язык XML Schema Definition (XSD)

Язык XML Schema Definition (XSD) основан на XML и обладает более широкими возможностями описания структуры документа, чем DTD. Он поддерживает типизацию данных, пространства имен, регулярные выражения.

XML Schema содержит описание элементов и атрибутов XMLдокумента, правила наследования элементов, включая порядок и количество потомков, тип содержимого элементов, типы данных элементов и атрибутов, значения элементов и атрибутов и дополнительные ограничения на значения. Кроме того, использование XML Schema обеспечивает

трансформацию XML-документа в иерархию объектов определенных типов, доступ к которым может быть осуществлен программным способом с помощью интерфейса (функциональность PSV1).

Основным преимуществом языка XML Schema является поддержка строго типизированных данных. При обмене данными между различными приложениями и базами данных задача согласования типов данных всегда остается актуальной, поскольку в разных системах определения типов данных могут отличаться. К таким отличиям относятся: максимальное и минимальное возможные значения, наибольшая длина, поддержка дробных чисел, внутренняя кодировка и внешний формат (например, для даты и времени). Таким образом, несмотря на возможное совпадение названий типов данных, их реализация в различных продуктах может отличаться. Применение типов данных в схемах позволяет проводить необходимую верификацию данных документа при обмене или совместном использовании данных несколькими системами.

Данные пособие не является подробным руководством по языку XML Schema, поэтому здесь мы ограничимся только базовыми сведениями о языке XSD, которые необходимы для понимания последующего материала.

XML Schema всегда создается в отдельном файле, имеющем расширение xsd. Файл XML связывается с соответствующей схемой с помощью атрибута schemaLocation пространства имен схемы. Для того чтобы использовать атрибут schemaLocation, необходимо определить пространство имен схемы. Все эти определения указываются в корневом элементе документа XML.

```
<КорневойЭлемент
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="Схема.xsd">
```

Рассмотрим основные элементы структуры XML Schema.

Корневым элементом всегда является элемент <schema>. Описание атрибутов элемента <schema> приводится в табл. 2.10.

Таблица 2.10 Описание атрибутов элемента <schema>

Атрибут	Обязательный	Описание
elementFormDefault	Нет	Принимает значения «qualified» и «unqualified» (по умолчанию). Значение «qualified» указывет на то, что элементы документа должны уточняться префиксом пространства имен
attributeFormDefault	Нет	Принимает значения «qualified» и «unqualified» (по умолчанию). Значение «qualified» указывет на то, что атрибуты элементов документа должны уточняться префиксом пространства имен
xmlns: xs	Да	Всегда принимает значение xmlns: xs="http://www.w3.org/2001/XMLSchema" Указывает на пространство имен языка XMLSchema для элементов и типов данных схемы. При наличии префикса (в данном примере – xs) все элементы и типы данных схемы должны уточняться этим префиксом (xs: schema). Если префикс отсутствует, то атрибут xmlns указывает для схемы пространство имен по умолчанию. Элемент <schema> может содержать несколько атрибутов xmlns с разными префиксами, указывающими на используемые в документе пространства имен
targetNamespace	Нет	Пространство имен для элементов XML-документа, определенных данной схемой
version	Нет	Версия схемы
xml: lang	Нет	Язык для всех комментариев схемы

Корневой элемент <schema> может содержать следующие дочерние элементы:

- <element> – используется для определения элементов XML-документа;

- <attribute> – используется для определения атрибутов XML-документа;
- <group> – необходим для определения группы элементов, предназначенной для повторного использования в рамках схемы по ссылке на имя группы;

- <attributeGroup> – используется для определения атрибутов группы элементов;
- <annotation> – позволяет включать в XML-документ документацию;
- <import> – позволяет использовать компоненты указанной внешней схемы в основной схеме (обеспечивает модульность схем);
- <include> – добавляет все компоненты указанной внешней схемы в основную схему (обеспечивает модульность схем);
- < notation> – содержит определение нотации, описывающей формат не-XML-данных в XML-документе;
- <redefine> – переопределяет компоненты внешней схемы, имеющей то же пространство имен, что и основная схема;
- <simpleType> – объявляет простой тип содержимого элемента. Элементы с простым типом данных могут содержать только символьные данные и не могут включать атрибуты и дочерние элементы;
- <complexType> – объявляет сложный тип содержимого элемента, который может включать атрибуты и другие элементы.

XML Schema поддерживает три основные категории типов данных:

1) предопределенные примитивные типы – фундаментальные типы данных, на которые можно ссылаться и применять их к элементам и атрибутам. Примерами примитивных типов данных являются String, Float, Double, Time, Date, Decimal, AnyURI;

2) предопределенные производные типы – встроенные типы, полученные на основании примитивных типов. Примерами производных типов данных являются Integer, Long, Byte, Short, nonPositiveInteger,

nonNegativeInteger, ID и др.;

3) нестандартные типы – определяемые пользователем типы данных, которые создаются на основании примитивных или производных типов путем введения дополнительных ограничений. Поддержка нестандартных типов данных исключительно полезна для верификации данных с учетом бизнес-логики.

Для описания элементов и атрибутов, имеющих предопределенные (примитивные и производные) типы данных1, в XML Schema используются следующие синтаксические конструкции:

```
<xs:element name="ИмяЭлемента" type="ТипДанных" />
<xs:attribute name="ИмяАтрибута" type="ТипДанных"/>
```

Дополнительно для элементов и атрибутов можно указать атрибуты fixed или default для задания фиксированных значений элементов/атрибутов или значений по умолчанию.

```
<xs:element name="Пример" type="xs:string" default="Пример описания
элемента"/>
```

Если необходимо описать нестандартный тип данных для элемента или атрибута, то это следует делать с помощью тега <sympleType>, описав в нем новый тип данных.

```
<xs:element name="ИмяЭлемента">
    <xs:simpleType>
        Описание нестандартного типа данных
    </xs:simpleType>
</xs:element>
```

Новые нестандартные простые типы данных получают путем:

- сужения (restriction) встроенного или ранее определенного простого типа с помощью задания дополнительных ограничений;
- объединения (union) простых типов;
- использования списка (list) простых типов.

Пример использования нового простого типа данных, полученного путем сужения предопределенного типа (на базовый тип String накладываются

```
<xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:maxLength value="20"/>
                <xs:minLength value="10"/>
            </xs:restriction>
</xs:simpleType>
```

ограничения на максимально и минимально допустимую длину строки):

Пример использования нового простого типа данных, полученного путем объединения базовых типов (элемент или атрибут могут принимать неотрицательные или неположительные целые значения):

```
<xs:simpleType>
<xs:union memberTypes="xs:nonNegativeInteger
xs:nonPositiveInteger"/>
</xs:simpleType>
```

Пример использования списка простых типов (атрибут shoeSizes объявляется в качестве списка, содержащего десятичные значения 10.5, 9, 8 и 11):

```
<xs:simpleType name="Sizes">
        <xs:restriction base="xs:decimal">
            <xs:enumeration value="10.5"/>
            <xs:enumeration value="9"/>
            <xs:enumeration value="8"/>
            <xs:enumeration value="11"/>
        </xs:restriction>
</xs:simpleType>

    <xs:attribute name="shoeSizes">
        <xs:simpleType>
            <xs:list itemType="Sizes"/>
        </xs:simpleType>
    </xs:attribute>
```

Язык XML Schema использует различные типы ограничений на данные (см. табл. 2.8):

- ограничения длины (количество символов);
- границы значений (наибольшее и наименьшее значения как диапазон или порог);

▪ ограничения количества цифр десятичного числа (общее количество цифр или количество цифр после запятой);

▪ список допустимых значений;

▪ шаблоны;

▪ обработка символов пробела.

Примеры использования различных ограничений приведены в табл. 2.11.

Таблица 2.11

Примеры использования ограничений

Тип ограничения	Теги, задающие ограничения
Ограничения длины	<length> – фиксированное количество символов: <xs:simpleType> <xs:restriction base="xs:string"> <xs:length value="4"/> </xs:restriction> </xs:simpleType>
Тип ограничения	Теги, задающие ограничения
	<maxLength> – наибольшая длина значений определяемого типа: <xs:simpleType> <xs:restriction base="xs:string"> <xs:mfxLength value="20"/> </xs:restriction> </xs:simpleType> <minLength> – наименьшая длина значений определяемого типа: <xs:simpleType> <xs:restriction base="xs:string"> <xs:minLength value="2"/> </xs:restriction> </xs:simpleType>

Границы значений	<maxInclusive> – максимально допустимое значение, <minInclusive> – минимально допустимое значение (в примере допустимы целые значения в диапазоне от 5 до 10, включая 5 и 10): <xs:simpleType> <xs:restriction base="xs:integer"> <xs:maxInclusive value="10"/> <xs:minInclusive value="5"/> </xs:restriction> </xs:simpleType> <maxExclusive> – максимально допустимое значение, <minExclusive> – минимально допустимое значение (в примере допустимы целые значения в диапазоне от 5 до 10, исключая 5 и 10): <xs:simpleType> <xs:restriction base="xs:integer"> <xs:maxExclusive value="10"/> <xs:minExclusive value="5"/> </xs:restriction> </xs:simpleType>
Ограничения количества и типа цифр	<totalDigits> – общее количество цифр в определяемом числовом типе – сужении типа decimal, <fractionDigits> – количество цифр в дробной части числа: <xs:simpleType> <xs:restriction base="xs:decimal"> <xs:totalDigits value="8"/> <xs:fractionDigits value="3"/> </xs:restriction> </xs:simpleType>
Тип ограничения	Теги, задающие ограничения
Список допустимых значений	<enumeration> – элемент списка допустимых значений: <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="один"/> <xs:enumeration value="два"/> <xs:enumeration value="три"/> </xs:restriction> </xs:simpleType>

Шаблоны	<pattern> – регулярное выражение, используемое для ограничения внешнего вида или формы значений данных: <xs:simpleType> <xs:restriction base="xs:string" <xs:pattern value="[Ю|Ф][0–9]{3}"/> </xs:restriction> </xs:simpleType
Обработка символов пробела	<whitespace> – применяется при сужении типа string и определяет способ преобразования пробельных символов (пробел, табуляция, перевод строки). Используется в трех вариантах: 1) все пробельные символы сохраняются: <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="preserve"/> </xs:restriction> </xs:simpleType> 2) XML-процессор заменяет все пробельные символы на пробелы: <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="replace"/> </xs:restriction> </xs:simpleType> 3) XML-процессор заменяет пробельные символы пробелами, затем убирает начальные и конечные пробелы, а из нескольких подряд идущих пробелов оставляет только один: <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="collapse"/> </xs:restriction> </xs:simpleType>

Элементы, имеющие простой тип или предопределенные стандартные типы, могут содержать только данные (не могут содержать атрибутов и дочерних элементов).

Любой простой тип данных может содержать произвольный набор ограничений, который определяется бизнес-логикой приложения, работающего с данными.

Если простому типу данных присвоено имя, то ссылка на новый нестандартный тип данных может быть использована многократно в пределах

```
<xs:simpleType name="Код">
     <xs:restriction base="xs:string">
          <xs:length fixed="true" value="4"/>
          <xs:pattern value="[Ю|Ф][0-9]{3}"/>
     </xs:restriction>
</xs:simpleType>
<xs:element name="Код1" type="Код"/>
<xs:element name="Код2" type="Код"/>
```

данной схемы (аналогично ссылке на предопределенные типы данных).

В данном примере определен нестандартный тип данных с именем «Код», базирующийся на типе «string»: он использован как тип данных для элементов «Код1» и «Код2».

Для описания элементов XML-документа, содержащих дочерние элементы и атрибуты, в схеме используется сложный тип данных, который задается с помощью тега <complexType>.

```
<xs:element name="ИмяЭлемента">
     <xs:complexType>
          Описание сложного типа данных
     </xs:complexType >
</xs:element>
```

При описании сложного типа указываются порядок вхождения дочерних элементов (с помощью специальных тегов – индикаторов порядка, см. табл. 2.11), а также степень кардинальности повторяющихся элементов (с использованием атрибутов minOccurs и maxOccurs). Атрибут minOccurs определяет минимальную степень кардинальности, то есть наименьшее возможное количество повторений дочернего элемента. Значение minOccurs, равное нулю, указывает на необязательность (опциональность) элемента.

Атрибут maxOccurs определяет максимальную степень кардинальности, или наибольшее количество повторений элемента. Максимальная и минимальная степени кардинальности задаются определенными значениями. Для maxOccurs может быть указано значение unbounded (элемент встречается

любое количество раз).

```
<xs:element name="Книга">
   <xs:complexType>
        <xs:sequence>
             <xs:element name="Название" type="xs:string"/>
             <xs:element name="Автор" type="xs:string" maxOccurs="4"/>
             <xs:element name="Код" type="xs:string"/>
             <xs:element name="Цена" type="xs:string"/>
         </xs:sequence>
    </xs:complexType>
</xs:element>
```

В данном примере описан сложный тип данных для элемента «Книга», содержащего дочерние элементы «Название», «Автор», «Код», «Цена». Тег <sequence> является индикатором порядка вхождения дочерних элементов (табл. 2.12), а атрибут maxOccurs показывает максимально допустимое количество повторений элемента «Автор».

```
<xs:complexType name="Цена">
    <xs:choice>
        <xs:element name="Рубли" type="xs:double"/>
        <xs:element name="Доллары" type="xs:double"/>
    </xs:choice>
</xs:complexType >
```

Таблица 2.12 Теги индикатора порядка

Тег индикатора порядка	Описание
sequence	Дочерние элементы должны встречаться в указанном порядке. Степень кардинальности определяет, может ли дочерний элемент повторяться
all	Все дочерние элементы должны присутствовать в XML-документе. Дочерние элементы могут появляться в любом порядке, но должны встречаться только один раз. Нельзя задать значение степени кардинальности maxOccurs и minOccurs, отличное от единицы
choice	Из всех указанных дочерних элементов должен встречаться только один

Индикатор порядка choice указывает, что элемент этого типа «Цена» может содержать либо элемент «Рубли», либо элемент «Доллары», но не оба.

Язык RELAX NG

Язык RELAX NG также использует XML-синтаксис и является упрощенным вариантом XML Schema, сочетает в себе простоту DTD и возможности XML Schema. RELAX NG, как и XML Schema, поддерживает типизацию данных, регулярные выражения, пространства имен и возможность ссылаться на сложные определения.

Приложения, работающие с XML-документами, получают доступ к их содержимому и структуре путем использования специального программного компонента – XML-процессора (XML-анализатора). Следует отметить, что приложения работают не непосредственно с XML-документом, а с его информационным пространством XML Infoset, получаемым в результате разбора XML-документа XML-процессом. Таким образом, XML-процессор предназначен для анализа XML-документа, извлечения данных и передачи их на обработку в прикладную программу. XML-процессоры поддерживают механизм XML Namespace (спецификация W3C XML Namespaces 1.0) и обеспечивают проверку структурной и синтаксической корректности XML-документов.

XML-процессоры

Обрабатывая XML-документ, XML-процессоры представляют его структуру в виде упорядоченной модели данных, доступ к которой осуществляется с помощью стандартных интерфейсов прикладного программирования. Существуют два основных типа XML-процессоров – объектные (DOM) и потоковые (SAX).

Объектный анализатор строит в собственном пространстве памяти иерархическую модель разбираемого документа (Document Object Model, DOM), доступ к элементам которой прикладная программа получает с помощью DOM-интерфейсов. Основным преимуществом объектного анализатора является то, что он сразу предоставляет прикладной программе всю структуру XML-документа, позволяя ее анализировать в произвольном порядке.

Потоковый процессор является событийно управляемым и анализирует документ последовательно в режиме реального времени, что позволяет существенно экономить ресурсы памяти. Для доступа к элементам структуры документа прикладная программа в этом случае использует SAX (Simple API for XML). Следует отметить, что использование потоковых анализаторов утяжеляет логику прикладных программ и усложняет навигацию по документу.

Обобщенная схема работы XML-процессора представлена на рис. 2.4.

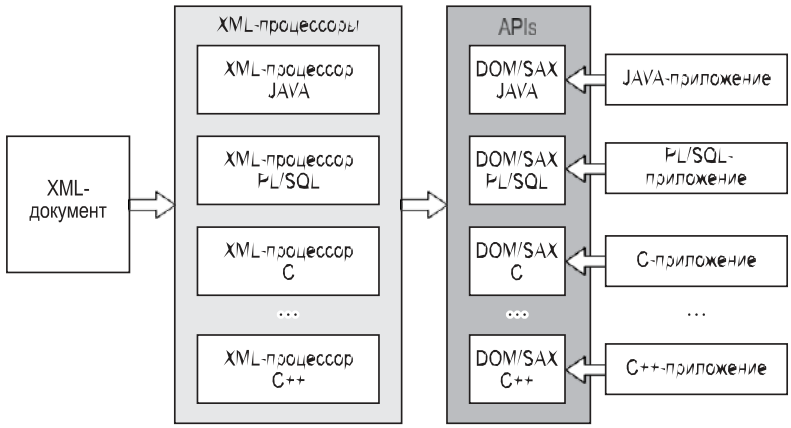


Рис. 2.4. Обобщенная схема работы XML-процессора

Программная обработка XML-документов выполняется также с использованием XSL Transformation (XSLT) процессоров, предназначенных для преобразования XML-документов в документы другой структуры и формата (XML, HTML, текстовый формат и т.д.). XSLT-процессоры соответствуют спецификациям W3C XSL Transform Proposed Recommendation 1.0 и Xpath Proposed Recommendation 1.0, поддерживают различные языковые и программные платформы.

Модульность XML-документов, позволяющая объединять данные из разных источников и собирать документы и их схемы из имеющихся блоков, широко используется в интеграционных проектах.

Так, при построении модели данных SOA-проектов, как правило, создается несколько XML-схем, основанных на концепции повторного

использования. Общая схема (мастер-схема) собирается из модульных подсхем, которые можно повторно использовать в разных контекстах и различных приложениях.

Например, схему Person.xsd можно использовать в контекстах имени сотрудника, имени покупателя или имени представителя поставщика. В результате достигается снижение затрат на проектирование и разработку.

Пусть схема Person.xsd имеет вид:

```
<?xml version="1.0"?>
<xsd:schema xmlns: xsd=http://www.w3.org/2001/XMLSchema
targetNamespace="http://www.company.org"
xmlns=http://www.person.org elementFormDefault="qualified">
     <xsd:complexType name="PersonType">
          <xsd:sequence>
               <xsd:element name="Name" type="xsd:string"/>
               <xsd:element name="SSN" type="xsd:string"/>
          </xsd:sequence>
     </xsd:complexType>
</xsd:schema>
```

Мастер-схема Company.xsd использует механизм include для ссылки на модульную подсхему:

```
<?xml version="1.0"?>
<xsd:schema xmlns: xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.company.org" xmlns="http://www.company.org"
elementFormDefault="qualified">
<xsd:include schemaLocation="Person.xsd"/>
     <xsd:element name="Company">
          <xsd:complexType>
               <xsd:sequence>
     <xsd:element name="Person" type="PersonType" maxOccurs="unbounded"/>
               </xsd:sequence>
          </xsd:complexType>
     </xsd:element>
</xsd:schema>
```

Синтаксис схемы позволяет определить в подсхеме элементы, группы, сложные и простые типы данных, которые можно подключить и на которые можно сослаться из другой схемы. В виде повторно используемой подсхемы обычно определяют стандартные для предприятия типы данных, международные и отраслевые коды, допустимые значения, устойчивые структуры данных (типы документов, блоки адресов, структуры продуктов и др.).

Уровень вложенности ссылок на подсхемы не ограничен. Ссылка на подсхему может иметь одну из следующих форм:

▪ включение (include) – в этом случае включаемая подсхема становится частью пространства имен основной схемы. Данный вариант используется, если основная схема имеет единый контекст;

▪ импорт (import) – используется, когда основная схема относится к нескольким контекстам, и ссылка на подсхему должна сохранить уникальность пространства имен подсхемы;

▪ переопределение (redefine) – включает в мастер-схему подсхему и

```
<?xml version="1.0"?>
<xsd:schema xmlns: xsd=http://www.w3.org/2001/XMLSchema
targetNamespace=http://www.company.org xmlns=http://www.company.org
elementFormDefault="unqualified" xmlns:per=http://www.person.org
xmlns: pro="http://www.product.org">
    <xsd:import namespace=http://www.person.org schemaLocation=
    "Person.xsd"/>
    <xsd:import namespace=http://www.product.org
    schemaLocation="Product.xsd"/>
    <xsd:element name="Company">
        <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="Person" type="per:PersonType"
            maxOccurs="unbounded"/>
            <xsd:element name="Product" type="pro:ProductType"
            maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

позволяет переопределить некоторые конструкции.

Модульные схемы широко используются в электронной коммерции при обмене данными между информационными системами предприятий-партнеров, а также в системах автоматизации государственного управления для решения задач централизованного сбора данных и межведомственного обмена информацией. Наглядным примером могут служить унифицированные форматы электронных банковских сообщений для безналичных расчетов, используемые для обмена с кредитными организациями и другими клиентами Банка России.

XSLT (Extensible Stylesheet Language Transformations) – это расширяемый язык стилевых преобразований, предназначенный для описания способа преобразования документов XML в документы других форматов. С помощью XSLT можно трансформировать XML-документ в любой формат (HTML, TXT, RTF, PDF и т.п.), а также в XML-документ другой структуры.

Последняя возможность широко используется при передаче информации между разными информационными системами, оперирующими с одной и той же информацией в различном представлении.

Язык XSLT основан на XML. Все элементы языка XSLT относятся к пространству имен http://www.w3.org/1999/XSL/Transform (обычно они имеют префикс xsl). На языке XSLT записывается инструкция по обработке XML-документа, предназначенная для XSLT-процессора. На сегодняшний день известны две спецификации языка – XSLT 1.0 и XSLT 2.0.

Инструкции по обработке, как правило, сохраняют в отдельном файле с расширением xslt, в этом случае ссылку на таблицу стилей помещают в документ XML как одну из инструкций по обработке.

Приведем пример ссылки:

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml-stylesheet type="text/xsl" href="simple.xslt"?>
<Корневой элемент>
     <!-содержание -->
</КорневойЭлемент>
```

XSLT-файл содержит описание правил преобразования исходного дерева элементов в конечное дерево. Преобразование строится путем сопоставления образцов и шаблонов. Образец сравнивается с элементами исходного дерева, а шаблон используется для создания частей конечного дерева. Структура конечного дерева может полностью отличаться от структуры исходного дерева.

XSLT-процессор выполняет преобразование, заданное в указанном XSLT-файле. XSLT совместно с входным XML-документом интерпретируется процессором, и на выходе получается документ другого формата.

Обычно XSLT используется совместно с XPath (XML Path Language) – языком запросов к элементам XML-документа. Запросы предназначены для организации доступа к узлам в XML-документе, осуществляют навигацию по DOM в XML.

XSLT-файл имеет следующую структуру:

1) обязательный корневой элемент stylesheet:

```
<xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/
    Transform>
    <Описание преобразования>
</xsl:stylesheet>
```

2) элементы шаблона, предназначенные для поиска и преобразования отдельных элементов документа;

3) конструкции для сериализации данных в выходной документ.

Описание основных элементов языка XSLT представлено в табл. 2.13.

Таблица 2.13 Описание основных элементов языка XSLT

Элемент	Описание
xsl:output	Элемент, управляющий форматом преобразованных данных. Обычно этот элемент объявляется сразу после элемента xsl:stylesheet. Обязательный атрибут method определяет тип выходного значения для таблицы стилей XSLT и может принимать одно их трех значений: xml, html или text. Пример преобразования XML в XML: <xsl:output method="xml" indent="yes"/>
xsl:template	Основной элемент языка. Используется для записи шаблонов преобразования. Корневой элемент xsl:stylesheet должен иметь хотя бы один дочерний элемент xsl:template. В содержимом элемента xsl:template задается конструктор последовательности преобразованных узлов XML-документа. xsl:template имеет следующие атрибуты:
	▪ match – обязательный атрибут, задает последовательность исходных узлов, для преобразования которых предназначен шаблон. Представляет собой выражение XPath, выделяющее узлы документа XML. Шаблон выполняется, когда обнаруживает совпадение (match); ▪ name – необязательный атрибут, используется элементом xsl:call-template для ссылки на шаблон; ▪ priority –необязательный атрибут, указывает порядок выполнения шаблонов, соответствующих одному и тому же выражению XPath; ▪ mode – необязательный атрибут. С его помощью можно сопоставлять шаблоны на основе значения, указанного в

	атрибуте mode, и выражения, указанного в выражении match. <xsl:template match="/book"> <Последовательность преобразованных узлов> </xsl:template> Шаблон выполняется для каждого узла <book> в документе XML
xsl:value-of	Используется для вывода значения узла. Значение выводится в виде строки. xsl:value-of представляет собой пустой элемент и не имеет никаких дочерних элементов. Выходное значение задается в обязательном атрибуте select этого элемента как выражение XPath: <xsl:template match="name"> <xsl:value-of select="."/> </xsl:template> Шаблон выполняется для элемента name. В выходной документ выводится значение элемента name
xsl:apply-templates	Задается чаще всего внутри элемента xsl:template, предписывает обработать рекурсивно узлы – потомки узлов, отобранных родительским элементом xsl:template: <xsl:template match="/"> <xsl:apply-templates/> </xsl:template> Пример рекурсивной обработки всех потомков корневого элемента XML-документа. Необязательный атрибут select ограничивает обработку только указанными в нем узлами. Значение атрибута задается как выражение XPath: <xsl:template match="product"> <xsl:apply-templates select="name"/> </xsl:template> <xsl:template match="name"> <xsl:value-of select="."/> </xsl:template>
xsl:if	Условный оператор, управляющий преобразованием. Конструктор преобразования задается внутри тега xsl:if и запускается только в том случае, если истинно выражение, указанное в атрибуте test: <xsl:template match="name"> <xsl:if test="@country="Russia""> <xsl:value-of select="."/> </xsl:if> </xsl:template> В выходной документ выводится значение элемента name

	только в том случае, если значение атрибута country равно Russia
xsl:for-each	Элемент используется для многократной обработки набора выбранных узлов. В атрибуте select задается выражение XPath, позволяющее отобрать необходимые узлы. После того как узлы выбраны, для каждого из них выполняются операторы, указанные в элементе xsl:for-each: <xsl:for-each select="Контрагенты/Контрагент"> <tr> <td><xsl:value-of select="Код"/></td> <td> <xsl:value-of select="Наименование"/> </td> </tr> </xsl:for-each> В выходной HTML-документ выводится таблица, содержащая столько строк, сколько элементов <Контрагент> содержит родительский тег <Контрагенты>
xsl:sort	Элемент используется для сортировки xml-тегов. Элемент должен размещаться внутри элементов xsl:apply-templates или xsl:for-each. Для задания атрибута или элемента, по которому выполняется сортировка, используется атрибут select. Для определения порядка сортировки используется атрибут order: <xsl:for-each select="Контрагенты/Контрагент"> <xsl:sort select="Наименование" order="ascending"/> <tr> <td><xsl:value-of select="Код"/></td> <td> <xsl:value-of select="Наименование"/> </td> </tr> </xsl:for-each> В выходной HTML-документ выводится таблица, строки которой отсортированы по наименованию контрагента (сортировка по возрастанию)
xsl:import	Позволяет повторно использовать шаблоны, определенные в других таблицах стилей. Записывается в корневом элементе xsl:stylesheet. Элементов xsl:import может быть несколько: xsl:import href="URI Адрес Внешнего XSLT"/> XSLT-процессор отыскивает внешнюю таблицу стилей и

	подставляет ее на место элемента xsl:import перед преобразованием. Если правила преобразования из внешних файлов XSLT конфликтуют с правилами, определенными в самой таблице стилей, то применяются те правила, которые записаны последними, поэтому важен порядок записи элементов xsl:import в таблицу стилей
xsl:include	Позволяет повторно использовать шаблоны, определенные в других таблицах стилей: <xsl:include href="URI Адрес Внешнего XSLT"/> Его можно записать не только в корневом элементе xsl:stylesheet, но и в любом месте таблицы стилей. Порядок записи элементов xsl:include в таблице стилей не имеет значения

В табл. 2.14 представлены выражения XPath, используемые в качестве значений атрибутов match и select.

Таблица 2.14

Выражения XPath, используемые в качестве значений атрибутов match и select

Выражение XPath	Описание
/	Корневой узел
.	Текущий узел. Может быть записано как self::node ()
..	Родительский узел текущего узла. Может быть записано как parent::node ()
Products	Узел Products
Products / Product1	Подузел Product1 узла Products
Product/*	Все потомки узла Product
/Product	Узел Product, являющийся прямым потомком корневого узла
@name	Атрибут name текущего узла
@*	Все атрибуты текущего узла
Product @ Price	Атрибут Price узла Product
Products / Product @ Price	Атрибут Price узла Product, являющегося подузлом узла Products
..@ Price	Атрибут Price родительского узла
//	Любое количество промежуточных узлов
Products // Product	Все узлы Product, имеющие предка Products
|	Знак разделения конкретных узлов
Product | Product2	Узел Product1и узел Product2

[]	Предикатное выражение
Products [Product]	Узел Products, имеющий потомка Product
Products [Products="qq"]	Узел Products, имеющий потомка Product, значение которого равно qq
Product [@Price]	Узел Product, имеющий атрибут Price
Product [@Price ="5"]	Узел Product, имеющий атрибут Price, значение которого равно 5
count (Product/*)	Количество потомков узла Product
name ()	Имя текущего узла

Рассмотрим практический пример использования языка XSLT. Пусть необходимо связать информационную систему, выполняющую регистрацию заказов клиентов (система А), с системой обработки заказов (система В), причем внутренние форматы данных в этих системах существенно отличаются.

Простейшей технологией, обеспечивающей слабое связывание приложений и создающей основу для эффективного управления изменениями, является передача сообщений. Пусть система А передает сообщение о заказе клиента следующего формата:

```
<Order>
    <date>01/11/2013</date>
    <OrderNumber>4554654</OrderNumber>
    <Customer>
        <Id>123</Id>
        <Name>Smith</Name>
    </Customer>
    <OrderItems>
        <Item>
            <Quantity>10</Quantity>
            <ItemNo>444</ItemNo>
            <Discription>IPhone</Discription>
        </Item>
        <Item>
            <Quantity>20</Quantity>
            <ItemNo>555</ItemNo>
            <Discription>IPad</Discription>
        </Item>
    </OrderItems>
</Order>
```

Система В готова принять и обработать заказ клиента в следующем формате:

```
<?xml version="1.0" encoding="utf-8"?>
<OrderItems>
    <OrderItem>
        <date>01/11/2013</date>
        <OrderNumber>4554654</OrderNumber>
        <CustomerID>123</CustomerID>
        <Quantity>10</Quantity>
        <ItemNo>444</ItemNo>
        <Discription>IPhone</Discription>
    </OrderItem>
    <OrderItem>
        <date>01/11/2013</date>
        <OrderNumber>4554654</OrderNumber>
        <CustomerID>123</CustomerID>
        <Quantity>20</Quantity>
        <ItemNo>555</ItemNo>
        <Discription>IPad</Discription>
    </OrderItem>
</OrderItems>
```

То есть заказ клиента, содержащий несколько позиций, должен быть разбит на части, каждая из которых дублирует информацию о дате заказа, номере заказа и идентификаторе заказчика. На рис. 2.5 показана логика преобразования данных в промежуточном слое интеграционного решения.

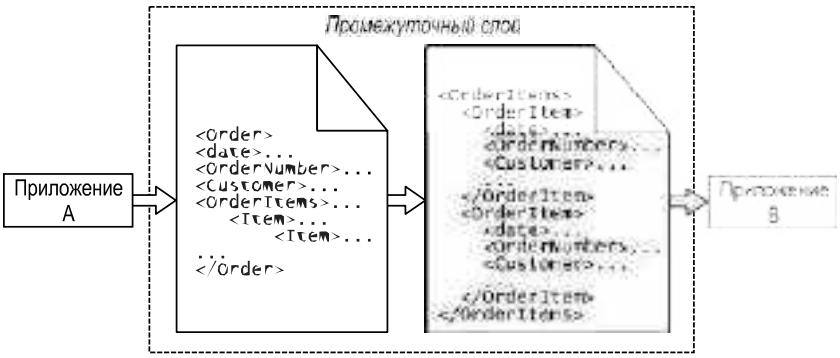


Рис. 2.5. Преобразование данных

Одним из способов подобного преобразования является использование преобразования XSLT. Следующий XSLT-код выполнит преобразование исходного XML-документа в требуемый формат:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform"
     xmlns:msxsl="urn: schemas-microsoft-com:xslt" exclude-
     resultprefixes="msxsl">
     <xsl:output method="xml" indent="yes"/>
     <xsl:template match="Order">
     <OrderItems>
          <xsl:apply-templates select="OrderItems/Item"/>
     </OrderItems>
     </xsl:template>
     <xsl:template match="Item">
        <OrderItem>
             <date>
                  <xsl:value-of select="parent::node()/
                  parent::node()/date"/>
              </date>
              <OrderNumber>
                  <xsl:value-of select="parent::node()/
                  parent::node()/OrderNumber"/>
              </OrderNumber>
              <CustomerID>
                  <xsl:value-of select="parent::node()/
                  parent::node()/Customer/Id"/>
              </CustomerID>
              <xsl:apply-templates select="*"/>
          </OrderItem>
     </xsl:template>
     <xsl:template match="*">
          <xsl:copy>
              <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

Преобразование находит в исходном документе корневой элемент Order, после чего создает корневой элемент <OrderItems> и начинает обрабатывать все элементы <Item>, найденные внутри элемента <OrderItems>. Для каждого из элементов <Item> создается новый шаблон, который копирует элементы date, OrderNumber и CustomerID из элемента Order исходного документа, находящегося на два уровня выше, и присоединяет к ним все остальные элементы, вложенные в Item.

Альтернативным вариантом является реализация программного разбора исходного XML-документа с копированием необходимых тегов в результирующий документ. Такое решение намного сложнее поддерживать в случае возможных изменений формата результирующего документа. Кроме того, практика показывает, что XSLT-преобразование выполняется намного быстрее, чем перенос элементов из исходного документа в результирующий программным путем.

2.4. Интеграция на уровне сервисов

Технология Web-сервисов разрабатывалась как основанная на открытых стандартах технология взаимодействия между приложениями. В отличие от плохо сочетаемых друг с другом стандартов удаленного вызова (DCOM, CORBA, RMI), технология Web-сервисов полностью снимает проблемы взаимодействия приложений, созданных и функционирующих на разных платформах, и позволяет строить хорошо масштабируемые слабо связанные интеграционные решения.

Web-сервисы могут выполнять два вида функций:

1) обмен данными между различными компонентами распределенной системы или несколькими взаимодействующими приложениями;

2) предоставление разнообразных сервисов (выполнение различных бизнес-функций).

Понятие Web-сервиса и его характеристики

Web-сервисы – это программные компоненты, взаимодействие с которыми осуществляется по сети Интернет с использованием открытых протоколов.

Каждый Web-сервис характеризуется URI-адресом и набором функций, которые могут быть вызваны удаленно. Web-сервис не предназначен для обслуживания конечных пользователей, он предоставляет услуги другим приложениям – клиентам Web-сервисов, поэтому у Web-сервиса нет пользовательского интерфейса – он имеет программный интерфейс, описанный в формате, пригодном для компьютерной обработки.

Интерфейс Web-сервиса определяет его абстрактную функциональность, а конкретная реализация интерфейса Web-сервиса обеспечивает отправку, обработку и получение сообщений. Обычно интерфейс Web-сервиса описывается в WSDL-формате. WSDL-описание Web-сервиса определяет формат пересылаемых сообщений, тип передаваемых данных, протокол передачи сообщений и адреса доставки сообщений.

Кроссплатформенность технологии Web-сервисов обеспечивается XML-форматом их описания и XML-форматом сообщений, отправляемых и получаемых Web-сервисами.

Взаимодействие с Web-сервисами осуществляется при помощи сообщений, передаваемых с использованием открытых протоколов, чаще всего протокола HTTP.

Таким образом, Web-сервис обладает следующими характеристиками:

- поддерживает единые методы обращения к нему;
- интерфейс Web-сервиса не зависит от платформы;
- способы внутренней реализации Web-сервиса обычно неизвестны инициатору запроса (клиенту сервиса). Web-сервис может быть реализован практически на любом языке программирования;
- Web-сервис является повторно используемым компонентом. Несмотря на то что название «Web-сервис» подразумевает ис-

пользование сервиса в сети Интернет, Web-сервисы могут работать во внутренней сети предприятия, обеспечивая связь между прикладными программами (бизнес-функция одного приложения становится доступна другим приложениям).

Web-сервисы могут обслуживать запросы на доступ к данным, преобразование данных или обмен данными. В этом случае Web-сервис обращается к источнику данных, выбирает данные, преобразует их и перемещает от источника к получателю. При перемещении данных важно правильно перевести данные из одного формата в другой, то есть отобразить метаданные источника на метаданные получателя. Для описания метаданных широко используются XML-схемы, реализованные в виде общих словарей предприятия.

Следует понимать, что построение интеграционного решения на основе Web-сервисов не всегда является оптимальным решением. Главными недостатками Web-сервисов являются меньшая производительность и больший размер сетевого трафика по сравнению с такими технологиями, как

RMI, CORBA, DCOM, за счет использования текстовых XML-сообщений. Технологию Web-сервисов целесообразно применять для объединения компонентов, работающих в распределенной среде на различных платформах или в том случае, если потребитель Web-сервиса заранее неизвестен.

Классификация Web-сервисов

Классификация Web-сервисов может быть выполнена на основе следующих критериев:

- по типу взаимодействия с клиентом;
- по типу клиента Web-сервиса;
- по модели обработки запроса Web-сервисом.

На основе модели обработки клиентского запроса выделяют следующие типы Web-сервисов:

- метод-ориентированные;
- документ-ориентированные;
- ресурс-ориентированные.

Метод-ориентированные Web-сервисы (или RPC-Web-сервисы) – это Web-сервисы, взаимодействие с которыми производится по протоколу SOAP (Simple Object Access Protocol) с использованием XML-сообщений. Интерфейс метод-ориентированных Web-сервисов описан в формате WSDL. Такое описание интерфейса сервиса обеспечивает автоматическую генерацию кода, необходимого для связи с сервисом, на стороне клиента. SOAP-протокол может использовать различные транспортные протоколы (HTTP, FTP, SMTP). SOAPсообщения, участвующие в обмене между клиентом и Web-сервисом, имеют строго определенную структуру и передают имя и параметры удаленно вызываемой процедуры, а также результат вызова.

Недостатком подобного подхода является достаточно тесная связь между клиентом и Web-сервисом (клиент должен знать все детали интерфейса Web-сервиса, при изменении имени или параметров вызываемой процедуры необходимо изменять клиентов).

Документ-ориентированные Web-сервисы – это XML-Web-сервисы, ориентированные на сообщения. Они обеспечивают низкоуровневую обработку XML-сообщений, содержание которых не связано с процедурами интерфейса, а полностью определяется XML-схемой. При получении сообщения от клиента Web-сервис обрабатывает полученные данные целиком и формирует ответное сообщение. XML-Web-сервисы могут принимать и передавать сообщения как в формате SOAP, так и в чистом XML-формате.

XML-Web-сервисы создают более слабую связь между клиентом и Web-сервисом, так как возможные изменения затрагивают XMLсхему, а не WSDL-описание сервиса. Однако при этом увеличивается сложность реализации Web-сервиса, поскольку требуется проводить анализ структуры сообщения.

Ресурс-ориентированные Web-сервисы – это RESTful-Web-сервисы, предоставляющие доступ к удаленным ресурсам с помощью HTTP-запросов. Сервисы данного типа основаны на архитектуре REST (Representational State Transfer) и обеспечивают взаимодействие с удаленными ресурсами, передавая клиенту их представление. RESTful-Web-сервисы поддерживают четыре операции – GET, PUT, POST и DELETE, с помощью которых производятся запрашиваемые клиентом действия с ресурсами. Технология REST Web-сервисов также может использовать WSDL-описание и SOAP-протокол для передачи сообщений, но может обходиться и без них.

Архитектура REST упрощает анализ клиентского запроса, поскольку HTTP-методы запроса напрямую связываются с операциями Web-сервиса. Недостаток технологии заключается в небольшом количестве доступных операций и ограничении «один ресурс – один URL-адрес».

Спецификация WSDL

Спецификация WSDL (Web Service Definition Language) определяет язык описания сервиса и обеспечивает способ описания технической информации о Web-сервисе и его интерфейсе, включая имя и адрес сервиса, способ и протокол взаимодействия с сервисом, форматы сообщений, функциональность сервиса. В настоящий момент существуют спецификации

WSDL 1.1 и WSDL 2.0. Подробнее о спецификациях см..

Протоколы передачи данных

Для обеспечения кроссплатформенности сервиса запрос, посылаемый клиентом Web-сервису, и ответ, который сервис возвращает инициатору запроса, представляет собой сообщение XML-формата. SOAP (Simple Object Access Protocol) – это протокол обмена сообщениями в распределенной среде, то есть соглашение о структуре документов, участвующих в обмене сообщениями. SOAP-протокол может использоваться поверх разнообразных транспортных протоколов для передачи символьной и двоичной информации. В настоящий момент существуют спецификации SOAP 1.1 и SOAP 1.2.

Типы взаимодействия с клиентом

Web-сервисы поддерживают синхронную и асинхронную модели взаимодействия с клиентом. В случае синхронного взаимодействия после отправки запроса Web-сервису клиент блокируется до получения ответа. При асинхронном взаимодействии работа клиента не блокируется, обработка ответа производится после его получения.

В рамках синхронной и асинхронной моделей могут быть реализованы следующие сценарии взаимодействия клиента с Webсервисом:

- однонаправленный запрос – клиент посылает сообщение одному или нескольким Web-сервисам, которые его обрабатывают, но не генерируют ответ;
- синхронный запрос-ответ – синхронный обмен сообщениями между клиентом и Web-сервисом;
- асинхронный запрос-ответ – асинхронный обмен сообщениями между клиентом и Web-сервисом;
- RPC-вызов – клиент вызывает процедуру интерфейса Webсервиса путем сериализации параметров процедуры в сообщение и отправки его Web-сервису;
- сообщение об ошибке – в случае, если выполнение процедуры интерфейса Web-сервиса завершается с ошибкой, клиент получает сообщение

об ошибке;

- запрос с подтверждением – клиент получает от Web-сервиса сообщение об успешной или неуспешной доставке запроса;
- передача через посредников – клиентское сообщение передается целевому Web-сервису через посредников (промежуточные Web-сервисы);
- кеширование – в случае схожего клиентского запроса ответ ему посылается из кеша;
- дополнительная функциональность – при обмене сообщениями может быть дополнительно реализованы шифрование данных, аутентификация пользователей, поддержка транзакций и т.п.

Клиенты Web-сервисов

Клиентами, пользующимися услугами Web-сервисов, могут быть приложения двух типов:

1) клиентские приложения, взаимодействие которых с Web-сервисом инициирует конечный пользователь (такие приложения могут быть как «тонким», так и «толстым» клиентом);
2) программные компоненты, автоматически совершающие запрос к Web-сервису (без участия пользователя).

Репозитории Web-сервисов

Если Web-сервис предоставляет услуги для широкого круга заранее неизвестных потребителей, то он должен быть зарегистрирован в каком-либо из открытых репозиториев (реестров), чтобы потенциальные пользователи смогли его отыскать.

При использовании Web-сервисов в целях интеграции корпоративных приложений создаются специальные закрытые реестры, позволяющие осуществлять поиск нужного Web-сервиса.

Наиболее распространенными стандартами для создания корпоративных реестров являются UDDI, ebXML и DISCO.

UDDI (Universal Discription, Discovery and Integration) – стандарт для создания платформонезависимого, основанного на XML реестра Web-

сервисов, позволяющего публиковать и находить их WSDL-описание. UDDI-реестр хранит информацию о поставщиках, функциональности и деталях реализации Web-сервисов.

UDDI-реестр логически разделен на три типа каталогов:

1) белые страницы – содержат информацию о поставщиках Web-сервисов (имя поставщика, описание, контактная информация);

2) желтые страницы – классифицируют Web-сервисы по сферам применения;

3) зеленые страницы – содержат информацию о деталях реализации Web-сервисов, необходимую для доступа к ним.

Данные каталогов хранятся в виде XML-файлов, структура которых определяется моделью данных UDDI. Каждый XML-файл с данными имеет уникальный UDDI-идентификатор и структуру, определяемую XML-схемами. Администрирование UDDI-реестров осуществляется с помощью специальных приложений UDDI-серверов, которые, по сути, представляют собой набор Web-сервисов для поиска, публикации, хранения и репликации данных реестра. Примером UDDI-сервера с открытым исходным кодом является http://ws.apache.org/juddi.

ebXML (Electronic Business using eXtensible Markup Language) – основанный на XML стандарт, обеспечивающий инфраструктуру для создания единого электронного рынка. Стандарт позволяет создавать бизнес-документы и описания бизнес-процессов, хранить их в едином реестре и предоставлять информацию для поиска бизнес-партнеров.

ebXML-реестр может быть создан как для отдельной корпорации, так и для предприятий отдельной отрасли. Примеры реализации технологии ebXML можно найти на официальном сайте проекта http://www.ebxml.org.

DISCO – это технология компании Microsoft, предназначенная для публикации и поиска Web-сервисов. Суть технологии заключается в том, что в определенном каталоге сервера, на котором развернут Web-сервис, создается XML-документ определенной структуры (DISCO-документ), содержащий

ссылку на WSDL-описание Web-сервиса. Поиск сервисов осуществляется с помощью специальной утилиты, генерирующей файл с результатами поиска, на основании которого автоматически создаются клиентские заглушки Web-сервиса.

Обобщенная схема взаимодействия поставщика сервиса, клиента сервиса и репозитория сервисов представлена на рис. 2.6.

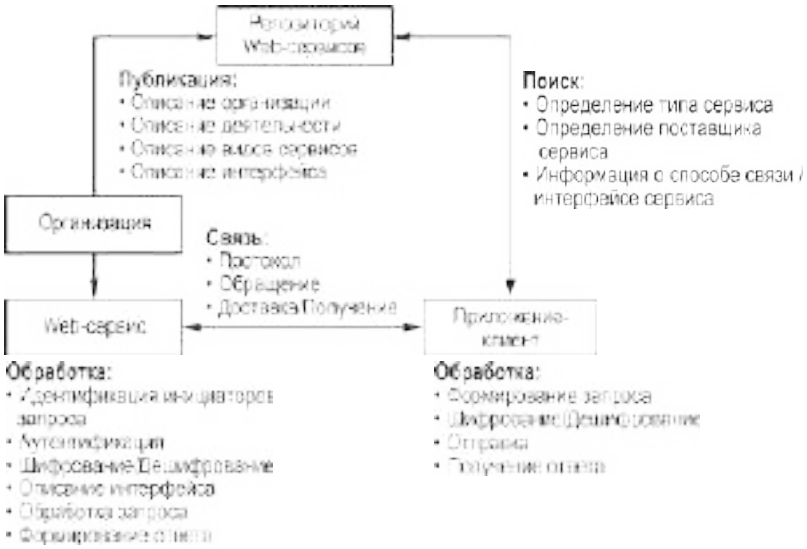


Рис. 2.6. Взаимодействие поставщика и клиента Web-сервиса с репозиторием сервисов

Функционирование интеграционных решений, использующих Web-сервисы (SOA-сценарий взаимодействия приложений), основано на выполнении бизнес-процессов путем последовательного взаимодействия с рядом Web-сервисов.

Существует два подхода к описанию бизнес-процессов и интеграции Web-сервисов – это оркестровка и хореография.

Оркестровка Web-сервисов – это описание корпоративного бизнеспроцесса в виде набора взаимодействий внутренних и внешних Webсервисов. Контроль над бизнес-процессом осуществляется данной компанией с помощью центрального координатора, который в общем случае также может быть реализован как Web-сервис. Центральный координатор управляет взаимодействием Web-сервисов и определяет порядок выполнения

ими необходимых функций. Оркестровка представляет точку зрения на бизнес-процесс со стороны его владельца. Хореография Web-сервисов – это описание глобального бизнеспроцесса, в который вовлечены несколько участников (например, разные компании), обменивающихся сообщениями с целью достижения общего бизнес-результата. В данном случае не требуется центральный координатор, так как каждый участник взаимодействия обладает информацией о том, когда выполнять свои операции и с каким внешним Web-сервисом требуется взаимодействовать. Хореография позволяет описать бизнес-процесс как совместное действие нескольких участников, каждый из которых знает в нем свою роль. Хореография Web-сервисов может быть реализована набором оркестровок. Различия между оркестровкой и хореографией иллюстрируют рис. 2.7, 2.8.

Рис. 2.7. Оркестровка сервисов

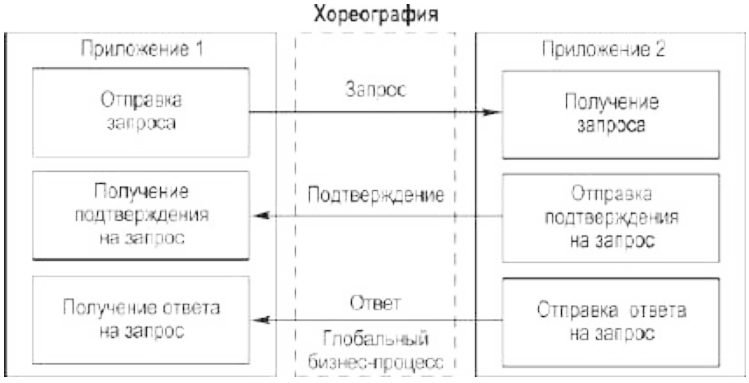


Рис. 2.8. Хореография сервисов

Для проектирования SOA-сценария взаимодействия информационной

системы необходимо получить формальное описание бизнес-процессов и описание их связи с Web-сервисами. Описания бизнес-процессов в контексте технологии Web-сервисов выполняется с использованием языков WS-BPEL (Web Services Business Process Execution Language) и WS-CDL (Web Services Choreography Discription Language).

Язык WS-BPEL

Язык WS-BPEL предназначен для описания бизнес-процессов, представляющих собой связанную последовательность Web-сервисов. Язык описывает оркестровку Web-сервисов при выполнении бизнес-процесса и представляет собой подмножество языка разметки XML, а также позволяет декларативно описывать связи со взаимодействующими системами и порядок обращения к сервисам, предоставляемым ими.

XML-описание бизнес-процесса, как правило, создается с помощью BPEL-визуального редактора, а затем исполняется BPEL-сервером. Язык WS-BPEL поддерживается различными серверами и средами разработки (WebLogic, Eclipse, Oracle BPEL Process Manager, Oracle Application Server, Web Sphere, Microsoft Windows Workflow Foundation и др.).

Различают абстрактное и исполняемое BPEL-описания. Абстрактное BPEL-описание используется на первоначальном этапе разработки BPEL-процесса и описывает бизнес-процесс частично, без излишней детализации. Исполняемое BPEL-описание полностью декларирует процесс и предназначено для выполнения BPEL-сервером.

Поскольку BPEL предназначен для определения последовательности вызова сервисов, он использует объявления действий. Причем BPEL-действия могут быть двух типов: структурные (содержащие другие действия и определяющие логику их выполнения) и базовые (элементарные действия).

Примеры структурных и базовых действий представлены в табл. 2.15.

С полной спецификацией языка можно ознакомиться на сайте разработчика стандарта Organization for the Advancement of Structured Information Standards (OASIS).

Таблица 2.15 Примеры структурных и базовых действий

BPEL-действия	Описание
Структурные BPEL-действия	
Sequence	Содержит действия, выполняемые последовательно
If	Задает условия, для которых выполняются вложенные действия
while	Условие повторения вложенного действия
Pick	Выполняет действие при получении сообщения, связанного с действием
Flow	Обеспечивает параллельное и синхронизированное выполнение действий
For Each	Циклически повторяет действия
Базовые BPEL-действия	
Invoke	Вызов Web-сервиса
Receive	Получение сообщения от бизнес-партнера
Reply	Посылает ответ после получения сообщения от партнера
wait	Определяет задержку BPEL-процесса
exit	Завершает экземпляр BPEL-процесса

Язык WS-CDL

Web Services Choreography Description Language (WS-CDL) – это XML-язык для описания взаимодействия отдельных сервисов между собой (хореографии сервисов). Язык описывает наборы правил для определения порядка обмена сообщениями между бизнеспартнерами.

WS-CDL не является исполняемым языком описания бизнеспроцессов, он позволяет только строить модель взаимодействия бизнес-партнеров, которую затем можно реализовать с помощью исполняемого языка (BPEL) или любого другого языка программирования.

3. Лабораторный практикум

Лабораторная работа №1

Вопросы для проработки:

1. *Разбор концепции собственного цифрового сервиса для выбранной отрасли*
2. *Формирование требований к цифровому сервису*

Методические указания по проведению занятия

Занятия проводятся в три этапа. На первом этапе преподаватель дает краткую характеристику понятия «требования к программному обеспечению», формулирует основные виды требований и их отличия.

На втором этапе происходит обсуждение концепции собственного цифрового сервиса, сформированной обучающимися в ходе самостоятельной работы по теме 1.1 теоретического раздела.

На третьем этапе происходит формулировка требований к цифровому сервису. В учёт берутся полученные в ходе обсуждения предложения и замечания, а также теоретическое введение, данное преподавателем.

По результатам выполнения занятия формируется отчет, содержащий результаты каждого из трёх этапов.

Командная работа в аудитории

Командная работа в аудитории начинается с первого этапа. Преподаватель вводит обучающихся в проблематику работы с требованиями. На практических примерах приводится зависимость качества результата от точной формулировки требований.

Вводятся такие понятия, как: высокоуровневые требования, бизнес-требования, пользовательские требования, функциональные требования.

На конкретном примере, предложенном преподавателем, студенты в командном режиме формируют 3–5 примеров требований по каждому их виду.

По окончании ознакомления с требованиями к системе начинается

второй этап – обсуждение концепции собственного цифрового сервиса. Обучающиеся объединяются в команды по 4 человека для формирования предложений и замечаний к разработанной концепции цифрового сервиса. Роли в команде распределяются следующим образом: один обучающийся выступает в роли заказчика системы (автора концепции), трое других – в роли группы разработки.

В ходе обсуждения автор концепции должен донести до группы разработки основную идею своего сервиса и, совместно с группой разработки, сформировать к нему бизнес-требования.

По окончании обсуждения концепции обучающиеся меняются ролями до тех пор, пока не будут обсуждены все концепции авторов, входящих в состав команды.

Индивидуальная работа в аудитории

В ходе самостоятельной работы задача обучающегося состоит в формировании высокоуровневых, пользовательских и функциональных требований.

Необходимо учесть замечания и пожелания, высказанные группой разработки по концепции цифрового сервиса и, а также ориентироваться на сформулированные бизнес-требования.

Требования к отчету

Отчёт по работе должен содержать следующие разделы:

1. Цель работы, сформулированная с учётом выбранной предметной области
2. Теоретическое введение, сформулированное на основе обсуждения проблематики требований с преподавателем.
3. Описание первоначальной концепции цифрового сервиса, сформулированной студентом самостоятельно
4. Описание отредактированной по результатам обсуждения в команде концепции цифрового сервиса, в том числе таблицу с произведёнными изменениями. В таблице, помимо самих изменений,

должно быть отражено обоснование принятых решений.

5. Модель бизнес-требований, сформированную в результате обсуждения концепции цифрового сервиса в команде.
6. Модель высокоуровневых требований к цифровому сервису.
7. Модель пользовательских требований к цифровому сервису.
8. Модель функциональных требований к цифровому сервису.

Лабораторная работа №2

Вопросы для проработки:

1. *Системный проект бизнес-приложения*
2. *Функциональная модель бизнес-приложения.*
3. *Структурная модель бизнес-приложения*

Методические указания по проведению занятия

Занятия проводятся в четыре этапа. На первом этапе преподаватель даёт разъяснения теоретических положений и на практическом примере показывает, что из себя представляет системный проект бизнес-приложения, а также то, каким образом использовать унифицированный язык моделирования UML для описания функциональной и структурной модели бизнес-приложения.

На втором этапе обучающиеся должны сформировать системный проект для своего бизнес-приложения, опираясь на теоретические положения и примеры, изложенные преподавателем.

На третьем этапе происходит разработка функциональной модели системы, в которой отражается логика работы создаваемого бизнес-приложения, а также структурной модели системы, в которой отражается распределение функций по элементам, а также распределение элементов по рабочим местам и пользователям.

На четвертом этапе производится верификация модели в соответствии с требованиями и системным проектом.

Для разработки моделей используется унифицированный язык моделирования UML.

Командная работа в аудитории

Командная работа в аудитории начинается с первого этапа. Преподаватель вводит обучающихся в проблематику проектирования систем. Рассматривается пример на одну из выбранных обучающимися тематик. Преподаватель демонстрирует этапность работ, а также зависимость моделей друг от друга. В результате коллективной работы должен быть создан типовой системный проект на выбранную тематику.

Второй этап является командным и подразумевает разработку системного проекта бизнес-приложения. Обучающиеся вновь объединяются в группы, в том же составе, что и на первом занятии. Выполняя роль заказчика своего проекта, и, тем самым, определяя ограничения и требования к реализации продукта, обучающийся совместно с командной разработки описывает основные аспекты дальнейшей реализации системы.

Четвёртый этап подразумевает возвращение обучающихся к командному режиму. На данном этапе обучающийся-разработчик моделей демонстрирует их группе разработки, которая выполнят роль экспертов. Задача группы разработки оценить модели и провести их верификацию на предмет соответствия требованиям, полученным в рамках предыдущей работы. Все выявленные несоответствия документируются.

Индивидуальная работа в аудитории

Третий этап является индивидуальным, в ходе которого каждый из студентов разрабатывает функциональную и структурную модели своего бизнес-приложения. Функциональная модель должна отражать следующие аспекты:

1. Возможности взаимодействия пользователя с системой.
2. Логику выполнения системной отдельных функций (функция авторизации в расчет не берётся, если не используется оригинальный, разработанный студентом подход).

3. Хронологию взаимодействия системы с пользователем в ходе выполнения функций.
4. Хронологию взаимодействия компонентов системы в ходе выполнения функций.

Структурная модель должна отражать следующие аспекты:

1. Внутреннюю структуру системы: перечень её основных компонентов и связей между ними.
2. Размещение компонентов системы по физическим узлам.

Требования к отчету

Отчёт по работе должен содержать следующие разделы:

1. Цель работы, сформулированная с учётом выбранной предметной области
2. Теоретическое введение, сформулированное на основе обсуждения проблематики проектирования информационных систем с преподавателем.
3. Системный проект, сформированный по результатам командной работы
4. Функциональную модель бизнес-приложения, в том числе:
    a. Возможности взаимодействия пользователя с системой.
    b. Логику выполнения системной отдельных функций (функция авторизации в расчет не берётся, если не используется оригинальный, разработанный студентом подход).
    c. Хронологию взаимодействия системы с пользователем в ходе выполнения функций.
    a. Хронологию взаимодействия компонентов системы в ходе выполнения функций
5. Структурную модель бизнес-приложения, в том числе:
    a. Внутреннюю структуру системы: перечень её основных компонентов и связей между ними.
    b. Размещение компонентов системы по физическим узлам.

6. Протокол верификации модели.

Лабораторная работа №3

Вопросы для проработки:

1. *Разработка бизнес-приложения.*
2. *Тестирование бизнес-приложения.*

Методические указания по проведению занятия

Занятия проводятся в четыре этапа. На первом этапе преподаватель осуществляет разработку бизнес-приложения на основе сформированной совместно с коллективом обучающихся документации. Первоочередная задача преподавателя на данном этапе не реализовать полноценную версию системы, а отработать с обучающимися типовые ошибки, показать на примере то, каким образом требования и проектные модели преобразуются в готовое решение.

Второй этап подразумевает разработку обучающимися ядра своего бизнес-приложения, которое выполняет основные функции в соответствии с требованиями и проектом.

Третий этап подразумевает тестирование разработанного бизнес-приложения и проверку корректности его функционирования.

Четвертый этап подразумевает устранение выявленных ошибок и недочётов (в случае наличия таковых) и документирование разработки.

Командная работа в аудитории

Командная работа в аудитории выполняется на третьем этапе. Команды, сформированные на первом занятии в составе только группы разработки (исключая на время разработчика бизнес-приложения) осуществляют тестирование системы.

Задача команды на данном этапе выявить в первую очередь несоответствие приложения требованиям и проекту. В части тестирования работоспособности приложения следует обращать внимание только на грубые ошибки (неожиданное завершение работы приложения, необрабатываемые

исключения и пр.).

По окончании проверки команда разработки знакомит разработчика бизнес-приложения с её результатами. Результаты должны быть задокументированы таким образом, чтобы разработчик бизнес-приложения мог в случае наличия критических замечаний устранить их.

Индивидуальная работа в аудитории

Индивидуальная работа в аудитории двухэтапная, выполняется поочерёдно с командной.

На первом этапе индивидуальной работы обучающийся реализует своё бизнес-приложение в соответствии с требованиями и проектной документацией, полученными в ходе выполнения предыдущих работ.

Стек технологий выбирается обучающимся самостоятельно, исходя из следующих предпосылок:

1. Стопроцентная реализуемость требований в рамках выбранного стека технологий.
2. Понимание собственных возможностей при работе с выбранным стеком технологий.
3. Доступность стека технологий в учебном заведении и конкретной лаборатории, где проводятся занятия.

Обучающийся в праве в качестве отдельных компонентов бизнес-приложения использовать готовые решения, если это не противоречит лицензионному соглашению.

Задача обучающегося реализовать как можно больше полезных функций в рамках отведённого времени. Под полезными функциями понимается осуществление пользователем или самостоятельно системой действий, приводящих к обработке информации в базе данных.

Второй этап индивидуальной работы посвящён доработке в соответствии с замечаниями (при наличии) группы разработки. Задача обучающегося привести бизнес-приложение в соответствие требованиям, устранив максимальное количество замечаний в рамках отведённого времени.

Требования к отчету

Отчёт по работе должен содержать следующие разделы:

1. Цель работы, сформулированная с учётом выбранной предметной области
2. Спецификацию стека используемых технологий с кратким обоснованием выбора. При формировании обоснования следует избегать общих фраз из разряда «удобный интерфейс» и пр.
3. Описание процесса реализации бизнес-приложения. Обучающемуся следует уделить внимание отражению требований и проектных моделей в реализуемых компонентах. Листинг модулей следует приводить в приложении к отчёту.
4. Протокол верификации системы, полученный от группы разработки
5. Описание процесса устранения замечаний (при наличии), отраженных в протоколе группы разработки.

Лабораторная работа №4

Вопросы для проработки:

1. *Выбор и интеграция цифровой технологии в качестве поставщика данных*
2. *Выбор и интеграция цифровой технологии в качестве обработки данных, расположенных в базе данных бизнес-приложения*

Методические указания по проведению занятия

Занятия проводятся в четыре этапа. На первом этапе преподаватель даёт разъяснения теоретических положений и на практическом примере показывает, каким образом осуществляется интеграция цифровых технологий в бизнес-приложение. Задача преподавателя разобрать как минимум два кейса интеграции: когда в исходном бизнес-приложении изначально заложены возможности взаимодействия с цифровыми сервисами и когда такой возможности нет.

На втором этапе обучающиеся самостоятельно выбирают цифровую

технологию, которая в их бизнес-приложении будет выполнять роль поставщика данных.

На третьем этапе обучающиеся самостоятельно выбирают цифровую технологию, которая в их бизнес-приложении будет выполнять роль дополнительного модуля для обработки данных, размещённых в базе данных.

Технологии для второго и третьего этапов выбираются из следующих категорий:

1. Нейротехнологии и Искусственный интеллект
2. Технологии виртуальной и дополненной реальностей
3. Технологии распределенного реестра
4. Технологии беспроводной связи

По согласованию с преподавателем и, в случае наличия соответствующей материально-технической базы студентом могут быть выбраны цифровые технологии, связанные с обширным использованием аппаратных составляющих: Новые производственные технологии, Квантовые технологии и Компоненты робототехники и сенсорика.

На четвертом этапе обучающиеся должны подготовить питч по своему бизнес-приложению. Задача питча – убедить аудиторию в лице преподавателя и других обучающихся, выполняющих роль инвесторов, в том, что разработанное бизнес-приложение имеет перспективы, может быть приобретено как готовый продукт или требует инвестиций для коммерциализации.

Командная работа в аудитории

Командная работа в аудитории в рамках данного занятия состоит в оценке всей группой проекта, разработанного обучающимся. Задача этапа – дать объективную оценку перспектив проекта и степени его готовности в соответствии с требованиями, разработанными на предыдущих занятиях.

Индивидуальная работа в аудитории

Индивидуальная работа обучающегося состоит из двух этапов. На первом этапе она заключается в реализации интегрированного бизнес-

приложения, в состав которого включены как минимум две цифровых технологии.

Первая цифровая технология должна отвечать за поставку данных, которые в дальнейшем обрабатываются системой. В качестве такой технологии может рассматриваться, например, Интернет вещей как поставщик актуальных данных о складских запасах и состоянии их хранения (температуре, влажности). Альтернативой может быть вектор входных признаков, полученный от внешних библиотек распознавания образов.

Вторая цифровая технология должна отвечать за обработку данных, которые уже имеются в системе. В качестве такой технологии могут быть рассмотрены, например, нейронные сети, с помощью которых реализуются прикладные алгоритмы искусственного интеллекта.

Главная задача обучающегося при выполнении работы – интегрировать технологию. По этой причине допускается и приветствуется использование готовых библиотек и/или сторонних реализованных сервисов.

Второй этап индивидуальной работы подразумевает подготовку питча по своему бизнес-приложению. В зависимости от уровня готовности обучающийся самостоятельно принимает решение о цели питча: привлечение инвестора или продажа готового продукта потенциальным клиентам. Основные требования к питчу: длительность не более 3 минут, количество слайдов не более 5.

Требования к отчету

Отчёт по работе должен содержать следующие разделы:

1. Цель работы, сформулированная с учётом выбранной предметной области
2. Спецификацию выбранных цифровых технологий с кратким обоснованием выбора.
3. Спецификацию библиотек или готовых сервисов, с помощью которых в работе планируется реализовать цифровые технологии.
4. Описание процесса интеграции цифровых технологий в бизнес-

приложение. Описание также должно включать обновлённую проектную документацию в части, затрагиваемой расширением функционала бизнес-приложения

5. Слайды презентации и речь, использованную в питче по своему бизнес-приложению.

Системная интеграция и корпоративные информационные системы

Презентации к лекциям

Цель и задачи курса

Цель курса:
Формирование у студентов фундаментальных теоретических знаний по вопросам методологии разработки, внедрения, интеграции и использования корпоративных информационных систем.

В ходе изучения курса у магистранта должно сформироваться представление об актуальных направлениях в автоматизации деятельности предприятий.

Определения

Информационные системы – это проекция систем реального мира на область ИТ. Проекция должна отображать любое состояние системы реального мира в тот или иной момент времени. (Wang, Wand)

Информационная система (ИС) — это система, реализующая информационную модель предметной области, чаще всего — какой-либо области человеческой деятельности. ИС должна обеспечивать: получение (ввод или сбор), хранение, поиск, передачу и обработку (преобразование) информации. [ГОСТ 34].

Информационная система – совокупность содержащейся в базах данных информации (сведения/сообщения/данные независимо от формы их представления) и обеспечивающих ее обработку информационных технологий (процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов) и технических средств. [149-ФЗ “ОБ ИНФОРМАЦИИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ И О ЗАЩИТЕ ИНФОРМАЦИИ” 27.07.2006]

Определения

Автоматизированная информационная система - такая информационная система, которая объединяет персонал и комплекс средств автоматизации деятельности персонала, реализующая информационную технологию выполнения установленных функций [ГОСТ 34].

Автоматизированная информационная система - комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а так же системный персонал, который обеспечивает поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователя .

Определения

АСУ — Автоматизированные системы управления
АСУП — Автоматизированные системы управления предприятия
АСУ ТП — Автоматизированные системы управления технологическими процессами

Автоматизи´рованная систе´ма (АС) — система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций.

Автоматизи´рованная систе´ма(АС) — это организованная совокупность средств, методов и мероприятий, используемых для регулярной обработки информации для решения задачи.

Если автоматизируемый процесс связан в основном с обработкой информации, то такая система называется автоматизированной информационной системой.

Определения

Корпоративная информационная система (КИС) – автоматизированная информационная система и вся инфраструктура предприятия, обеспечивающая автоматизацию процессов корпорации и поддержку принятия решений.

Корпоративные информационные системы (КИС) - это интегрированные системы управления территориально распределенной корпорацией, основанные на углубленном анализе данных, широком использовании систем информационной поддержки принятия решений, электронных документообороте и делопроизводстве. КИС призваны объединить стратегию управления предприятием и передовые информационные технологии.

КИС - масштабируемая система, предназначенная для комплексной автоматизации всех видов хозяйственной деятельности больших и средних предприятий, в том числе корпораций, состоящих из группы компаний, требующих единого управления.

Определения

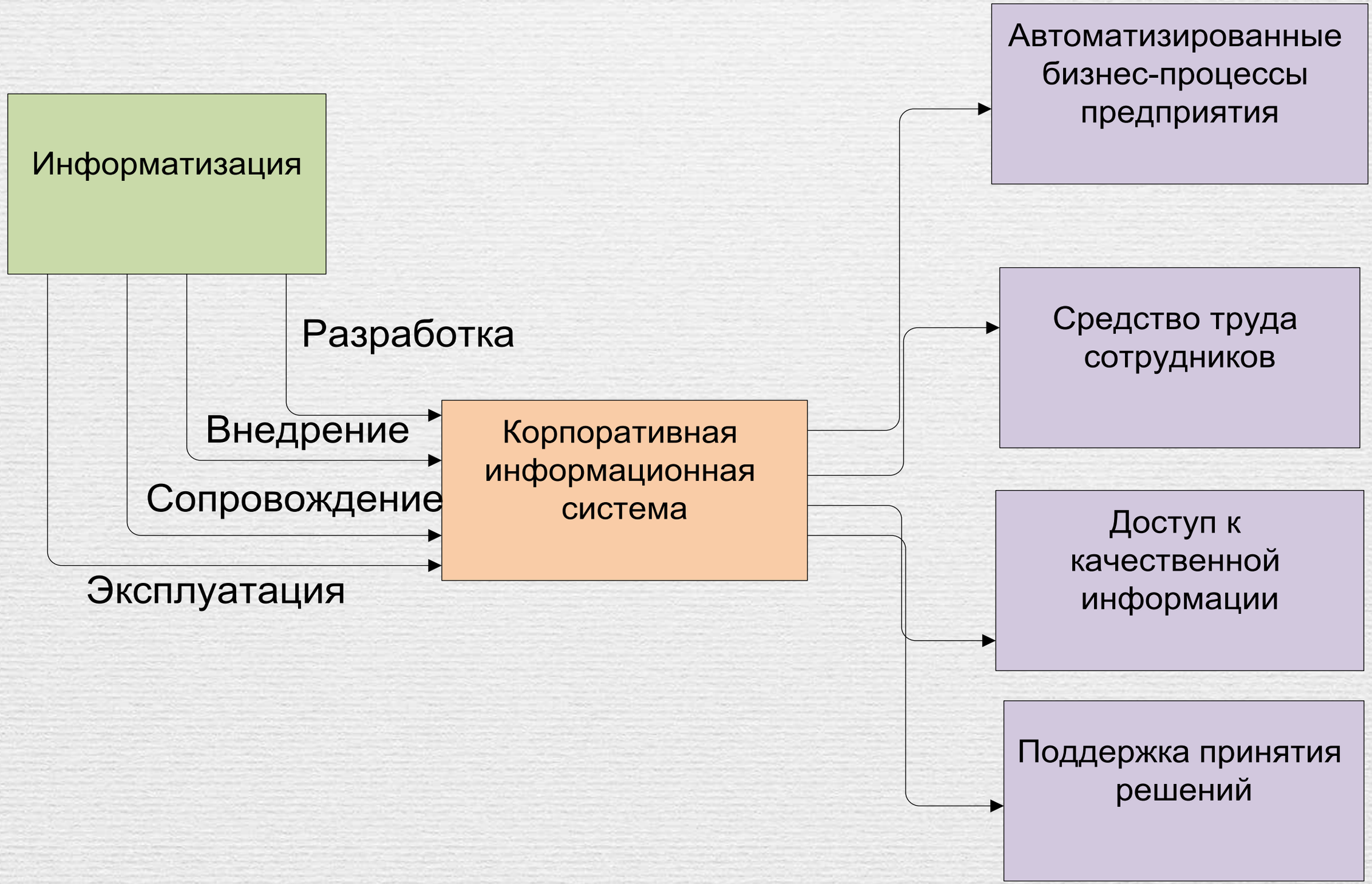
Бизнес-процесс - это совокупность действий (операций), направленных на достижение какой-либо определенной цели .

Всю деятельность предприятия можно представить как совокупность бизнес-процессов

Элементарный бизнес-процесс - это действие (операция), имеющее вход, выход, механизм исполнения, ограничения или условия на выполнение и продолжительность.

Составной бизнес-процесс – иерархия элементарных и других составных бизнес-процессов

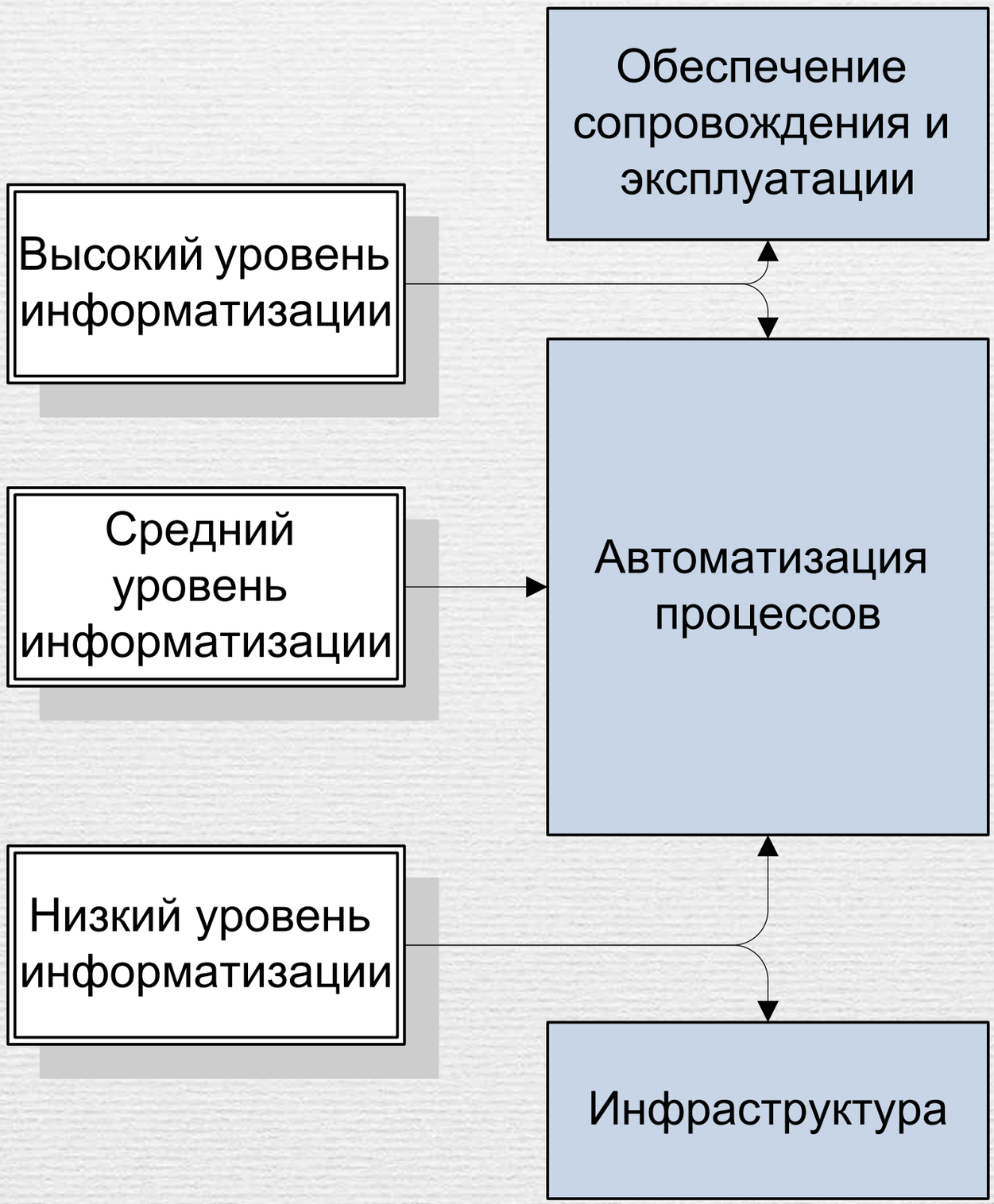
Выгоды использования КИС



Цели внедрения КИС

Области информатизации

Уровни информатизации



Требования к составу КИС

В состав КИС должны войти

- автоматизация процессов предметных областей деятельности предприятия
- средства для документационного обеспечения управления
- коммуникационное программное обеспечение
- средства организации коллективной работы сотрудников
- другие вспомогательные (технологические) продукты.

КИС должна поддерживать интеграцию большого числа программных продуктов.

КИС объединяет CASE, ERP, CRM, MRP, WMS, ECM, СЭД и др.

MRP

MRP (Material Requirement Planning) — система планирования потребностей в материалах

Автоматизация планирования потребностей в материалах состоит из логически связанных процедур
решающих правил и требований
MRP переводит производственное расписание в «цепочку требований», синхронизированных во времени, и запланированных «покрытий» этих требований для каждой единицы запаса компонентов, необходимых для выполнения производственного расписания.

Цель MRP
удовлетворение потребности в материалах, компонентах и продукции для планирования производства и доставки потребителям;
поддержка низких уровней запасов;
планирование производственных операций, расписаний доставки, закупочных операций.

MRP II

MRP II (Manufacturing Resource Planning) — Планирование производственных ресурсов - методология, направленная на более широкий охват ресурсов предприятия, нежели MRP. В отличие от MRP, в системе MRP II производится планирование не только в материальном, но и в денежном выражении.

Включает следующие алгоритмы:

- Планирование продаж и производств
- Управление спросом
- Составление плана производства
- Планирование материальных потребностей
- Спецификации продуктов
- Управление складом
- Плановые поставки
- Планирование производственных мощностей
- Управление финансами
- Моделирование
- Оценка результатов деятельности
- Планирование распределения ресурсов
- Материально-техническое снабжение

ERP

ERP - система (Enterprise Resource Planning) - система планирования ресурсов предприятия. Интегрированная система на базе ИТ для управления внутренними и внешними ресурсами предприятия (физические активы, финансовые, материально-технические и человеческие ресурсы).

Цель системы — содействие потокам информации между всеми хозяйственными подразделениями (бизнес-функциями) внутри предприятия и информационная поддержка связей с другими предприятиями.

ERP-система обычно имеет единую базу данных и формирует стандартизованное единое информационное пространство предприятия.

CRM

CRM - система (Customer Relationship Management System) — система управления взаимодействием с клиентами.

Функции
Учет клиентов (проектов, контрактов и т.п.)
Учет истории взаимоотношений с клиентами
Планирование взаимодействий с клиентами
Анализ клиентов и результатов взаимодействий (воронка продаж, оценка эффективности продаж, кластеризация клиентов и т.п.)

CRM системы представлены
Системами Управления продажами
Управление маркетингом
Управление сервисом и call-центром

MES

MES (Manufacturing Execution System) — производственная исполнительная система. Системы решают задачи синхронизации, координируют, анализируют и оптимизируют выпуск продукции в рамках какого-либо производства.

MES- включает :

Активация производственных мощностей

Отслеживание производственных мощностей

Сбор информации, связанной с производством

Отслеживание и контроль параметров качества

Обеспечение персонала и оборудования информацией, необходимой для начала процесса производства

Установление связей между персоналом и оборудованием в рамках производства

Установление связей между производством и поставщиками, потребителями, инженерным отделом, отделом продаж и менеджментом

ECM

Enterprise content management (ECM) — управление информационными ресурсами предприятия или управление корпоративной информацией

ECM — это стратегическая инфраструктура и техническая архитектура для поддержки единого жизненного цикла неструктурированной информации (контента) различных типов и форматов.

ECM-системы реализуют :

Управление документами — экспорт/импорт, контроль версий, безопасность, архивирование.

Управление потоками работ (Workflow) — поддержка бизнес-процессов, передача контента по маршрутам, назначение рабочих задач и состояний, создание журналов аудита.

Управление веб-контентом (WCM) — автоматизация роли веб-мастера, управление динамическим контентом и взаимодействием пользователей.

Управление мультимедиаконтентом (DAM) — управление графическими, видео и аудиофайлами.

Управление знаниями (Knowledge Management) — поддержка систем для накопления и доставки релевантной для бизнеса информации.

Документо-ориентированное взаимодействие (Collaboration) — совместное использование документов пользователями и поддержка проектных команд.

СЭД

Система автоматизации документооборота, система электронного документооборота (СЭД) — автоматизированная многопользовательская система, сопровождающая процесс управления работой иерархической организации с целью обеспечения выполнения этой организацией своих функций. При этом предполагается, что процесс управления опирается на человеко-читаемые документы, содержащие в слабоформализованной форме инструкции для сотрудников организации, необходимые к исполнению.

Основные принципы СЭД

Однократная регистрация документа и единая база документов

Возможность параллельного выполнения операций, позволяющая сократить время движения документов и повышения оперативности их исполнения

Непрерывность движения документа, позволяющая идентифицировать ответственного за исполнение документа (задачи) в каждый момент времени жизни документа (процесса).

Эффективно организованная система поиска документа, позволяющая находить документ, обладая минимальной информацией о нём.

Развитая система отчётности по различным статусам и атрибутам документов, позволяющая контролировать движение документов по процессам документооборота и принимать управленческие решения, основываясь на данных из отчётов.

BPMS

BPM - концепция процессного управления организацией, рассматривающая бизнес-процессы как особые ресурсы предприятия, непрерывно адаптируемые к постоянным изменениям, и полагающаяся на такие принципы, как понятность и видимость бизнес-процессов в организации за счёт моделирования бизнес-процессов с использованием формальных нотаций, использования программного обеспечения моделирования, симуляции, мониторинга и анализа бизнес-процессов, возможность динамического перестроения моделей бизнес-процессов силами участников и средствами программных систем[.

Business-Process Management System (BPMS) -технологическое программное обеспечение для поддержки концепции BPM. Среди нотаций моделирования бизнес-процессов в различных решениях используются языки BPMN, IDEF0 и другие. Среди известных нотаций выполнения бизнес-процессов, применяемых в программных системах — BPEL и её диалекты.

Аутсорсинг

Интеграция

Разработка

Интеграция…

Интеграция – объединение КИС, связывающее множество документов и отношений в данных системах

Системная интеграция — комплекс услуг по созданию и сопровождению эффективной с точки зрения взаимодействия используемых информационных систем ИТ-инфраструктуры предприятия.

Системный интегратор в отрасли информационных технологий, реже — в оборонной промышленности, массовых коммуникациях — компания-подрядчик (в американской экономической терминологии — Value Added Reseller), выполняющая функцию головной организации в осуществлении той или иной целевой программы или проекта и извлекающая прибыль из создаваемой добавленной стоимости компании-заказчика. Добавочная стоимость возникает за счёт интеграции продуктов и снижения издержек.

Частная интеграция

Это традиционный и наиболее распространенный в прошлом подход к интеграции систем, заключающийся в создании специализированных интерфейсов к приложениям для каждой отдельной ситуации. Проблема в том, что в результате частной интеграции возникает путаница из связующих звеньев интеграции приложений, что, в конечном счете, заводит ее в тупик.

Интеграция на уровне пользовательских интерфейсов

Идея достаточно оригинальна: приложения могут взаимодействовать так же, как их используют люди, а именно через пользовательский интерфейс.

Простейшая форма такой интеграции под названием screen scraping уже более десяти лет используется для расширения возможностей мэйнфрейм-приложений. Расшифровывая символьные дисплеи, связанные со многими старыми приложениями, инструменты "соскабливания экрана" предлагают быстрый и порой весьма надежный интерфейс для доступа к системам, которые в противном случае были бы полностью закрытыми.

Более новый вариант интеграции на уровне пользовательских интерфейсов «HTML-scraping» основан на результатах работы веб-приложений, выведенных в браузере. Инструменты этого класса идентифицируют компоненты HTML-документа, полученного в результате работы веб-приложения, и предоставляют эти компоненты для повторного использования и интеграции.

Такой подход предлагает доступ к интегрированным системам "только для чтения". Кроме того, этот подход сильно ограничен, так как для остальных (не веб) приложений имитация пользователя в качестве парадигмы интеграции представляется более сложным делом.

Интеграция на уровне данных

Подход хранилищ данных (datawarehouses) подразумевает поддержку данных в специальных хранилищах независимо от бизнес-логики, их породившей. Доступ к хранилищам могут получать различные приложения.

Данные в хранилищах подразумевают использование общей модели данных для всего набора приложений, что препятствует разработчикам и ограничивает гибкость. Даже в четко определенной области, например в HR, приложения самообслуживания, предназначенные для сотрудников, и инструменты поддержки принятия решений, нацеленные на менеджеров, скорее всего будут использовать различные модели данных.
Интеграция на уровне данных сама по себе не предлагает средств для связи с традиционными приложениями. Иногда слои данных таких систем хорошо представлены и документированы, однако чаще для связи с ними требуются специальные коннекторы, что влечет за собой дополнительные расходы, организацию внесения изменений и "привязку" к поставщику.
Компоненты приложений не всегда хорошо взаимодействуют в реальном времени, что является критически важным требованием сегодняшних стратегических бизнес-приложений

Архитектура предприятия (EA)

Содержание

- Общие характеристики понятий «архитектура ИТ» и «архитектура предприятия».
- Эволюция представлений об архитектуре предприятия
- Методики описания АП

Общие характеристики понятий «архитектура ИТ» и «архитектура предприятия»

Что не является архитектурой ИТ

Список продуктов и их поставщиков типа:

- серверная ОС MS Windows 2007,
- СУБД MS SQL,
- серверы на платформе Intel
- телекоммуникационное оборудование Cisco.

Развитие архитектурного подхода

Архитектура

50-е	60-е	70-е
Аппаратные средства	Программно-аппаратный комплекс	Человеко-машинный комплекс

Основные принципы человеко-машинного комплекса:

- ЧМК не только имеет цели и поведение, но может их менять, изменяя при этом внешнюю среду, другие системы и/или саму себя,
- обладает социальным устройством своей внутренней среды,
- может иметь весьма приблизительную границу между внутренней и внешней средой

Развитие архитектурного подхода

Подходы к формулировке архитектуры

- По мнению Gartner Group, подход к формулировке архитектуры ИТ должен основываться на анализе общекорпоративных процессов и переоценке своих бизнес-процессов и поддерживающих их приложений.

- Современные подходы к формулировке понятия архитектуры являются попытками **предоставить язык, понятный и полезный одновременно для бизнес-руководства и для специалистов в области ИТ.**

Основные определения. Система:

Система - это «совокупность взаимодействующих элементов, упорядоченная для достижения одной или нескольких поставленных целей». (ISO/IEC 15288:2002)

Определение распространяется на самые разные по характеру и масштабу системы - от локального устройства (например, мобильного телефона) до целой отрасли (включая ее работников).

Другие стандарты дополнительно указывают, что в составе систем рассматриваются машины, люди, программное обеспечение.

Эберхард Речтин (основатель понятия «системное мышление»): **система**, когда "целое составляет нечто большее, чем механическая сумма составляющих, т.е. система обладает свойствами, которые отсутствуют у составляющих ее элементов"

Основные определения. Предприятие:

Под термином "**Предприятие**" мы будем иметь в виду формальное объединение, не обязательно связанное с коммерческой деятельностью. Это может быть и государственная организация, и общественное, в том числе неформальное, объединение участников, связанных общей целью.

Международные стандарты определяют **Предприятие** как некое образование, состоящее из одной или нескольких организаций либо их частей, разделяющих общую миссию и цели по предоставлению некоторого выхода, например услуги или продукта.

Согласно **более общему определению**, **Предприятие** представляет собой комплексную систему культурных, технологических и процессных компонент, организованных для достижения целей организации. Мы можем применять архитектурные подходы как к целому предприятию, так и к подразделению или даже к отдельной прикладной системе.

АП и организационные изменения

Архитектура предприятия является одним из инструментов организационных изменений предприятия в целом с использованием ИТ.

Гуру в области бизнеса отмечают, что существуют два основных подхода к организационным изменениям.

Первый подход связан с реорганизацией, реинжинирингом процессов,

второй подход – с управлением знаниями.

Архитектура предприятия – это прежде всего управление знаниями.

Включение в архитектуру предприятия представлений о бизнес-архитектуре обеспечивает связь с возможностями оптимизации бизнес-процессов.

Архитектура предприятия частично затрагивает и процессы управления ИТ в организации. В этом плане она дополняет достаточно эффективные методики организации и реорганизации процессов внутри ИТ-службы, такие как ITIL, COBIT и другие

Взаимосвязь АИТ и АП с целями организации

Архитектура информационных технологий и архитектура предприятия в целом являются основным механизмом интерпретации и реализации целей организации через адекватные ИТ-инфраструктуру и системы.

Это достигается через создание определенного количества взаимосвязанных архитектурных представлений.

Элементы архитектуры предприятия

Бизнес - модели

Описывают:

- стратегию организации,
- структуры управления,
- требования,
- ограничения и правила,
- основные бизнес-процессы, включая взаимосвязи и зависимости между ними.

Бизнес-архитектура описывает на уровне предприятия в целом то, **как реализуются основные функции организации, включая организационные и функциональные структуры, роли и ответственности**

Архитектура информации

Определяет ключевые активы, связанные со **структурированной** и **неструктурированной информацией**, требующейся для бизнеса, включая:

- расположение,
- время,
- типы файлов и баз данных;
- других информационных хранилищ.

Архитектура прикладных систем

Описывает те системы, которые обеспечивают:

необходимый **функционал для реализации логики бизнес-процессов** организации.

Технологическая архитектура

Включает описание ИТ-сервисов, которые требуются для реализации перечисленных выше областей архитектуры.

Логические модели ИТ-сервисов построены в абстрактной, технологически независимой форме и оставляют свободу для оптимального выбора конкретных технологий.

Физические модели определяются:

- технологиями,
- аппаратными платформами
- программными платформами, выбранными для реализации ИТ-сервисов.

Итак:

- Архитектура системы состоит из нескольких компонент, внешних свойств и интерфейсов, связей и накладываемых ограничений, а также архитектуры этих внутренних компонент".
- Такое рекурсивное определение удобно тем, что является достаточно общим, применимым практически к любой системе, а не обязательно только к системе, использующей информационные технологии, и при этом позволяет ограничить степень детализации на нужном уровне.
- Упоминание внутренних компонент используется для отражения того факта, что "**хорошая**" **архитектура позволяет обеспечить повторное использование или модернизацию/замену таких внутренних компонент без изменения внешней охватывающей системы.** Итеративное, иерархическое построение архитектуры позволяет решить и еще одну важную задачу – облегчить ее восприятие человеком.
- Оптимальным числом элементов на отдельном уровне любой схемы абстракции или в каком-либо списке является 7 плюс или минус 2 объекта. Именно поэтому большинство подходов к описанию архитектуры включает в себя ее разбиение на предметные области (или представления), общее количество которых как раз и находится в этом диапазоне.
- Примерами таких предметных областей являются архитектура прикладных систем, архитектура данных, технологическая архитектура и т.д. Старый, как мир, принцип "разделяй и властвуй" как нельзя лучше подходит для того, чтобы справляться с объективной сложностью корпоративных информационных систем. При этом такое разбиение позволяет рабочим группам, специализирующимся на различных предметных областях, работать параллельно, что делает проблему осознаваемой с интеллектуальной точки зрения.

Итак:

- Giga Group :

  "в индустрии ИТ нет одного, единственно правильного стандарта на определение архитектуры ИТ, поэтому общие соглашения внутри организации важнее теоретической точности".

Итак, важна не столько академическая точность определения того, что такое архитектура ИТ, сколько реальный процесс использования архитектурных принципов.

Основные определения. Архитектура:

Gardner Group:

- общий план или концепция, используемая для создания системы, такой как здание или информационная система, или "абстрактное описание системы, ее структуры, компонентов и их взаимосвязей";
- семейство руководящих принципов, концепций, правил, шаблонов, интерфейсов и стандартов, используемых при построении совокупности информационных технологий предприятия.

Основные определения. Архитектура:

Giga Group:

- Архитектура – это инвестиция в стандарты процессов, технологий и интерфейсов в целях улучшения возможностей организаций и уменьшения стоимости разработки и сопровождения информационных систем. Преимущества инвестиций в архитектуру распространяются на несколько проектов сразу, но не все эти проекты могут быть известны в момент разработки архитектуры;

- Корпоративная архитектура ИТ – это видение, принципы и стандарты, которыми организации руководствуются при разработке и внедрении технологий

Основные определения. Архитектура:

Глоссарий ФОСТАС:

1) многоаспектное описание или план задуманной или развиваемой системы на уровне ее компонентов, детализированное в достаточной мере для руководства ее воплощением, а также принципы и руководящие материалы, определяющие руководство конструированием и развитием системы во времени;

2) структура существующей системы как совокупность ее компонентов и их взаимосвязей.

3 измерения

Рассмотрим теперь более подробно, какие отдельные понятия в рамках представления об "архитектуре" существуют, и как они связаны между собой. Можно выделить, по крайней мере, 3 различных "измерения" в данном континууме определений:

- иерархия архитектур различных организационных систем;

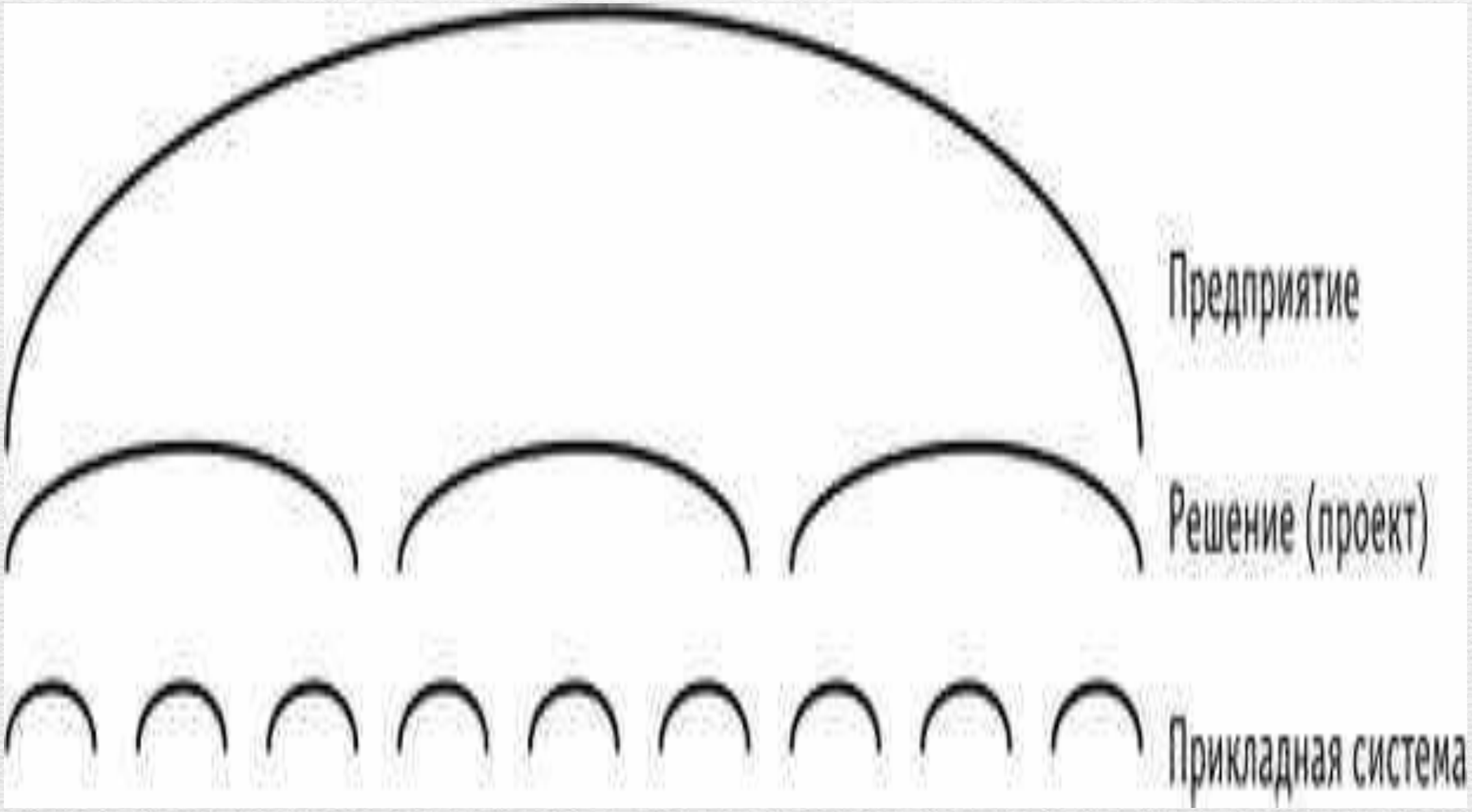
- соотношения между объективной реальностью и субъективным восприятием;

- соотношения между общесистемной архитектурой и частными архитектурами.

- Точно так же, как и в строительстве, существуют различные уровни архитектуры (план города, план застройки района, планы отдельных зданий), требуется дальнейшая детализация высокоуровневых определений и классификация архитектуры бизнеса и информационных технологий на различных уровнях.

- Таким образом, мы можем говорить об архитектуре предприятия в целом, архитектуре уровня отдельных проектов или семейства продуктов, можем говорить об архитектуре отдельной прикладной системы. И в первом, и во втором, и в третьем случае – это все архитектуры. Вопрос заключается в декомпозиции сложных систем и в том, на каком уровне принимаются те или иные архитектурные решения.

Уровни принятия архитектурных решений



Архитектура предприятия

Определяет общую структуру и функции систем (бизнес и ИТ) в рамках всей организации в целом (включая партнеров и другие организации, формирующие так называемое "расширенное предприятие") и обеспечивает общую рамочную модель (framework), стандарты и руководства для архитектуры уровня отдельных проектов.

Общее видение, обеспечиваемое архитектурой предприятия, создает возможность единого проектирования систем, адекватных, с точки зрения обеспечения потребностей организации, и способных к взаимодействию и интеграции там, где это необходимо.

Архитектура уровня отдельных проектов

Определяет структуру и функции систем (бизнес и ИТ) на уровне проектов и программ (совокупностей проектов), но в контексте всей организации в целом, т.е. не в изолированном рассмотрении индивидуальных систем.

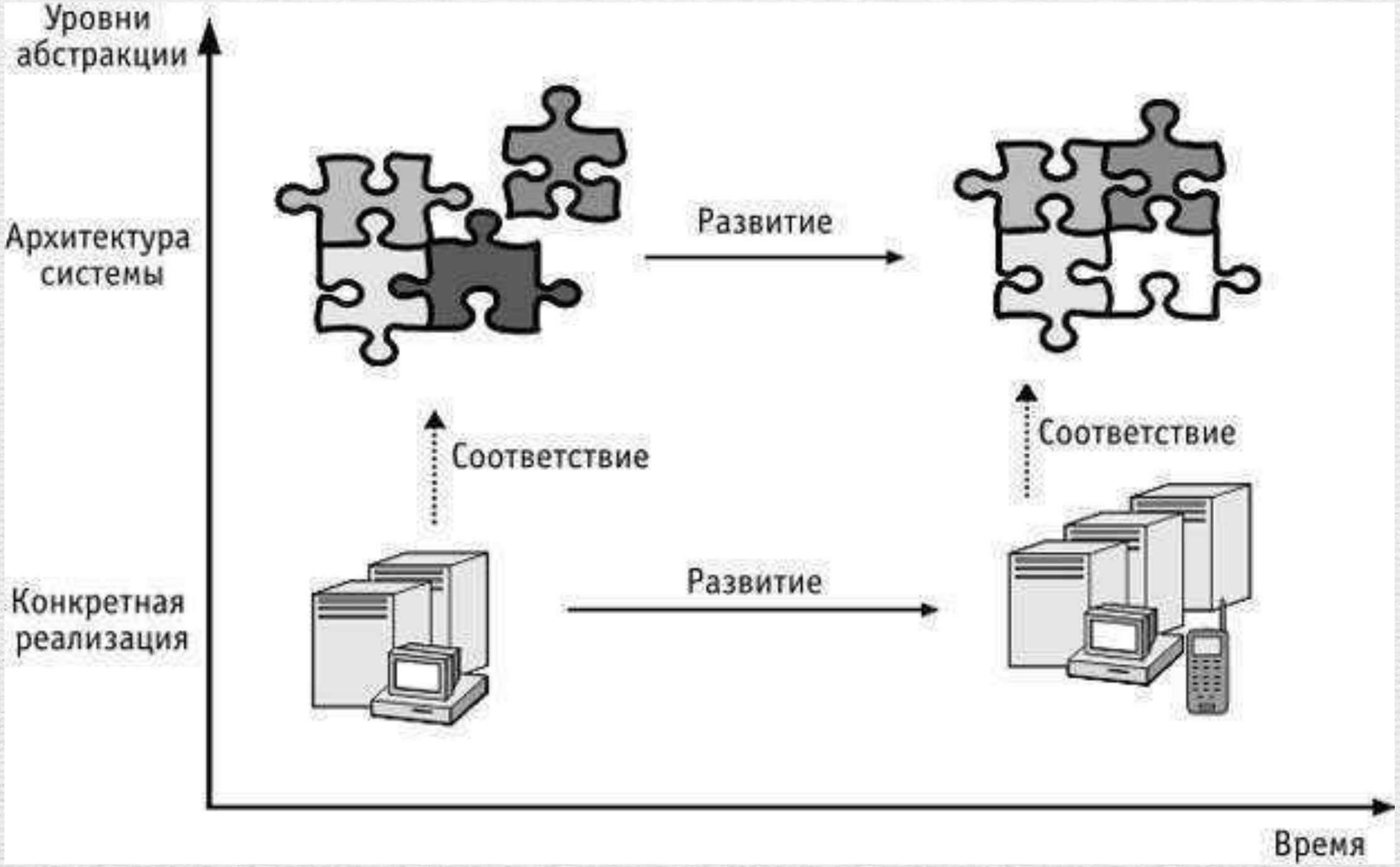
Архитектура уровня отдельных проектов детализирует, соответствует и существует в рамках архитектуры предприятия.

Архитектура прикладных систем

Определяет структуру и функции приложений, которые разрабатываются с целью обеспечения требуемой функциональности.

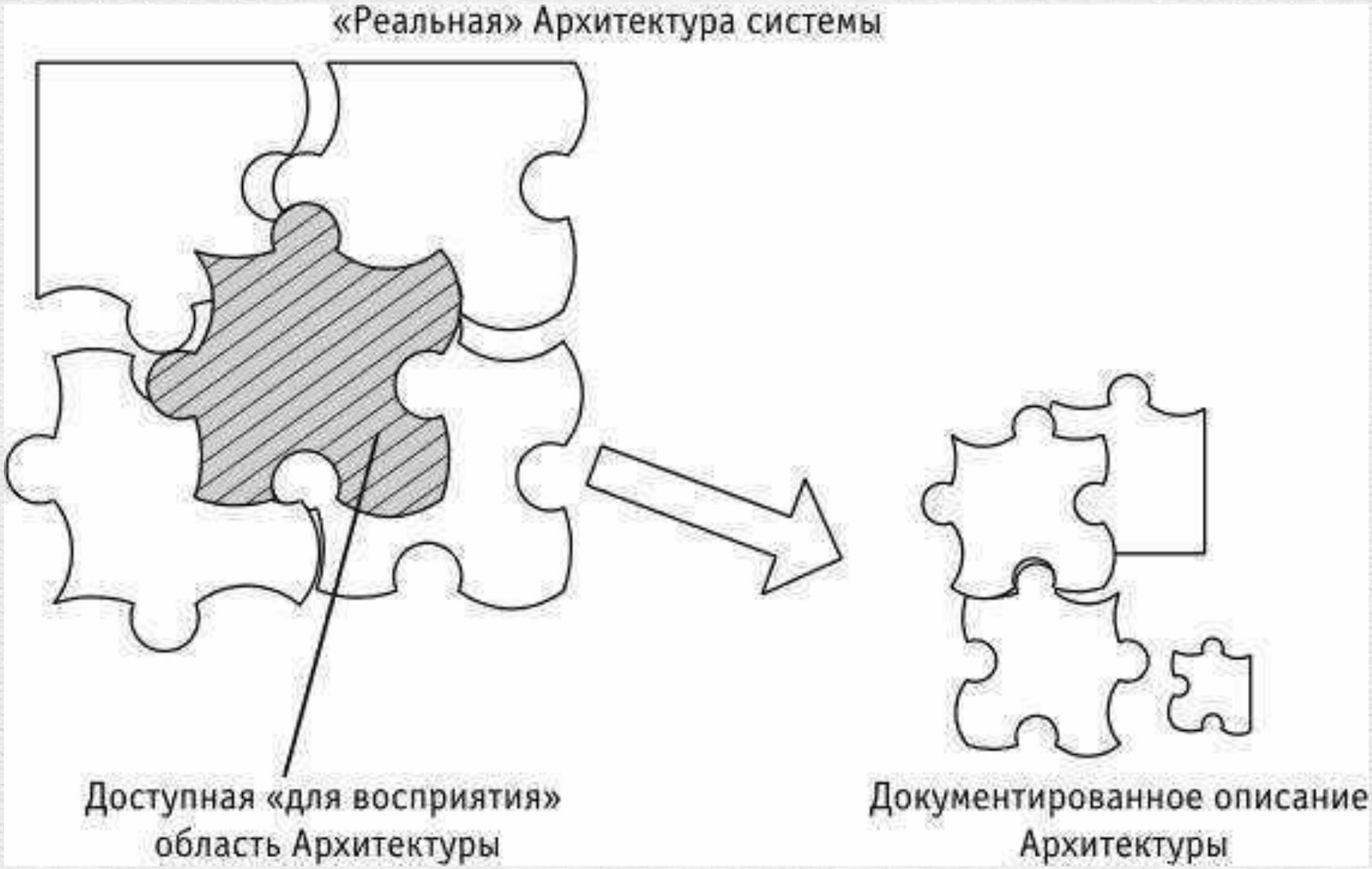
Некоторые элементы этой архитектуры могут быть определены на уровне архитектуры предприятия или архитектуры отдельных проектов (в форме стандартов и руководств) в целях использования лучшей практики и соответствия принципам всей архитектуры в целом.

Гносеологический аспект проблемы: 1. Архитектура как модель ИС



Гносеологический аспект проблемы:

2. Описание архитектуры как проекция реальности



Вывод:

- Архитектура системы (внешняя область) по определению является бесконечно сложным, глубоким и неявным понятием. Только часть этого общего понятия, которая в принципе доступна для восприятия архитекторами, может быть переведена в явную документируемую форму – модель или набор моделей с неизбежными упрощениями, ограничениями и субъективными искажениями. Такая проекция и представляет собой "описание архитектуры". Поэтому при использовании подобных описаний при проектировании систем необходимо всегда помнить об их неизбежной неполноте. При этом совершенно уместным является остроумное замечание о том, что "все модели являются, в общем-то, неверными, но некоторые из них при этом являются полезными".
- **Таким образом, ИТ-архитектура существует независимо от предпринимаемых в организации проектов по ее описанию, упорядочиванию и развитию.**
- Отсутствие решений в области ИТ или бессистемное их принятие на практике приводят к появлению "зоопарка" аппаратных средств и приложений, напоминающих спонтанную застройку в условиях отсутствия градостроительных планов, появление вагончиков и "шанхаев" со всеми вытекающими последствиями.

Определение Архитектуры: Рамочная модель разработки архитектуры по IEEE 1471

Система обладает некоторой архитектурой, которая может быть определенным образом описана с различных точек зрения в зависимости от интереса тех людей (участников), которые рассматривают архитектуру системы.

Каждой точке зрения на архитектуру системы соответствует определенное представление, основу которого составляет некоторый набор моделей.

Подход системного проектирования

- Выделяются такие подмножества, как системная архитектура (архитектура систем – System Architecture) и программная архитектура (архитектура программного обеспечения – Software Architecture).

Уровни архитектур

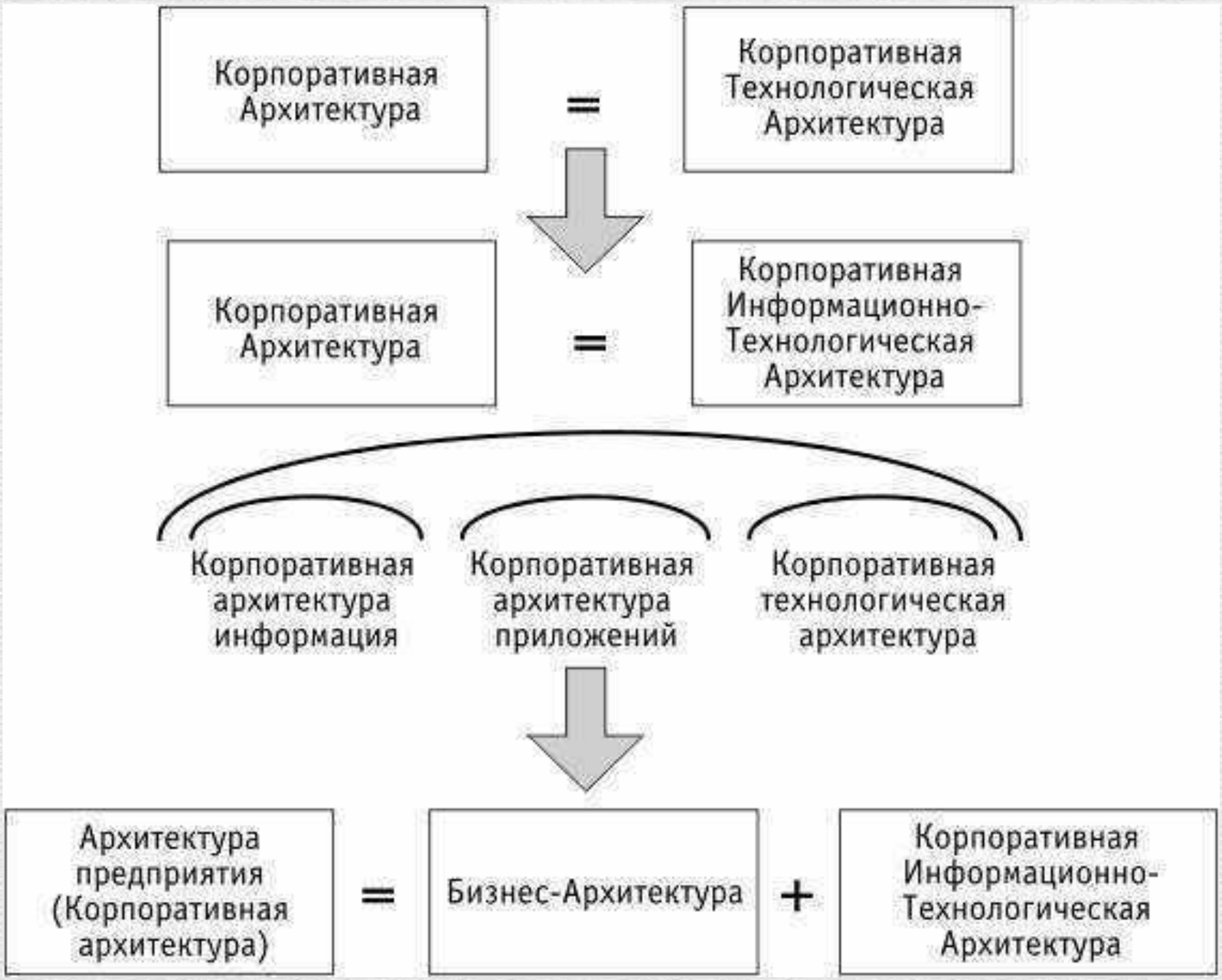
- **концептуальная архитектура** определяет компоненты системы и их назначения, обычно в неформальном виде. Это представление часто используется для обсуждения с нетехническими специалистами, такими как руководство, бизнес-менеджеры и конечные пользователи функциональных характеристик системы (что система должна уметь делать, в основном, с точки зрения конечного пользователя);

- **логическая архитектура** выделяет, прежде всего, вопросы взаимодействия компонент системы, интерфейсы и используемые протоколы. Это представление позволяет эффективно организовать параллельную разработку;

- **физическая реализация**, которая описывает привязку к конкретным узлам размещения, типам оборудования, характеристикам окружения, таким как, например, используемые операционные системы и т.п.

ЭВОЛЮЦИЯ ПРЕДСТАВЛЕНИЙ ОБ АРХИТЕКТУРЕ ПРЕДПРИЯТИЯ

Эволюция представлений об архитектуре предприятия



Корпоративная архитектура = корпоративная технологическая архитектура

Работы по описанию архитектуры были сосредоточены на формировании технологических стандартов и принципов, включая проведение инвентаризации различных технологий, используемых в организации.

Такой подход позволяет добиться **определенных частных выгод**, связанных прежде всего с уменьшением стоимости закупок и эксплуатации информационных систем и уменьшением затрат на разработку приложений и обучение персонала.

Однако он является **заведомо ограниченным**, так как не подразумевает ориентацию на решение бизнес - задач, таких как, например, формирование единых в масштабе компании данных по клиентам.

Корпоративная архитектура = корпоративная информационно-технологическая архитектура

Основное направление работ состоит в совместном использовании общих данных, исключении дублирования бизнес-функций, координации управления пользователями, ресурсами, информационной безопасностью за счет улучшений в управлении портфелем прикладных систем.

Корпоративная информационно-технологическая архитектура масштаба предприятия описывает то, как компоненты информационной системы связаны между собой. Такой подход обеспечивает более эффективное взаимодействие различных структурных подразделений организации:

- совместный доступ к информации различных подразделений, а также внешних организаций (клиентов, партнеров, поставщиков);

- уменьшение дублирования с точки зрения параллельной реализации близких по функционалу прикладных систем для различных бизнес - подразделений;

- решение проблем, которые затрагивают интересы нескольких подразделений, например, интеграция и взаимодействие информационных систем.

Получаемые преимущества носят, в основном, тактический характер

Корпоративная архитектура (АП) = бизнес-архитектура + корпоративная информационно-технологическая архитектура

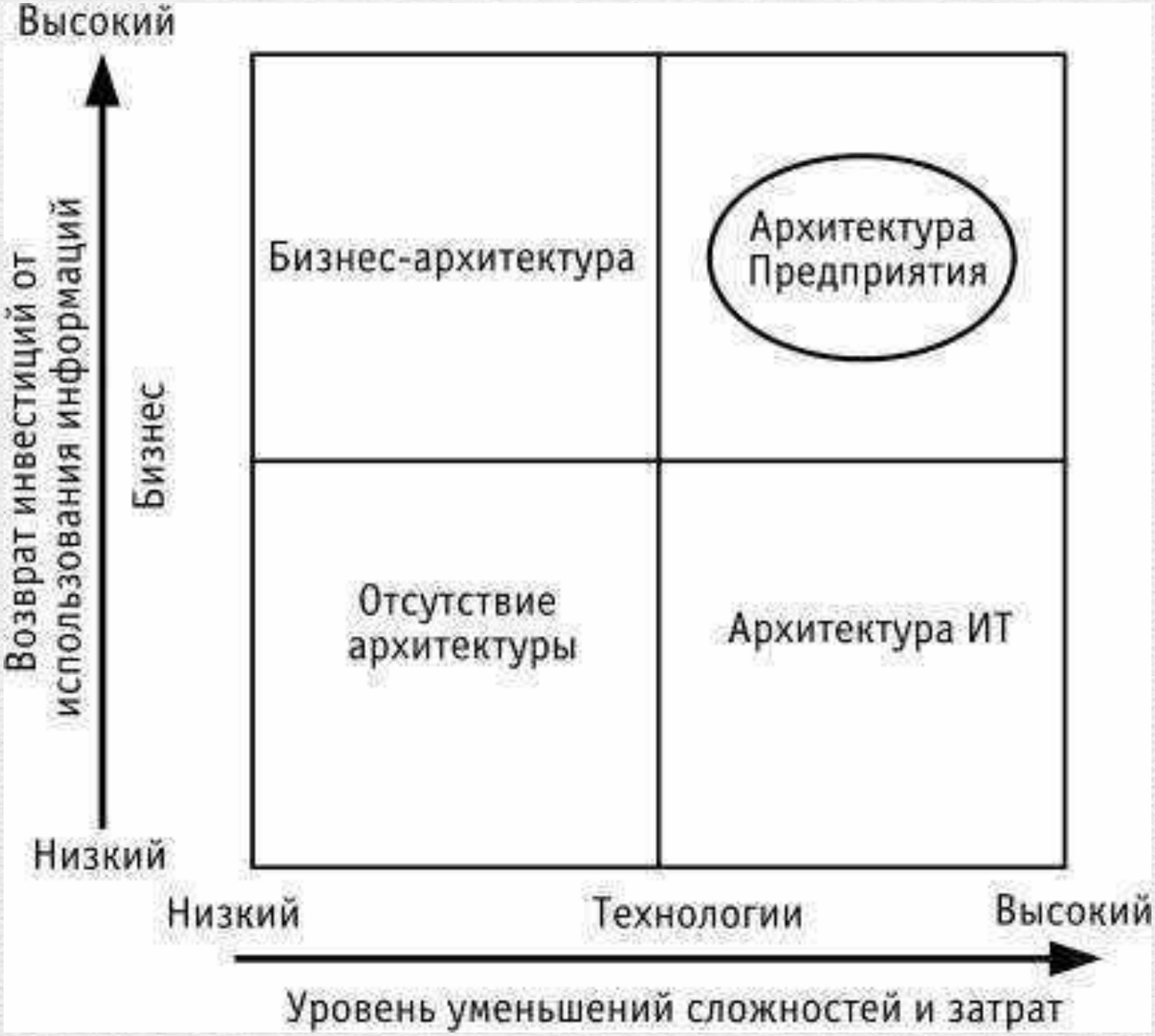
Архитектура предприятия (Enterprise Architecture) объединяет корпоративную ИТ - архитектуру масштаба предприятия с бизнес - архитектурой и позволяет обеспечить достижение стратегических целей предприятия.

Преимуществами такого включения бизнес - архитектуры в контекст рассмотрения целостной архитектуры предприятия являются большая способность организации к изменениям или динамичность и синхронизация возможностей информационных технологий с бизнес -стратегией:

- обеспечение вариативности бизнес -стратегии за счет возможности изменений в обеспечивающих процессах и технологических решениях;

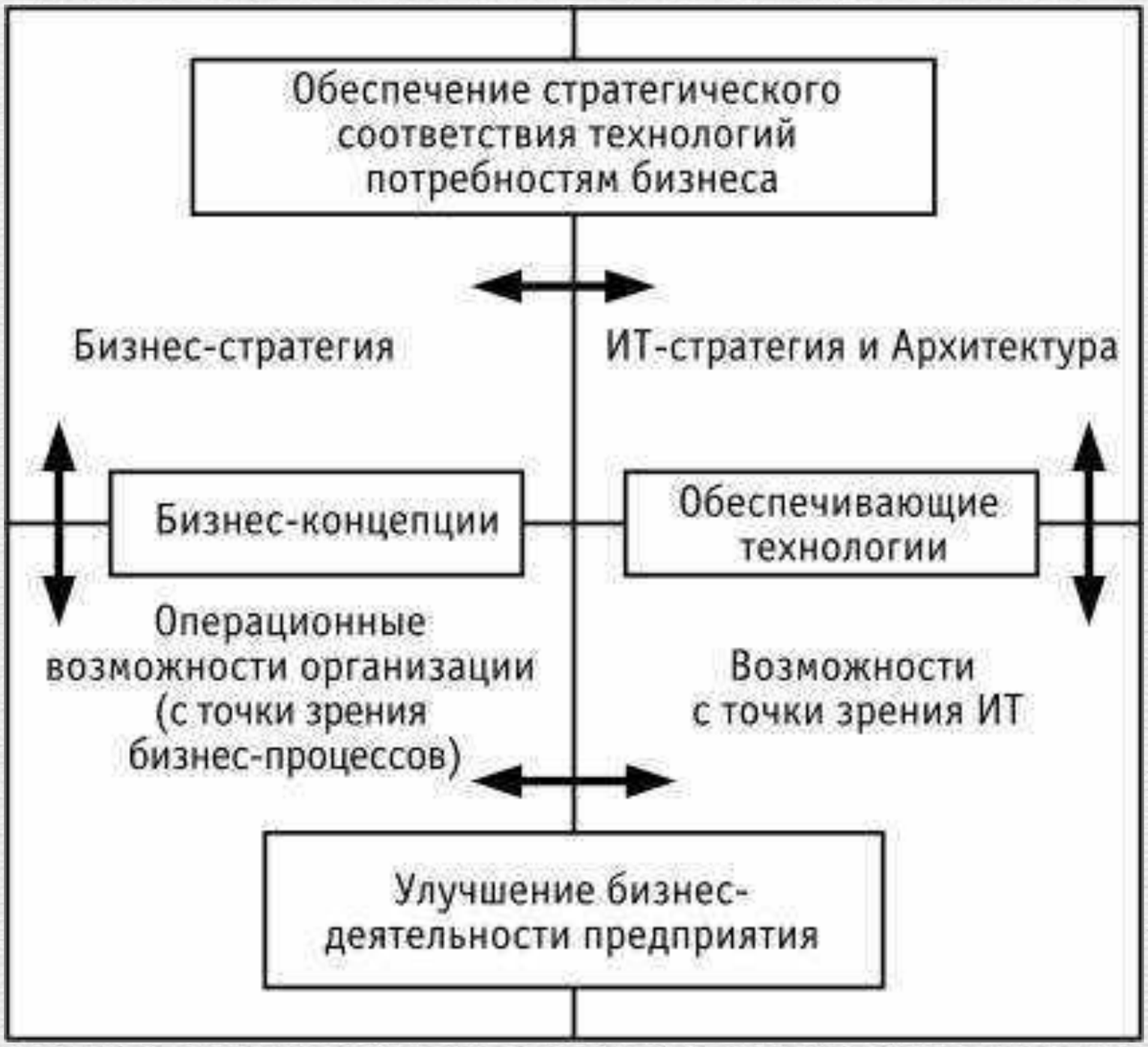
- лучшие перспективы, с точки зрения использования возможностей информационных технологий по формированию самой бизнес -стратегии.

Позиционирование понятия АП



Расширение представлений об "архитектуре предприятия" и дополнительные преимущества: целостный подход, взгляд на организацию в целом

Интегрированная концепция АП: "...концепция архитектуры предприятия – это план реализации миссии организации через оптимальное выполнение своих ключевых бизнес-процессов в условиях формирования эффективной инфраструктуры информационных технологий



Формальное определение АП

- Формальное описание архитектуры предприятия впервые было сформулировано в стандарте ISO 15704.

- Идея состояла в том, чтобы разработать максимально общую, так называемую эталонную (reference) модель архитектуры предприятия, которая охватывала бы дополнительно процесс развития предприятия во времени как проект, а также учитывала бы роль человеческого фактора.

- Архитектуры отдельных подсистем, в том числе ИТ-системы предприятия, могут быть тогда разработаны как специфические уточнения такой общей модели. Разработанная как приложение к данному стандарту, такая общая эталонная модель архитектуры получила название GERAM (Generalized Enterprise Reference Architecture and Methodology).

Контекст АП

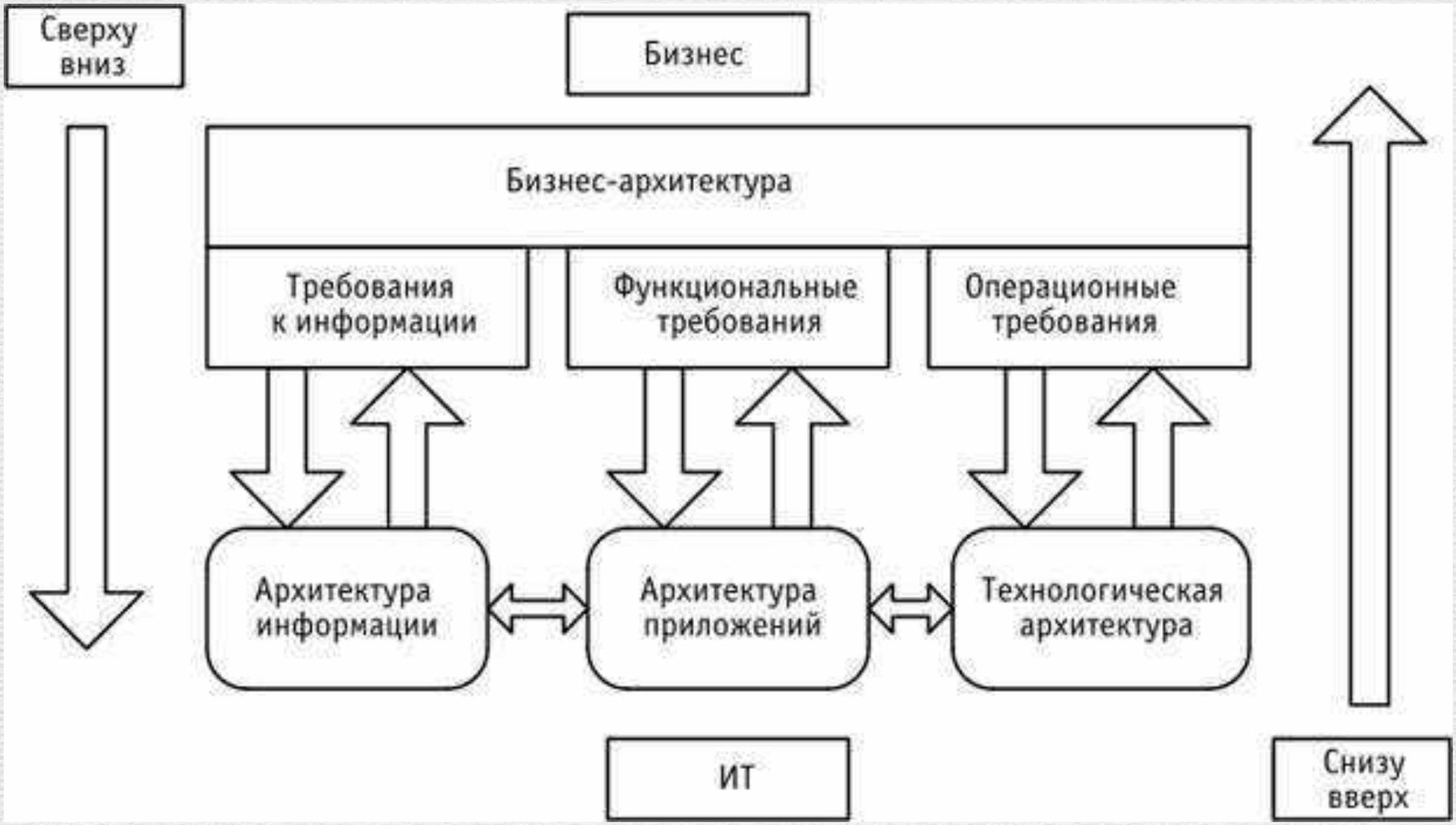
Эффективная архитектура предприятия должна обеспечивать целостный и всеобъемлющий взгляд на следующие аспекты:

- бизнес, включая движущие силы (ключевые факторы), видение и стратегию;
- организационные структуры и сервисы, которые требуются для реализации этого видения и стратегии;
- информация, системы и технологии, которые требуются для эффективной реализации этих сервисов.

Разработка архитектуры предприятия является сложным процессом, поскольку эта концепция включает в себя информационные системы в контексте всей организации.

Для упрощения процесса архитектура предприятия обычно структурируется за счет рассмотрения бизнеса и систем в виде набора компонент (или сервисов) с взаимосвязями между ними, при этом опускаются определенные детали отдельных компонент.

Связь требований бизнеса и различных областей архитектуры ИТ



Разработка архитектуры предприятия

- Разработка архитектуры предприятия включает в себя компоненты, связанные с функциональной архитектурой (бизнесом), информационными технологиями и управлением архитектурным процессом.

- Диаграмма отражает подход NASCIO (Национальной Ассоциации CIO США), на то, как различные компоненты взаимодействуют и влияют друг на друга.

МЕТОДИКИ ОПИСАНИЯ АП

Методики описания АП

Архитектура предприятия является целостным описанием ключевых стратегий организации, связанных с бизнесом, информацией, прикладными системами и технологиями, а также их влиянием на функции и бизнес-процессы организации.

Имеется множество методик описания архитектуры, и все они разбивают архитектуру предприятия на различное количество моделей и определений, которые относятся к таким областям, как **бизнес, информация, прикладные системы, технологическая инфраструктура**.

- методики, опубликованные аналитическими компаниями, такими как Gartner, Giga Group, META Group и другими;
- модель Захмана;
- методика TOGAF;
- методика POSIX 1003.23, которая основывается на разработках компании Cap Gemini, переданных для публичного использования в 1996 году.

Модель Захмана

- 1987 год - появились первая статья Джона Захмана,

- 1992 год - вторая (в соавторстве с Дж. Сова) статья Джона Захмана (1. Sowa J. F., Zachman J. A. Extending and Formalizing the Framework for Information System Architecture // IBM Systems Journal. 1992. V. 31. № 3.

предложен вариант обобщенной схемы или структуры (framework, или «фреймвок») для описания и анализа архитектуры: формально (по названию) еще архитектуры ИС, но по содержанию - уже предприятия.

Матрица Захмана

	ПОЧЕМУ? Мотивация	КТО? Люди	ЧТО? Данные	КАК? Функции	ГДЕ? Место	КОГДА? Время
Контекст (рамки) — *Стратег*	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса (концептуальная) — *Владелец*	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели	Модели бизнес-процессов	Система логистики	Газовый график работ
Системная модель (логическая) — *Проектировщик*	Модель правил обработки событий	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Распределенная архитектура	Структура обработки
Технологическая модель (физическая) — *Разработчик*	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление (реализация) — *Программист*	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Бизнес-архитектура

Архитектура правил | **Организа-ционная архитектура** | **Архитектура данных** | **Архитектура приложений** | **Инфра-структура** | **ВременнАя архитектура**

Этапы создания информационной системы

взгляд (видение)

Матрица Захмана

- **Ряд 1 – Контекст**
  Внешние требования и движущие факторы
  Моделирование бизнес-функций

- **Ряд 2 – Модель бизнеса**
  Модели бизнес-процессов

- **Ряд 3 – Системная модель**
  Логические модели

- **Ряд 4 – Технологическая модель**
  Физические модели/ Определение и разработка решения

- **Ряд 5 – Детальное представление**
  Как выстроено/ Внедрение

- **Ряд 6 – Работающая организация**
  Функционирование организации Оценка

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обрабстки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Правила

- ***Правило 1:***

Нет заданного порядка расположения колонок

- ***Правило 2:***

Каждая колонка имеет в основе простую, базовую модель

- ***Правило 3:***

Базовая модель в каждой колонке уникальна

- ***Правило 4:***

Каждый ряд представляет различную точку зрения (взгляд на систему)

- ***Правило 5:***

Каждая клетка уникальна

- ***Правило 6:***

Совокупность клеток одного ряда формирует полное описание системы с соответствующей точки зрения

Матрица Захмана – Ряд 1
Контекст/Уровень Владельца процесса

- **Мотивация/Почему**

Цели бизнеса, задачи и результаты деятельности

Меры, относящиеся к каждой функции

- **Данные/Что**

Классы данных верхнего уровня, связанные с каждой функцией

- **Люди/Кто**

Держатели акций, имеющие отношение к каждой функции

- **Функции/Как**

Бизнес-функции верхнего уровня

- **Место/Где**

Местоположения, связанные с каждой функцией

- **Время/Когда**

Циклы и события, относящиеся к каждой функции

- **Внешние требования и движущие факторы**
- **Моделирование бизнес-функций**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Матрица Захмана – Ряд 2
Модель организации/Уровень Аналитика

- **Мотивация/Почему**

Политики, процедуры и стандарты для каждого процесса

- **Люди/Кто**

Роли и ответственность в каждом процессе

- **Данные/Что**

Информация о бизнесе

- **Функции/Как**

Бизнес-процессы

- **Место/Где**

Местоположения, связанные с каждым процессом

- **Время/Когда**

Действия в рамках каждого процесса и последовательность интеграции и оптимизации процессов

- **Модели бизнес-процессов**
- **Окружение бизнес-функций**
- **Ислючение пересечения и дублирования функций**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Матрица Захмана – Ряд 3
Системная модель/Уровень Архитектора

- **Мотивация/Почему**

Политики, процедуры и стандарты в рамках модели бизнес-правил

- **Люди/Кто**

Логическое представление прав доступа в зависимости от роли и ответственности

- **Данные/Что**

Логические модели данных и взаимосвязи между данными

- **Функции/Как**

Логическое представление информационных систем и их взаимосвязей

- **Место/Где**

Логическое представление распределения системной архитектуры по местам

- **Время/Когда**

Логические события и их следствия в рамках бизнес-событий и их следствий

- **Логические модели**
- **Управление проектами**
- **Определение требований**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Матрица Захмана – Ряд 4
Технологическая модель/Уровень Проектировщика

- **Мотивация/Почему**

Бизнес-правила в рамках стандартов информационных систем

- **Люди/Кто**

Спецификация прав доступа в рамках выбранных платформ и технологий

- **Данные/Что**

Требования к типам систем управления базами данных в рамках логических моделей данных

- **Функции/Как**

Спецификация приложений, функционирующих на основе выбранных технологических платформ

- **Место/Где**

Спецификация сетевых устройств и их взаимос в пределах физических границ системы

- **Время/Когда**

Спецификация «переключателей» событий в системе в рамках выбранных платформ и технологий

- **Физические модели**
- **Управление технологиями**
- **Выбор решений и их реализация**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Матрица Захмана – Ряд 5
Как выстроено/Уровень Программиста

- **Мотивация/Почему**

Бизнес-правила в рамках выбранных технологических стандартов

- **Люди/Кто**

Права доступа, созданные для контроля доступа к выбранным платформам и технологиям

- **Данные/Что**

Определение данных в рамках физических моделей данных

- **Функции/Как**

Программы, написанные для работы на основе выбранных технологических платформ

- **Место/Где**

Сетевые устройства, формируемые для соответствия спецификациям узлов

- **Время/Когда**

Программирование временных промежутков для упорядочивания последовательности действий в рамках выбранных платформ и технологий

- **Как выстроено**
- **Управление конфигурацией**
- **Внедрение**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Матрица Захмана – Ряд 6
Работа организации/Уровень Пользователя

- **Мотивация/Почему**

Использование возможностей специальных технологий в рамках стандартов

- **Люди/Кто**

Сотрудники и ключевые акционеры, работающие с системой в рамках своих ролей и уровня ответственности

- **Данные/Что**

Внесение данных и их хранение в активных базах данных

- **Функции/Как**

Функционирующие компьютерные инструкции

- **Место/Где**

Отправка и получение сообщений

- **Время/Когда**

Установление временных промежутков для задания последовательности событий

- **Работа организации**
- **Управление операциями**
- **Оценка**

	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса	Важные для бизнеса организации	Вещи, значимые для бизнеса	Основные бизнес-процессы	География бизнеса	События и периоды, важные для бизнеса
Модель бизнеса	Бизнес план, частные цели и стратегии	Модели потоков работ	Семантические модели Бизнес-сущности и их связи	Модели бизнес-процессов	Система логистики	Базовый график работ
Системная модель	Модель бизнес-правил	Архитектура пользовательского интерфейса	Концептуальная модель данных	Архитектура приложений	Архитектура распределенной системы	Структура обработки событий
Технологическая модель	Модель правил обработки событий	Архитектура представления	Физическая модель данных	Архитектура программно-аппаратной системы	Технологическая архитектура	Структура циклов управления
Детальное представление	Спецификации правил работы системы	Спецификации ролей и прав доступа	Спецификации форматов данных	Код программных компонентов	Спецификации архитектуры сети	Спецификации обработки событий и прерываний
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Метод Захмана
Концептуально важные идеи:

- рекурсивность логики формирования моделей и метамоделей на основе одной обобщенной схемы;

- использование репозитория архитектурной информации для работы с разными моделями и их состояниями;

- управление архитектурой и изменениями предприятия на основе репозитория.

Метод Захмана позволяет:

- концентрироваться на отдельных аспектах предприятия или его конкретной системы и в то же время не терять взгляда на него как на целое;

- использовать одну понятную и бизнес-руководителям, и компьютерным специалистам концептуальную основу для совместных обсуждений и планирования;

- планировать соответствие друг другу описаний-ячеек, обеспечивая тем самым согласование бизнеса и ИТ;

- сохранять при этом независимость от какого-либо программного продукта (инструмента) с его формализмами, особенностями и ограничениями.

Метод Спивака. EAP (Enterprise Architecture Planning)

Метод Спивака. EAP (Enterprise Architecture Planning)

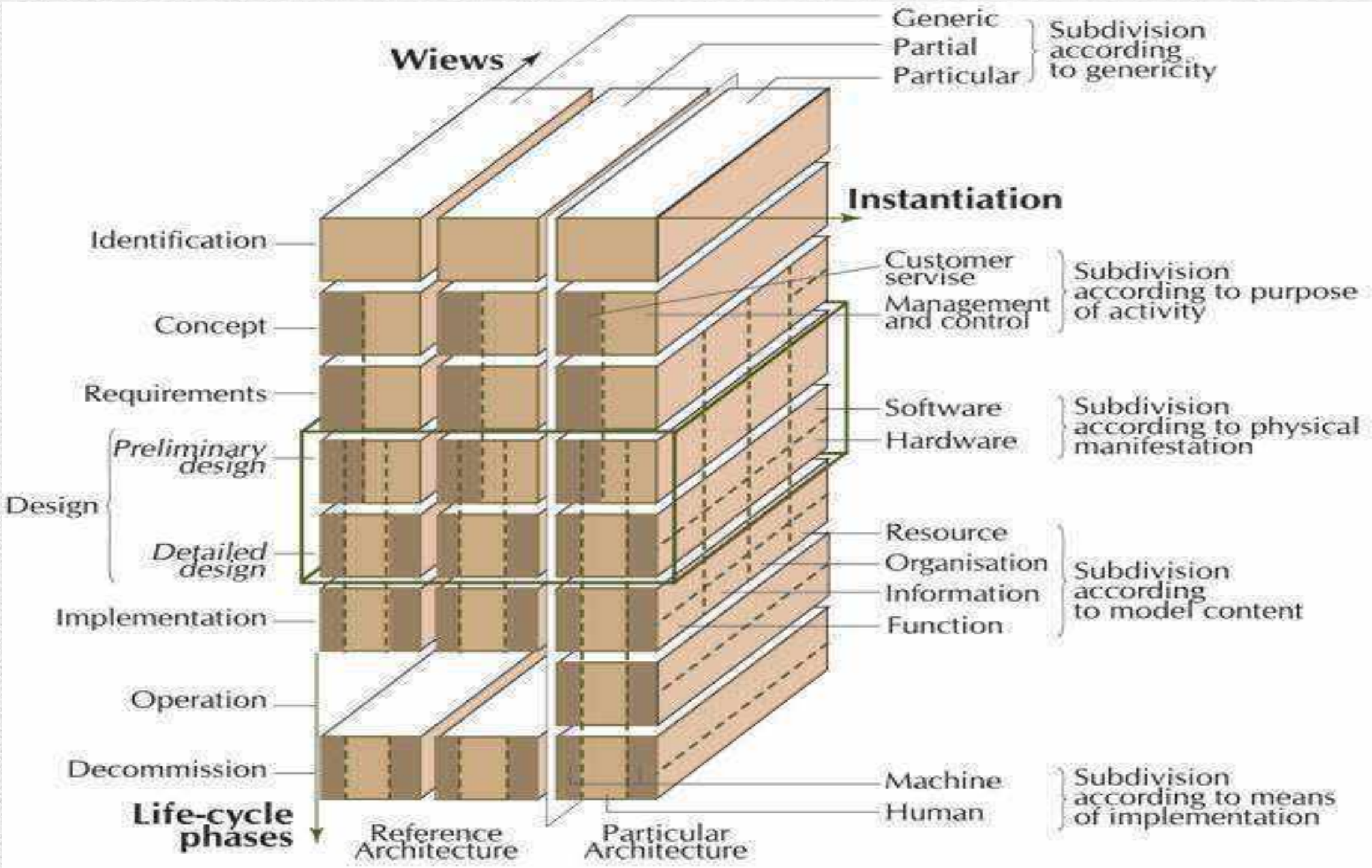
Процесс EAP Спивака ориентирован на планирование ИС. При этом EAP представлялся как процесс, определяемый бизнесом или данными, поскольку:

- основанием для формируемых архитектур являлась устойчивая бизнес-модель;
- данные (понимаемые как бизнес-информация, описываемая в стиле «сущности-связи») определялись до определения приложений;
- зависимости в данных определяли последовательность внедрения прикладных систем.

GERAM (Generalised Enterprise Reference Architecture and Methodology)

GERAM (Generalised Enterprise Reference Architecture and Methodology) - Обобщенная референсная архитектура и методология предприятия.
GERAM включено в качестве приложения в действующий базовый стандарт - ISO 15704:2000 «Requirements for enterprise-reference architectures and methodologies».

Обобщенная схема GERAM



Обобщенная схема GERAM

- четыре группы аспектов архитектуры предприятия, названных представлениями (Views) - типы моделей («функции», «данные», «ресурсы», «организация», что уже, чем шесть аспектов Захмана), назначения (может быть ассоциировано со столбцом «ЗАЧЕМ» Захмана), реализации и «физические представления» (аппаратура, ПО) и возможность определять дополнительные аспекты;

- описание всех аспектов или какой-то их части на каждой из семи или восьми фаз формирования архитектуры и функционирования предприятия;

- конкретизацию модели архитектуры на трех уровнях - обобщенном, уровне частичных моделей (они же повторно используемые референсные, reference) и конкретных моделей.

ISO 15704:2000. Industrial automation systems - Requirements for enterprise-reference architectures and methodologies.

Стандарт **предназначен** для определения требований к архитектурам и методологиям предприятия (enterprise-reference architectures and methodologies).

Стандарт **нацелен** на решение задач трех типов: создание предприятия, его реструктуризация и инкрементальные изменения.

Стандарт **ориентирован** как на людей, так и на технологии.

ISO 15704:2000.

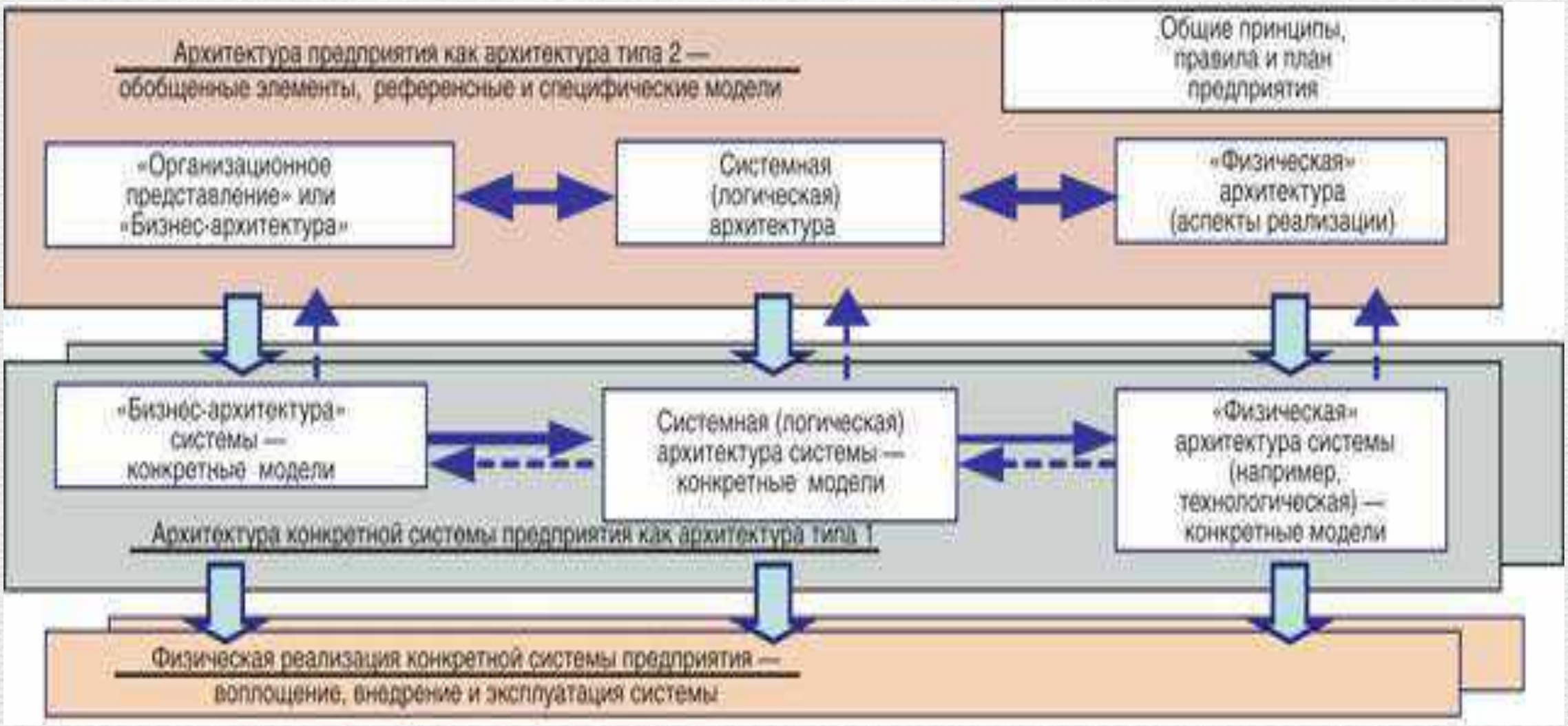
Архитектура - это описание (модель) основной компоновки и взаимодействия частей системы (будь то физический либо абстрактный объект или сущность).

Примечание. Есть два и только два типа архитектур, относящихся к интеграции предприятий:

- архитектуры систем (иногда называемые архитектурами типа 1), которые имеют дело с конструкцией некоторой системы, например, компьютерной системы управления как части всеобъемлющей системы интеграции предприятия;

- архитектуры (планы/проекты) предприятия (иногда называемые архитектурами типа 2), которые имеют дело с таким проектом, как интеграция всего предприятия, или с иной программой его развития».

Связи АП, архитектур конкретных систем предприятия и их физической реализации



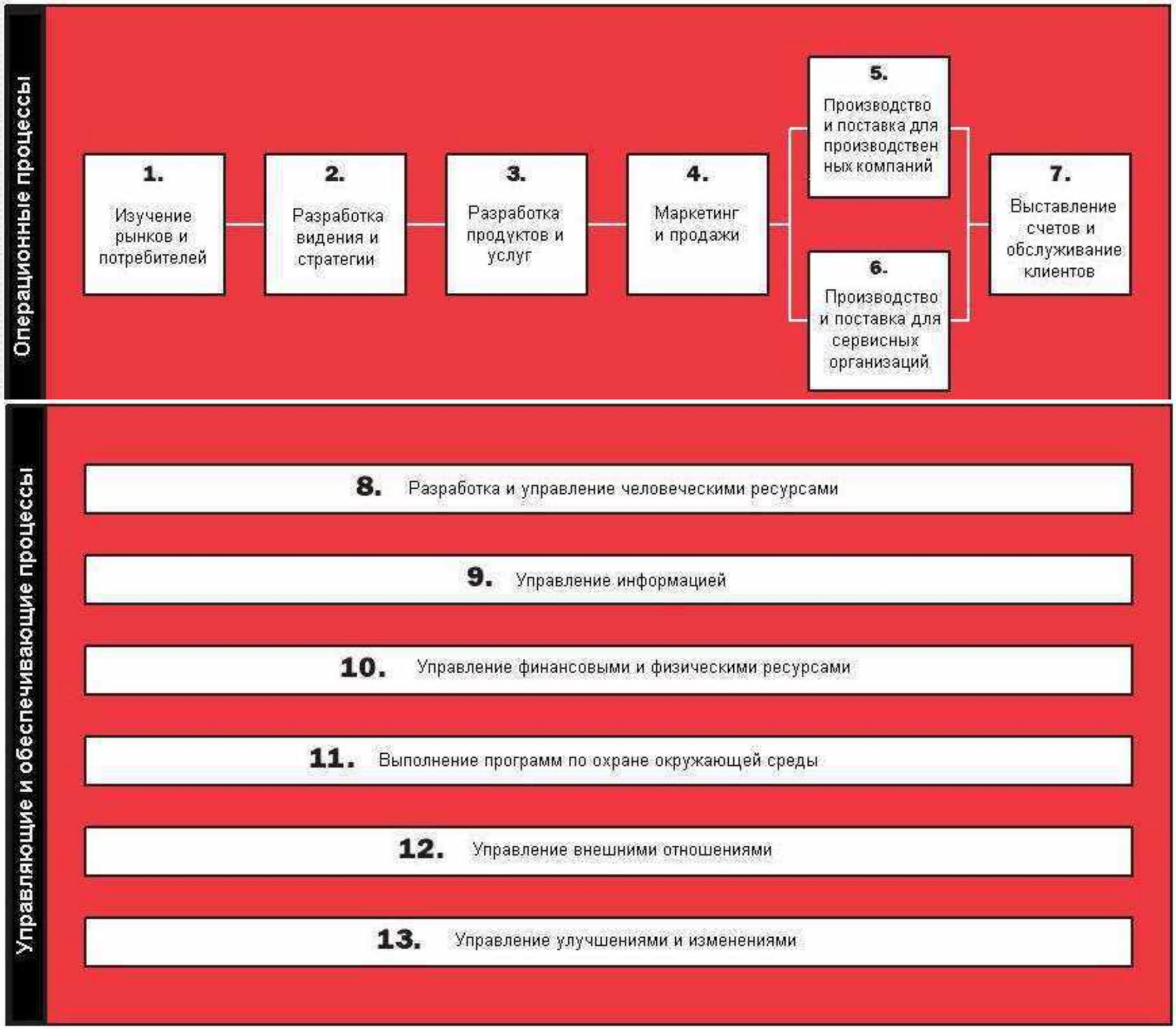
Референсные модели в стандарте ISO 15704:2000

АП, основанные на моделях, согласно стандарту должны поддерживать идею «многократно применимых референсных моделей» (reusable reference models). Такие модели вводятся стандартом для того, чтобы за счет использования концепций, применимых на многих предприятиях, можно было повышать эффективность моделирования. При этом указывается, что референсные модели требуют адаптации к конкретному предприятию, то есть не рассматриваются как эталон для непосредственного применения. Предусматривается также возможность создания специфических/детальных (particular) моделей, которые описывают некоторую сущность конкретного предприятия или его части.

Референтные модели

Референтная модель - модель эффективного делового процесса, созданная для предприятия конкретной отрасли, внедренная на практике и предназначенная для использования при разработке/реорганизации деловых процессов на других предприятиях.

Референтная модель типовых бизнес процессов для коммерческих организа



American Productivity & Quality Center, Arthur Andersen

Модель OSI

Уровни модели OSI/ISO и их характеристики

№	Название	Функции	Пример протокола	Реализация
1	Физический	Передача битов по физическим линиям связи. Определяет механические, электрические и оптические характеристики кабелей и разъемов физического интерфейса сети	Спецификация 10Base-T	Канал связи
2	Канальный	Обеспечивает взаимодействие между двумя компьютерами. Функции — проверка доступности среды передачи, реализация механизмов обнаружения и коррекции ошибок. В протоколах канального уровня, используемых в локальных сетях, заложены определенная структура связей между компьютерами и способы их адресации	Ethernet, Token Ring, FDDI, 100VG-AnyLAN	Операционная система
3	Сетевой	Реализует взаимодействие узлов сети. Этот уровень служит для образования единой транспортной системы, объединяющей несколько сетей с различными принципами передачи информации между конечными узлами	Протокол межсетевого взаимодействия IP стека TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell	
4	Транспортный	Обеспечивает возможность передачи более длинных сообщений между хостами	Протоколы TCP и UDP стека TCP/IP и протокол SPX стека Novell	
5	Сеансовый	Устанавливает и поддерживает соединение между компонентами распределенной системы. Использует механизм соединений транспортного уровня		Промежуточная среда
6	Представительский	Устраняет различия в представлении данных. Этот уровень гарантирует то, что информация, передаваемая прикладным уровнем, будет понятна прикладному уровню в другой системе. При необходимости уровень преобразовывает данные в некоторый общий формат представления или выполняет обратное преобразование	Secure Socket Layer (SSL)	
7	Прикладной	Обеспечивает функционирование приложения. Набор протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам	Протокол электронной почты	Компонент информационной системы

СОКЕТЫ

- Сокет — это программный интерфейс для обеспечения обмена данными между процессами; абстрактный объект, представляющий конечную точку соединения

- Обеспечивают примитивы низкого уровня для непосредственного обмена потоком байт между двумя процессами

- Могут быть использованы для организации простейшего взаимодействия удаленных приложений

Пусть клиенту банка необходимо обеспечить возможность перевода денег на свой счет из другого банка, используя Web-приложение (задача интеграции Web-приложения с финансовой системой).
Простейшее решение может быть получено с использованием протокола TCP/IP. Для определенности представим, что необходимо передать только имя клиента и сумму денежного перевода.
Приведенный ниже пример кода (C#) демонстрирует вариант реализации конечной точки Web-приложения, в котором создается сокет с адресом my.bank.com и по сети пересылается сумма и имя клиента

```
String hostName = "my.bank.com";
//Задание DNS-имени удаленного компьютера
int port = 80;
IPHostEntry hostInfo = Dns.GetHostByName (hostName);
//Преобразование имени удаленного компьютера в IP-адрес
IPAddress address = hostInfo.AddressList [0];
IPEndPoint endpoint = new IPEndPoint (address, port);
Socket socket = new Socket (address.AddressFamily, SocketType.
Stream, ProtocolType.Tcp);
socket.Connect (endpoint);
byte [] amount = BitConverter.GetBytes (1000);
//Преобразование данных в поток байтов
byte [] name = Encoding.ASCII.GetBytes ("Joe");
int bytesSent = socket.Send (amount);
bytesSent += socket.Send (name);
socket.Close ();
```

Проблемы и ограничения использования сокетов

- Связываемые системы должны иметь одинаковое внутреннее представление целых чисел (32 или 64 бит) и одинаковый порядок следования байтов (прямой или обратный). В противном случае передаваемые данные будут неверно интерпретированы приложением-получателем;

- Компьютер должен иметь неизменяемый адрес. Перемещение приложения получателя на другой компьютер потребует изменения программного кода;

- Интегрируемые приложения должны быть доступны в одно и то же время, поскольку протокол TCP/IP является протоколом с установкой соединения. Если одно из взаимодействующих приложений в момент установки соединения недоступно, то соединение не будет установлено;

- Использование соглашения о формате данных, предаваемых между интегрируемыми приложениями. Любое изменение в формате потребует изменений в программном коде как на стороне отправителя, так и на стороне получателя.

использование механизма сокетов приводит к получению плохо масштабируемого, неустойчивого, трудно поддерживаемого интеграционного решения, обеспечивающего «жесткую» связь между приложениями.

Промежуточная среда

Промежуточная среда (middleware — связующее программное обеспечение) выполняет функции сеан- сового и представительского уровней модели OSI/ISO.

Наличие промежуточной среды позволяет создать открытые, масштабируемые и отказоустойчивые распределенные системы.

Промежуточная среда предоставляет:

- единый, независимый от операционной системы механизм использования одними программными компонентами сервисов других компонентов;
- безопасность — аутентификация и авторизация всех пользователей сервисов компонента и защита передаваемой между компонентами информации от искажения и чтения третьей стороной;
- целостность данных — управление транзакциями, распределенными между удаленными компонентами системы;
- балансировку нагрузки на серверы с программными компонентами;
- бнаружение удаленных компонентов.

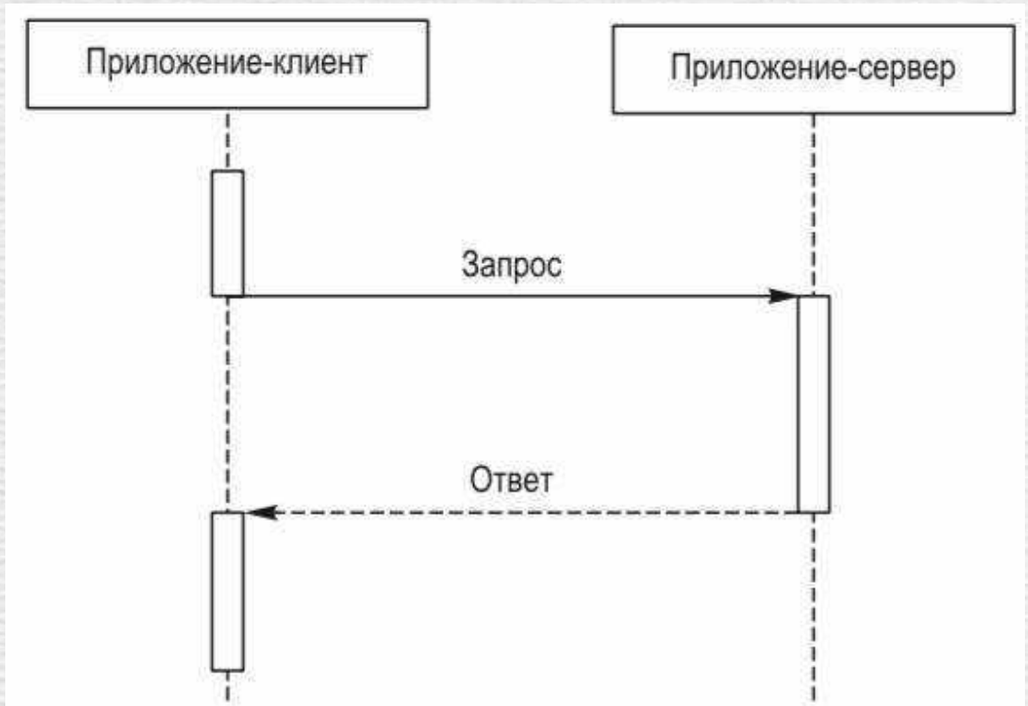
Типы промежуточных сред

- ориентированные на посылку сообщений — поддерживают связь между компонентами распределенной системы посредством обмена сообщениями (Microsoft Message Queuing, IBM MQSeries, Sun Java System Message Queue);

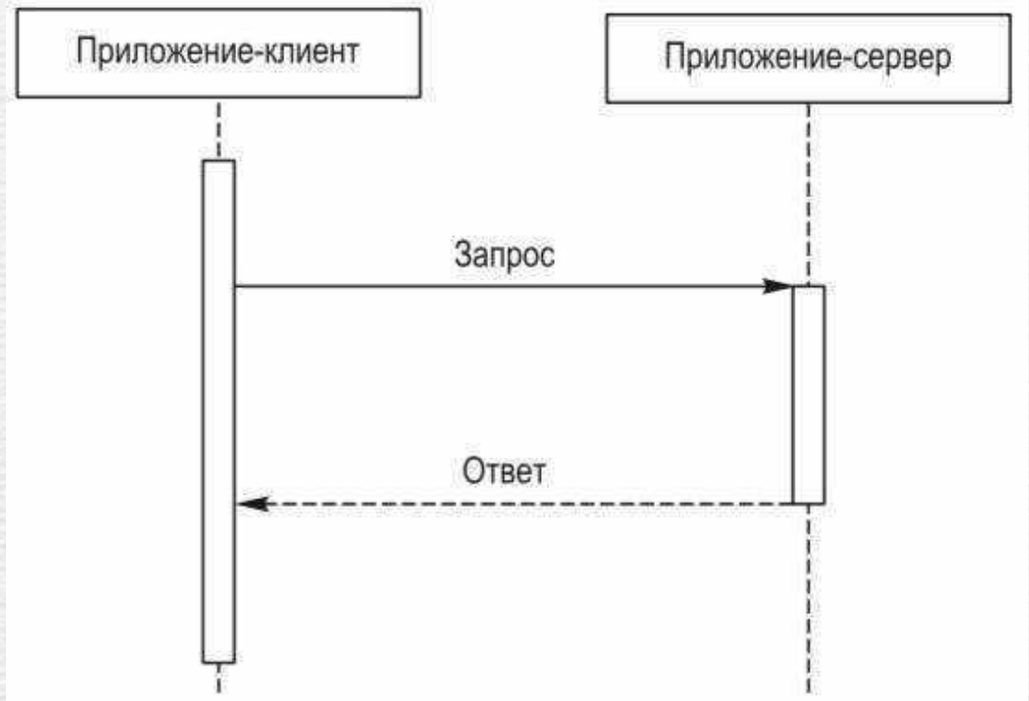
- объектно-ориентированные — построены на модели удаленного обращения к методу (J2EE, .NET, CORBA);

- транзакционно-ориентированные — ориентированы на поддержку транзакций в системах распределенных баз данных (мониторы транзакций, все современные промышленные СУБД, например MS SQL SERVER).

Модели взаимодействия приложений



Синхронным называется такое взаимодействие, при котором приложение, выступающее в роли клиента, отослав запрос, блокируется и может продолжать работу только после получения ответа от приложения, выступающего в роли сервера. Иногда такой вид взаимодействия называют блокирующим.



В случае асинхронного взаимодействия приложение-клиент после отправки запроса приложению-серверу может продолжать работу, даже если ответ на запрос еще не пришел. Иногда такой вид взаимодействия называют неблокирующим

Сложности реализации асинхронного взаимодействия

Сложности реализации асинхронного взаимодействия обусловлены следующими обстоятельствами:

- возможность использования нескольких потоков исполнения, увеличивающая производительность решения, одновременно затрудняет его отладку;

- приложение-клиент должно быть готово принять ответ в любой момент, даже в процессе выполнения другой задачи;

- поскольку асинхронные процессы могут выполняться в любом порядке, приложение-клиент должно уметь обрабатывать результат запроса с учетом времени получения.

Стандарты объектно-ориентированного взаимодействия

- Включают в себя COM (DCOM, COM+), RMI, CORBA

- Идея заключается в создании виртуального компонента, являющегося прокси-объектом (локальным представителем) удаленного метода или процедуры. Приложение-клиент обращается к прок-си-объекту, который и передает сообщение к удаленному объекту.

Удаленный вызов процедур RPC

Впервые был реализован в начале 1980-х гг. компанией Sun Microsystems и явился прообразом объектно-ориентированного промежуточного слоя применительно к структурной парадигме программирования.

Суть технологии RPC заключается в том, что вызов функции на удаленном компьютере осуществляется с помощью промежуточного программного обеспечения так же, как и на локальном

Для того чтобы организовать удаленный вызов процедуры необходимо:

- описать интерфейс процедуры, которая будет использоваться для удаленного вызова при помощи языка определения интерфейсов (Interface Definition Language, IDL);

- скомпилировать определение процедуры (при этом будут созданы описание этой процедуры на том языке программирования, на котором будут разрабатываться клиент и сервер, и два дополнительных компонента — клиентский и серверный переходники). Клиентский переходник представляет собой процедуру-заглушку (stub), которая будет вызываться клиентским приложением. Клиентский переходник размещается на той же машине, где находится компонент-клиент, а серверный переходник — на той же машине, где находится компонент-сервер.

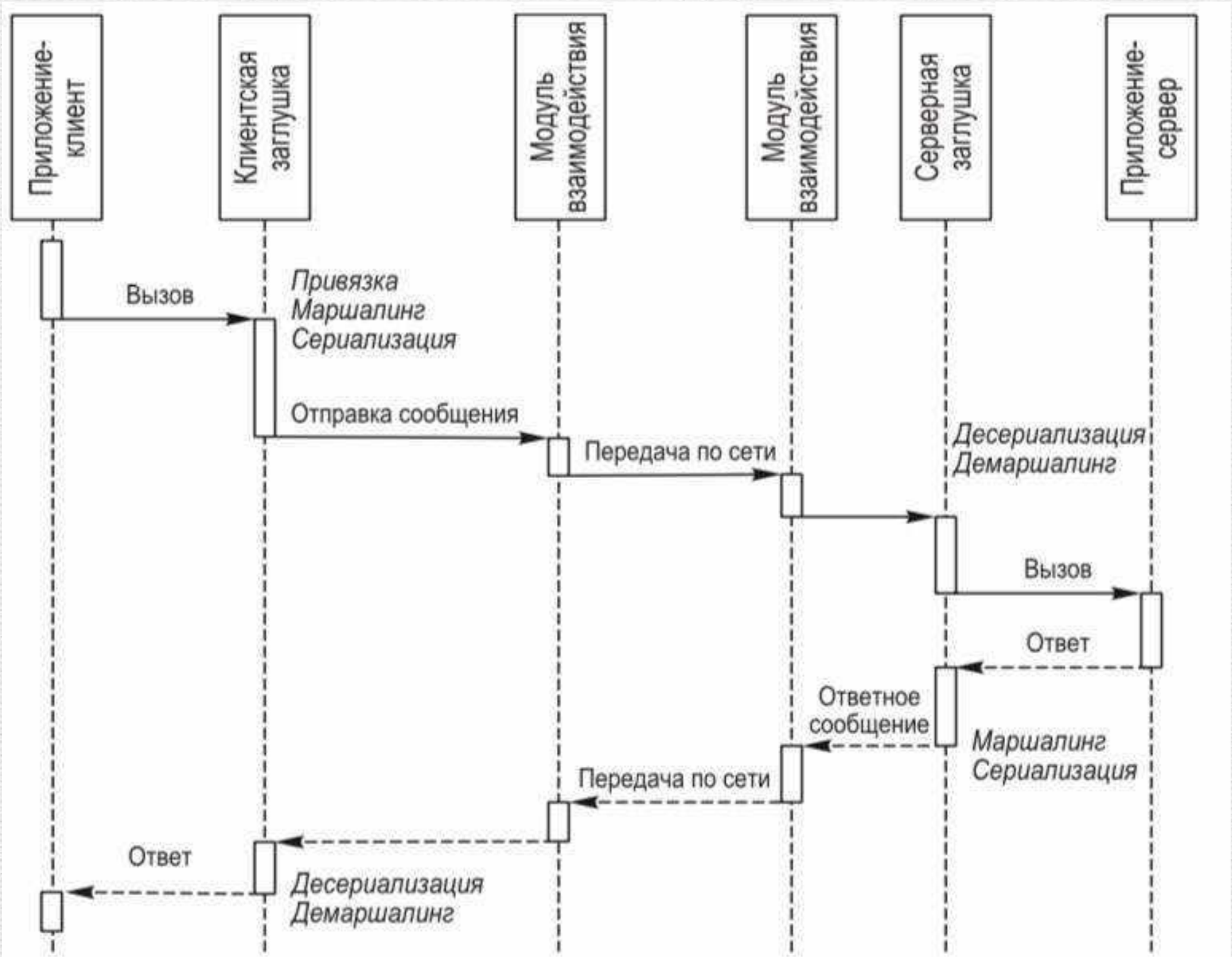
Организация работы RPC

- клиентский переходник выполняет привязку к серверу (определяет физическое местонахождение (адрес) сервера);
- клиентский переходник выполняет маршалинг (упаковывает вызов процедуры и ее аргументы в сообщение в некотором стан- дартном для данной системы формате);
- клиентский переходник выполняет сериализацию сообщения (преобразует полученное сообщение в поток байтов) и отсылает с помощью какого-либо протокола (транспортного или более высокого уровня) на машину, на которой помещен серверный компонент;
- серверный переходник принимает сообщение, содержащее пара- метры вызова процедуры, распаковывает эти параметры при помощи десериализации и демаршалинга, вызывает локально соответствующую функцию серверного компонента, получает ее результат, упаковывает его и посылает по сети на клиентскую машину;
- после получения ответа от сервера выполняется процедура десериализации.

Процесс преобразования параметров для передачи их между процессами при удаленном вызове называется маршализацией (marshaling).

Процесс преобразования экземпляра какого-либо типа данных в набор байт называется сериализацией.

Схема работы RPC



Удаленный вызов метода

- Является модификацией метода удаленного вызова процедуры для объектной парадигмы программирования.
- В момент вызова удаленного метода на стороне клиента создается клиентская заглушка, называемая посредником (proxy).
- Клиентская заглушка в своем интерфейсе имеет все методы объекта-сервера, выполняет маршалинг и сериализацию параметров вызываемых методов и передает их по сети.
- Поскольку один объект сервера может предоставлять несколько методов, информация о том, какой именно метод вызывается, также упаковывается вместе с аргументами вызова.
- Промежуточная среда на стороне сервера десериализует параметры и передает их заглушке, называемой каркасом, или скелетоном (skeleton), на стороне сервера.

Стандарты объектно-ориентированного взаимодействия

Стандарт	Назначение	Достоинства	Ограничения
Com	Средство взаимодействия приложений, функционирующих на одном компьютере	Высокий уровень абстракции Простота использования Очень высокая производительность	COM-приложения способны функционировать только под управлением Windows (двоичная структура объектов компонентной модели Microsoft) Распределенные решения плохо масштабируются (жесткая связь между клиентом и сервером) Невысокая устойчивость к сбоям COM-систем (жесткая привязка сервера к клиенту, двухуровневая архитектура)
Com+	Промежуточная среда для создания распределенных систем, действующих в локальной сети	Поддержка синхронного и асинхронного взаимодействий программных компонентов Динамическая балансировка нагрузки Поддержка логической целостности данных за счет работы с координатором распределенных транзакций Возможность ограничения доступа к компоненту в разрезе ролей, связанных с учетными записями пользователей	Использование ограничено взаимодействием компонентов внутри локальной или виртуальной частной сети, построенной на базе Microsoft Windows, в пределах одного предприятия

Стандарты объектно-ориентированного взаимодействия

Стандарт	Назначение	Достоинства	Ограничения
DCOM	Промежуточная среда для создания распределенных систем, действующих в локальной сети и сети Интернет	Независимость от языка программирования Простота Поддержка распределенных транзакций	Использование ограничено платформами Windows
CORBA	Набор спецификаций для промежуточного программного обеспечения объектного типа. Создавалась консорциумом OMG как универсальная методология создания распределенных систем в гетерогенных средах	Кроссплатформенность Высокий уровень устойчивости к внутренним сбоям за счет большей изоляции клиентов и серверов (трехуровневая архитектура) Передача данных между компонентами происходит при помощи протокола UDP (обеспечивает экономию системных ресурсов) Набор сервисов (управление транзакциями, безопасностью, жизненным циклом объектов, событиями и т.д.)	Сложность технологии
RMI	Архитектура (Remote Method Invocation, то есть вызов удаленного метода), которая интегрирована с JDK1.1 и реализует распределенную модель вычислений	Самый простой и быстрый способ создания распределенных систем Распределенное автоматическое управление объектами	Поддержка одного языка программирования Java Собственный протокол взаимодействия Трудность интегрирования с существующими приложениями

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Чеченский государственный университет им. А.А. Кадырова»
ИНСТИТУТ МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра программирования и ИКТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
«Технологии Big Data»

Направление подготовки (специальности)	Информатика и вычислительная техника
Код направления подготовки (специальности)	09.04.01
Профиль подготовки	Технологии интеллектуальных автоматизированных систем
Квалификация выпускника	Магистр
Форма обучения	Заочная

Грозный, 2024 г.

Введение

Big Data – один из наиболее часто упоминаемых сегодня терминов в ИТ-публикациях. Однако, как это часто бывает с новыми понятиями, при краткости термина его смысл весьма расплывчат. Согласно англоязычной Википедии, Big Data – это наборы данных такого объема, что традиционные инструменты не способны осуществлять их захват, управление и обработку за приемлемое для практики время. Но, согласитесь, в этом смысле проблема «больших данных» существовала на протяжении всей истории развития ИТ. Всегда возникало желание обработать большие массивы данных за минимальное время, и всегда для этого оказывалось недостаточно мощности существующей ИТ-инфраструктуры. Почему же этот термин стал так актуален именно сейчас? И почему его толкования различаются от публикации к публикации? Попробуем разобраться.

Прежде всего отметим, что под термином Big Data в разном контексте могут подразумеваться данные большого объема, технология их обработки, проекты, рынок и даже компании, активно использующие эту технологию. Для того чтобы прояснить смысловое наполнение термина, необходимо рассмотреть все перечисленные аспекты, что мы и постараемся сделать в настоящей статье.

Очевидно, что так или иначе термин связан с проблемой накопления огромных массивов данных. Статистика поражает. За последние три года человечество произвело информации больше, чем за всю историю своего существования до 2008 года. И рост продолжается экспоненциально.

Таким образом, к 2013 году количество хранящейся информации в мире составило 1,2 зеттабайта, из которых на нецифровую информацию приходится менее 2 %. Трудно представить себе такой объем данных. Если записать данные в книгах, ими можно было бы покрыть всю поверхность Соединенных Штатов в 52 слоя. Если записать данные на компакт-диски и сложить их в пять стопок, то каждая из них будет высотой до Луны. В III веке до н. э. считалось, что весь интеллектуальный багаж человечества хранится в великой

Александрийской библиотеке, поскольку египетский царь Птолемей II стремился сохранить копии всех письменных трудов. Сейчас же в мире накопилось столько цифровой информации, что на каждого живущего ее приходится в 320 раз больше, чем хранилось в Александрийской библиотеке.

Рис. 1. Объем создаваемой информации в мире (в эксабайтах) и доступные ресурсы хранения данных (источник: IDC)

Поскольку в повседневной практике мы не привыкли оперировать эксабайтами и зеттабайтами, для упрощения восприятия дальнейшего текста напомним читателям терминологию обозначения объема данных (табл. 1 и табл. 2).

Как видно из рис. 1, в 2006 году объем созданной информации был соизмерим с объемом ресурсов, доступных для ее хранения, однако уже в 2007 году информации было произведено больше, чем средств для ее хранения, и данная тенденция стала усиливаться. К 2025 году планируется получить объем превышающий значение в 160 зеттабайт.

Рис. 2 иллюстрирует динамику роста данных, объясняя ее экспоненциальный характер ростом вычислительных средств, приложений и пользователей — от миллионов в эпоху мэйнфреймов до сотен миллионов в эпоху ПК и миллиардов пользователей в эпоху мобильных устройств, мобильного Интернета, социальных сетей, «облачных» технологий и построения всевозможных решений «умной» экономики.

Рис. 2. Схема роста компьютерных приложений и их пользователей

Действительно, сегодня поток данных растет с невероятной скоростью. Данные поступают с разного рода устройств безопасности, проводится мониторинг износостойкости и прочности ответственных изделий. Огромные массивы данных генерируются в научных экспериментах, таких как регистрация метеорологических данных и астрономических наблюдений. Один только архив телескопа «Хаббл», накопленный за 15 лет, занимает около 25 Тбайт. В повседневной жизни мы сталкиваемся со всё возрастающим объемом данных — они поступают с экрана ПК, планшетника, мобильного телефона, приборных панелей автомобиля, радиочастотных идентификаторов, даже бытовые предметы, такие как кроссовки, и те могут генерировать данные (рис. 3). Множатся мобильные устройства и медицинские приложения, способные измерять пульс, давление и режим тренировки. По данным Terradata, в 2011 году примерно 31 млн американцев стали пользователями приложений для мобильных устройств, предназначенных для мониторинга здоровья.

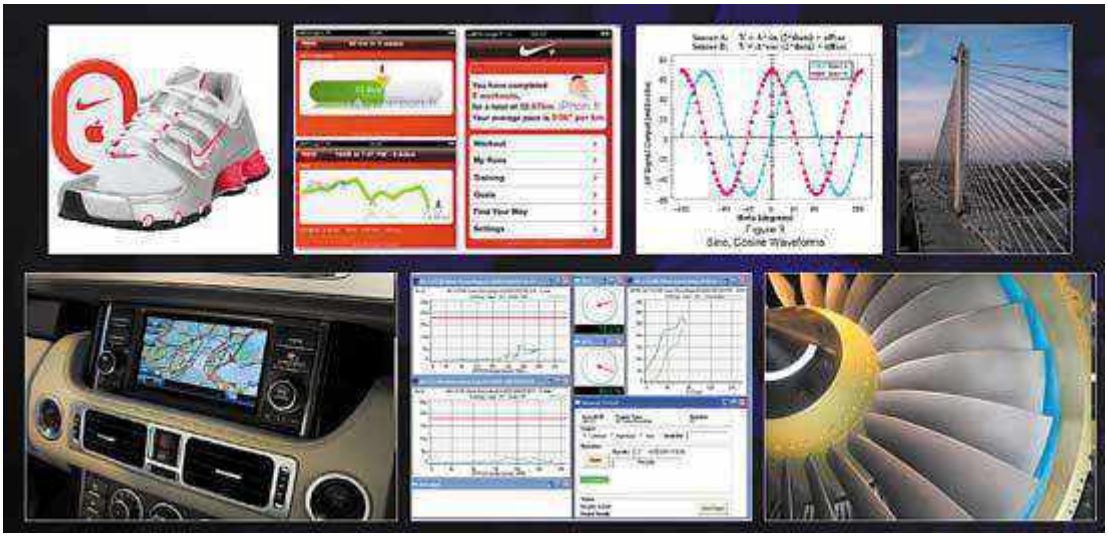


Рис. 3. Количество устройств, генерирующих потоки данных, растет с каждым годом (источник: McKinsey)

Для оптимизации любых рабочих процессов требуется информация. В развитых странах даже в сельском хозяйстве информатизация достигла невиданных масштабов. На ферме в США одна корова генерирует 200 Мбайт информации в год. Мониторинг и оптимизация содержания животного позволяют увеличить надои молока.

Отдельная тема – Интернет и мобильные технологии. Потоки информации генерируются всё новыми интернет-сервисами, социальными сетями, приложениями электронной торговли, приложениями о местонахождении абонентов сетей и т.п. Количество e-mail, отправляемых каждую секунду в мире, – 2,9 млн. Объем видео, закачиваемого на YouTube каждую минуту, – 20 часов. Объем данных, обрабатываемых Google за день, – 24 петабайт. Количество сообщений на Твиттере в день — 50 млн. Количество минут, проведенных в Facebook в месяц, – около 700 млрд. Объем данных, переданных/полученных на мобильные устройства, – 1,3 эксабайт. Количество продуктов, заказываемых в Amazon в секунду, – 72,9.

При огромном объеме данных постоянно растущего числа вебстраниц это лишь вершина айсберга, большую же его часть составляют данные о данных. Аналитикам и интернет-маркетологам интересно не только содержание веб-сайтов, но и обширная информация о пользователях: их привычки, история посещенных страниц, рекламные предпочтения, круг общения и знакомства.

Именно это позволяет продвигать товары и рекламу в Интернете. Мы можем только предполагать, насколько подробно можно проанализировать наше поведение по истории посещений в Сети. Очевидно, что для этого нужно обрабатывать действительно гигантские объемы данных (рис. 4).

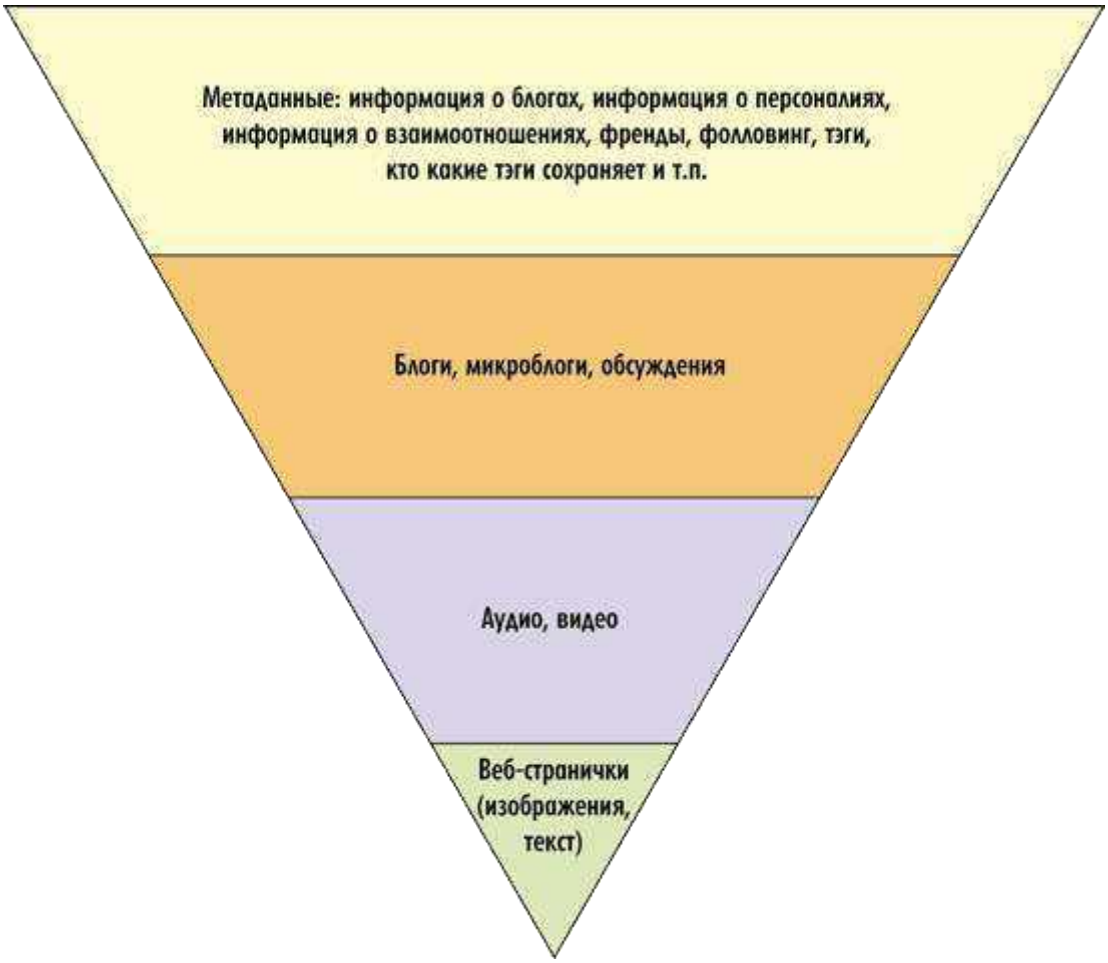


Рис. 4. Объемы данных, накапливаемых в интернет-приложениях

Анализируя данные, можно создавать новые сервисы и продукты, оптимизировать бизнес, а следовательно, зарабатывать деньги. По сути технология Big Data не есть что-то принципиально новое. Просто на современном этапе развития технологий появились средства, позволяющие хранить, обрабатывать и анализировать такие объемы данных, что стал меняться принцип подхода к их анализу. Аналитики рассуждают следующим образом: «Мы не знаем, нужна ли нам информация, а если нужна, то какая, до тех пор, пока не проанализируем, насколько она взаимосвязана». Стоимость хранения информации настолько снизилась, что появилась возможность собирать всё больше данных и анализировать, казалось бы, не связанные друг

с другом факторы. Данные собираются исходя из принципа «мы не знаем, чего мы не знаем». Человеческий мозг не может обнаружить такие закономерности, какие отмечает компьютер, выдавая совершенно неожиданные количественные взаимосвязи. Например, он может количественно связать площадь того или иного цвета на обложке журнала с вероятностью его продаж в таком-то месяце года.

Технология Big Data предоставляет услуги, помогающие раскрыть коммерческий потенциал мега-массивов данных за счет поиска ценных закономерностей и фактов путем объединения и анализа больших объемов данных.

Говоря об инструментарии в подходах Big Data, следует отметить, что необходимость обработки качественно новых объемов структурированных и неструктурированных данных показала: традиционные подходы к их хранению и обработке стали неэффективными, а следовательно, необходимы новые технологии. Учитывая масштабность задач, перед бизнесом встала задача не только выбора адекватного инструментария по анализу информации, но и построения оптимальной вычислительной инфраструктуры, которая была бы эффективной и не очень дорогой. Всё это подводит к более полному определению Big Data.

Характеристики технологии

Forrester определяет понятие Big Data как технологию в области аппаратного и программного обеспечения, которая объединяет, организует, управляет и анализирует данные, характеризующиеся «пятью V»: объемом (Volume), разнообразием (Variety), достоверностью (Veracity), скоростью (Velocity) и ценностью (Value) – рис. 5. Рассмотрим каждую из этих составляющих.

У компаний, которым необходимо хранить и осуществлять доступ к большему объему данных, есть два варианта: первый — приобрести более мощный компьютер с большим количеством процессоров, оперативной памяти, дискового пространства и т.д. Это называется масштабированием по

вертикали – scale vertically или scale up, то есть добавление ресурсов на один вычислительный узел.

Рис. 5. Границы применения традиционных BI- и Big Data-технологий (источник: Forrester)

Второй вариант – горизонтальное масштабирование (scale horizontally или scale out), которое базируется на добавлении дополнительных вычислительных узлов, то есть предоставляет возможность добавлять в систему дополнительные компьютеры и распределять работу между ними. Горизонтальное масштабирование позволяет построить высоконадежное решение с обеспечением должной степени резервирования на базе недорогих стандартных компьютеров, каждый из которых обладает невысокой надежностью. Сотни и даже тысячи маломощных компьютеров, объединенных в кластер, могут обеспечивать вычислительную мощность суперкомпьютеров.

Объем данных (Volume)

Мы описали лавинообразный рост объема данных в научных и персональных приложениях. Дополним картину информацией о том, какие объемы данных накопили корпорации. Только в США это более 100 Тбайт данных. При этом в разных вертикальных индустриях объем данных существенно различается, а следовательно, актуальность применения технологии Big Data в них различна (рис. 6).

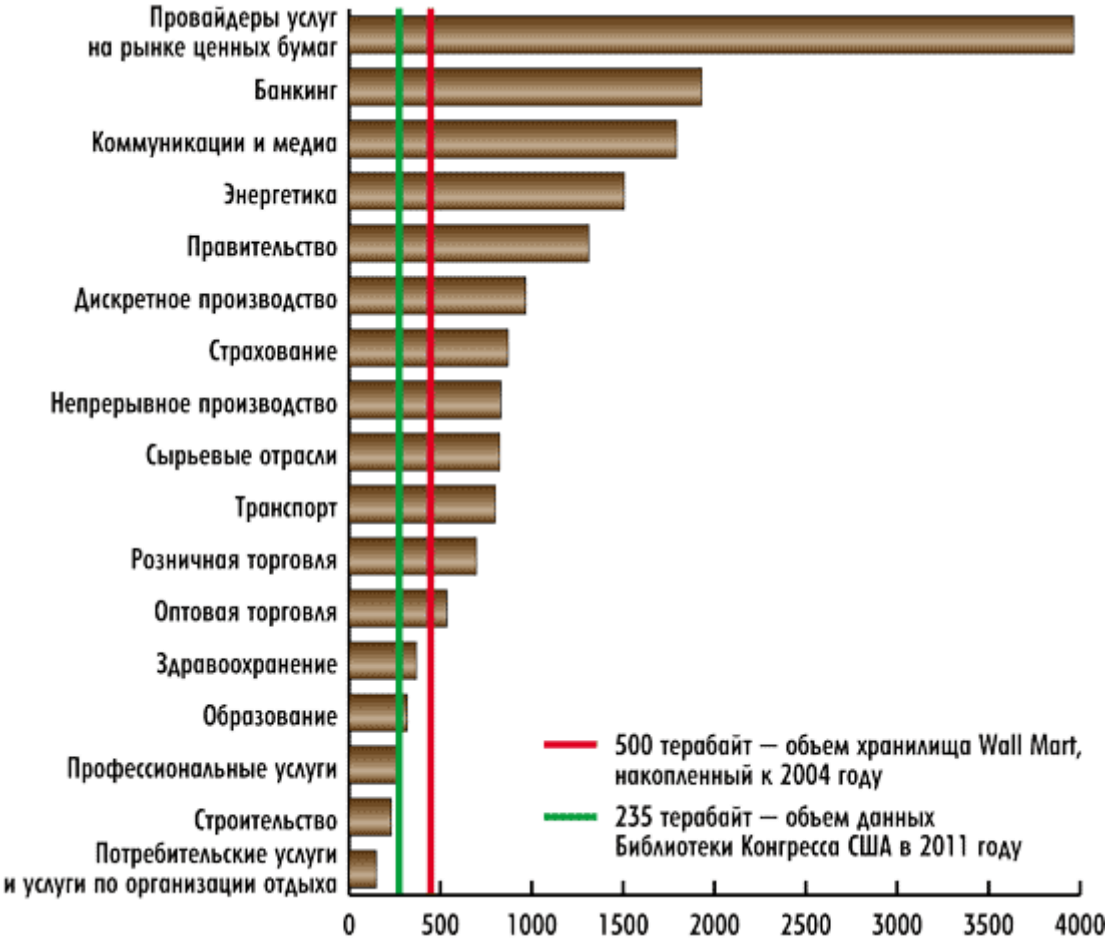


Рис. 6. Объем накопленных данных в корпорациях из разных сфер деятельности (источник: McKinsey)

Разнообразие форматов данных (Variety)

Способность приложения обрабатывать большие массивы данных, поступающие из разных источников в различных форматах, является главным критерием отнесения его к технологии Big Data. Обычно Big Data-приложения объединяют данные из разных источников (как внутренних, так и внешних по отношению к организации) и разной степени структурированности

(структурированные, слабоструктурированные и неструктурированные). Многие бизнес-задачи и научные эксперименты требуют совместной обработки данных различных форматов – это могут быть табличные данные в СУБД, иерархические данные, текстовые документы, видео, изображения, аудиофайлы и т.д. Пример подобного рода задачи из области медицины: как найти оптимальный курс лечения для конкретного пациента, базируясь на огромном количестве историй болезней пациентов (которые постоянно меняются), а также на базе данных медицинских исследований и геномных данных? Другой пример – из области оптимизации бизнес-процессов: как провести анализ структурированных данных из ERP-приложения, а также слабоструктурированных данных в виде логфайлов и неструктурированного текста из отзывов покупателей? Третий пример – из сферы прогнозирования погоды: как выполнить анализ климата на базе многолетних метеорологических данных и данных, поступающих со спутника в реальном времени?

Скорость поступления и обработки информации

В ряде задач может потребоваться очень высокая скорость обработки данных. Например, биржевым игрокам иногда нужно мгновенно принять решение, основываясь на большом количестве данных о состоянии рынка, — за пару секунд ситуация уже может измениться. Существует также целый ряд задач, когда решение нужно принимать в реальном времени, например обработка биометрических данных, получаемых в огромном потоке людей, которые необходимо сверить с базой данных о злоумышленниках. Очень большая скорость поступления данных характерна для многих научных задач. Например, проект по запуску гигантского радиотелескопа с суммарной площадью антенн 1 км^2, который запущен в 2015 году, предполагает передачу сигналов с одной антенны со скоростью 160 Гбит/с, что в 10 раз превышает весь предыдущий интернет-трафик.

Ценность для бизнеса

Компания IDC тоже выделяет «четыре V», характеризующие данные, однако это уже несколько иной набор: объем (Volume), разнообразие (Variety), скорость (Velocity) и ценность (Value). То есть параметр Variety (изменчивость), который применяет компания Forrester, она заменяет на параметр Value (ценность). IDC подчеркивает, что параметр Value — один из основных, позволяющих выделить Big Data как новое явление. Он относится к экономическому эффекту, который технология Big Data обеспечивает пользователям. Действительно, большие хранилища данных в сфере финансовых услуг, телекоммуникаций, розничной торговли и государственных организаций существовали на протяжении многих лет. Применялись решения по обработке данных в реальном времени для управления бизнес-процессами, например в торговле, а также высокопроизводительные вычислительные системы для научных исследований. Различие их состоит в том, что те системы, которые раньше решали отдельные проблемы бизнеса на больших предприятиях, сегодня становятся основой осуществления их бизнес-стратегии.

Технология Big Data позволяет уменьшить расходы на ИТ-инфраструктуру и ПО, сократить затраты на рабочую силу за счет более эффективных методов интеграции данных, управления, анализа и выработки решения; обеспечить увеличение дохода и прибыли путем новых или более эффективных способов ведения бизнеса. То есть на современном этапе те же самые технологии представляют качественно новую ценность для предприятия. Возрастание степени влияния Big Data в современных корпорациях иллюстрирует табл. 3.

При этом сочетание использования ПО с открытым исходным кодом и снижение цен аппаратных систем сделало эти технологии более доступными. Системы, доступ к которым предоставлялся ранее только государственным учреждениям или немногим крупнейшим компаниям, теперь стали доступны

для гораздо более широкого числа пользователей, что сформировало сравнительно массовый рынок на подобные услуги.

Очевидно, что компании, которые сумеют извлечь из доступных им данных больше полезной для себя и своего бизнеса информации, окажутся в выигрыше. MacKinsey приводит количественную характеристику данного положения (рис. 7).

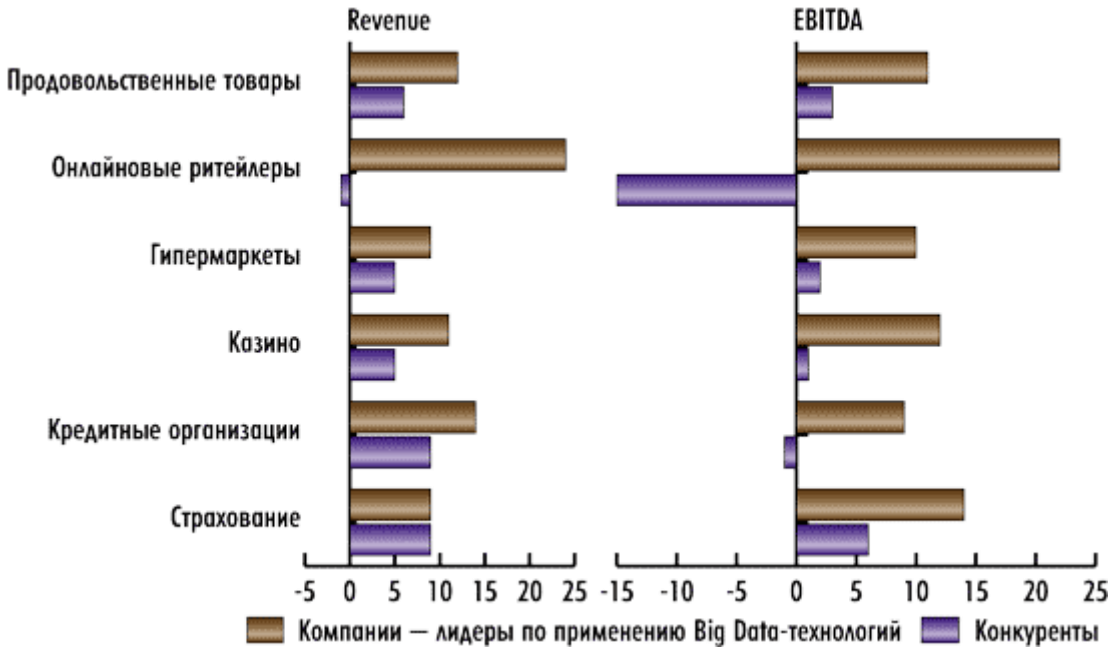


Рис. 7. Лидеры по применению Big Data-технологий и их конкуренты (источник: MacKinsey)

На рис. 7 показаны оборот и EBITDA компаний, которые применяют Big Data-технологии, и насколько это помогает им получить конкурентные преимущества.

Big Data как рынок

Если рассматривать Big Data как рынок, то полезно ответить на вопрос, кто является пользователями и потребителями на нем, какие процессы они автоматизируют и какие технологии для этого используются.

Потребители – это организации, являющиеся, как правило, и пользователями решения, и производителями данных, которые должны быть обработаны, а в большинстве случаев – еще и исполнителями работ по аналитической обработке данных. Отметим, что по мере удешевления

технологии Big Data к числу ее пользователей добавляется всё больше заказчиков из средних предприятий.

Автоматизируемые процессы: сбор данных, их обработка, поддержка принятия и исполнения решений. Эти шаги состоят из множества подопераций, таких как мониторинг, обнаружение, измерение, оповещение, очистка, анализ и архивирование.

Таким образом, технологию Big Data можно рассматривать как некий стек технологий (рис. 8).

Рис. 8. Стек Big Data

ИТ-инфраструктура

ИТ-инфраструкутра для задач класса Big Data всё чаще строится на базе стандартных серверов, сетей, СХД, гипервизоров и кластерного ПО, что позволяет удешевить решение. Комплексы, построенные путем масштабирования стандартных x86-серверов наряду с использованием сетевых технологий Ethernet 10GbE, позволяют достигать вычислительных мощностей, которые в прошлом были доступны только на специализированных суперкомпьютерах. Следует отметить, что «облачная» инфраструктура – это удобная технология для работы с большими объемами данных.

Организация и управление Big Data

Технологии для организации и управления данными относятся к программному обеспечению, которое обрабатывает и готовит все виды структурированных и неструктурированных данных для анализа. Эти приложения отвечают за извлечение, очистку, нормализацию и интеграцию данных. Они включают подходы реляционных баз данных, но всё-таки в большей мере – NoSQL-подходы. Такой подход направлен на реализацию моделей баз данных, отличных от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Подход NoSQL не является полным отрицанием языка SQL и реляционной модели и исходит из того, что SQL – это полезный инструмент, но отнюдь не оптимальный при работе с данными очень большого объема и в проектах с разнородными данными. Основные положения при разработке такого типа систем – нереляционная модель данных, открытый исходный код, хорошая горизонтальная масштабируемость.

Аналитическая обработка Big Data и выявление закономерностей

ПО для аналитической обработки Big Data и выявление закономерностей – это большая группа приложений, которая может быть классифицирована по разным принципам. Приложения для офлайновой обработки или онлайновой обработки по запросу, средства выявления закономерностей в данных, приложения для различных вертикальных областей, например решения для розничной торговли, оптимизации транспортных потоков и т.п. Данное ПО также может быть классифицировано по типу данных, которые анализируются: текстовые, аудио, видео, сетевые структуры. Кроме того, приложения можно разделить по степени сложности задач: базовая агрегация или сложные прогнозные задачи.

Средства поддержки принятия решения

В большинстве практических приложений анализ данных не является самоцелью. Если мы говорим об автоматизации бизнес-задач, то решение должно включать замкнутый цикл модели принятия решений, который

содержит такие шаги, как мониторинг, анализ, поддержка принятия решения и автоматизация его исполнения. Следует выделять два класса ПО поддержки принятия решений: ПО поддержки принятия решений в транзакционных и проектных управленческих задачах. Первые требуют высокой степени автоматизации, функционирования в режиме реального времени и потоковых данных. Принятие решений базируется на политиках – на основе выбора действий, предписанных той или иной ситуацией. В качестве примера можно привести выявление случаев мошенничества, оптимизацию торговли ценными бумагами, оптимизацию цен на авиабилеты, рекомендацию товаров в системах электронной коммерции и т.п.

Второй тип ПО – обычно это анализ по запросу, включающий выявление закономерностей в данных, прогнозирование некоторых событий и принятие решений на основе данного интеллектуального предсказания. Примеры включают приложения для сегментации клиентов, исследование закономерностей в проектировании фармацевтических препаратов, исследование закономерностей в залегании природных ресурсов, прогнозирование погоды.

Технологии Big Data могут использоваться как в транзакционных, так и в проектных задачах.

Как видно из рис. 9, в разных индустриях перспективы внедрения технологии Big Data различны. Например, потенциал применения Big Data в правительственных организациях один из наиболее высоких, однако индекс легкости захвата данных минимален.

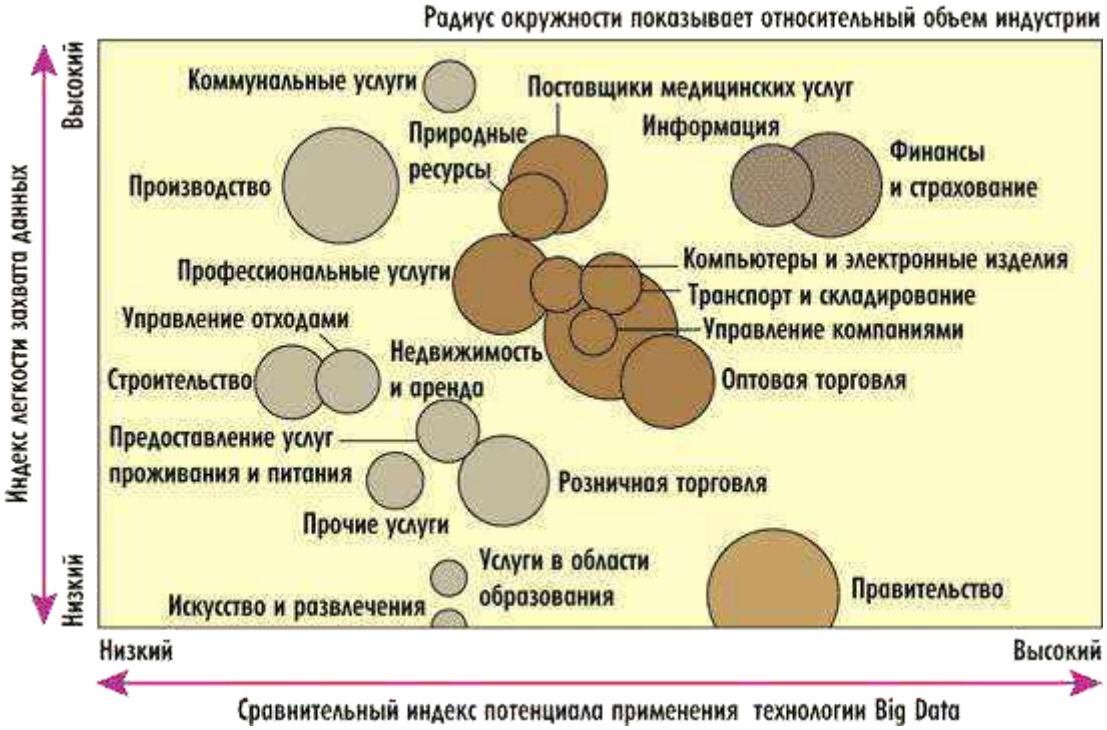


Рис. 9. Сравнительный индекс потенциала применения технологии Big Data и индекс легкости захвата данных в разных вертикальных индустриях

Примеры реализации

Из примеров реализации технологии Big Data наиболее часто упоминается проект Hadoop – по осуществлению распределенных вычислений для обработки больших объемов данных, который создается в рамках Apache Software Foundation. Коммерческую поддержку проекта осуществляет компания Cloudera. В проекте участвуют разработчики со всего мира. С технологической точки зрения Apache Hadoop – это свободный Java-фреймворк, поддерживающий выполнение распределенных приложений, работающих на больших кластерах, которые построены на стандартном оборудовании. Поскольку обработка данных организуется на кластере серверов, если один из них выходит из строя, работа перераспределяется между оставшимися. В Hadoop реализована технология MapReduce, которая обеспечивает автоматическое распараллеливание данных и их обработку на кластерах. Ядром Hadoop служит отказоустойчивая распределенная файловая система HDFS (Hadoop Distributed File System), оперирующая системами хранения. Она разбивает входящие данные на блоки, каждый из которых попадает на отведенное ему место в пуле серверов. Система

позволяет приложениям масштабироваться до уровня тысяч узлов и петабайт данных.

Позднее на рынке появился целый ряд решений, в которых использовались принципы, реализованные в MapReduce. К ним можно отнести продукты компаний Teradata, Aster Data, Netezza, DATAllegro, Microsoft (SQL Server, Project Madison), Dataupia, Vertica, ParAccel, Neoview, Greenplum, IBM (DB2, проект Database Partitioning Feature) и Oracle (проект Exadata).

Об интересе крупных вендоров к технологии Big Data свидетельствуют их приобретения последних лет. Так, в 2010 году IBM купила Netezza – производителя программно-аппаратных комплексов для BI-систем и аналитических хранилищ данных – приблизительно за 1,7 млрд долл. Сегодня IBM предлагает решение в области хранилищ данных семейства IBM Netezza, предназначенное специально для выполнения сложной аналитики на сверхбольших объемах данных. В 2010 году EMC приобрела компанию Greenplum Software, занимающуюся организацией информационных хранилищ. Фирма Greenplum разрабатывает программные средства баз данных для организации хранилищ данных и интеллектуальных ресурсов предприятий. У Greenplum есть несколько значительных инвесторов, в том числе Sun Microsystems и SAP Ventures. База клиентов компании включает Skype, Equifax и T-Mobile Fox Interactive Media.

Сегодня EMC позиционирует себя как производителя, обладающего полным стеком решений для построения «облачной» инфраструктуры для хранения и работы с «большими данными».

Оба игрока – IBM/Netezza и EMC/Greenplum – отмечены компанией Gartner как представители квадранта лидеров (рис. 10).

Рис. 10. Магический квадрант провайдеров решений в области систем управления хранилищами данных (источник: Gartner)

Итак, объем данных, производимых отдельным человеком, растет. Но еще важнее другое: объем данных, производимых машинами, растет еще быстрее. Журналы, системы отслеживания RFID, сенсорные сети, данные GPS, розничные сделки – все это вносит свой вклад в растущую гору данных. Объем общедоступных данных тоже растет с каждым годом. Организации уже не ограничиваются обработкой только собственных данных; в будущем успех будет в значительной степени определяться их способностью извлекать полезную информацию из данных других организаций.

В общем, Большие Данные – это хорошо. Плохо то, что с их хранением и анализом возникает множество проблем.

Чтение всех данных с одного диска выполняется слишком медленно, а запись – и того медленнее. Очевидный способ сокращения времени чтения заключается в одновременном чтении данных с нескольких дисков.

На первый взгляд идея использования одной сотой части диска кажется расточительной. Но мы можем хранить сто наборов данных, каждый из которых занимает один терабайт, и организовать совместный доступ к ним. Возможно, пользователи такой системы охотно согласятся на общий доступ в обмен на ускорение анализа данных; кроме того, статистически их задания по анализу данных с большей вероятностью окажутся распределенными по времени и будут в меньшей степени мешать друг другу.

Однако концепция параллельного чтения и записи данных на нескольких дисках не так проста.

Hadoop: MapReduce

Во-первых, необходимо учесть возможность сбоев оборудования; как только вы начинаете использовать много устройств вместо одного, вероятность сбоя на одном из них значительно повышается. Стандартным способом предотвращения потери данных является репликация: система хранит избыточные копии данных, чтобы в случае сбоя была доступна другая копия. Например, так работают массивы RAID, хотя, как вы вскоре узнаете, файловая система Hadoop – HDFS (Hadoop Distributed Filesystem) — использует несколько иной подход.

Во-вторых, в большинстве задач анализа данных требуется каким-то образом объединять данные. Может оказаться, что данные, прочитанные с одного диска, должны объединяться с данными остальных дисков.

Собственно, в этом и заключается функциональность **Hadoop**: система надежного общего хранения и анализа данных. HDFS обеспечивает хранение, а MapReduce – анализ. Hadoop содержит и другие компоненты, но эти возможности образуют ядро системы.

MapReduce – модель программирования, ориентированная на обработку данных. Эта модель проста, но не настолько, чтобы в ее контексте нельзя было

реализовать полезные программы. Hadoop позволяет запускать программы MapReduce, написанные на разных языках, например, Java, Ruby, Python и C++. Но самое важное заключается в том, что программы MapReduce параллельны по своей природе, а следовательно, крупномасштабный анализ данных становится доступным для всех, у кого в распоряжении имеется достаточно компьютеров. Достоинства MapReduce в полной мере проявляются в работе с большими наборами данных.

- Работа MapReduce состоит из двух шагов: Map и Reduce.
- На Map-шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом – masternode) получает входные данные задачи, разделяет их на части и передает другим компьютерам (рабочим узлам – workernode) для предварительной обработки. Название данный шаг получил от одноименной функции высшего порядка.
- На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат.

При правильной архитектуре приложения, информация о том, на каких машинах расположены блоки данных, позволяет запустить на них же вычислительные процессы Workers и выполнить большую часть вычислений локально, т.е. без передачи данных по сети. Именно эта идея лежит в основе парадигмы MapReduce и её конкретной реализации в Hadoop.

Классическая конфигурация кластера Hadoop состоит из одного сервера имён, одного мастера MapReduce (JobTracker) и набора рабочих машин, на каждой из которых одновременно крутится сервер данных (DataNode) и воркер (TaskTracker). Каждая MapReduce работа (рис. 11) состоит из двух фаз:

1. map – выполняется параллельно и (по возможности) локально над каждым блоком данных. Вместо того, чтобы доставлять терабайты данных к программе, небольшая, определённая пользователем программа копируется на

сервера с данными и делает с ними всё, что не требует перемешивания и перемещения данных (shuffle).

2. reduce – дополняет map агрегирующими операциями.

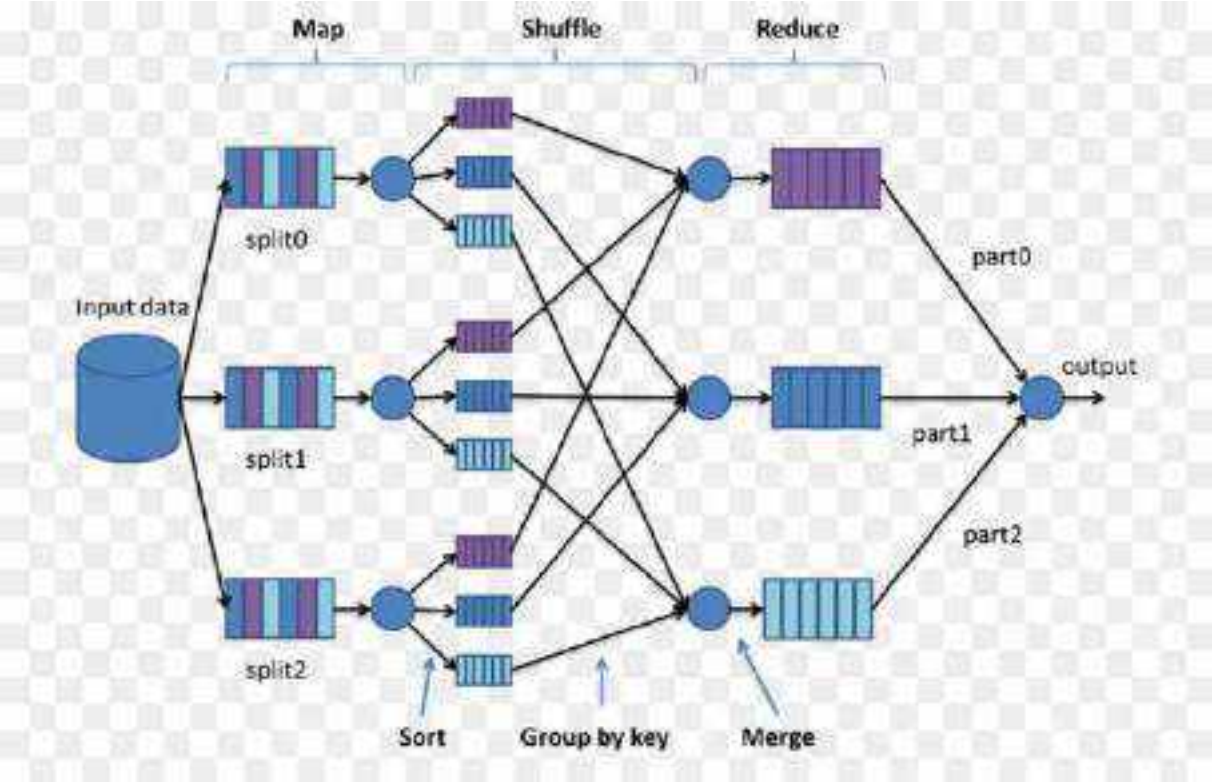


Рис. 11. Парадигма MapReduce

Проблема в том, что большинство реальных задач гораздо сложней одной работы MapReduce. В большинстве случаев мы хотим делать параллельные операции, затем последовательные, затем снова параллельные, затем комбинировать несколько источников данных и снова делать параллельные и последовательные операции. Стандартный MapReduce спроектирован так, что все результаты – как конечные, так и промежуточные – записываются на диск. В итоге время считывания и записи на диск, помноженное на количество раз, которые оно делается при решении задачи, зачастую в несколько (да что там в несколько, до 100 раз!) превышает время самих вычислений.

И здесь появляется Spark. Спроектированный в университете Berkeley, Spark использует идею локальности данных, однако выносит большинство вычислений в память вместо диска. Ключевым понятием в Spark-е является RDD (resilient distributed dataset) – указатель на ленивую распределённую

коллекцию данных. Большинство операций над RDD не приводит к каким-либо вычислениям, а только создаёт очередную обёртку, обещая выполнить операции только тогда, когда они понадобятся.

Такая модель оказалась настолько эффективной и удобной, что проекты из экосистемы Hadoop начали один за другим переводить свои вычисления на Spark, а над самим движком сейчас работает больше людей, чем над морально устаревшим MapReduce.

Вы видели, как MapReduce работает для малых входных данных; пришло время взглянуть на систему «издалека» и рассмотреть, как происходит передача данных для больших входных наборов. Чтобы картина была полноценной, данные должны храниться в распределенной файловой системе (обычно в системе HDFS, о которой вы узнаете позже). Это позволит Hadoop разместить вычисления MapReduce на каждом компьютере, обрабатывающем свою часть данных. Давайте посмотрим, как работает эта схема.

Для начала немного терминологии. Задание (job) MapReduce представляет собой единицу работы, которую хочет выполнить клиент: оно состоит из входных данных, программы MapReduce и конфигурационной информации. Чтобы выполнить задание, Hadoop разбивает его на задачи (tasks), которые делятся на два типа: задачи отображения и задачи свертки. Узлы, управляющие процессом выполнения задания, делятся на два типа: трекер заданий (jobtracker) и несколько трекеров задач (tasktracker). Трекер заданий координирует все задания, выполняемые системой; для этого он планирует выполнение задач на трекерах задач. Трекеры задач выполняют задачи и отправляют отчеты о ходе работы трекеру заданий, который отслеживает общий прогресс каждого задания. Если попытка выполнения задачи завершается неудачей, трекер может заново спланировать ее на другом трекере.

Hadoop делит входные данные заданий MapReduce на фрагменты фиксированного размера, называемые сплитами (splits). Hadoop создает для

каждого сплита одну задачу отображения, которая выполняет определенную пользователем функцию отображения для каждой записи в сплите.

Наличие многих сплитов означает, что время, затраченное на обработку каждого сплита, относительно мало по сравнению с временем обработки всех входных данных. Таким образом, если сплиты будут обрабатываться параллельно, нагрузка будет лучше сбалансирована при малом размере сплитов, поскольку более быстрая машина сможет обработать больше сплитов, чем медленная. Даже если машины идентичны, сбойные процессы или другие одновременно выполняемые задания заставляют обратить внимание на распределение нагрузки, а качество распределения нагрузки возрастает с уменьшением сплитов.

С другой стороны, при слишком малом размере сплита затраты на управление сплитами и создание задач отображения занимают слишком большую долю общего времени выполнения. Для большинства заданий желательный размер сплита обычно соответствует размеру блока HDFS – 128 Мбайт по умолчанию, хотя эту величину можно изменить для кластера (для всех вновь создаваемых файлов) или задать при создании файла.

Hadoop старается по возможности выполнять задачи отображения в узле, в котором входные данные хранятся в HDFS. Этот принцип, называемый оптимизацией локальности данных, стремится снизить нагрузку на ценную пропускную способность кластера. Тем не менее в некоторых случаях все три узла, на которых размещены реплики блока HDFS с входным сплитом задачи отображения, заняты выполнением других задач отображения; тогда планировщик заданий ищет свободный слот отображения на узле в том же сегменте (rack), что и один из блоков. Крайне редко даже такое решение оказывается невозможным, и тогда используется внесегментный узел, что приводит к межсегментной передаче данных по сети. Эти три возможности представлены на рис. 12.

Понятно, почему оптимальный размер сплита совпадает с размером блока: это максимальный размер данных, которые могут гарантированно

храниться на одном узле. Если сплит занимает два блока, вряд ли найдется узел HDFS, на котором хранятся оба блока, и часть сплита придется передавать по сети узлу, на котором выполняется задача отображения. Конечно, в этом случае выполнение станет менее эффективным, чем при выполнении всей задачи отображения с использованием локальных данных.

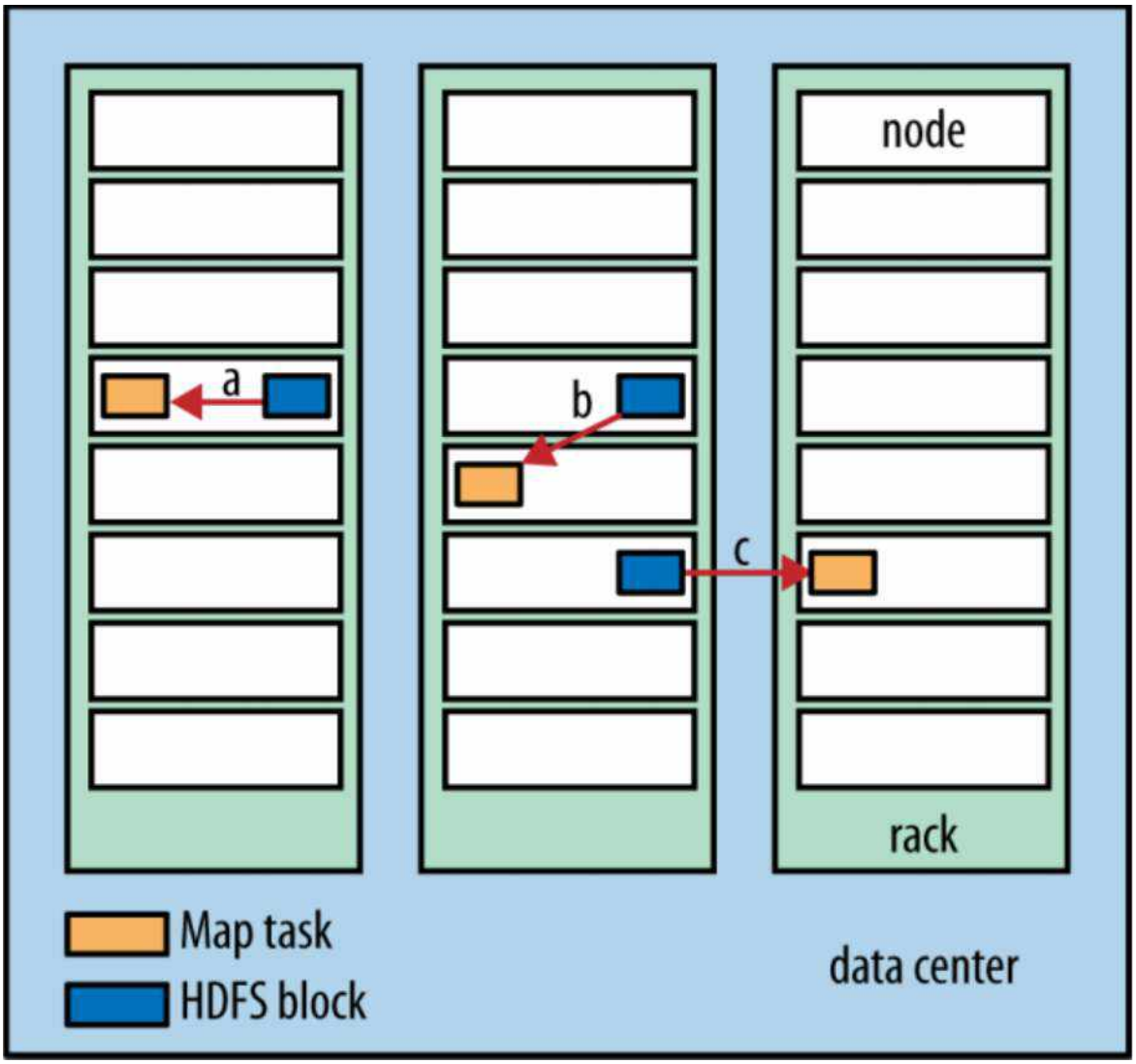


Рис. 12. Возможности split

Задачи отображения записывают свои выходные данные на локальный диск, а не в HDFS. Почему? Вывод отображений является промежуточным: он обрабатывается задачами свертки для получения итогового вывода, и после завершения задания вывод задач отображения можно удалить. Таким образом, хранение его в HDFS с репликацией неэффективно. Если на узле, на котором

работает задача отображения, произойдет сбой до того, как вывод отображения будет получен задачей свертки, Hadoop автоматически заново выполнит задачу отображения на другом узле, чтобы воссоздать вывод задачи отображения.

Задачи свертки не пользуются преимуществами локальности данных; входные данные одной задачи свертки обычно образуются из выходных данных всех задач отображения. В нашем примере одна задача свертки получает данные от всех задач отображения. Таким образом, отсортированный вывод отображений должен быть передан по сети узлу, на котором работает задача свертки; там данные объединяются и передаются функции свертки, определенной пользователем. Вывод свертки обычно хранится в HDFS по соображениям надежности.

Для каждого блока HDFS, содержащего выходные данные свертки, первая реплика хранится в локальном узле, а другие реплики хранятся во внесегментных узлах. Соответственно, запись вывода свертки приводит к расходованию пропускной способности сети, но не больше, чем для обычного конвейера записи HDFS. Весь поток данных для одной задачи свертки изображен на рис. 13.

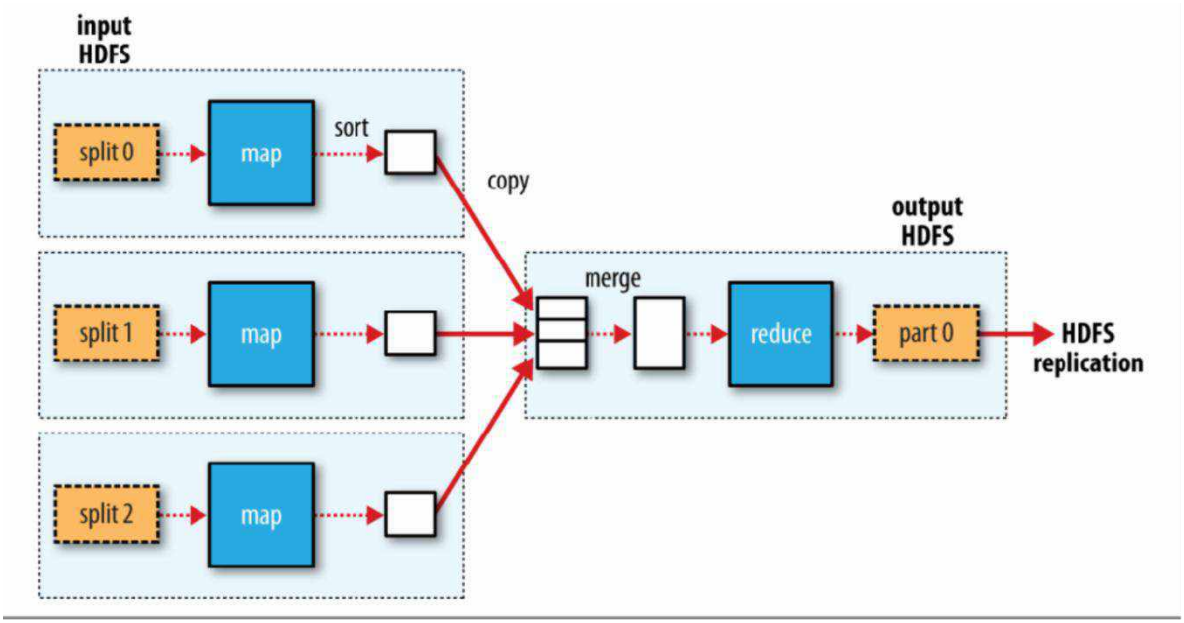


Рис. 13. Свертка

Пунктирные прямоугольники обозначают узлы, пунктирные стрелки – передачу данных узлам, а жирные стрелки – передачу данных между узлами.

Количество задач свертки не определяется размером входных данных, а задается независимо.

При наличии нескольких функций свертки задачи отображения разделяют свои выходные данные, при этом каждая создает один раздел (partition) для одной задачи свертки. В каждом разделе может быть много ключей (и связанных с ними значений), но все записи для заданного ключа находятся в одном разделе. Разделением можно управлять при помощи функции, определяемой пользователем, но обычно стандартная реализация разделения (с хеш-функцией) работает очень хорошо.

Поток данных для общего случая нескольких задач свертки показан на рис. 14. Из диаграммы ясно видно, почему поток данных между задачами свертки и отображения известен под неформальным названием «тасовка» (shuffle) – каждая задача свертки получает данные от многих задач отображения. Процесс тасовки сложнее, чем может показаться из диаграммы, а его оптимизация может оказать большое влияние на время выполнения задания.

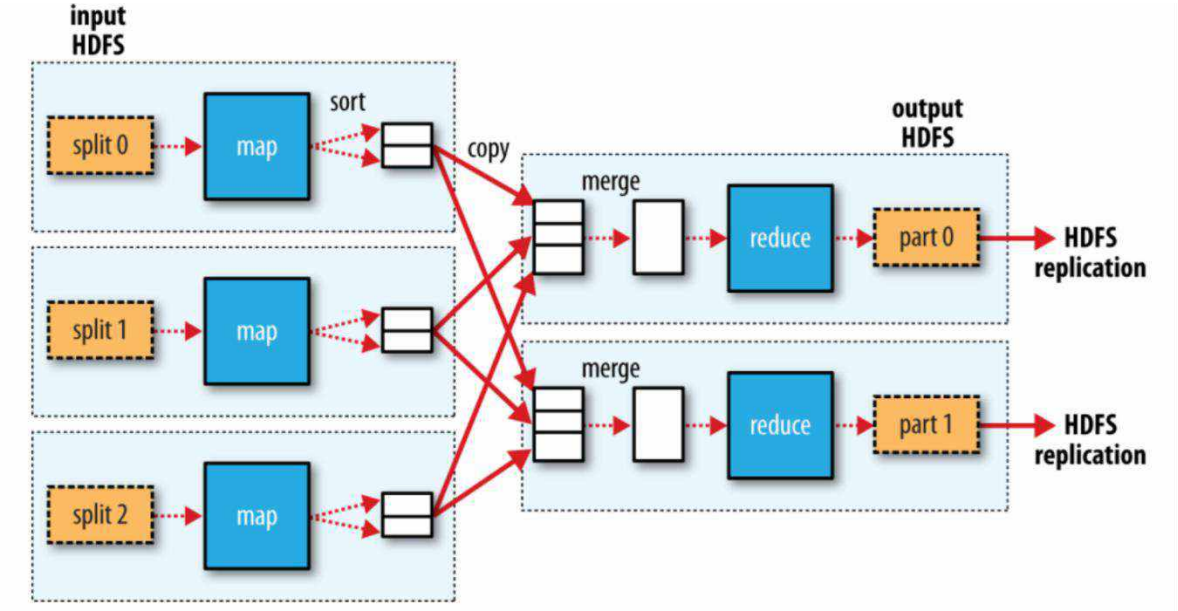


Рис. 14. Несколько задач свертки

Наконец, задание может содержать нуль задач свертки. Это может быть уместно, когда тасовка не нужна, потому что обработка осуществляется полностью параллельно. В этом случае внеузловая пересылка данных происходит только при выполнении записи задачами отображения в HDFS (рис. 15).

Многие задания MapReduce ограничиваются по пропускной способности, доступной в кластере, поэтому передачу данных между задачами отображения и свертки желательно свести к минимуму. Hadoop позволяет пользователю задать комбинирующую функцию, которая будет выполняться для выходных данных отображения; выходные данные комбинирующей функции образуют ввод функции свертки.

Так как комбинирующая функция является оптимизацией, Hadoop не предоставляет гарантий относительно того, сколько раз она будет вызвана для конкретной записи выходных данных отображения (и будет ли вызвана вообще). Иначе говоря, при вызове комбинирующей функции нуль, один или несколько раз функция свертки должна выдавать одинаковый результат.

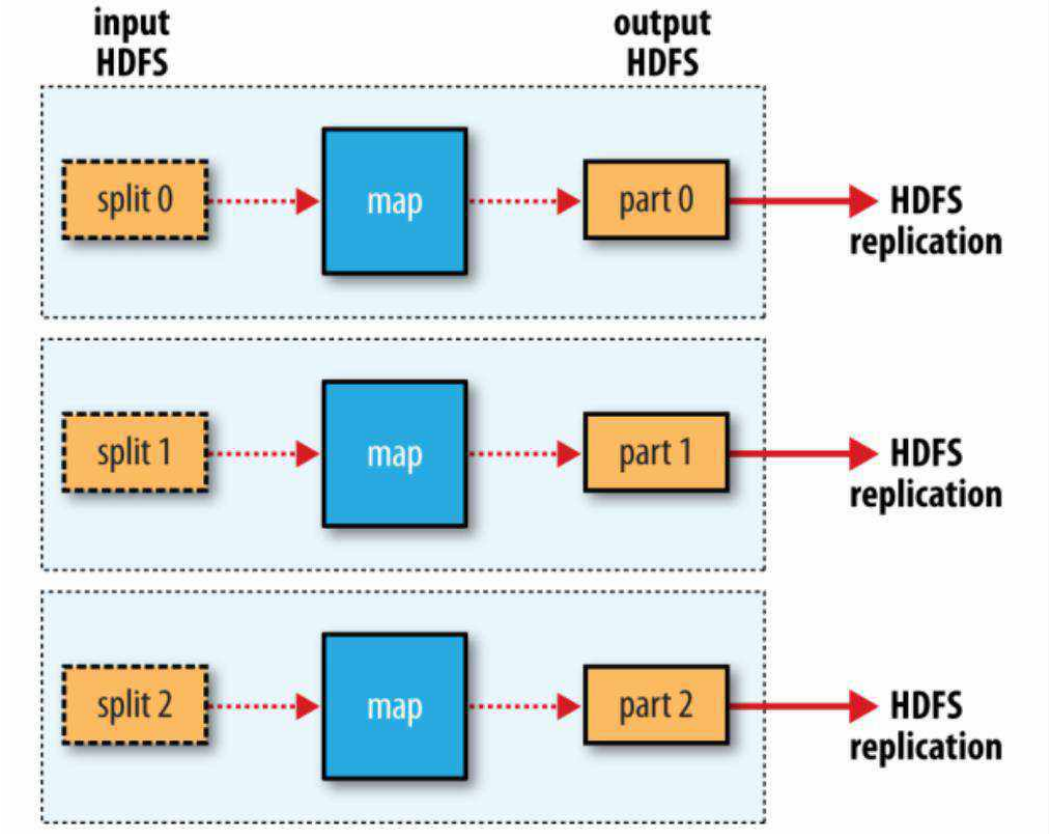


Рис. 15. Параллельная обработка

Hadoop предоставляет API для MapReduce, позволяющий записывать функции отображения и свертки на других языках, кроме Java. Технология Hadoop Streaming использует стандартный поток Unix для организации взаимодействия Hadoop с программами, так что при написании программ MapReduce можно использовать любой язык, поддерживающий чтение из стандартного входного потока (стандартного ввода) и запись в стандартный выходной поток (стандартный вывод). Функция свертки читает строки из стандартного ввода (которые, как гарантирует инфраструктура, отсортированы по ключу) и записывает свои результаты в стандартный вывод.

Источник: *Hadoop: The Definitive Guide, Fourth Edition by Tom White Copyright © 2015 Tom White. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.*

HDFS

АРХИТЕКТУРА HDFS

Кластер HDFS включает следующие компоненты (рис. 16):

1. **Управляющий узел, узел имен или сервер имен (NameNode)** – отдельный, единственный в кластере, сервер с программным кодом для управления пространством имен файловой системы, хранящий дерево файлов, а также мета-данные файлов и каталогов. **NameNode** – обязательный компонент кластера HDFS, который отвечает за открытие и закрытие файлов, создание и удаление каталогов, управление доступом со стороны внешних клиентов и соответствие между файлами и блоками, дублированными (реплицированными) на узлах данных. Сервер имён раскрывает для всех желающих расположение блоков данных на машинах кластера.
2. **Secondary NameNode — вторичный узел имен,** отдельный сервер, единственный в кластере, который копирует образ HDFS и лог транзакций операций с файловыми блоками во временную папку,

применяет изменения, накопленные в логе транзакций к образу HDFS, а также записывает его на узел NameNode и очищает лог транзакций. Secondary NameNode необходим для быстрого ручного восстанавления NameNode в случае его выхода из строя.

3. **Узел или сервер данных (DataNode, Node)** – один их множества серверов кластера с программным кодом, отвечающим за файловые операции и работу с блоками данных. **DataNode** – обязательный компонент кластера HDFS, который отвечает за запись и чтение данных, выполнение команд от узла NameNode по созданию, удалению и репликации блоков, а также периодическую отправку сообщения о состоянии (heartbeats) и обработку запросов на чтение и запись, поступающих от клиентов файловой системы HDFS. Стоит отметить, что данные проходят с остальных узлов кластера к клиенту мимо узла NameNode.
4. **Клиент (client)** – пользователь или приложение, взаимодействующий через специальный интерфейс (API – Application Programming Interface) с распределенной файловой системой. При наличии достаточных прав, клиенту разрешены следующие операции с файлами и каталогами: создание, удаление, чтение, запись, переименование и перемещение. Создавая файл, клиент может явно указать размер блока файла (по умолчанию 64 Мб) и количество создаваемых реплик (по умолчанию значение равно 3-ем).

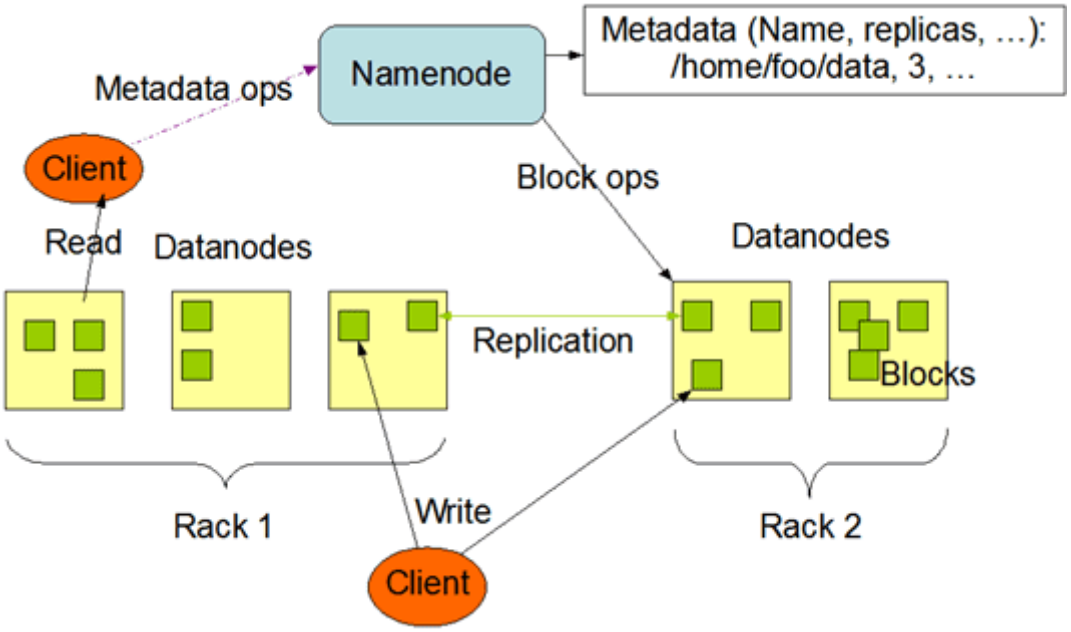


Рис. 16. Архитектура HDFS (Hadoop Distributed File System)

Взаимодействие узлов имен, узлов данных и клиентов осуществляется по протоколам на основе TCP/IP.

Благодаря репликации блоков по узлам данных, распределенная файловая система Hadoop обеспечивает высокую надежность хранения данных и скорость вычислений. Кроме того, **HDFS обладает рядом отличительных свойств**:

- **большой размер блока** по сравнению с другими файловыми системами (>64MB), поскольку HDFS предназначена для хранения большого количества огромных (>10GB) файлов;
- **ориентация на недорогие и, поэтому не самые надежные сервера** – отказоустойчивость всего кластера обеспечивается за счет репликации данных;
- **зеркалирование и репликация** осуществляются на уровне кластера, а не на уровне узлов данных;
- **репликация происходит в асинхронном режиме** – информация распределяется по нескольким серверам прямо во время загрузки, поэтому выход из строя отдельных узлов данных не повлечет за собой полную пропажу данных;

- **HDFS оптимизирована для потоковых считываний файлов**, поэтому применять ее для нерегулярных и произвольных считываний нецелесообразно;
- **клиенты могут считывать и писать файлы HDFS напрямую** через программный интерфейс Java;
- **файлы пишутся однократно,** что исключает внесение в них любых произвольных изменений;
- **принцип WORM (Write-once and read-many**, один раз записать – много раз прочитать) полностью освобождает систему от блокировок типа «запись-чтение». Запись в файл в одно время доступен только одному процессу, что исключает конфликты множественной записи.
- **HDFS оптимизирована под потоковую передачу данных**;
- **сжатие данных и рациональное использование дискового пространства** позволило снизить нагрузку на каналы передачи данных, которые чаще всего являются узким местом в распределенных средах;
- **самодиагностика** – каждый узел данных через определенные интервалы времени отправляет диагностические сообщения узлу имен, который записывает логи операций над файлами в специальный журнал;
- **все метаданные сервера имен хранятся в оперативной памяти**.

В связи с особенностями архитектуры и принципом действия, **для HDFS характерны следующие недостатки**:

- сервер имен является центральной точкой всего кластера и его отказ повлечет сбой системы целиком;
- отсутствие полноценной репликации Secondary NameNode;
- отсутствие возможности дописывать или оставить открытым для записи файлы в HDFS, за счет чего в классическом дистрибутиве Apache Hadoop невозможно обновлять блоки уже записанных данных;
- отсутствие поддержки реляционных моделей данных;
- отсутствие инструментов для поддержки ссылочной целостности данных, что не гарантирует идентичность реплик. HDFS перекладывает

проверку целостности данных на клиентов. При создании файла клиент рассчитывает контрольные суммы каждые 512 байт, которые в последующем сохраняются на сервере имен. При считывании файла клиент обращается к данным и контрольным суммам. В случае их несоответствия происходит обращение к другой реплике.

Когда набор данных перерастает емкость одной физической машины, его приходится распределять по нескольким разным машинам. Файловые системы, управляющие хранением данных в сети, называются распределенными файловыми системами. Поскольку они работают в сетевой среде, проектировщику приходится учитывать все сложности сетевого программирования, поэтому распределенные файловые системы сложнее обычных дисковых файловых систем. Например, одна из самых серьезных проблем – сделать так, чтобы файловая система переживала сбои отдельных узлов без потери данных.

Hadoop поставляется с распределенной файловой системой, которая называется **HDFS (Hadoop Distributed File System)**.

Файловая система HDFS спроектирована для хранения очень больших файлов с потоковой схемой доступа к данным в кластерах обычных машин. Рассмотрим это утверждение более подробно.

Строение HDFS

Очень большие файлы

Под «очень большими» в этом контексте подразумеваются файлы, размер которых составляет сотни мегабайт, гигабайт и терабайт. Сейчас существуют кластеры Hadoop, в которых хранятся петабайты данных.

Потоковый доступ к данным

В основу HDFS заложена концепция однократной записи/многократного чтения как самая эффективная схема обработки данных. Набор данных обычно генерируется или копируется из источника, после чего с ним выполняются различные аналитические операции. В каждой операции задействуется

большая часть набора данных (или весь набор), поэтому время чтения всего набора данных важнее задержки чтения первой записи.

Обычное оборудование

Hadoop не требует дорогостоящего оборудования высокой надежности. Система спроектирована для работы на стандартном оборудовании (общедоступное оборудование, которое может быть приобретено у многих фирм) с достаточно высокой вероятностью отказа отдельных узлов в кластере (по крайней мере, для больших кластеров). Технология HDFS спроектирована таким образом, чтобы в случае отказа система продолжала работу без сколько-нибудь заметного прерывания.

Также следует *выделить области применения, для которых в настоящее время HDFS подходит не лучшим образом* (при том, что в будущем ситуация может измениться):

Быстрый доступ к данным

Приложения, требующие доступа к данным с минимальной задержкой (в диапазоне десятков миллисекунд), плохо сочетаются с HDFS. Напомним, что система HDFS оптимизирована для обеспечения высокой пропускной способности передачи данных, за которую приходится расплачиваться замедлением доступа. HBase в настоящее время лучше подходит для организации доступа к данным с минимальной задержкой.

Многочисленные мелкие файлы

Так как узел имен хранит метаданные файловой системы в памяти, предел количества файлов в файловой системе определяется объемом памяти узла имен. Как показывает опыт, каждый файл, каталог и блок занимают около 150 байт. Таким образом, например, если у вас имеется миллион файлов, каждый из которых занимает один блок, для хранения информации потребуется не менее 300 Мбайт памяти. Хранение миллионов файлов еще приемлемо, но миллиарды файлов уже выходят за пределы возможностей современного оборудования.

Множественные источники записи, произвольные модификации файлов

Запись в файлы HDFS может выполняться только одним источником. Запись всегда осуществляется в конец файла. Поддержка множественных источников записи или модификации с произвольным смещением в файле отсутствует. (Может быть, эти возможности будут поддерживаться в будущем, но, скорее всего, они будут относительно неэффективными).

Основные концепции HDFS

Блоки

С дисковым устройством связывается размер блока – минимальный объем данных, используемых в операции чтения или записи. Однодисковые файловые системы используют это обстоятельство, возвращая данные в блоках, размер которых кратен размеру дискового блока. Размер блока файловой системы обычно составляет несколько килобайт, тогда как размер дискового блока обычно равен 512 байтам. Как правило, все эти нюансы полностью прозрачны для пользователей файловых систем, которые просто читают или записывают в файлы данные произвольной длины. Однако программы сопровождения файловых систем (такие, как df и fsck) работают на уровне блоков файловой системы.

В HDFS тоже существует концепция блока, но он имеет существенно больший размер – по умолчанию 128 Мбайт. Как и в однодисковых файловых системах, файлы HDFS разбиваются на блочные фрагменты, хранимые независимо друг от друга. В отличие от однодисковых файловых систем, файл HDFS, меньший одного блока, не занимает все пространство, выделенное под блок. Если в тексте не указано обратное, термином «блок» в книге обозначаются блоки HDFS.

Почему блоки HDFS такие большие?

Блоки HDFS существенно больше дисковых блоков. Это сделано для того, чтобы свести к минимуму количество операций позиционирования. При достаточно большом размере блока время передачи данных с диска может быть намного больше времени позиционирования к началу блока. Таким образом, время передачи большого файла, состоящего из многих блоков,

определяется скоростью передачи данных. Несложные вычисления показывают, что если время позиционирования составляет около 10 миллисекунд, а скорость передачи составляет 100 Мбайт/с, то, чтобы время позиционирования составляло 1% от времени передачи, размер блока должен составлять примерно 100 Мбайт. По умолчанию используется значение 128 Мбайт, хотя во многих установках HDFS размер блока увеличен. Можно ожидать, что с ростом скорости передачи данных в новых поколениях дисковых накопителей размер блока будет расти. Впрочем, с увеличением размера блока не стоит заходить слишком далеко. Задачи отображения в MapReduce обычно работают с одним блоком, так что при малом количестве задач (меньшем числа узлов в кластере) ваши задания будут выполняться медленнее, чем могли бы.

Абстракция блоков в распределенной файловой системе имеет несколько преимуществ.

Первое преимущество наиболее очевидно: **файл может быть больше любого отдельного диска в сети**. Блоки файла совершенно необязательно хранить на одном диске; они могут использовать любые диски в кластере. Более того, при хранении файла в кластере HDFS его блоки могут быть распределены по всем дискам кластера (хотя эта ситуация несколько нетипична).

Второе: **использование в качестве абстрактной единицы блока вместо файла упрощает подсистему хранения**. Простота – желанное свойство любых систем, но она особенно важна в распределенных системах с их разнообразными возможными режимами отказов. Использование блоков в подсистеме хранения данных упрощает хранение (так как блоки имеют фиксированный размер, система может легко вычислить количество блоков, которые могут храниться на заданном диске) и устраняет проблемы с метаданными (блок – всего лишь фрагмент данных, предназначенный для сохранения, с ним не нужно сохранять метаданные файла – скажем,

информацию о разрешениях доступа, которые могут отдельно обрабатываться другой системой).

Кроме того, **блоки хорошо вписываются в механизм репликации – они улучшают отказоустойчивость и доступность системы**. Для защиты от поврежденных блоков и сбоев дисков/машин, каждый блок реплицируется на небольшом количестве физически разделенных машин (как правило, трех). Если блок становится недоступным, его копия читается из другого места, причем это происходит прозрачно для клиента. Блок, ставший недоступным из-за повреждения данных или сбоя оборудования, копируется из альтернативных хранилищ на другие работоспособные машины, чтобы довести фактор репликации до нормального уровня.

Кроме того, некоторые приложения могут установить высокий фактор репликации для блоков часто запрашиваемого файла, чтобы улучшить распределение нагрузки по чтению в пределах кластера.

Узлы имен и узлы данных

Кластер HDFS состоит из двух типов узлов, работающих по схеме «управляющий-подчиненный»: узел имен (управляющий) и несколько узлов данных (подчиненные). Узел имен управляет пространством имен файловой системы. Он поддерживает дерево файловой системы и метаданные всех файлов и каталогов в дереве. Информация хранится на локальном диске в виде двух файлов: образа пространства имен и журнала изменений. Узел имен также знает, на каких узлах данных хранятся все блоки заданного файла; однако информация о местонахождении блоков не хранится постоянно, а строится заново по сведениям узлов данных при запуске системы.

Чтобы обратиться к файловой системе по поручению пользователя, клиент связывается с узлом имен и узлами данных. Клиент предоставляет интерфейс файловой системы, сходный с интерфейсом POSIX (Portable Operating System Interface), так что пользовательскому коду не нужно ничего знать об узлах имен и узлах данных.

Узлы данных – основная «рабочая сила» файловой системы. Они читают и записывают блоки (по требованию клиентов или узла имен), а также периодически передают узлу имен список сохраняемых ими блоков.

Без узла имен файловая система становится неработоспособной. Более того, при уничтожении машины, на которой работает узел имен, все файлы в файловой системе будут потеряны, потому что восстановить их по блокам узлов данных будет невозможно.

По этой причине важно сделать узел имен устойчивым к возможным сбоям. Hadoop предоставляет два механизма решения этой задачи.

Первый способ – архивация файлов, определяющих устойчивое состояние метаданных файловой системы. Hadoop можно настроить таким образом, чтобы узел имен записывал свое устойчивое состояние в несколько файловых систем. Операции записи выполняются синхронно и атомарно. Обычно в конфигурации настраивается запись на локальный диск и в удаленный том NFS.

Также можно запустить вторичный узел имен, который, несмотря на название, не выполняет функции узла имен. Его основная функция – периодическое слияние образа пространства имен с журналом изменений, чтобы предотвратить чрезмерное разрастание последнего. Вторичный узел имен обычно работает на отдельной физической машине, потому что для выполнения слияния необходимы значительная вычислительная мощность и столько же памяти, сколько на узле имен. Вторичный узел имен хранит копию объединенного образа пространства имен, который может использоваться в случае сбоя основного узла. Тем не менее состояние вторичного узла имен несколько отстает от состояния основного узла, так что в случае полного отказа последнего потеря данных практически неизбежна. Обычно в таких случаях файлы метаданных узла имен копируются из NFS на вторичный узел, который начинает работать как новый основной.

Блокировка кеширования

Обычно узел данных считывает блоки с диска, но для часто используемых файлов блоки могут быть явно кэшированы в памяти узла данных, в блочном кэше вне кучи. По умолчанию блок кэшируется только в памяти одного узла данных, хотя число настраивается для каждого файла. Планировщики заданий (для MapReduce, Spark и других фреймворков) могут использовать преимущества кэшированных блоков, выполняя задачи на узле данных, где кэшируется блок, для повышения производительности чтения. Например, небольшая таблица поиска, используемая в объединении, является хорошим кандидатом для кэширования.

Пользователи или приложения указывают namenode, какие файлы кэшировать (и на какой срок), добавляя директиву cache в пул кеша. Пулы кэша – это административная группа для управления разрешениями кеширования и использованием ресурсов.

Технология HDFS Federation

Узел имен хранит ссылки на все файлы и блоки файловой системы в памяти; это означает, что в очень больших кластерах с множеством файлов память становится ограничивающим фактором масштабирования. Технология HDFS Federation, появившаяся в серии выпусков 2.x, обеспечивает возможность масштабирования кластера за счет добавления узлов имен, каждый из которых управляет частью пространства имен файловой системы. Например, один узел имен может управлять всеми файлами иерархии /user, а второй – файлами иерархии /share.

При использовании федерации каждый узел имен управляет томом пространства имен, состоящим из метаданных пространства имен и пула блоков, содержащего все блоки файлов в пространстве. Тома пространств имен независимы друг от друга; это означает, что узлы имен не обмениваются данными друг с другом, а сбой одного узла имен не влияет на доступность пространств имен, находящихся под управлением других узлов. Вместе с тем,

пул блоков разбиению не подвергается, так что узлы данных регистрируются у каждого узла имен в кластере и хранят блоки из разных пулов.

Высокая доступность HDFS

Сочетание репликации метаданных узлов имен в нескольких файловых системах и использования вторичного узла имен для создания контрольных точек защищает от потери данных, но не обеспечивает высокой доступности системы. Узел имен по-прежнему остается единой точкой сбоя. Если в этой точке происходит сбой, все клиенты – включая задания MapReduce – теряют возможность чтения, записи или получения списка файлов, потому что узел имен является единственным хранилищем метаданных и информации о соответствии между файлами и блоками. В этом случае вся система Hadoop фактически утрачивает работоспособность, пока не будет запущен новый узел.

Чтобы восстановиться после сбоя узла имен в такой ситуации, администратор запускает новый основной узел имен с одной из реплик метаданных файловой системы и настраивает узлы данных и клиентов на использование нового узла. Новый узел имен не может обслуживать запросы до того, как 1) в память будет загружен образ пространства имен, 2) будет воспроизведен журнал изменений и 3) от узлов данных будет получено достаточно отчетов для выхода из безопасного режима. В больших кластерах с множеством файлов и блоков время запуска узла имен может составлять 30 и более минут.

Долгое восстановление также создает проблемы для повседневного сопровождения. Более того, непредвиденные сбои узлов имен настолько редки, что на практике запланированные простои играют более важную роль.

В серии выпусков Hadoop 2.x эта ситуация была исправлена добавлением поддержки высокой доступности (HA, High Availability) HDFS. В этой реализации используются два узла имен в конфигурации «активный/резервный». В случае сбоя активного узла имен резервный узел берет на себя обязанности по обслуживанию клиентских запросов без сколько-нибудь заметного перерыва.

Чтобы это стало возможно, потребовались некоторые архитектурные изменения:

- **Узлы имен должны использовать общее хранилище данных с высокой доступностью для хранения журнала изменений.** При активизации резервный узел имен читает данные до конца общего журнала изменений, чтобы синхронизировать свое состояние с активным узлом имен, после чего продолжает читать новые записи по мере того, как они будут записываться активным узлом имен.
- **Узлы данных должны отправлять блочные отчеты обоим узлам имен,** потому что информация о соответствии блоков хранится в памяти узла имен, а не на диске.
- **Клиенты должны быть настроены на преодоление сбоев узлов имен с использованием механизма, прозрачного для пользователя.**

Если активный узел имен перестает работать, резервный узел может активизироваться очень быстро (за десятые доли секунды), потому что у него в памяти хранится последняя информация состояния: как последние записи журнала изменений, так и актуальная информация соответствия блоков. На практике время преодоления сбоя будет больше (около минуты), потому что решение об отказе активного узла имен должно приниматься системой по возможности консервативно.

В маловероятном случае, когда резервный узел имен оказывается неработоспособным в момент сбоя активного узла, администратор запускает резервный узел «с нуля». Ситуация, по крайней мере, не хуже, чем без высокой доступности; с организационной точки зрения она лучше, потому что процесс представляет собой стандартную процедуру, встроенную в Hadoop.

Преодоление сбоев и изоляция

Переходом от активного узла имен к резервному управляет новый компонент системы, называемый контроллером преодоления сбоев. Контроллеры преодоления сбоев заменяемы; первая реализация использует ZooKeeper для проверки того, что активен только один узел имен. На каждом

узле имен работает облегченный процесс контроллера преодоления сбоев, задача которого состоит в отслеживании сбоев на узле (с использованием простого механизма «проверки пульса») и инициировании преодоления сбоя в случае отказа.

Преодоление сбоев также может инициироваться вручную администратором – например, при регулярном техническом обслуживании. Это называется корректным преодолением сбоев, так как контроллер преодоления сбоев обеспечивает организованное переключение ролей узлов имен.

Однако в случае аварийного преодоления сбоев невозможно быть полностью уверенным в том, что сбойный узел имен перестал работать. Например, медленная работа сети или сетевого раздела может привести к переходу в состояние преодоления сбоев, хотя ранее активный узел имен продолжает работать и полагает, что он все еще является активным. Реализация высокой доступности прикладывает большие усилия, чтобы предотвратить возможные повреждения данных со стороны ранее активного узла имен – этот метод называется изоляцией.

Диапазон механизмов ограждения включает в себя отмену доступа namenode к каталогу общего хранилища и отключение его сетевого порта с помощью команды удаленного управления. В крайнем случае, ранее активный namenode может быть огражден с помощью техники, довольно наглядно известной как STONITH, или «выстрелить другому узлу в голову», которая использует специализированный блок распределения мощности для принудительного выключения главной машины.

Преодоление сбоев на стороне клиента выполняется прозрачно клиентской библиотекой. Простейшая реализация использует для управления преодолением сбоев данные конфигурации на стороне клиента. HDFS URI использует логическое имя хоста, отображаемое на пару адресов узлов имен (в конфигурационном файле), а клиентская библиотека поочередно опробует каждый адрес, пока операция не завершится успехом.

Поток данных. Чтение файла

Чтобы получить представление о том, как организована передача данных между клиентом, взаимодействующим с HDFS, узлом имен и узлами данных, взгляните на рисунок 17. На нем изображена основная последовательность событий при чтении файла.

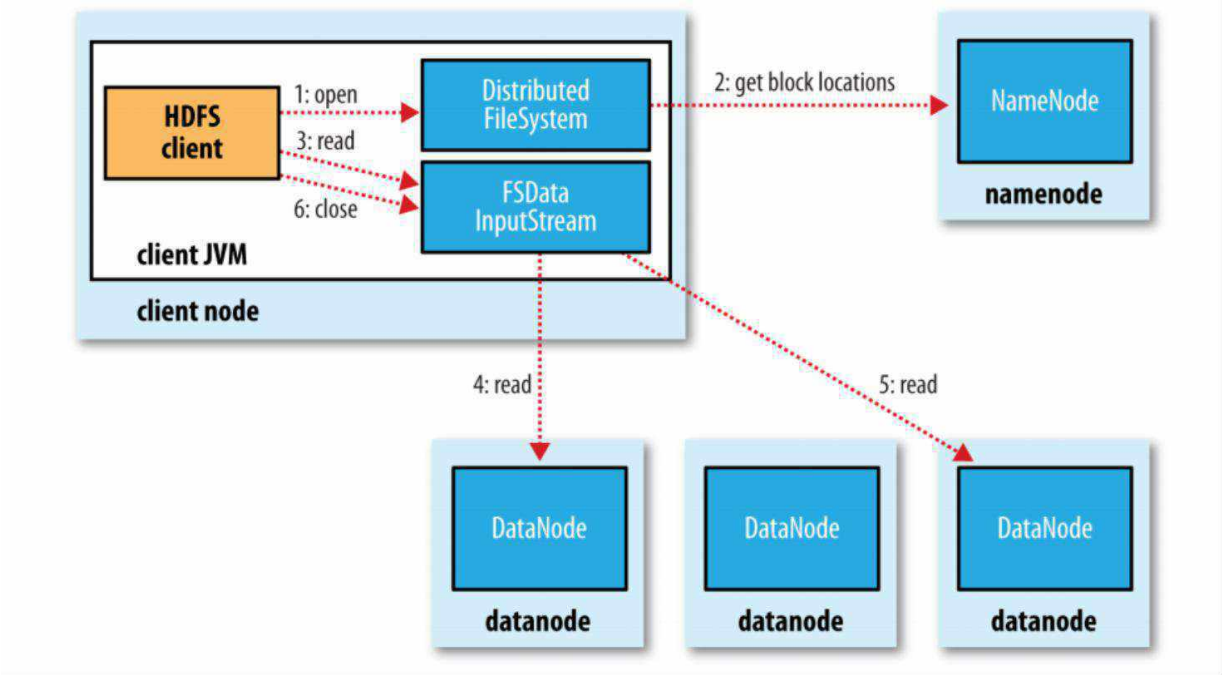


Рис. 17. Основная последовательность событий при чтении файла

Клиент открывает файл, который он хочет прочитать, вызывая open () для объекта FileSystem, который для HDFS является экземпляром DistributedFileSystem (шаг 1 на рисунке 17). DistributedFileSystem вызывает узел имен, используя удаленную процедуру, чтобы определить расположение первых нескольких блоков в файле (шаг 2). Для каждого блока namenode возвращает адреса узлов данных, у которых есть копия этого блока. Кроме того, узлы данных сортируются по их близости к клиенту. Если клиент сам является узлом данных (например, в случае задачи MapReduce), клиент будет читать с локального узла данных, если этот узел данных содержит копию блока.

Объект DistributedFileSystem возвращает объект FSDataInputStream (входной поток с поддержкой позиционирования), из которого клиент читает

данные. В свою очередь, FSDataInputStream инкапсулирует объект DFSInputStream, управляющий вводом/выводом узлов данных и узла имен.

Затем клиент вызывает для потока read() (шаг 3). Объект DFSInputStream, располагающий адресами узлов данных нескольких начальных блоков файла, подключается к первому (ближайшему) узлу данных для получения первого блока. Данные передаются с узла данных клиенту, который многократно вызывает read() для потока (шаг 4). При достижении конца блока DFSInputStream закрывает подключение к узлу данных, а затем находит лучший узел данных для следующего блока (шаг 5). Все происходящее прозрачно для клиента, с точки зрения которого данные просто читаются из непрерывного потока.

Блоки читаются по порядку, объект DFSInputStream открывает новые подключения к узлам данных по мере чтения. При этом он обращается к узлу имен за информацией о местонахождении узлов данных для следующей группы блоков, когда в них возникнет необходимость. Завершив чтение, клиент вызывает close() для FSDataInputStream (шаг 6).

Если в процессе чтения при взаимодействии с узлом данных происходит ошибка, DFSInputStream пытается использовать следующий ближайший узел данных для этого блока. Он также запоминает сбойные узлы данных и не пытается снова использовать их для последующих блоков. DFSInputStream также проверяет контрольные суммы данных, полученных им от узла данных. Обнаружив поврежденный блок, он сообщает о нем узлу имен, прежде чем пытаться читать копию блока с другого узла данных.

Важная особенность этой архитектуры заключается в том, что клиент напрямую обращается к узлам данных для получения данных, а узел имен помогает ему подобрать лучший узел данных для каждого блока. Такая архитектура обеспечивает масштабирование HDFS для большого количества одновременно обслуживаемых клиентов, потому что трафик данных по всем узлам данных в кластере. При этом узел имен только обслуживает запросы местонахождения блоков (делая это чрезвычайно эффективно, поскольку

информация хранится в памяти) и не пытается предоставлять данные, чтобы не создавать «узких мест» при увеличении количества клиентов.

Поток данных. Запись в файлы.

Перейдем к рассмотрению записи в файлы HDFS. Мы рассмотрим ситуацию с созданием нового файла, записью данных в него и закрытием.

Клиент создает файл, вызывая метод create() объекта DistributedFileSystem (шаг 1 на рис. 18). DistributedFileSystem обращается с вызовом RPC к узлу имен для создания в пространстве имен файловой системы нового файла, с которым не связан ни один блок (шаг 2). Узел имен выполняет различные проверки: он убеждается в том, что файл не существует, а клиент обладает необходимыми разрешениями для его создания.

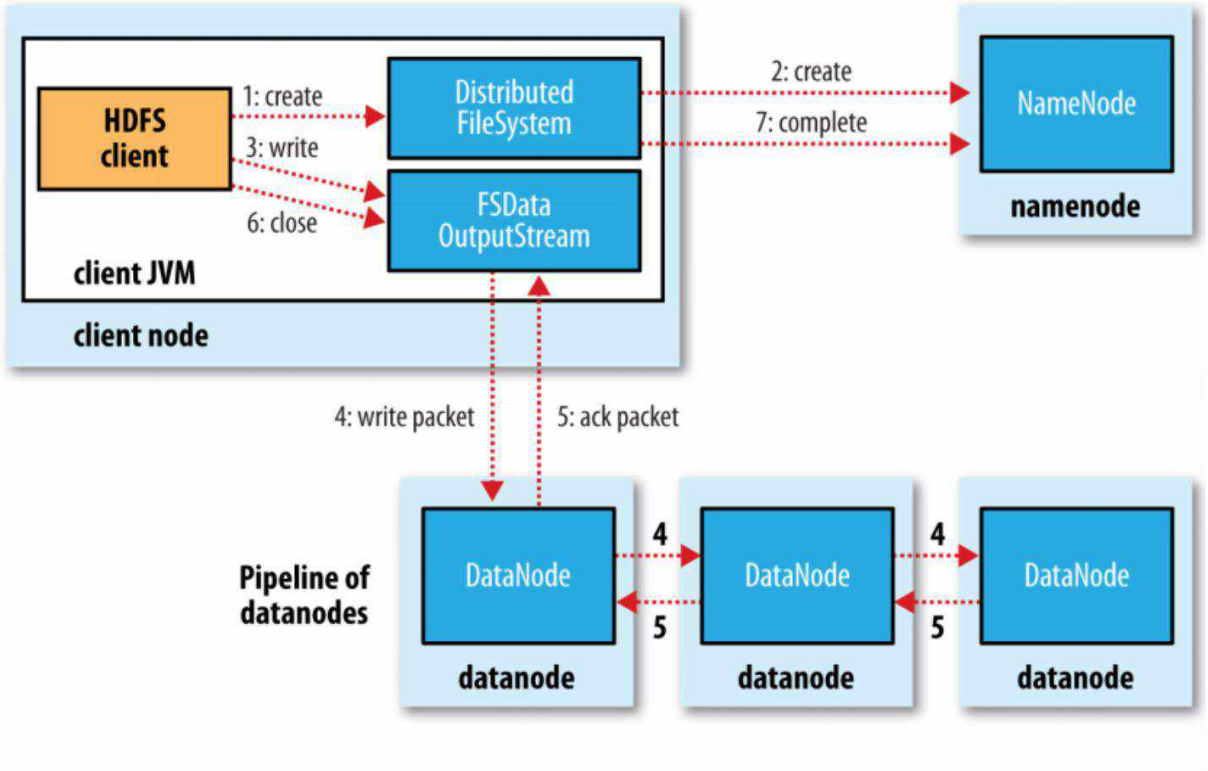


Рис. 18. Запись в файлы

Если все проверки проходят успешно, узел имен сохраняет информацию о новом файле; в противном случае операция создания файла завершается неудачей и у клиента происходит исключение IOException. DistributedFileSystem возвращает клиенту объект FSDataOutputStream, в который можно записывать данные. Как и при чтении, FSDataOutputStream

содержит объект DFSOutputStream, который обеспечивает все взаимодействия с узлами данных и узлом имен.

Во время записи данных клиентом (шаг 3) DFSOutputStream разбивает их на пакеты, которые ставятся во внутреннюю очередь, называемую очередью данных. Очередь данных обслуживается объектом DataStreamer, который обращается к узлу имен с запросом на выделение новых блоков; для этого узел имен должен выбрать узлы данных, подходящие для хранения реплик. Список узлов данных образует конвейер. Будем считать, что коэффициент репликации равен 3, так что конвейер состоит из трех узлов. DataStreamer отправляет пакеты первому узлу данных в конвейере, тот сохраняет пакеты и пересылает их второму узлу данных в конвейере. Аналогичным образом второй узел данных сохраняет пакеты и пересылает их третьему (и последнему) узлу данных в конвейере (шаг 4).

DFSOutputStream также поддерживает внутреннюю очередь пакетов, ожидающих подтверждения со стороны узлов данных; она называется очередью подтверждения. Пакет удаляется из очереди подтверждения только после того, как он будет подтвержден всеми узлами данных в конвейере (шаг 5).

Если в узле данных произойдет сбой во время записи данных, выполняются следующие действия (полностью прозрачные для клиента, записывающего данные): прежде всего, конвейер закрывается, а все пакеты в очереди подтверждения добавляются в начало очереди данных, чтобы избежать потери пакетов узлами данных за сбойным узлом. Текущий блок на работоспособных узлах данных получает новую идентификационную информацию, которая передается узлу имен, так что частично записанный блок на сбойном узле данных будет удален при последующем восстановлении работоспособности узла. Сбойный узел данных удаляется из конвейера, а оставшаяся часть данных блока записывается в два работоспособных узла данных в конвейере. Узел имен замечает, что количество реплик блока

недостаточно, и организует создание дополнительной реплики на другом узле. После этого все последующие блоки обрабатываются по обычной схеме.

Во время записи блока теоретически возможен (хотя и маловероятен) сбой сразу нескольких узлов данных. При условии записи dfs.replication.min реплик (по умолчанию 1) операция завершается успешно, а блок асинхронно реплицируется в кластере до достижения целевого коэффициента репликации (dfs.replication, по умолчанию используется значение 3).

Завершив запись данных, клиент вызывает для потока close() (шаг 6). Все оставшиеся пакеты направляются в конвейер узлов данных, а связь с узлом имен осуществляется после подтверждения завершения записи файла (шаг 7). Узел имен уже знает, из каких блоков состоит файл (информация передается DataStreamer при запросах на распределение блоков); остается лишь дождаться минимальной репликации блоков, после чего возвращается признак успешного завершения операции.

Источник: *Hadoop: The Definitive Guide, Fourth Edition by Tom White Copyright © 2015 Tom White. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.*

NoSQL

Когда люди используют термин «база данных NoSQL», они обычно используют его для обозначения любой нереляционной базы данных. Некоторые говорят, что термин «NoSQL» означает «не SQL», в то время как другие говорят, что он означает «не только SQL». В любом случае, большинство согласны с тем, что базы данных NoSQL – это базы данных, которые хранят данные в формате, отличном от реляционных таблиц.

Базы данных NoSQL появились в конце 2000-х годов, когда стоимость хранения резко снизилась. Прошли те времена, когда нужно было создавать сложную, трудноуправляемую модель данных, чтобы избежать дублирования данных. Разработчики (а не хранилище) становились основной статьей

расходов при разработке программного обеспечения, поэтому базы данных NoSQL были оптимизированы для повышения производительности разработчиков.

Поскольку затраты на хранение быстро снижались, объем данных, которые приложения должны были хранить и запрашивать, увеличивался. Эти данные были самых разных форм и размеров – структурированные, полуструктурированные и полиморфные – и заранее определить схему стало практически невозможно. Базы данных NoSQL позволяют разработчикам хранить огромные объемы неструктурированных данных, что дает им большую гибкость.

Кроме того, популярность Agile Manifesto росла, и инженеры-программисты переосмысливали свои методы разработки программного обеспечения. Они признавали необходимость быстрой адаптации к изменяющимся требованиям. Им нужна была возможность быстро выполнять итерации и вносить изменения в свой программный стек – вплоть до базы данных. Базы данных NoSQL предоставили им такую гибкость.

Популярность облачных вычислений также возросла, и разработчики начали использовать общедоступные облака для размещения своих приложений и данных. Им нужна была возможность распределять данные между несколькими серверами и регионами, чтобы сделать их приложения отказоустойчивыми, масштабировать, а не увеличивать, и интеллектуально распределять данные по географическому положению.

База данных NoSQL предоставляет механизм для хранения и извлечения данных, которые моделируются средствами, отличными от табличных отношений, используемых в реляционных базах данных. Базы данных NoSQL все чаще используются в больших данных и веб-приложениях реального времени. Системы NoSQL также иногда называют не только SQL, чтобы подчеркнуть, что они могут поддерживать SQL – подобные языки запросов или располагаться рядом с базами данных SQL в многоязычных персистентных архитектурах.

Мотивы для этого подхода включают простоту дизайна, более простое «горизонтальное» масштабирование до кластеров машин (что является проблемой для реляционных баз данных), более точный контроль над доступностью. Структуры данных, используемые базами данных NoSQL отличаются от используемых по умолчанию в реляционных базах данных, что ускоряет выполнение некоторых операций в NoSQL. Конкретная пригодность данной базы данных NoSQL зависит от проблемы, которую она должна решить. Иногда структуры данных, используемые базами данных NoSQL, также рассматриваются как «более гибкие», чем таблицы реляционных баз данных.

Многие хранилища NoSQL компрометируют согласованность в пользу доступности, устойчивости к разделам и скорости. Препятствия для более широкого внедрения хранилищ NoSQL включают использование низкоуровневых языков запросов (например, вместо SQL), отсутствие возможности выполнять специальные соединения между таблицами, отсутствие стандартизированных интерфейсов и огромные предыдущие инвестиции в существующие реляционные базы данных. В большинстве хранилищ NoSQL отсутствуют настоящие ACID-транзакции, хотя некоторые базы данных сделали их центральными в своих разработках.

Вместо этого большинство баз данных NoSQL предлагают концепцию «согласованности в конечном счете», при которой изменения в базе данных распространяются на все узлы «в конце концов» (обычно в течение миллисекунд), поэтому запросы данных могут не возвращать обновленные данные немедленно или могут приводить к чтению данных, которые не точны, проблема, известная как устаревшие чтения. Кроме того, в некоторых системах NoSQL могут наблюдаться потерянные записи и другие формы потери данных. Некоторые системы NoSQL поддерживают такие концепции, как ведение журнала с упреждающей записью, чтобы избежать потери данных. Для распределенной обработки транзакций в нескольких базах данных согласованность данных является еще более серьезной проблемой, которая

сложна как для NoSQL, так и для реляционных баз данных. Немногие системы поддерживают как транзакции ACID, так и стандарты X/Open XA для распределенной обработки транзакций. Ограничения в интерфейсной среде преодолеваются с помощью протоколов семантической виртуализации, благодаря чему службы NoSQL доступны для большинства операционных систем.

Каждая база данных NoSQL имеет свои уникальные особенности. На высоком уровне многие базы данных NoSQL имеют следующие функции:

- Гибкие схемы
- Горизонтальное масштабирование
- Быстрые запросы благодаря модели данных
- Простота использования для разработчиков

Со временем появилось четыре основных типа баз данных NoSQL: document databases (базы данных документов), key-value databases (базы данных «ключ-значение»), wide-column stores (хранилища с широкими столбцами) и graph databases (базы данных графов).

Рассмотрим некоторые из них.

key-value databases

Базы данных «ключ-значение», также известные как хранилища «ключ-значение», – это типы баз данных, в которых данные хранятся в формате «ключ-значение» и оптимизированы для чтения и записи этих данных. Данные извлекаются с помощью уникального ключа или ряда уникальных ключей для получения связанного значения с каждым ключом. Значения могут быть простыми типами данных, такими как строки и числа, или сложными объектами.

Пара «ключ-значение» не являются новой концепцией и уже были с нами в течение последних нескольких десятилетий. Одним из известных хранилищ является старый реестр Windows, позволяющий системе/приложениям хранить данные в структуре «ключ-значение», где ключ может быть представлен в виде уникального идентификатора или уникального пути к

значению. Данные записываются (вставляются, обновляются и удаляются) и запрашиваются на основе ключа для сохранения/получения его значения.

В своей простейшей форме хранилище ключ-значение похоже на объект словаря/массива, который существует в большинстве парадигм программирования, но хранится постоянно и управляется системой управления базами данных. Базы данных типа "ключ-значение" используют компактные, эффективные структуры индексов, чтобы иметь возможность быстро и надежно находить значение по его ключу, что делает их идеальными для систем, которым необходимо иметь возможность находить и извлекать данные за постоянное время.

Существует несколько вариантов использования, в которых выбор хранилища значений ключей является оптимальным решением:

- Произвольный доступ к данным в режиме реального времени, например, к атрибутам сеанса пользователя в онлайн-приложении, таком как игры или финансы.
- Механизм кэширования часто используемых данных или конфигурации на основе ключей.
- Приложение разработано на основе простых запросов на основе ключей.

document databases

Базы данных документов предлагают множество преимуществ, в том числе:

- Интуитивно понятная модель данных, с которой разработчикам легко и быстро работать.
- Гибкая схема, позволяющая модели данных развиваться по мере изменения потребностей приложения.
- Возможность горизонтального масштабирования.

Благодаря этим преимуществам базы данных документов являются базами данных общего назначения, которые можно использовать в самых разных случаях и отраслях.

Базы данных документов считаются нереляционными базами данных. Вместо хранения данных в фиксированных строках и столбцах базы данных документов используют гибкие документы. Базы данных документов являются наиболее популярной альтернативой табличным реляционным базам данных.

Документ – это запись в базе данных документов. Документ обычно хранит информацию об одном объекте и любых связанных с ним метаданных.

Документы хранят данные в парах поле-значение. Значения могут быть различных типов и структур, включая строки, числа, даты, массивы или объекты. Документы могут храниться в таких форматах, как JSON, BSON и XML.

Ниже представлен документ JSON, в котором хранится информация о пользователе по имени Антон.


```
{
    "_id": 1,
    "first_name": "Anton",
    "email": "Anton@sfedu.ru",
    "cell": "8-555-555-55-55",
    "likes": [
      "IT",
      "sport",
      "shopping"
    ],
    "classmate": [
      {
        "name": "Aleksandr",
        "status": "student",
        "date_founded": {
          "$date": "2021-05-19T04:00:00Z"
        }
```

```
    },
    {
      "name": "Andrey",
      "date_founded": {
        "$date": "2021-11-01T04:00:00Z"
      }
    }
  ]
}
```

Коллекция – это группа документов. В коллекциях обычно хранятся документы с похожим содержимым. Не все документы в коллекции должны иметь одинаковые поля, поскольку базы данных документов имеют гибкую схему. Некоторые базы данных документов обеспечивают проверку схемы, поэтому при необходимости схему можно заблокировать.

Коллекция используя гибкую схему для хранения информации и поэтому документы могут не содержать одинаковых полей.

Базы данных документов имеют следующие ключевые особенности:

Модель документа: данные хранятся в документах (в отличие от других баз данных, которые хранят данные в таких структурах, как таблицы или графики). Документы сопоставляются с объектами в большинстве популярных языков программирования, что позволяет разработчикам быстро разрабатывать свои приложения.

Гибкая схема. Базы данных документов имеют гибкую схему, что означает, что не все документы в коллекции должны иметь одинаковые поля.

Распределенная и отказоустойчивая. Базы данных документов распределены, что обеспечивает горизонтальное масштабирование (как правило, дешевле, чем вертикальное масштабирование) и распределение данных. Базы данных документов обеспечивают отказоустойчивость за счет репликации.

Запросы через API или язык запросов. Базы данных документов имеют API или язык запросов, который позволяет разработчикам выполнять операции CRUD (Create, Read, Update, Delete) в базе данных. Разработчики могут запрашивать документы на основе уникальных идентификаторов или значений полей.

in-memory database

База данных в памяти – это программное обеспечение для хранения данных, которое хранит все свои данные в памяти хоста. Основное различие между традиционной базой данных и базой данных в памяти заключается в том, где хранятся данные. Даже по сравнению с твердотельными накопителями (SSD) оперативная память (ОЗУ) на несколько порядков быстрее, чем доступ к диску. Поскольку база данных в памяти использует последнюю для хранения, доступ к данным происходит намного быстрее, чем в традиционной базе данных, использующей дисковые операции.

Базы данных в памяти обеспечивают быстрый доступ к своему содержимому; с другой стороны, они подвержены высокому риску потери данных в случае сбоя сервера, поскольку данные нигде не сохраняются. Если произойдет сбой или отключение сервера, все, что в настоящее время находится в памяти этого компьютера, будет потеряно из-за нестабильной природы ОЗУ. Также стоит отметить, что стоимость памяти намного выше стоимости жестких дисков. Вот почему на современных компьютерах обычно гораздо больше места на жестком диске, чем памяти. Этот фактор делает базы данных в оперативной памяти намного более дорогими. Они также более подвержены риску нехватки места для хранения данных.

Примером использования может быть сайт электронной коммерции с высоким трафиком, на котором необходимо хранить содержимое корзины покупок для сотен тысяч клиентов в любой момент времени. Время отклика в таком масштабе было бы слишком медленным для многих баз данных на дисках, поэтому базы данных в памяти используются, чтобы не отставать от нагрузки и обеспечивать положительный опыт работы с клиентами.

Cassandra

Apache Cassandra – это нереляционная или NoSQL база данных с открытым исходным кодом, которая обеспечивает непрерывную доступность, огромный масштаб и распределение данных по нескольким центрам обработки данных и облачным зонам доступности. Проще говоря, Cassandra предоставляет высоконадежный механизм хранения данных для приложений, требующих огромного масштаба.

Apache Cassandra был разработан Авинашем Лакшманом и Прашантом Маликом, когда они оба работали инженерами в Facebook.

База данных была разработана для поддержки функции поиска в папке «Входящие» Facebook, что позволяет пользователям быстро находить разговоры и другой контент, который они ищут.

Архитектура объединила модель распределения, чтобы обеспечить горизонтальное масштабирование между несколькими узлами, с механизмом хранения с логической структурой. В результате получилась масштабируемая база данных, способная справиться с самыми большими объемами данных и высокими требованиями к производительности.

Cassandra была разработана с учетом масштабируемости, производительности и постоянной доступности в качестве основных принципов архитектуры. Cassandra работает с использованием кольцевой архитектуры без хозяина – она не полагается на отношения ведущий-подчиненный. В Cassandra все узлы играют одинаковую роль; отсутствует концепция главного узла, когда все узлы взаимодействуют друг с другом через распределенный масштабируемый протокол. Записи распределяются между узлами с помощью хеш-функции, а чтения направляются на определенные узлы. Cassandra хранит данные, равномерно распределяя данные по своему кластеру узлов. Каждый узел отвечает за часть данных.

Apache Cassandra можно определить по нескольким основным характеристикам. Прежде всего, это столбцовая база данных. Она отличается высокой согласованностью, устойчивостью к ошибкам и масштабируемостью.

Базовая архитектура состоит из группы узлов. Каждый из этих узлов может принимать запросы на чтение или запись. Это важный аспект, потому что это означает, что нет «главного» узла. Все узлы взаимодействуют одинаково.

В данные находятся в узлах кластера, и кластер представляет собой набор серверов, расположенных в центрах обработки данных, где данные хранятся и обрабатываются. Узлы, связанные друг с другом, группируются в одном центре обработки данных.

Эта структура предназначена для расширения. Если нужно больше места, просто добавляются узлы. Таким образом, систему очень легко расширять по мере необходимости, особенно если количество одновременно работающих пользователей увеличивается.

Эта структура также обеспечивает защиту данных. «Журнал фиксации» – это метод резервного копирования для обеспечения целостности данных и предотвращения потери данных. Данные индексируются и записываются в memtable – структуру данных в памяти, куда Cassandra записывает. Каждый массив имеет свою активную таблицу памяти. Когда memtables достигают своего максимума из-за завершения журнала фиксации, они сбрасываются на диск и становятся неизменяемыми SSTables (таблица сортированных строк). Именно эта система обеспечивает безопасность данных.

CQL

Cassandra CQL, простая альтернатива языку структурированных запросов (SQL), представляет собой декларативный язык, разработанный для обеспечения абстракции при доступе к Apache Cassandra.

Основные конструкции Cassandra (CQL) включают:

Пространство ключей. Подобно базе данных СУБД, пространство ключей представляет собой контейнер для данных приложения, который должен иметь имя и набор связанных атрибутов. Ключевое пространство Cassandra – это база данных SQL.

Таблицы – пространство ключей состоит из нескольких таблиц. Семейство столбцов Cassandra представляет собой таблицу SQL.

Первичный ключ. Первичный ключ состоит из ключа строки/раздела и ключа кластера и позволяет пользователям однозначно идентифицировать внутренние строки данных. Ключ строки/раздела определяет узел, на котором хранятся данные. Кластерный ключ определяет порядок сортировки данных в конкретной строке.

Язык запросов CQL – это интерфейс NoSQL, намеренно похожий на SQL, предоставляющий пользователям, знакомым с реляционными базами данных, знакомый язык, который в конечном счете снижает входной барьер для Apache Cassandra.

Некоторые функции, связанные с SQL, но недоступные в CQL:

WHERE – предикат может содержать только столбцы, указанные в первичном ключе.

ORDER BY – порядок можно применять только к столбцу кластера.

GROUP BY – идентичные данные не могут быть сгруппированы в CQL.

JOIN – данные из разных семейств столбцов невозможно объединить в CQL без сторонних инструментов.

Сложные транзакции. Там, где SQL поддерживает сложные транзакции, CQL поддерживает простые транзакции.

CQL используется в нереляционных базах данных, совместим с Apache Cassandra, использует синтаксис запросов, подобный SQL, допускает взаимозаменяемость терминов «таблица» и «семейство столбцов» и использует собственный двоичный протокол CQL. Запрос CQL также может возвращать объекты. Доступ к объектам в коллекции можно получить через итерацию через поле коллекции в объекте результата. В отличие от сильно структурированных столбцов SQL, поля результатов могут быть коллекциями или самими объектами, а не просто значениями типа данных.

Apache Cassandra CQL поддерживает широкий набор типов данных, включая типы коллекций, пользовательские типы, собственные типы, типы кортежей и определяемые пользователем типы.

Смысл таков, что в NoSQL базах в отличие от реляционных структура данных не регламентирована (или слабо типизированна, если проводить аналогии с языками прогаммирования) – в отдельной строке или документе можно добавить произвольное поле без предварительного декларативного изменения структуры всей таблицы. Таким образом, если появляется необходимость поменять модель данных, то единственное достаточное действие – отразить изменение в коде приложения.

Приятное следствие отсутствия схемы – эффективность работы с разреженными (sparse) данными. Если в одном документе есть поле date_published, а во втором – нет, значит никакого пустого поля date_published для второго создано не будет. Это, в принципе, логично, но менее очевидный пример – column-family NoSQL базы данных, в которых используются знакомые понятия таблиц/колонок. Однако в силу отсутствия схемы, колонки не объявляются декларативно и могут меняться/добавляться во время пользовательской сессии работы с базой. Это позволяет в частности использовать динамические колонки для реализации списков.

У неструктурированной схемы есть свои недостатки – помимо упомянутых выше накладных расходов в коде приложения при смене модели данных – отсутствие всевозможных ограничений со стороны базы (not null, unique, check constraint и т.д.), плюс возникают дополнительные сложности в понимании и контроле структуры данных при параллельной работе с базой разных проектов (отсутствуют какие-либо словари на стороне базы). Впрочем, в условиях быстро меняющегося современного мира такая гибкость является все-таки преимуществом. В качестве примера можно привести Твиттер, который лет пять назад вместе с твиттом хранил лишь немного дополнительной информации (время, Twitter handle и еще несколько байтов

метаинформации), однако сейчас в дополнение к самому сообщению в базе сохраняется еще несколько килобайт метаданных.

Представление данных в виде агрегатов (aggregates)

В отличие от реляционной модели, которая сохраняет логическую бизнес-сущность приложения в различные физические таблицы в целях нормализации, NoSQL хранилища оперируют с этими сущностями как с целостными объектами (рис. 19).

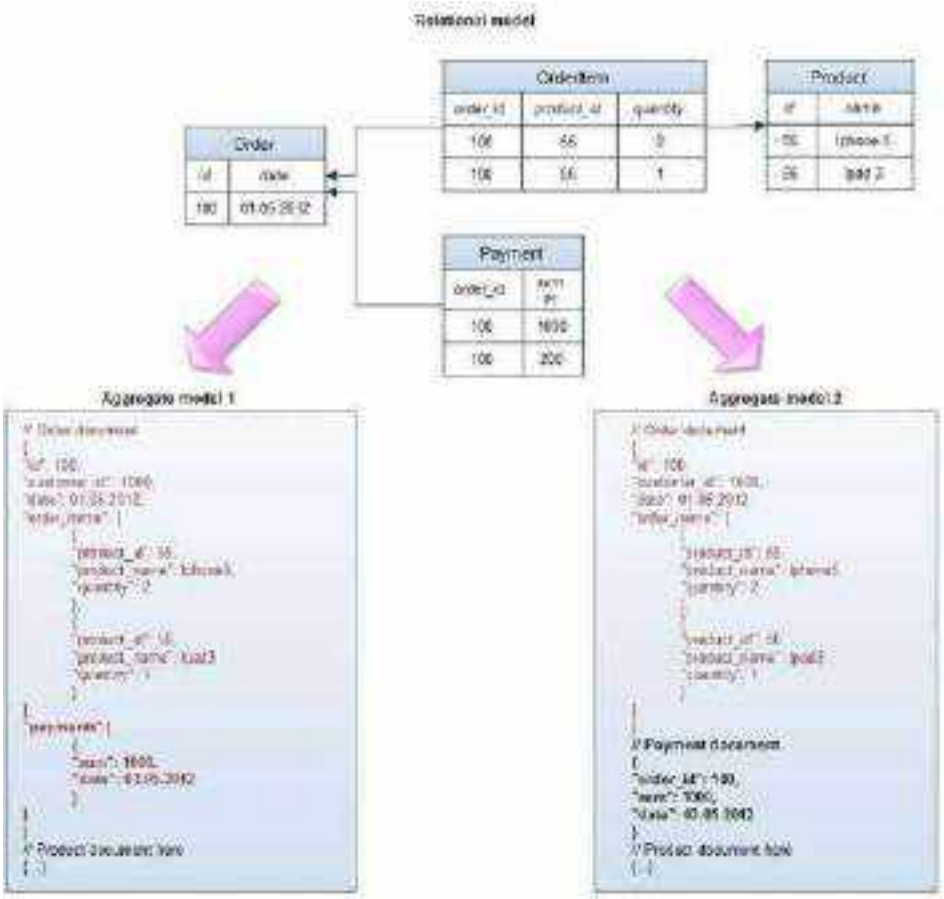


Рис. 19. Работа с агрегатами

В этом примере продемонстрированы агрегаты для стандартной концептуальной реляционной модели e-commerce “заказ – позиции заказа – платежи – продукт”. В обоих случаях заказ объединяется с позициями в один логический объект, при этом каждая позиция хранит в себе ссылку на продукт и некоторые его атрибуты, например, название (такая денормализация необходима, чтобы не запрашивать объект продукта при извлечении заказа — главное правило распределенных систем – минимум “джоинов” между объектами). В одном агрегате платежи объединены с заказом и являются

составной частью объекта, в другом – вынесены в отдельный объект. Этим демонстрируется главное правило проектирования структуры данных в NoSQL базах – она должна подчиняться требованиям приложения и быть максимально оптимизированной под наиболее частые запросы. Если платежи регулярно извлекаются вместе с заказом – имеет смысл их включать в общий объект, если же многие запросы работают только с платежами – значит, лучше их вынести в отдельную сущность.

Многие возразят, заметив, что работа с большими, часто денормализованными, объектами чревата многочисленными проблемами при попытках произвольных запросов к данным, когда запросы не укладываются в структуру агрегатов. Что, если мы используем заказы вместе с позициями и платежами по заказу (так работает приложение), но бизнес просит нас посчитать, сколько единиц определенного продукта было проданно в прошлом месяце? В этом случае вместо сканирования таблицы OrderItem (в случае реляционной модели) нам придется извлекать заказы целиком в NoSQL хранилище, хотя большая часть этой информации нам будет не нужна. К сожалению, это компромисс, на который приходится идти в распределенной системе: мы не можем проводить нормализацию данных как в обычной односерверной системе, так как это создаст необходимость объединения данных с разных узлов и может привести к значительному замедлению работы базы.

Плюсы и минусы обоих подходов показаны на рисунке 20.

Слабые ACID свойства

Долгое время консистентность (consistency) данных была “священной коровой” для архитекторов и разработчиков. Все реляционные базы обеспечивали тот или иной уровень изоляции – либо за счет блокировок при изменении и блокирующего чтения, либо за счет undo-логов. С приходом огромных массивов информации и распределенных систем стало ясно, что обеспечить для них транзакционность набора операций с одной стороны и получить высокую доступность и быстрое время отклика с другой –

невозможно. Более того, даже обновление одной записи не гарантирует, что любой другой пользователь моментально увидит изменения в системе, ведь изменение может произойти, например, в мастер-ноде, а реплика асинхронно скопируется на слейв-ноду, с которой и работает другой пользователь.

Нормализация данных	Данные в виде агрегатов
• Целостность информации при обновлении (меняем запись в одной таблице, а не в нескольких) • Ориентированность на широкий спектр запросов к данных	• Оптимизация только под определенный вид запросов • Сложности при обновлении денормализованных данных
• Неэффективна в распределенной среде • Низкая скорость чтения при использовании объединений (joins) • Несоответствие объектной модели приложения физической структуре данных (impedance mismatch, решается с помощью Hibernate etc.)	• Лучший способ добиться большой скорости на чтение в распределенной среде • Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании) • Родная (native) поддержка атомарности на уровне записей

Рис. 20. Преимущества и недостатки NoSQL

В таком случае он увидит результат через какой-то промежуток времени. Это называется eventual consistency и это то, на что идут сейчас все крупнейшие интернет-компании мира, включая Facebook и Amazon. Последние с гордостью заявляют, что максимальный интервал, в течение которого пользователь может видеть неконсистентные данные составляют не более секунды. Пример такой ситуации показан на рисунке 21.

Логичный вопрос, который появляется в такой ситуации – а что делать системам, которые классически предъявляют высокие требования к атомарности-консистентности операций и в то же время нуждаются в быстрых распределенных кластерах – финансовым, интернет-магазинам и т.д? Практика показывает, что эти требования уже давно неактуальны: вот что сказал один разработчик финансовой банковской системы: “Если бы мы действительно ждали завершения каждой транзакции в мировой сети ATM (банкоматов), транзакции занимали бы столько времени, что клиенты убегали бы прочь в ярости. Что происходит, если ты и твой партнер снимаете деньги

одновременно и превышаете лимит? Вы оба получите деньги, а мы поправим это позже.”

Рис. 21. Рассогласованность данных

Другой пример – бронирование гостиниц, показанный на картинке. Онлайн-магазины, чья политика работы с данными предполагает eventual consistency, обязаны предусмотреть меры на случай таких ситуаций (автоматическое решение конфликтов, откат операции, обновление с другими данными). На практике гостиницы всегда стараются держать “пул” свободных номеров на непредвиденный случай и это может стать решением спорной ситуации.

На самом деле слабые ACID свойства не означают, что их нет вообще. В большинстве случаев приложение, работающее с реляционной базой данных, использует транзакцию для изменения логически связанных объектов (заказ – позиции заказа), что необходимо, так как это разные таблицы. При правильном проектировании модели данных в NoSQL базе (агрегат представляет из себя заказ вместе с перечнем пунктов заказа) можно добиться такого же самого уровня изоляции при изменении одной записи, что и в реляционной базе данных.

Распределенные системы, без совместно используемых ресурсов (share nothing)

Опять же, это не касается графа баз данных, чья структура по определению плохо разносится по удаленным нодам.

Это, возможно, главный лейтмотив развития NoSQL баз. С лавинообразным ростом информации в мире и необходимости ее обрабатывать за разумное время встала проблема вертикальной масштабируемости – рост скорости процессора остановился на 3.5 Ггц, скорость чтения с диска также растет тихими темпами, плюс цена мощного сервера всегда больше суммарной цены нескольких простых серверов. В этой ситуации обычные реляционные базы, даже кластеризованные на массиве дисков, не способны решить проблему скорости, масштабируемости и пропускной способности. Единственный выход из ситуации – горизонтальное масштабирование, когда несколько независимых серверов соединяются быстрой сетью и каждый владеет/обрабатывает только часть данных и/или только часть запросов на чтение-обновление. В такой архитектуре для повышения мощности хранилища (емкости, времени отклика, пропускной способности) необходимо лишь добавить новый сервер в кластер — и все. Процедурами шардинга, репликации, обеспечением отказоустойчивости (результат будет получен даже если одна или несколько серверов перестали отвечать), перераспределения данных в случае добавления ноды занимается сама NoSQL база. Вкратце представлю основные свойства распределенных NoSQL баз.

Репликация – копирование данных на другие узлы при обновлении. Позволяет как добиться большей масштабируемости, так и повысить доступность и сохранность данных. Принято подразделять на два вида: *master-slave* (рис. 22):

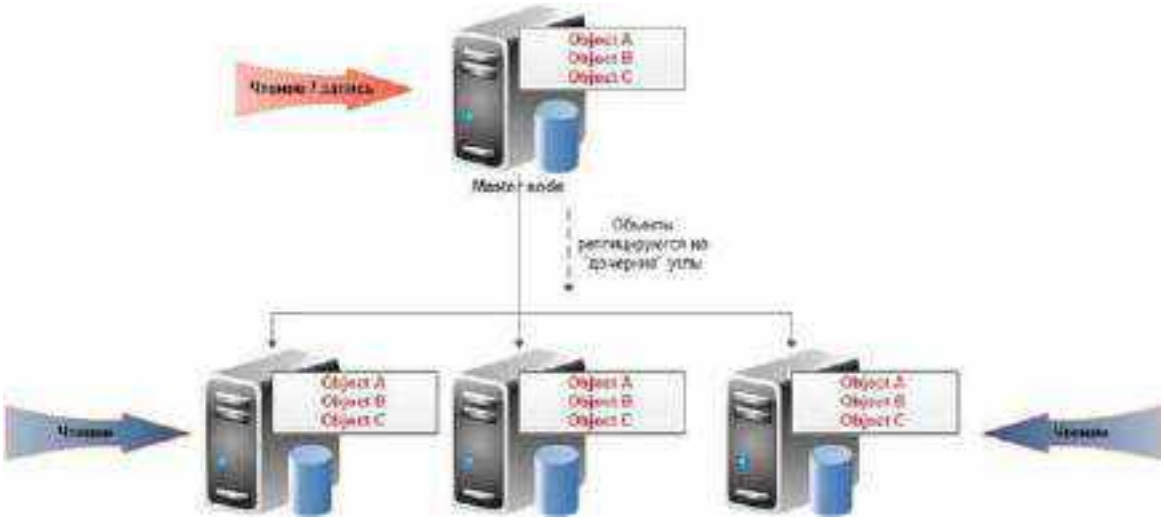


Рис. 22. Master-slave

peer-to-peer (рис. 23):

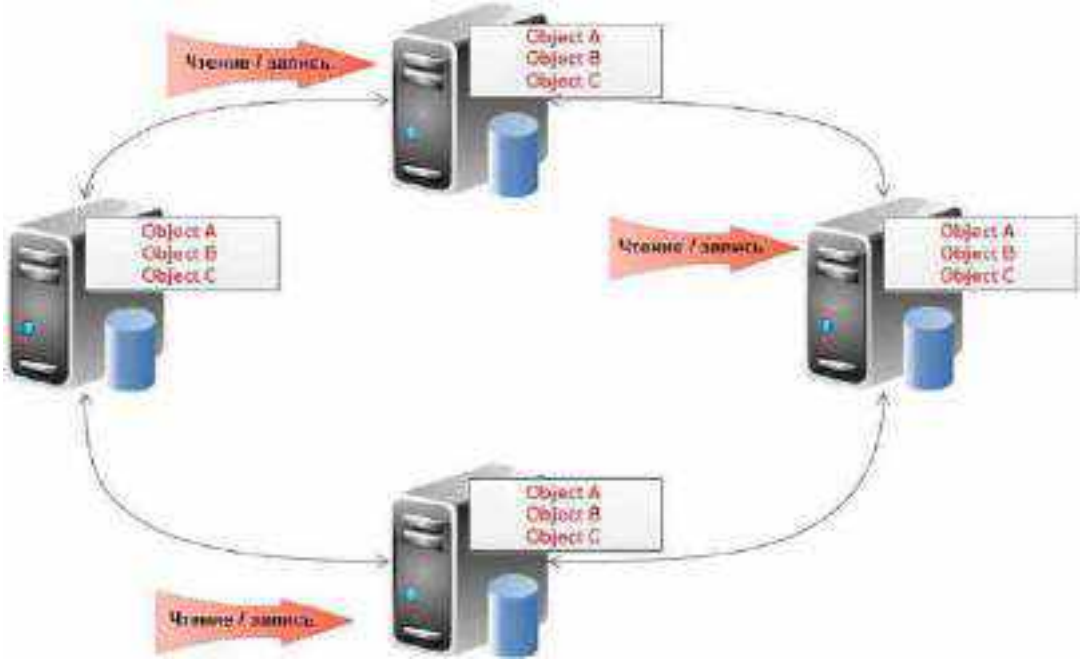


Рис. 23. Peer-to-peer

Первый тип предполагает хорошую масштабируемость на чтение (может происходить с любого узла), но немасштабируемую запись (только в мастер узел). Также есть тонкости с обеспечением постоянной доступности (в случае падения мастера либо вручную, либо автоматически на его место назначается один из оставшихся узлов). Для второго типа репликации предполагается, что все узлы равны и могут обслуживать как запросы на чтение, так и на запись. **Шардинг** — разделение данных по узлам (рис. 24).

Рис. 24. Шардинг

Шардинг часто использовался как “костыль” к реляционным базам данных в целях увеличения скорости и пропускной способности: пользовательское приложение партицировало данные по нескольким независимым базам данных и при запросе соответствующих данных пользователем обращалось к конкретной базе. В NoSQL базах данных шардинг, как и репликация, производятся автоматически самой базой и пользовательское приложение обособленно от этих сложных механизмов.

NoSQL движение набирает популярность гигантскими темпами. Однако это не означает, что реляционные базы данных становятся рудиментом или чем-то архаичным. Скорее всего они будут использоваться и использоваться по-прежнему активно, но все больше в симбиозе с ними будут выступать NoSQL базы. Мы вступаем в эру polyglot persistence — эру, когда для различных потребностей используются разные хранилища данных. Теперь нет монополизма реляционных баз данных, как безальтернативного источника данных. Все чаще архитекторы выбирают хранилище исходя из природы самих данных и того, как мы ими хотим манипулировать, какие объемы информации ожидаются. И поэтому все становится только интереснее.

Источник: https://habr.com/ru/post/152477/

Hbase

Apache Hbase – это нереляционная база данных (NoSQL), включающая в себя концепцию и функции Google BigTable. Разница в том, что HBase является открытым программным обеспечением. Эта база данных обычно работает в распределенной файловой системе Hadoop (HDFS). Она обеспечивает произвольный и последовательный доступ для записи и чтения в режиме реального времени к таблицам, содержащим миллиарды строк и миллионы столбцов.

Это также позволяет комбинировать источники данных на основе разных структур и разных схем. Поэтому это очень хороший выбор для многоструктурного хранения данных. Также можно делать запросы на конкретную метку времени, что позволяет делать запросы «флешбэк».

Изначально интегрированный с Hadoop, HBase работает с другими механизмами доступа к данным через YARN. Его язык программирования — Java.

Apache HBase берет на себя все функции Google BigTable. Таким образом, мы находим фильтры Блума или операции в памяти и сжатие. Таблицы в этой базе данных могут служить входными данными для заданий MapReduce в экосистеме Hadoop, а также могут служить выходными данными после обработки данных MapReduce.

Конкретно, HBase – это хранилище данных типа «ключ-значение», ориентированное на столбцы. Поэтому он может работать со всеми данными, обрабатываемыми Hadoop. Его нельзя использовать для замены базы данных SQL, но можно добавить слой SQL в эту базу данных, чтобы интегрировать ее с различными инструментами бизнес-аналитики и анализа данных.

Преимущества HBase многочисленны. Прежде всего, эта база данных обладает высокой отказоустойчивостью. Данные реплицируются на разные серверы центра обработки данных, а автоматическое аварийное переключение гарантирует высокую доступность. Кроме того, шардинг и балансировка нагрузок и таблиц выполняются автоматически.

Эта нереляционная база данных также работает быстро. Она предлагает поиск практически в реальном времени и обработку на стороне сервера с помощью фильтров и сопроцессоров. Кэширование в памяти также доступно. Последним его преимуществом является универсальность. Модели данных могут иметь множество вариантов использования. Можно экспортировать метрики через плагины File и Ganglia.

Распределенное хранилище, предлагаемое HBase, также позволяет выполнять запросы к огромным объемам данных. Наконец, можно «масштабировать» возможности чтения и записи этой технологии до миллионов в секунду, что позволяет преодолеть ограничения реляционных систем управления базами данных.

Данные организованы в **таблицы**, проиндексированные первичным ключом, который в Hbase называется **RowKey**. Для каждого **RowKey** ключа может храниться неограниченный набор **атрибутов (или колонок)**.

Колонки организованны в **группы колонок**, называемые **Column Family**. Как правило в одну **Column Family** объединяют колонки, для которых одинаковы паттерн использования и хранения (рис. 25). Для каждого **аттрибута** может храниться несколько различных **версий**. Разные версии имеют разный **timestamp**.

Записи физически хранятся в отсортированном по **RowKey** порядке. При этом данные соответствующие разным **Column Family** хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок. При удалении определённого **атрибута** физически он сразу не удаляется, а лишь маркируется специальным флажком **tombstone.** Физическое удаление данных произойдет позже, при выполнении операции Major Compaction. **Атрибуты**, принадлежащие одной **группе колонок** и соответствующие одному **ключу** физически хранятся как отсортированный список. Любой **атрибут** может отсутствовать или присутствовать для каждого ключа, при этом если **атрибут** отсутствует – это не вызывает накладных расходов на хранение пустых значений.

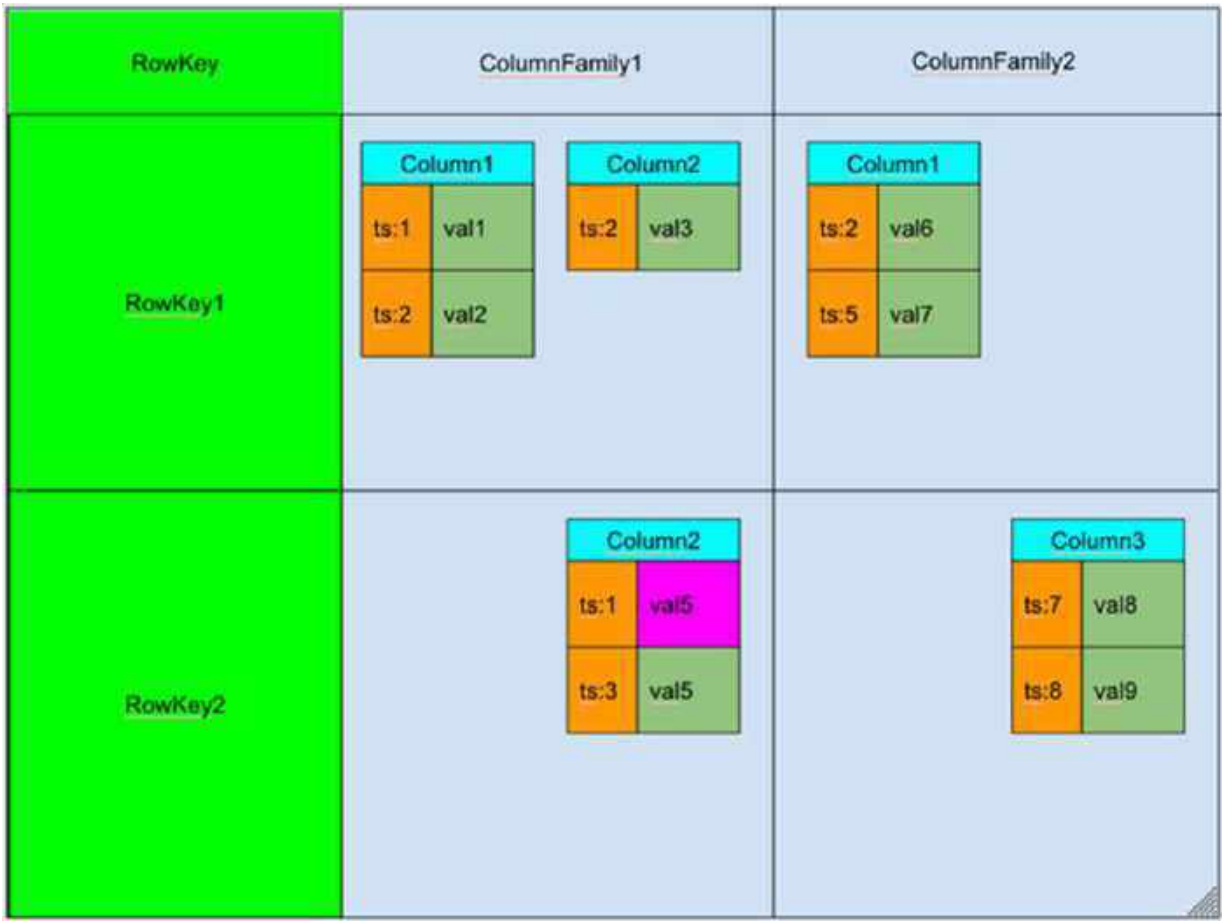


Рис. 25. Модель данных Hbase

Список и названия **групп колонок** фиксирован и имеет четкую схему. На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых **версий**. Если разница между **timestamp** для определенно **версии** и текущим временем больше TTL – запись помечается **к удалению**. Если количество **версий** для определённого **атрибута** превысило максимальное количество версий – запись также помечается **к удалению**.

Поддерживаемые операции:

Список поддерживаемых операций в hbase весьма прост. Поддерживаются 4 основные операции:

– Put: добавить новую запись в hbase. Timestamp этой записи может быть задан руками, в противном случае он будет установлен автоматически как текущее время.

– **Get**: получить данные по определенному RowKey. Можно указать Column Family, из которой будем брать данные и количество версий которые хотим прочитать.

– **Scan**: читать записи по очереди. Можно указать запись с которой начинаем читать, запись до которой читать, количество записей которые необходимо считать, Column Family из которой будет производиться чтение и максимальное количество версий для каждой записи.

– **Delete**: пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction.

Hbase является распределенной базой данных, которая может работать на десятках и сотнях физических серверов, обеспечивая бесперебойную работу даже при выходе из строя некоторых из них. Поэтому архитектура Hbase довольна сложна по сравнению с классическими реляционными базами данных (рис. 26).

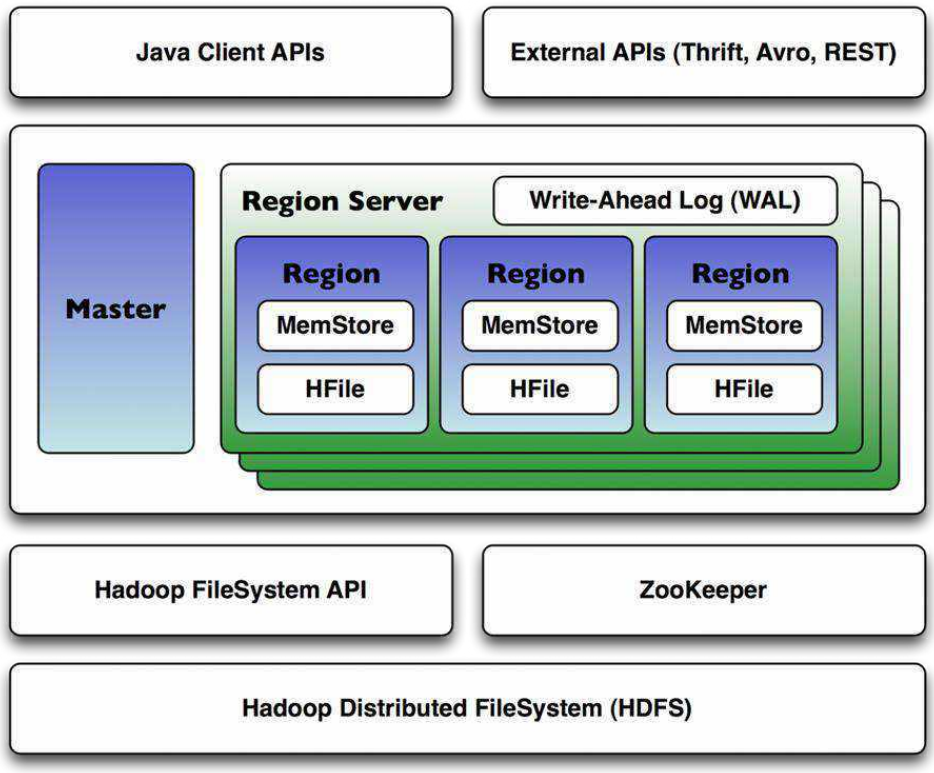


Рис. 26. Архитектура Hbase

Hbase для своей работы использует два основных процесса:

1. Region Server – обслуживает один или несколько **регионов.** Регион – это диапазон записей соответствующих определенному диапазону подряд идущих RowKey. Каждый регион содержит:

- **Persistent Storage** – основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате **HFile**. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle. **MemStore** – буфер на запись. Так как данные хранятся в HFile d отсортированном порядке – обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении MemStore до некоторого критического значения данные записываются в новый HFile.
- **BlockCache** – кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.
- **Write Ahead Log(WAL).** Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не произошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.

2. Master Server – главный сервер в кластере hbase. Master управляет распределением регионов по Region Server'ам, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.

Для координации действий между сервисами Hbase использует Apache ZooKeeper, специальный сервис предназначенный для управления конфигурациями и синхронизацией сервисов.

При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов – можно заранее задать границы регионов и увеличить их максимальный размер.

Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется **compaction.** Compaction'ы бывают двух видов:

- **Minor Compaction.** Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.
- **Major Compaction.** Запускается руками или по наступлению срабатыванию определенных триггеров (например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера. **Major Compaction** лучше делать во время, когда нагрузка на кластер небольшая. Во время Major Compaction также происходит физическое удаление данных, ране помеченных меткой tombstone.

Hbase довольно сложна в администрировании и использовании, поэтому прежде чем использовать hbase есть смысл обратить внимание на альтернативы:

• Реляционные базы данных. Очень неплохая альтернатива, особенно в случае когда данные влезают на одну машину. Также в первую очередь о реляционных базах данных стоит подумать в случае когда важны транзакции индексы отличные от первичного.

• Key-Value хранилища. Такие хранилища как Redis и Aerospike лучше подходят когда необходима минимизация latency и менее важна пакетная обработка данных.

• Файлы и их обработка при помощи MapReduce. Если данные только добавляются, и редко обновляются/изменяются, то лучше не использовать Hbase, а просто хранить данные в файлах. Для упрощения работы с файлами можно воспользоваться такими инструментами как Hive, Pig и Impala, о которых речь пойдет в следующих статьях.

Источник: ***https://habr.com/ru/company/dca/blog/280700/***

Hive

Hive – SQL-движок над MapReduce. Классические СУБД, такие как Postgres, MySQL или Oracle не имеют гибкости в масштабировании при обработке больших массивов данных и при достижении объема большего дальнейшая поддержка становится большой проблемой.

Apache Hive был придуман для того, чтобы объединить два этих достоинства:

- Масштабируемость MapReduce;
- Удобство использования SQL для выборок из данных.

Hive появился в недрах компании Facebook в 2007 году, а через год исходники hive были открыты и переданы под управление apache software foundation. Изначально hive представлял собой набор скриптов поверх hadoop streaming, позже развился в полноценный фреймворк для выполнения запросов к данным поверх MapReduce.

Актуальная версия apache hive (2.0) представляет собой продвинутый фреймворк, который может работать не только поверх фреймворка Map/Reduce, но и поверх Spark, а также Apache Tez.

Apache hive используют в production такие компании как Facebook, Grooveshark, Last.Fm и многие другие. Мы в Data-Centric alliance используем HIve в качестве основного хранилища логов нашей рекламной платформы.

Архитектура Hive

Hive представляет из себя движок, который превращает SQL-запросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запрсоы), Optimimer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение), Executor (запускает задачи на фреймворке MapReduce (рис. 27).

Для работы hive также необходимо хранилище метаданных. Дело в том что SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует

hive хранятся просто в виде файлов на hdfs – необходимо где-то хранить соответствие между объектами hive и реальными файлами.

В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

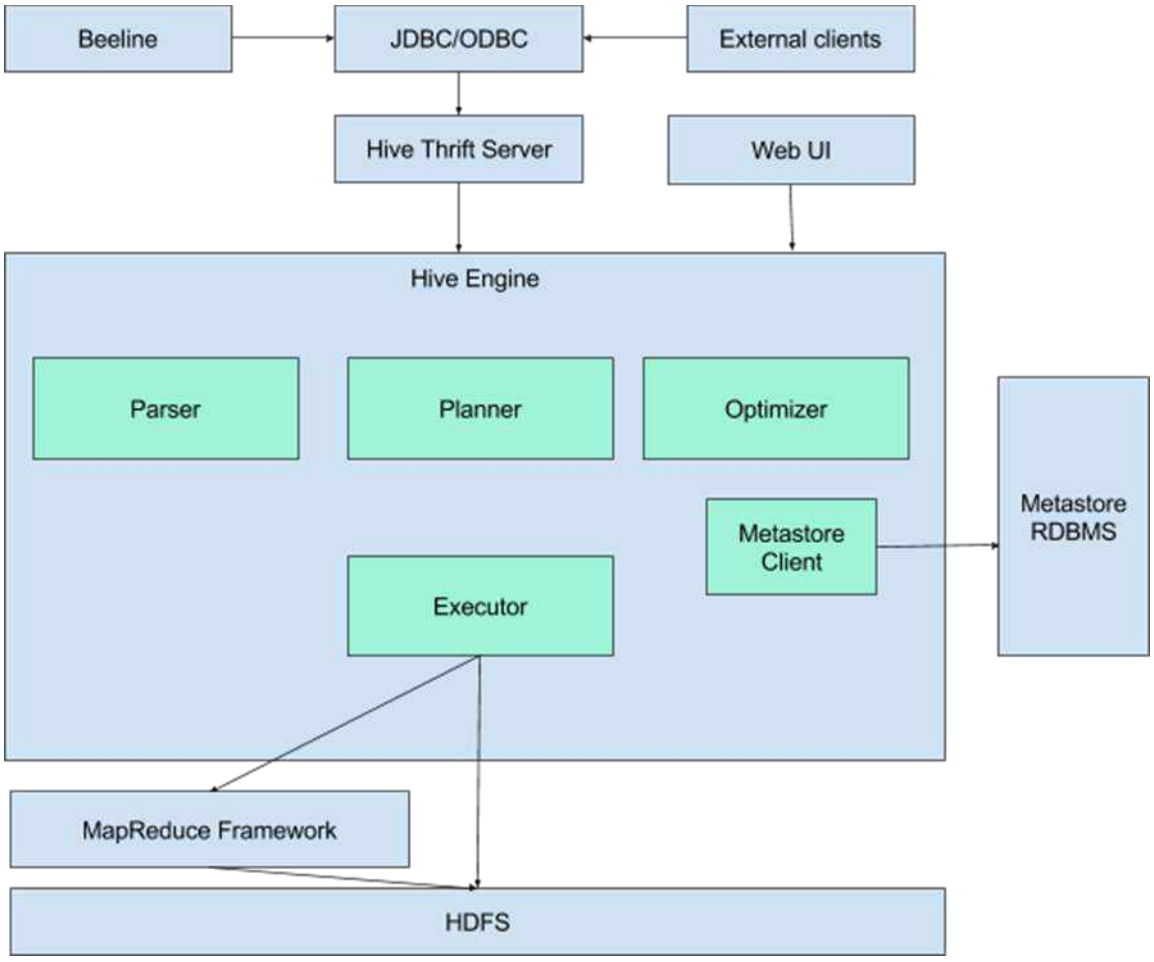


Рис. 27. Архитектура Hive [1]

Для того чтобы попробовать работу с hive проще всего воспользоваться его командной строкой. Современная утилита для работы с hive называется beeline. Для этого на любой машине в hadoop-кластере с установленным hive достаточно набрать команду.

```
$beeline
```

Далее необходимо установить соединение с hive-сервером:

```
beeline> !connect jdbc:hive2://localhost:10000/default root root
Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 1.1.0-cdh5.7.0)
Driver: Hive JDBC (version 1.1.0-cdh5.7.0)
```

Transaction isolation: TRANSACTION_REPEATABLE_READ

0: jdbc:hive2://localhost:10000/default>

root root — в данном контексте это имя пользователя и пароль. После этого вы получите командную строку, в которой можно вводить команды hive.

Также иногда бывает удобно не вводить sql-запросы в командную строку beeline, а предварительно сохранить и редактировать их в файле, а потом выполнить все запросы из файла. Для этого нужно выполнить beeline с параметрами подключения к базе данных и параметром -f указывающим имя файла, содержащего запросы:

beeline -u jdbc:hive2://localhost:10000/default -n root -p root -f sorted.sql

При работе с hive можно выделить следующие объекты которыми оперирует hive:

1. База данных;
2. Таблица;
3. Партиция (partition);
4. Бакет (bucket).

Разберем каждый из них подробнее:

База данных

База данных представляет аналог базы данных в реляционных СУБД. База данных представляет собой **пространство имён**, содержащее таблицы. Команда создания новой базы данных выглядит следующим образом:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Database и Schema в данном контексте это одно и тоже. Необязательная добавка **IF NOT EXISTS** как не сложно догадаться создает базу данных только в том случае если она еще не существует.

Пример создания базы данных:

CREATE DATABASE userdb;

Для переключения на соответствующую базу данных используем команду USE:

USE userdb;

Таблица

Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие – что данные hive'овских таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive.

Таблицы в hive бывают двух видов:

Классическая таблица, данные в которую добавляются при помощи hive. Вот пример создания такой таблицы:

```
CREATE TABLE IF NOT EXISTS employee ( eid int, name String,
salary String, destination String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Тут мы создали таблицу, данные в которой будут храниться в виде обычных csv-файлов, колонки которой разделены символом табуляции. После этого данные в таблицу можно загрузить. Пусть у нашего пользователя в домашней папке на hdfs есть (напоминаю, что загрузить файл можно при помощи **hadoop fs -put**) файл sample.txt вида:

```
1201    Gopal45000Technical manager
1202    Manisha     45000Proof reader
1203    Masthanvali40000Technical writer
1204    Kiran 40000Hr Admin
1205    Kranthi     30000Op Admin
```

Загрузить данные мы сможем при помощи следующей команды:

```
LOAD DATA INPATH '/user/root/sample.txt'
OVERWRITE INTO TABLE employee;
```

После hive переместит данные, хранящемся в нашем файле в хранилище hive. Убедиться в этом можно прочитав данные напрямую из файла в хранилище hive в hdfs:

[root@quickstart ~]# hadoop fs -text /user/hive/warehouse/userdb.db/employee/*

1201 Gopal 45000 Technical manager

1202 Manisha 45000 Proof reader

1203 Masthanvali 40000 Technical writer

1204 Kiran 40000 Hr Admin

1205 Kranthi 30000 Op Admin

Классические таблицы можно также создавать как результат select-запроса к другим таблицам:

0: jdbc:hive2://localhost:10000/default> CREATE TABLE big_salary as SELECT * FROM employee WHERE salary > 40000;

0: jdbc:hive2://localhost:10000/default> SELECT * FROM big_salary;

```
+----------------+-----------------+-------------------+-----------------------+--+
| big_salary.eid | big_salary.name | big_salary.salary | big_salary.destination |
+----------------+-----------------+-------------------+-----------------------+--+
| 1201           | Gopal           | 45000             | Technical manager     |
| 1202           | Manisha         | 45000             | Proof reader          |
+----------------+-----------------+-------------------+-----------------------+--+
```

Кстати говоря, SELECT для создания таблицы в данном случае уже запустит mapreduce-задачу.

Внешняя таблица, данные в которую загружаются внешними системами, без участия hive. Для работы с внешними таблицами при создании таблицы нужно указать ключевое слово EXTERNAL, а также указать путь до папки, по которому хранятся файлы:

CREATE EXTERNAL TABLE IF NOT EXISTS employee_external (eid int, name String,

salary String, destination String)

COMMENT 'Employee details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/user/root/external_files/';

После этого таблицей можно пользоваться точно так же как и обычными таблицами hive. Самое удобное в этом, что вы можете просто скопировать файл в нужную папочку в hdfs, а hive будет автоматом подхватывать новые файлы при запросах к соответствующей таблице. Это очень удобно при работе например с логами.

Партиция (partition)

Так как hive представляет из себя движок для трансляции SQL-запросов в mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS.

Для создания партиционированной таблицы необходимо указать по каким колонкам будет произведено партиционирование:

CREATE TABLE IF NOT EXISTS employee_partitioned (eid int, name String,

salary String, destination String)

COMMENT 'Employee details'

PARTITIONED BY (birth_year int, birth_month string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

При заливке данных в такую таблицу необходимо явно указать, в какую партицию мы заливаем данные:

```
LOAD DATA INPATH '/user/root/sample.txt' OVERWRITE
INTO TABLE employee_partitioned
PARTITION (birth_year=1998, birth_month='May');
```

Посмотрим теперь как выглядит структура директорий:

```
[root@quickstart ~]# hadoop fs -ls
/user/hive/warehouse/employee_partitioned/
Found 1 items
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03
/user/hive/warehouse/employee_partitioned/birth_year=1998
[root@quickstart ~]# hadoop fs -ls -R
/user/hive/warehouse/employee_partitioned/
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03
/user/hive/warehouse/employee_partitioned/birth_year=1998
drwxrwxrwx   - root supergroup          0 2016-05-08 15:03
/user/hive/warehouse/employee_partitioned/birth_year=1998/birth_month=May
-rwxrwxrwx   1 root supergroup        161 2016-05-08 15:03
/user/hive/warehouse/employee_partitioned/birth_year=1998/birth_month=May/sa
mple.txt
```

Видно, что структура директорий выглядит таким образом, что каждой партиции соответствует отдельная папка на hdfs. Теперь, если мы будем запускать какие-либо запросы, у казав в условии WHERE ограничение на значения партиций — mapreduce возьмет входные данные только из соответствующих папок.

В случае External таблиц партиционирование работает аналогичным образом, но подобную структуру директорий придется создавать вручную. Партиционирование очень удобно например для разделения логов по датам, так как правило любые запросы за статистикой содержат ограничение по датам. Это позволяет существенно сократить время запроса.

Бакет

Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какого-либо столбца. Однако оно не всегда применимо. Например, если количество значений в столбце очень велико. Например, это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.

В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает значение хэш-функции вычисленное по определенной колонке.

При любой работе с бакетированными таблицами необходимо не забывать включать поддержку бакетов в hive (иначе hive будет работать с ними как с обычными таблицами):

set hive.enforce.bucketing=true;

Для создания таблицы разбитой на бакеты используется конструкция CLUSTERED BY

set hive.enforce.bucketing=true;

CREATE TABLE employee_bucketed (eid int, name String, salary String, destination String)

CLUSTERED BY(eid) INTO 10 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

Так как команда Load используется для простого перемещения данных в хранилище hive – в данном случае для загрузки она не подходит, так как данные необходимо предобработать, правильно разбив их на бакеты. Поэтому их нужно загрузить при помощи команды INSERT из другой таблицы (например из внешней таблицы):

set hive.enforce.bucketing=true;

FROM employee_external INSERT OVERWRITE TABLE employee_bucketed SELECT *;

После выполнения команды убедимся, что данные действительно разбились на 10 частей:

[root@quickstart ~]# hadoop fs -ls /user/hive/warehouse/employee_bucketed

Found 10 items

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000000_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000001_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000002_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000003_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000004_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000005_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000006_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000007_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000008_0

-rwxrwxrwx 1 root supergroup 31555556 2016-05-08 16:04 /user/hive/warehouse/employee_bucketed/000009_0

Теперь при запросах за данными, относящимися к определенному пользователю, нам не нужно будет сканировать всю таблицу, а только 1/10 часть этой таблицы [2].

Источники: 1. http://www.tutorialspoint.com/hive/hive_create_table.htm

2. https://habr.com/ru/post/283212/

Impala

Impala – это массово-параллельный механизм интерактивного выполнения SQL-запросов к данным, хранящимся в Apache Hadoop (HDFS и HBase), написанный на языке C++ и распространяющийся по лицензии Apache 2.0. Также Импала называют MPP-движком (Massively Parallel Processing), распределенной СУБД и даже базой данных стека SQL-on-Hadoop.

Изначально рассматриваемый продукт был разработан компанией Cloudera и представлен на рынок в 2012 году, а 2 декабря 2015 года был принят в инкубатор фонда Apache Software Foundation. Поэтому сегодня Apache Импала обычно означает свободно распространяемое решение, а Cloudera Impala – коммерчески поддерживаемую версию от исходной компании-разработчика (Cloudera) [1].

Разумеется, Импала – это не единственное решение класса SQL-on-Hadoop. Помимо рассматриваемого продукта, такими аналитическими средствами для Big Data являются другие проекты Apache: Hive (Хайв), Drill, Phoenix. Потребность в разработке инструментов SQL-on-Hadoop возникла из-за необходимости аналитики больших данных, хранящихся в HDFS и HBase. Например, в рамках BI-приложений (Business Intelligence), когда требуется быстро ответить на сложный логический запрос, например, при поиске оптимального авиамаршрута или другой подобной задачи с непростой логистикой [2]. Благодаря автоматической трансляции запроса в исполнительный код, реализованной внутри средства SQL-on-Hadoop, разработчик Big Data системы может работать с данными, хранящимися в HBase или HDFS, как с реляционными таблицами, формируя различные выборки и условные фильтрации, а также изменяя значение данных.

Impala работает в распределенном режиме, когда экземпляры процессов выполняются на разных узлах кластера, получая, планируя и координируя запросы от клиентов. При этом возможно параллельное выполнение фрагментов SQL-запроса. **Клиенты** – это пользователи и приложения, которые отправляют SQL-запросы к данным, хранящимся в Apache Hadoop (HBase и HDFS) или Amazon S3. Взаимодействие с Импала происходит через веб-интерфейс HUE (Hadoop User Experience), ODBC, JDBC и оболочку командной строки Impala Shell.

Импала инфраструктурно зависит от другого популярного SQL-on-Hadoop инструмента, Apache Hive, используя его хранилище метаданных. В частности, Hive Metastore позволяет Impala знать о доступности и структуре баз данных. При создании, изменении и удалении объектов схемы или загрузке данных в таблицы через SQL-инструкции, соответствующие изменения метаданных автоматически передаются всем узлам Impala с помощью специализированной службы каталогов **[3]**.

Ключевыми компонентами Импала являются следующие исполняемые файлы [3]:

- **Impalad** (Impala daemon) – системная служба, которая планирует и выполняет запросы к данным HDFS, HBase и Amazon S3. Один процесс impalad работает на каждом узле кластера.
- **Statestored** – служба имен, которая отслеживает местоположение и статус всех impalad экземпляров в кластере. Один экземпляр этого системного сервиса работает на каждом узле, а также на главном сервере (Name Node).
- **Catalogd** – служба координации метаданных, которая передает изменения от операторов Impala DDL и DML всем затронутым узлам Impala, чтобы новые таблицы или недавно загруженные данные сразу отображались для любого узла кластера. Рекомендуется, чтобы один экземпляр Catalogd был запущен на том же хосте кластера, что и Statestored daemon (рис. 28).

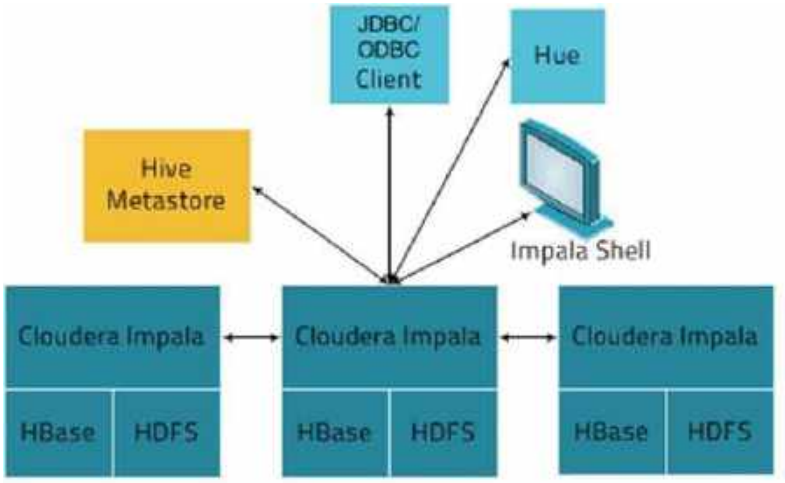


Рис. 28. Архитектура Impala

Cloudera Impala, как и Apache Hive вместо SQL использует аналогичный декларативный язык запросов Hive Query Language (HiveQL), который является подмножеством SQL92. Он немного отличается от стандартного SQL.

Само выполнение запроса в Импала происходит следующим образом **[3]**:

1. Клиентское приложение отправляет SQL-запрос, подключаясь к любому impalad через стандартизированные интерфейсы драйверов ODBC или JDBC. Подключенный impalad становится координатором текущего запроса.
2. Выполняется анализ SQL-запроса для определения задач для impalad-экземпляров в кластере, далее строится оптимальный план выполнения запроса.
3. Impalad напрямую обращается к HDFS и HBase с помощью локальных экземпляров системных сервисов для предоставления данных. Такое непосредственное взаимодействие существенно экономит время выполнения запроса, как отсутствие сохранения промежуточных результатов, в отличие от Apache Hive.
4. В ответ каждый daemon возвращает данные координирущему impalad, который далее отправляет результаты клиенту.

Благодаря MPP-механизму параллельного распределения запросов, кэшированию часто запрашиваемых данных в памяти, предварительному

запуску системных служб (при загрузке) и генерации программного кода во время исполнения (runtime), а не при компиляции (compile time), Импала работает быстрее надежной и отказоустойчивой Hive (рис. 29).

Вышеописанные архитектурные особенности обусловливают следующие преимущества Импала:

- высокая скорость работы, практически в режиме реального времени, в т.ч. в многопользовательской среде с высокой конкуренцией запросов. Особенная эффективность отмечается при обработке множества запросов с относительно простой логикой [4].
- встроенная поддержка безопасного сетевого протокола аутентификации Kerberos;
- приоритезация и возможность управления очередью запросов;
- поддержка популярных Big Data форматов (LZO, Avro, RCFile, Parquet, Sequence).

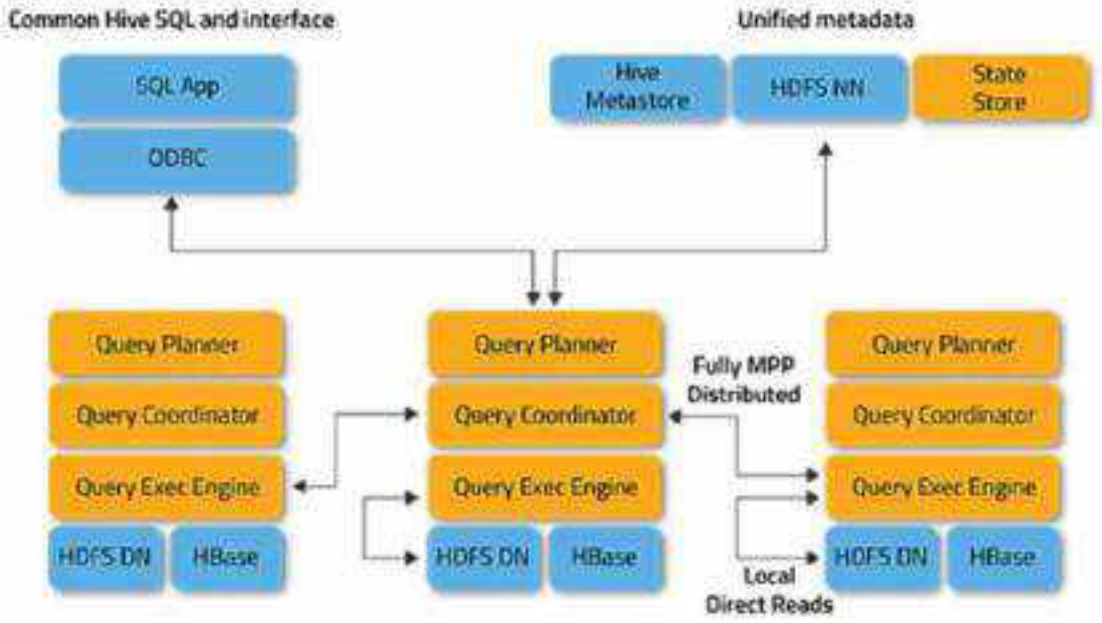


Рис. 29. Компоненты Impala для выполнения SQL-запросов к данным в Apache Hadoop

Обратной стороной всех этих достоинств Импала является снижение надежности, отказоустойчивости и пропускной способности по сравнению с Apache Hive. Также, в отличие от Hive, Импала не поддерживает сложные типы данных (map, list и struct) [5]. Несмотря на эти недостатки, Cloudera

Impala активно используется в различных Big Data проектах по всему миру. Благодаря своей быстроте, эта система востребована не только у аналитиков и инженеров по данным, но и в крупных production-решениях.

Источники:

1. https://www.cloudera.com/products/open-source/apache-hadoop/impala.html
2. https://www.dezyre.com/hadoop-tutorial/impala-case-study-flight-data-analysis
3. https://ru.bmstu.wiki/Apache_Impala
4. https://data-flair.training/blogs/impala-vs-hive/
5. https://www.educba.com/hive-vs-impala/

YARN

YARN – это система планирования заданий и управления кластером (Yet Another Resource Negotiator), которую также называют MapReduce 2.0 – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений. YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и аналитики больших данных.

История появления и роль YARN в экосистеме APACHE HADOOP

Будучи впервые выпущен в 2013 году под лицензией Apache 2.0, YARN является одним из основных модулей ядра экосистемы Hadoop, отвечая за планирование заданий и управление распределенными приложениями. YARN решает проблемы 1-ой версии технологии распределенных вычислений – MapReduce, предоставляя компоненты и API для разработки Big Data приложений, обеспечивая распределение ресурсов в ответ на запросы и отслеживания статусы выполнения заданий.

Поэтому очень часто этот фреймворк называют Apache Hadoop YARN или MapReduce 2.0, хотя он используется не только в классическом Hadoop

MapReduce. Благодаря более общей модели, чем в классическом Hadoop MapReduce, YARN позволяет запускать в кластере Apache Hadoop не только MapReduce-задания, но и любые распределенные приложения. В частности, в Apache Spark именно YARN обеспечивает распределённому приложению доступ к кластеру через свой диспетчер ресурсов – один из главных модулей YARN, которые мы рассмотрим далее.

Архитектура и принципы работы

YARN состоит из следующих компонентов:

- *ResourceManager (RM)* – менеджер ресурсов, которых отвечает за распределение ресурсов, необходимых для работы распределенных приложений, и наблюдение за узлами кластера, где эти приложения выполняются. ResourceManager включает планировщик ресурсов (Scheduler) и диспетчер приложений (ApplicationsManager, AsM).
- *ApplicationMaster (AM)* – мастер приложения, ответственный за планирование его жизненного цикла, координацию и отслеживание статуса выполнения, включая динамическое масштабирование потребления ресурсов, управление потоком выполнения, обработку ошибок и искажений вычислений, выполнение локальных оптимизаций. Каждое приложение имеет свой экземпляр ApplicationMaster. ApplicationMaster выполняет произвольный пользовательский код и может быть написан на любом языке программирования благодаря расширяемым протоколам связи с менеджером ресурсов и менеджером узлов.
- *NodeManager (NM)* – менеджер узла – агент, запущенный на узле кластера, который отвечает за отслеживание используемых вычислительных ресурсов (CPU, RAM и пр.), управление логами и отправку отчетов об использовании ресурсов планировщику. NodeManager управляет абстрактными контейнерами – ресурсами узла, доступными для конкретного приложения.

- *Контейнер (Container)* – набор физических ресурсов (ЦП, память, диск, сеть) в одном вычислительном узле кластера.

ApplicationsManager отвечает за прием задач и запуск экземпляров мастера приложений (ApplicationMaster), мониторингов узлов (контейнеров), на которых происходит выполнение распределенных заданий, и предоставляет сервис для перезапуска контейнера ApplicationMaster в случае сбоя. Планировщик менеджера ресурсов (Scheduler) является не ведет мониторинг и не отслеживает статус выполнения распределенного приложений, а также не дает гарантий перезапуска неудачных задач при аппаратных или программных отказах.

Таким образом, YARN разделяет функции управления ресурсами и планирования/мониторинга заданий на отдельные демоны, имея глобальный диспетчер ресурсов и мастер приложения для каждого отдельного задания или их конвейера в виде направленного ациклического графа (DAG, Directed Acyclic Graph).

ResourceManager и диспетчер узла образуют структуру вычисления данных, где менеджер ресурсов распределяет ресурсы между всеми приложениями в системе, а NodeManager на каждом узле отвечает за контейнеры, отслеживая использование ресурсов и сообщая об этом планировщику. Планировщик распределяет ресурсы между запущенными приложениями в виде контейнера с учетом их требований и известных ограничений емкости, очередей и пр.

Компоненты YARN взаимодействуют друг с другом через следующие протоколы:

- *ClientRMProtocol* – протокол взаимодействия клиента с ResourceManager для запуска, завершения и проверки статуса приложений;
- *AMRMProtocol* – протокол, который использует мастер приложения для регистрации или ее регистрации в менеджере ресурсов, а также для запроса ресурсов у планировщика;

- *ContainerManager* – протокол взаимодействия мастера приложения с диспетчером узлов для запуска или остановки и получения данных о необходимости обновления контейнеров под управлением NodeManager (рис. 30).

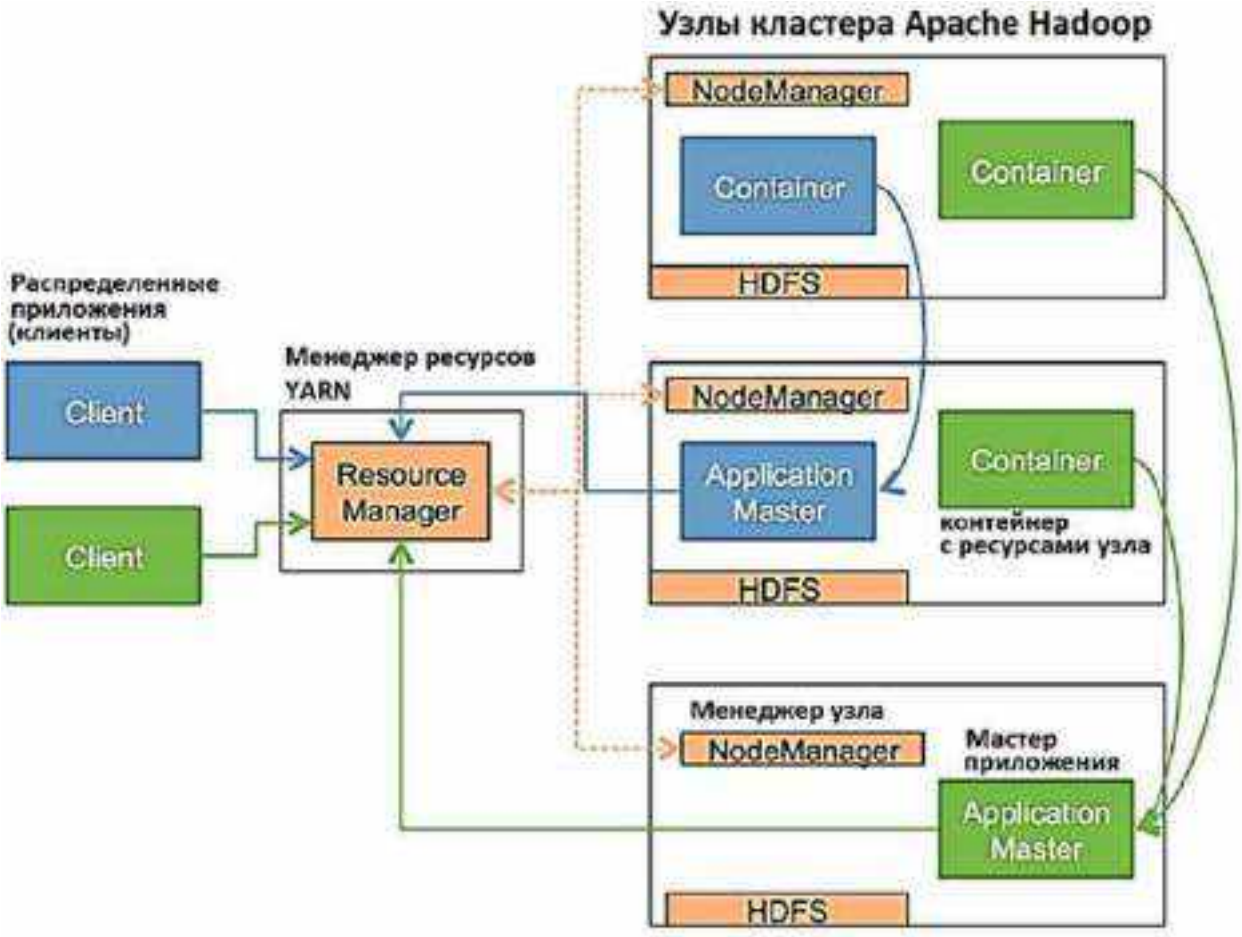


Рис. 30. Архитектура и принципы работы Apache Hadoop YARN

Принцип работы Hadoop YARN можно описать следующим образом:

- клиентское приложение отправляет запрос в кластер;
- менеджер ресурсов выделяет необходимые ресурсы для контейнера и запускает ApplicationMaster для обслуживания этого приложения;
- ApplicationMaster отправляет запрос менеджеру узла NodeManager, включая контекст запуска контейнера Container Launch Context (CLC);
- ApplicationMaster выделяет контейнеры для приложения в каждом узле и контролирует их работу до завершения работы приложения;
- для запуска контейнера менеджер узла копирует в локальное хранилище все необходимые зависимости (данные, исполняемые файлы, архивы);

- по завершении задачи мастер приложения отменяет выделенный контейнер в диспетчере ресурсов, завершая жизненный цикл распределенного задания;
- клиент может отслеживать состояние распределенного приложения, обращаясь к менеджеру ресурсов или сразу к мастеру приложения.

При необходимости клиент может самостоятельно завершить работу приложения. Являясь контейнером, работающим в кластере, мастер приложения должен быть отказоустойчивым. Хотя YARN и поддерживает восстановление при сбоях, но большая часть нагрузки по обеспечению отказоустойчивости все равно лежит на мастере приложения.

YARN предоставляет API-интерфейсы для запроса ресурсов кластера и работы с ними, но эти API-интерфейсы обычно не используются непосредственно пользовательским кодом. Вместо этого пользователи пишут в API более высокого уровня, предоставляемые распределенными вычислительными средами, которые сами построены на YARN и скрывают детали управления ресурсами от пользователя. Ситуация проиллюстрирована на рис. 31, где показаны некоторые распределенные вычислительные структуры (MapReduce, Spark и т. Д.), Работающие как приложения YARN на уровне кластерных вычислений (YARN) и уровне кластерного хранилища (HDFS и HBase).

Существует также уровень приложений, построенных на каркасах, показанных на рис. 31. Pig, Hive и Crunch - все это примеры сред обработки, которые работают на MapReduce, Spark или Tez (или на всех трех) и не взаимодействуют напрямую с YARN.

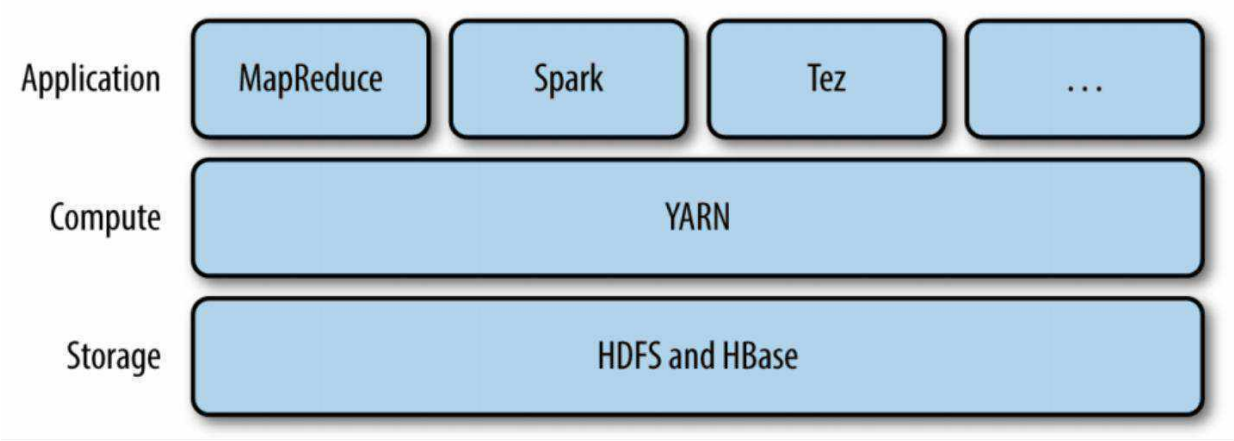


Рис. 31. Многоуровневость YARN

Анатомия запуска приложения YARN

YARN предоставляет свои основные услуги через два типа долго работающих демонов (*Для справки. Демон - это компьютерная программа в системах класса UNIX, которая запускается самой системой и работает в фоновом режиме без прямого взаимодействия с пользователем*): диспетчер ресурсов (по одному на кластер) для управления использованием ресурсов в кластере и диспетчеры узлов, работающие на всех узлах кластера для запуска и мониторинга контейнеров. Контейнер выполняет процесс, зависящий от приложения, с ограниченным набором ресурсов (память, ЦП и т. Д.). В зависимости от того, как настроена YARN, контейнер может быть процессом Unix или группой Linux. На рис. 32 показано, как YARN запускает приложение.

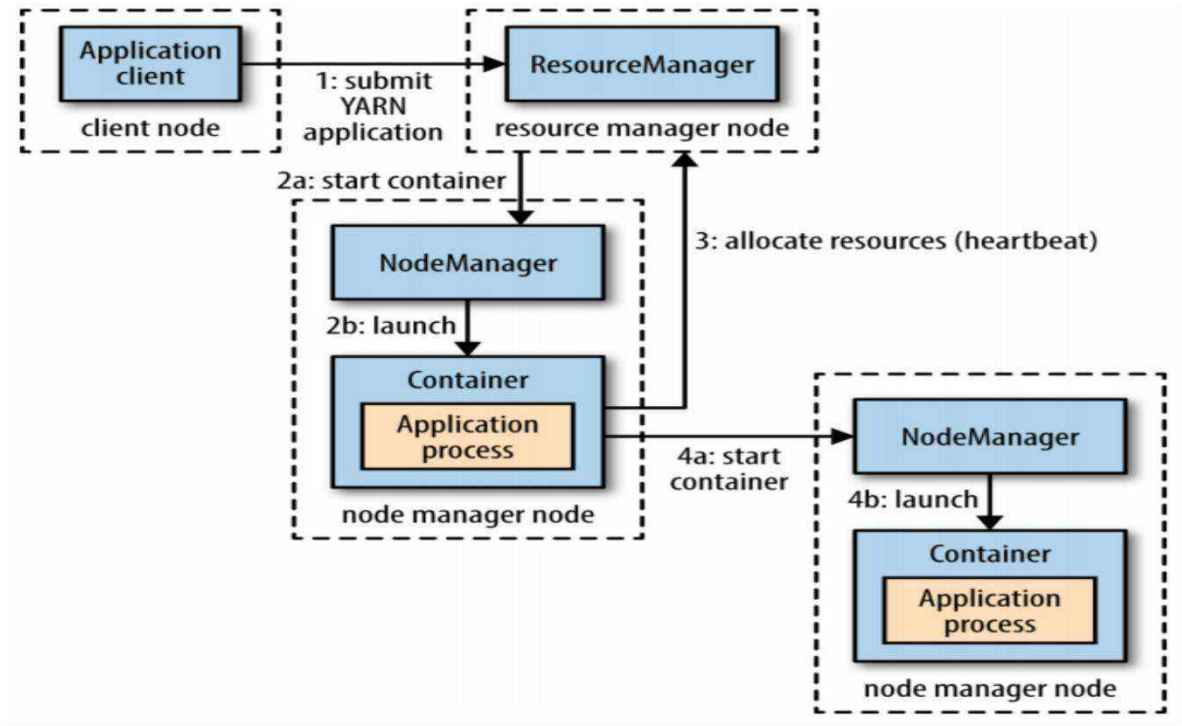


Рис. 32. Запуск приложения в YARN

Чтобы запустить приложение в YARN, клиент связывается с менеджером ресурсов и просит его запустить главный процесс приложения (шаг 1 на рисунке). Затем диспетчер ресурсов находит диспетчер узлов, который может запустить мастер приложения в контейнере (шаги 2a и 2b). Что именно делает мастер приложения после запуска, зависит от приложения. Он может просто запустить вычисление в контейнере, в котором оно выполняется, и вернуть результат клиенту. Или он может запросить дополнительные контейнеры у диспетчеров ресурсов (шаг 3) и использовать их для выполнения распределенных вычислений (шаги 4a и 4b). Последнее - то, что делает приложение MapReduce YARN, о чем мы поговорим более подробно.

Обратите внимание на рисунок 32 на этом слайде, что YARN сама по себе не предоставляет никакого способа для частей приложения (клиент, мастер, процесс) взаимодействовать друг с другом. Большинство нетривиальных приложений YARN используют ту или иную форму удаленной связи для передачи обновлений статуса и результатов обратно клиенту, но они специфичны для приложения.

Запросы ресурсов

YARN имеет гибкую модель для запросов ресурсов. Запрос на набор контейнеров может выражать количество компьютерных ресурсов, необходимых для каждого контейнера (память и ЦП), а также ограничения местоположения для контейнеров в этом запросе.

Локальность имеет решающее значение для обеспечения того, чтобы алгоритмы распределенной обработки данных эффективно использовали полосу пропускания кластера, поэтому YARN позволяет приложению указывать ограничения локальности для запрашиваемых контейнеров. Ограничения местоположения можно использовать для запроса контейнера на конкретном узле или стойке или в любом месте кластера (вне стойки).

Иногда ограничение местоположения не может быть выполнено, и в этом случае либо не выполняется распределение, либо, при желании, ограничение может быть ослаблено. Например, если был запрошен конкретный узел, но невозможно запустить на нем контейнер (потому что на нем работают другие контейнеры), то YARN попытается запустить контейнер на узле в той же стойке, или, если это невозможно ни на одном узле кластера.

В обычном случае запуска контейнера для обработки блока HDFS (например, для запуска задачи карты в MapReduce) приложение запросит контейнер на одном из узлов, на которых размещены три реплики блока, или на узле в одной из стойки, в которых размещаются реплики, или, в противном случае, на любом узле кластера.

Приложение YARN может запрашивать ресурсы в любое время во время работы. Например, приложение может выполнять все свои запросы заранее или может использовать более динамичный подход, при котором оно динамически запрашивает больше ресурсов для удовлетворения меняющихся потребностей приложения.

Срок службы приложения

Продолжительность жизни приложения YARN может сильно варьироваться: от недолговечного приложения в несколько секунд до

длительного приложения, которое работает в течение нескольких дней или даже месяцев. Вместо того, чтобы смотреть, как долго работает приложение, полезно категоризировать приложения с точки зрения того, как они сопоставляются с заданиями, выполняемыми пользователями. В простейшем случае одно приложение на одно пользовательское задание. Именно такой подход и использует MapReduce.

Вторая модель - запускать одно приложение на рабочий процесс или пользовательский сеанс (возможно, не связанных) заданий. Этот подход может быть более эффективным, чем первый, поскольку контейнеры можно повторно использовать между заданиями, а также существует возможность кэширования промежуточных данных между заданиями. Spark - это пример, использующий эту модель.

Третья модель – это долго работающее приложение, которое используется разными пользователями. Такое приложение часто играет роль координатора. Например, в Apache Slider есть долго работающий мастер приложений для запуска других приложений в кластере. Этот подход также используется Impala для предоставления прокси-приложения, с которым взаимодействуют демоны Impala для запроса ресурсов кластера. «Всегда включенный» мастер приложения означает, что пользователи получают ответы на свои запросы с очень малой задержкой, поскольку исключаются накладные расходы на запуск нового мастера приложения.

Создание приложений YARN

Написание приложения YARN с нуля довольно сложно, но во многих случаях в этом нет необходимости, поскольку часто можно использовать существующее приложение, которое отвечает всем требованиям. Например, если вы заинтересованы в запуске ориентированного ациклического графа (DAG) заданий, то вам подойдут Spark или Tez; или для потоковой обработки работает Spark, Samza или Storm.

Есть несколько проектов, упрощающих процесс создания приложения YARN. Упомянутый ранее Apache Slider позволяет запускать существующие

распределенные приложения на YARN. Пользователи могут запускать свои собственные экземпляры приложения (например, HBase) в кластере независимо от других пользователей, что означает, что разные пользователи могут запускать разные версии одного и того же приложения. Ползунок предоставляет элементы управления для изменения количества узлов, на которых выполняется приложение, а также для приостановки и возобновления работающего приложения.

Apache Twill похож на Slider, но, кроме того, предоставляет простую модель программирования для разработки распределенных приложений на YARN. Twill позволяет определять процессы кластера как расширение Java Runnable, а затем запускать их в контейнерах YARN в кластере. Twill также обеспечивает поддержку, среди прочего, ведения журнала в реальном времени (события журнала из исполняемых файлов передаются обратно клиенту) и командных сообщений (отправляемых от клиента к исполняемым объектам).

В случаях, когда ни один из этих вариантов не является достаточным - например, приложение, которое имеет сложные требования к планированию, - тогда приложение распределенной оболочки, которое является частью проекта YARN, само служит примером того, как написать приложение YARN. Он демонстрирует, как использовать клиентские API YARN для управления взаимодействием между клиентом или мастером приложения и демонами YARN.

YARN по сравнению с MapReduce 1

Распределенную реализацию MapReduce в исходной версии Hadoop (версия 1 и ранее) иногда называют «MapReduce 1», чтобы отличить ее от MapReduce 2, реализации, использующей YARN (в Hadoop 2 и более поздних версиях).

Важно понимать, что старый и новый API MapReduce - это не то же самое, что реализации MapReduce 1 и MapReduce 2. API-интерфейсы представляют собой клиентские функции, ориентированные на пользователя, и определяют способ написания программ MapReduce, тогда как реализации - это просто

разные способы запуска программ MapReduce. Поддерживаются все четыре комбинации: и старый, и новый API MapReduce работают как на MapReduce 1, так и на 2.

В MapReduce 1 есть два типа демонов, которые контролируют процесс выполнения задания: средство отслеживания заданий и одно или несколько средств отслеживания задач. Программа отслеживания заданий координирует все задания, выполняемые в системе, путем планирования задач для выполнения в средствах отслеживания задач. Средства отслеживания задач запускают задачи и отправляют отчеты о ходе выполнения в средство отслеживания заданий, которое ведет учет общего хода выполнения каждого задания. В случае сбоя задачи средство отслеживания заданий может перенести ее на другое средство отслеживания задач.

В MapReduce 1 средство отслеживания заданий заботится как о планировании заданий (сопоставление задач со средствами отслеживания задач), так и о мониторинге выполнения задач (отслеживание задач, перезапуск неудачных или медленных задач и ведение учета задач, например, ведение итоговых значений счетчика). Напротив, в YARN эти обязанности выполняются отдельными объектами: менеджером ресурсов и мастером приложения (по одному на каждое задание MapReduce). Программа отслеживания заданий также отвечает за хранение истории выполненных заданий, хотя можно запустить сервер истории заданий как отдельный демон, чтобы снять нагрузку с средства отслеживания заданий. В YARN эквивалентной ролью является сервер временной шкалы, на котором хранится история приложений.

Масштабируемость

YARN может работать на более крупных кластерах, чем MapReduce 1. MapReduce 1 устраняет узкие места масштабируемости в районе 4000 узлов и 40 000 задач, что связано с тем, что средство отслеживания заданий должно управлять как заданиями, так и задачами. YARN преодолевает эти ограничения благодаря своей архитектуре разделенного менеджера ресурсов /

главного приложения: он рассчитан на масштабирование до 10 000 узлов и 100 000 задач.

В отличие от средства отслеживания заданий, каждый экземпляр приложения – в данном случае задание MapReduce – имеет выделенный мастер приложения, который работает на протяжении всего приложения. Эта модель на самом деле ближе к исходной статье Google MapReduce, в которой описывается, как запускается главный процесс для координации карты и уменьшения количества задач, выполняемых на наборе рабочих.

Доступность

Высокая доступность (HA) обычно достигается путем репликации состояния, необходимого для другого демона, чтобы взять на себя работу, необходимую для предоставления службы, в случае сбоя демона службы. Однако из-за большого количества быстро меняющихся сложных состояний в памяти средства отслеживания заданий (например, состояние каждой задачи обновляется каждые несколько секунд) очень сложно модифицировать HA в службу отслеживания заданий.

Поскольку обязанности специалиста по отслеживанию вакансий разделены между менеджером ресурсов и мастером приложения в YARN, обеспечение высокой доступности сервиса стало проблемой разделения и владения: обеспечить высокую доступность для менеджера ресурсов, а затем для приложений YARN (для каждого приложения). И действительно, Hadoop 2 поддерживает высокую доступность как для диспетчера ресурсов, так и для главного приложения для заданий MapReduce.

Утилизация

В MapReduce 1 каждый трекер задач настроен со статическим распределением «слотов» фиксированного размера, которые делятся на слоты карты и сокращают слоты во время настройки. Слот карты можно использовать только для выполнения задачи карты, а слот уменьшения можно использовать только для задачи уменьшения.

В YARN менеджер узлов управляет пулом ресурсов, а не фиксированным количеством назначенных слотов. MapReduce, запущенный на YARN, не попадет в ситуацию, когда задача сокращения должна ждать, потому что в кластере доступны только слоты карты, что может произойти в MapReduce 1. Если ресурсы для запуска задачи доступны, тогда приложение будет иметь право на их.

Кроме того, ресурсы в YARN детализированы, поэтому приложение может запросить то, что ему нужно, а не неделимый слот, который может быть слишком большим (что приводит к расточительству ресурсов) или слишком маленьким (что может вызвать сбой) для конкретной задачи.

Мультитенантность – это возможность изолированно обслуживать пользователей из разных организаций в рамках одного сервиса

В некотором смысле, самым большим преимуществом YARN является то, что он открывает Hadoop для других типов распределенных приложений, помимо MapReduce. MapReduce – лишь одно из многих приложений YARN.

Пользователи даже могут запускать разные версии MapReduce в одном кластере YARN, что делает процесс обновления MapReduce более управляемым. (Обратите внимание, однако, что некоторые части MapReduce, такие как сервер истории заданий и обработчик тасования, а также сам YARN, по-прежнему нуждаются в обновлении в кластере.)

В идеальном мире запросы, которые делает приложение YARN, будут удовлетворяться немедленно. Однако в реальном мире ресурсы ограничены, и в загруженном кластере приложению часто приходится ждать выполнения некоторых своих запросов. Планировщик YARN должен распределять ресурсы между приложениями в соответствии с определенной политикой. Планирование в целом - сложная проблема, и не существует единой «лучшей» политики, поэтому YARN предоставляет выбор планировщиков и настраиваемых политик. Мы посмотрим на это дальше.

В YARN доступны три планировщика: FIFO, Capacity и Fair Schedulers. Планировщик FIFO помещает приложения в очередь и запускает их в порядке

отправки (первый пришел - первый ушел). Запросы на первое приложение в очереди распределяются первыми; как только его запросы удовлетворены, обслуживается следующее приложение в очереди и так далее.

Планировщик FIFO отличается тем, что прост для понимания и не требует какой-либо настройки, но он не подходит для общих кластеров. Большие приложения будут использовать все ресурсы кластера, поэтому каждое приложение должно ждать своей очереди. В общем кластере лучше использовать планировщик емкости или справедливый планировщик. Оба они позволяют выполнять длительные задания своевременно, в то же время позволяя пользователям, выполняющим параллельные небольшие специальные запросы, получать результаты в разумные сроки.

Разница между планировщиками проиллюстрирована на втором рисунке, который показывает, что в планировщике FIFO (показан на первом рисунке) небольшое задание блокируется до тех пор, пока не завершится большое задание.

В планировщике емкости (второй рисунок) отдельная выделенная очередь позволяет запускать небольшое задание сразу после его отправки, хотя это происходит за счет общего использования кластера, поскольку емкость очереди зарезервирована для заданий в этой очереди. Это означает, что большое задание завершается позже, чем при использовании планировщика FIFO.

С помощью Fair Scheduler (третий рисунок) нет необходимости резервировать установленный объем емкости, поскольку он будет динамически балансировать ресурсы между всеми выполняемыми заданиями. Сразу после запуска первого (большого) задания это единственное запущенное задание, поэтому оно получает все ресурсы в кластере. Когда начинается второе (небольшое) задание, ему выделяется половина ресурсов кластера, так что каждое задание использует свою справедливую долю ресурсов.

Обратите внимание, что существует задержка между запуском второго задания и получением его справедливой доли, поскольку ему приходится ждать, пока освободятся ресурсы в качестве контейнеров, используемых при завершении первого задания. После того, как небольшое задание завершается и больше не требует ресурсов, большое задание снова возвращается к использованию полной емкости кластера. Общий эффект - как высокая загрузка кластера, так и своевременное завершение небольших заданий.

Рисунок контрастирует с основными принципами работы трех планировщиков. В следующих двух разделах мы исследуем некоторые из более сложных вариантов конфигурации для Планировщиков Емкости и Справедливости.

Конфигурация планировщика емкости (Capacity Scheduler Configuration)

Планировщик емкости позволяет совместно использовать кластер Hadoop по организационным линиям, при этом каждой организации выделяется определенная емкость всего кластера. Каждая организация настроена с выделенной очередью, которая настроена на использование заданной части емкости кластера. Очереди могут быть дополнительно разделены иерархически, что позволяет каждой организации совместно использовать свои кластерные полномочия между различными группами пользователей внутри организации. В очереди приложения планируются с использованием расписания FIFO.

Как мы видели на последнем рисунке, отдельное задание не использует больше ресурсов, чем емкость его очереди. Однако, если в очереди находится более одного задания и доступны свободные ресурсы, то планировщик емкости может выделить резервные ресурсы для заданий в очереди, даже если это приведет к превышению емкости очереди. Такое поведение известно как эластичность очереди.

В нормальном режиме работы планировщик емкости не вытесняет контейнеры путем принудительного уничтожения их, поэтому, если очередь не хватает емкости из-за отсутствия спроса, а затем спрос увеличивается,

очередь будет возвращаться к емкости только по мере того, как ресурсы высвобождаются из других очередей в качестве контейнеров. Это можно смягчить, настроив очереди с максимальной емкостью, чтобы они не слишком сильно занимали емкости других очередей. Конечно, это происходит за счет эластичности очереди, поэтому разумный компромисс должен быть найден методом проб и ошибок.

Размещение очереди

Способ указания очереди, в которую помещается приложение, зависит от приложения. Например, в MapReduce вы устанавливаете свойство mapreduce.job.queue name равным имени очереди, которую хотите использовать. Если очередь не существует, во время отправки вы получите сообщение об ошибке. Если очередь не указана, приложения будут помещены в очередь с именем по умолчанию.

Для Планировщика емкости имя очереди должно быть последней частью иерархического имени, поскольку полное иерархическое имя не распознается. Итак, для конфигурации из предыдущего примера prod и eng подходят, но root.dev.eng и dev.eng не работают.

Конфигурация очереди

Fair Scheduler настраивается с использованием файла распределения с именем fair-scheduler.xml, который загружается из пути к классам. (Имя можно изменить, установив свойство yarn.scheduler.fair.allocation.file.) При отсутствии файла распределения Fair Scheduler работает, как описано ранее: каждое приложение помещается в очередь, названную в честь пользователя и очередей создаются динамически, когда пользователи подают свои первые заявки.

Конфигурация для каждой очереди указывается в файле распределения. Это позволяет настраивать иерархические очереди, подобные тем, которые поддерживаются планировщиком емкости. Например, мы можем определить очереди prod и dev, как мы это делали для планировщика емкости, используя файл распределения в следующем примере.

Иерархия очереди определяется с помощью вложенных элементов очереди. Все очереди являются дочерними по отношению к корневой очереди, даже если они не вложены в корневой элемент очереди. Здесь мы подразделяем очередь разработчиков на очередь с именем eng и очередь с именем science.

Очереди могут иметь веса, которые используются при расчете справедливой доли. В этом примере распределение кластеров считается справедливым, когда оно делится на пропорции 40:60 между prod и dev. Для очередей по английскому языку и науке не указан вес, поэтому они делятся поровну. Веса - это не совсем то же самое, что проценты, хотя в примере для простоты используются числа, которые в сумме дают 100. Мы могли бы указать веса 2 и 3 для очередей prod и dev, чтобы добиться того же веса очереди.

При установке весов не забывайте учитывать очередь по умолчанию и динамически создаваемые очереди (например, очереди, названные в честь пользователей). Они не указаны в файле распределения, но имеют вес 1.

Очереди могут иметь разные политики планирования. Политика по умолчанию для очередей может быть установлена в элементе defaultQueueSchedulingPolicy верхнего уровня; если он опущен, используется справедливое планирование. Несмотря на свое название, Fair Scheduler также поддерживает политику FIFO (fifo) для очередей, а также Dominant Resource Fairness (drf), описанную далее в этой главе.

Политику для конкретной очереди можно переопределить с помощью элемента schedulingPolicy для этой очереди. В этом случае очередь prod использует планирование FIFO, поскольку мы хотим, чтобы каждое производственное задание выполнялось последовательно и выполнялось в кратчайшие сроки. Обратите внимание, что справедливое совместное использование по-прежнему используется для разделения ресурсов между очередями prod и dev, а также между (и внутри) очередями eng и science.

Хотя это не показано в этом файле распределения, очереди могут быть настроены с минимальными и максимальными ресурсами и максимальным количеством запущенных приложений. Настройка минимальных ресурсов не является жестким пределом, а скорее используется планировщиком для определения приоритетов распределения ресурсов. Если две очереди ниже их справедливой доли, то ресурсы, находящиеся дальше всего ниже минимального, выделяются в первую очередь. Настройка минимального ресурса также используется для приоритетного прерывания, что обсуждается в данный момент.

Упреждение

Когда задание отправляется в пустую очередь в занятом кластере, задание не может быть запущено до тех пор, пока ресурсы не освободятся от заданий, которые уже выполняются в кластере. Чтобы время, необходимое для запуска задания, было более предсказуемым, Fair Scheduler поддерживает приоритетное прерывание.

Вытеснение позволяет планировщику уничтожать контейнеры для очередей, которые работают с большей, чем их справедливая доля ресурсов, так что ресурсы могут быть выделены очереди, которая находится под его справедливой долей. Обратите внимание, что вытеснение снижает общую эффективность кластера, так как завершенные контейнеры необходимо выполнить повторно.

Вытеснение включается глобально, если для yarn.scheduler.fair.preemption установлено значение true. Существует два соответствующих параметра тайм-аута приоритетного прерывания: один для минимального распределения и один для справедливого распределения, оба указаны в секундах. По умолчанию тайм-ауты не установлены, поэтому вам нужно установить хотя бы один, чтобы разрешить вытеснение контейнеров.

Если очередь ожидает до тех пор, пока не будет получена минимальная гарантированная доля, пока не будет получена минимальная гарантированная доля, то планировщик может вытеснить другие контейнеры. Тайм-аут по

умолчанию устанавливается для всех очередей с помощью элемента верхнего уровня defaultMinSharePreemptionTimeout в файле распределения и для каждой очереди путем установки элемента minShare PreemptionTimeout для очереди.

Аналогичным образом, если очередь остается ниже половины своей справедливой доли в течение всего времени ожидания приоритетного прерывания справедливой доли, то планировщик может вытеснить другие контейнеры. Тайм-аут по умолчанию устанавливается для всех очередей с помощью элемента верхнего уровня defaultFairSharePreemptionTimeout в файле распределения и для каждой очереди путем установки fairSharePreemptionTimeout для очереди. Пороговое значение также может быть изменено со значения по умолчанию 0,5, установив defaultFairSharePreemptionThreshold и fairSharePreemptionThres для удержания (для каждой очереди).

График задержки

Все планировщики YARN стараются учитывать запросы местоположения. В загруженном кластере, если приложение запрашивает конкретный узел, высока вероятность, что во время запроса на нем работают другие контейнеры. Очевидный способ действий - немедленно ослабить требования к месторасположению и разместить контейнер на той же стойке. Однако на практике было замечено, что короткое ожидание (не более нескольких секунд) может значительно увеличить шансы на выделение контейнера на запрошенном узле и, следовательно, повысить эффективность кластера. Эта функция называется планированием задержки и поддерживается как планировщиком мощности, так и планировщиком справедливости.

Каждый диспетчер узлов в кластере YARN периодически отправляет контрольный запрос диспетчеру ресурсов - по умолчанию один в секунду. Тактовые импульсы несут информацию о запущенных контейнерах диспетчера узлов и ресурсах, доступных для новых контейнеров, поэтому

каждый тактовый сигнал является потенциальной возможностью планирования для приложения для запуска контейнера.

При использовании планирования с задержкой планировщик не просто использует первую возможность планирования, которую он получает, но ожидает до заданного максимального количества возможностей планирования, прежде чем ослабить ограничение местоположения и воспользоваться следующей возможностью планирования.

Для Планировщика емкости планирование задержки настраивается путем установки yarn.scheduler.capacity.node-locality-delay положительного целого числа, представляющего количество возможностей планирования, которые он готов пропустить, прежде чем ослабить ограничение узла, чтобы соответствовать любому узлу в том же стойка.

Fair Scheduler также использует количество возможностей планирования для определения задержки, хотя она выражается как пропорция размера кластера. Например, установка yarn.scheduler.fair.locality.threshold.node на 0,5 означает, что планировщик должен дождаться, пока половина узлов в кластере не представит возможности планирования, прежде чем принимать другой узел в той же стойке. Существует соответствующее свойство, yarn.scheduler.fair.locality.threshold.rack, для установки порога до того, как другая стойка будет принята вместо запрошенной.

Доминирующая справедливость ресурсов

Когда планируется только один тип ресурса, например, память, тогда легко определить понятие емкости или справедливости. Если два пользователя запускают приложения, вы можете измерить объем памяти, который каждый из них использует, чтобы сравнить два приложения. Однако, когда в игре задействовано несколько типов ресурсов, все становится более сложным. Если приложению одного пользователя требуется много ЦП, но мало памяти, а другому - мало ЦП и много памяти, как сравнивать эти два приложения?

Планировщики в YARN решают эту проблему, чтобы посмотреть на доминирующий ресурс каждого пользователя и использовать его как меру

использования кластера. Такой подход называется Dominant Resource Fairness, сокращенно DRF.

Источник: *Hadoop: The Definitive Guide, Fourth Edition by Tom White Copyright © 2015 Tom White. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.*

Выполнение заданий в MapReduce

Весь процесс показан на рисунке 33. На самом высоком уровне есть пять независимых организаций:

• Клиент, отправляющий задание MapReduce.

• Менеджер ресурсов YARN, который координирует распределение вычислительных ресурсов в кластере.

• Диспетчеры узлов YARN, которые запускают и контролируют вычислительные контейнеры на машинах в кластере.

• Мастер приложения MapReduce, который координирует задачи, выполняющие задание MapReduce. Мастер приложения и задачи MapReduce выполняются в контейнерах, которые планируются менеджером ресурсов и управляются менеджерами узлов.

• Распределенная файловая система, которая используется для обмена файлами заданий между другими объектами.

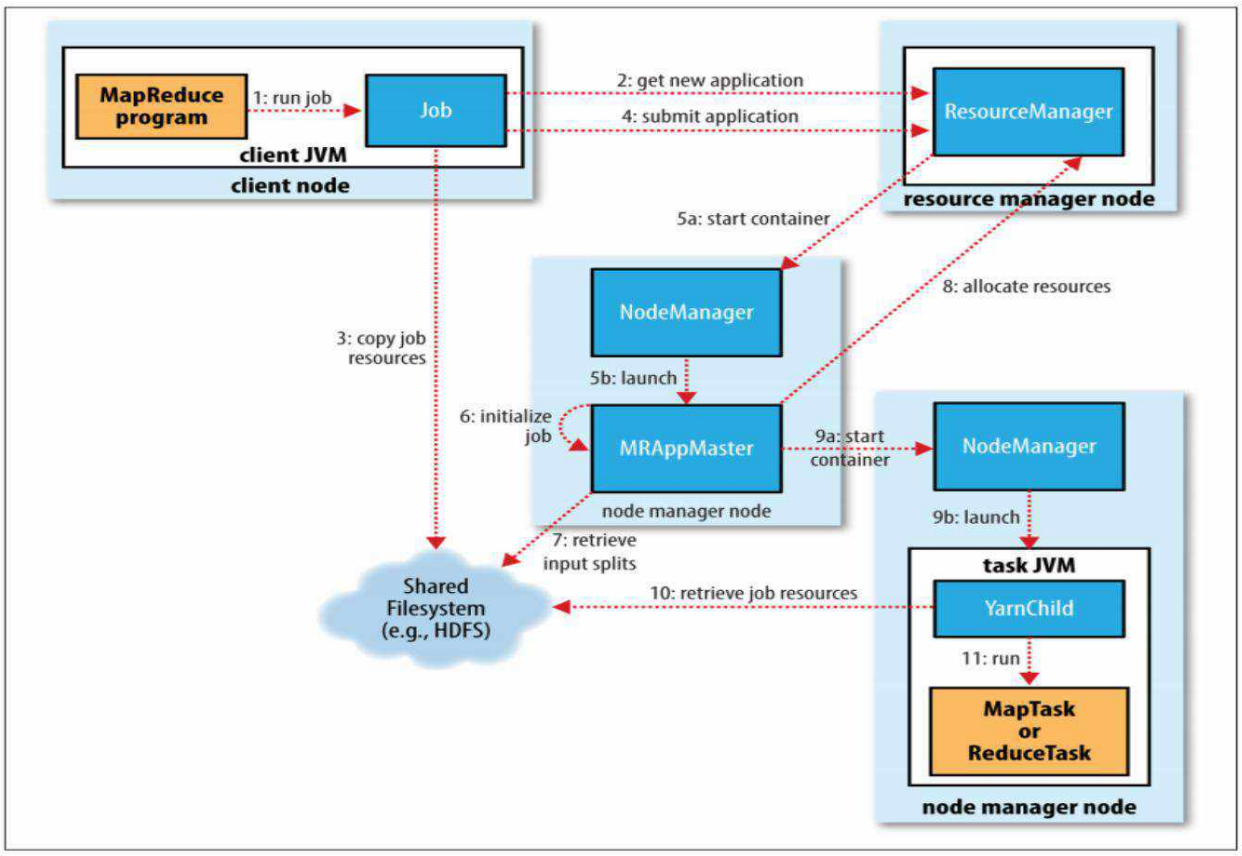


Рис. 33. Процессы MapReduce

Метод submit () в Job создает внутренний экземпляр JobSubmitter и вызывает для него submitJobInternal () (шаг 1 на рисунке). Отправив задание, wait For Completion () опрашивает ход выполнения задания раз в секунду и сообщает о ходе выполнения на консоль, если оно изменилось с момента последнего отчета. После успешного завершения задания отображаются счетчики заданий. В противном случае ошибка, которая привела к сбою задания, записывается в консоль.

Процесс отправки работы, реализованный JobSubmitter, выполняет следующие действия:

• Запрашивает у менеджера ресурсов новый идентификатор приложения, используемый для идентификатора задания MapReduce (шаг 2).

• Проверяет выходные характеристики задания. Например, если выходной каталог не указан или уже существует, задание не отправляется, а программе MapReduce выдается ошибка.

• Вычисляет входные разбиения для задания. Если расщепления не могут быть вычислены (например, из-за того, что входные пути не существуют), задание не отправляется, и программе MapReduce выдается ошибка.

• Копирует ресурсы, необходимые для выполнения задания, включая файл JAR задания, файл конфигурации и вычисленные входные разбиения, в общую файловую систему в каталог, названный по идентификатору задания (шаг 3). JAR-файл задания копируется с высоким коэффициентом репликации, поэтому в кластере имеется множество копий, к которым менеджеры узлов могут получить доступ при выполнении задач для задания.

• Отправляет задание, вызывая submitApplication () в диспетчере ресурсов (шаг 4).

Инициализация задания

Когда диспетчер ресурсов получает вызов своего метода submitApplication (), он передает запрос планировщику YARN. Планировщик выделяет контейнер, и менеджер ресурсов затем запускает там процесс мастера приложения под управлением менеджера узлов (шаги 5a и 5b).

Мастер приложения для заданий MapReduce - это приложение Java, основным классом которого является MRAppMaster. Он инициализирует задание, создавая ряд объектов бухгалтерского учета для отслеживания хода выполнения задания, поскольку он будет получать отчеты о ходе выполнения и завершении задач (шаг 6).

Затем он извлекает входные разбиения, вычисленные на клиенте, из общей файловой системы (шаг 7). Затем он создает объект задачи карты для каждого разбиения, а также ряд объектов задачи уменьшения, определенных свойством mapreduce.job.reduces. На этом этапе задачам присваиваются идентификаторы.

Мастер приложения должен решить, как выполнять задачи, составляющие задание MapReduce. Если задание небольшое, мастер приложения может решить запускать задачи в той же JVM, что и он сам. Это происходит, когда он определяет, что накладные расходы на выделение и

выполнение задач в новых контейнерах перевешивают выигрыш, который можно получить от их параллельного выполнения, по сравнению с их последовательным запуском на одном узле. Такое задание называется уберизированным или выполняется как сверхзадача.

Что считается небольшой работой? По умолчанию небольшое задание - это задание, в котором меньше 10 преобразователей, только один редуктор и размер входных данных меньше размера одного блока HDFS. Задачи Uber должны быть включены явно (для отдельного задания или для всего кластера), задав для mapreduce.job.ubertask.enable значение true.

Наконец, перед запуском каких-либо задач мастер приложения вызывает метод setupJob () для OutputCommitter. Для FileOutputCommitter, который используется по умолчанию, он создаст каталог окончательного вывода для задания и временное рабочее пространство для вывода задачи.

Назначение задачи

Если задание не соответствует требованиям для выполнения в качестве сверхзадачи, то мастер приложения запрашивает контейнеры для всей карты и сокращает количество задач в задании из диспетчера ресурсов (шаг 8). Запросы для задач карты выполняются первыми и имеют более высокий приоритет, чем запросы для задач сокращения, поскольку все задачи карты должны быть выполнены до начала фазы сортировки сокращения. Запросы на сокращение задач не выполняются до тех пор, пока не будут выполнены 5% задач карты.

Задачи сокращения могут выполняться в любом месте кластера, но запросы для задач карты имеют ограничения локальности данных, которые планировщик пытается соблюдать. В оптимальном случае задача является локальной для данных, то есть выполняется на том же узле, на котором находится разбиение. В качестве альтернативы задача может быть локальной стойкой: на той же стойке, но не на том же узле, что и разделение. Некоторые задачи не являются ни локальными данными, ни локальными стойками, и получают свои данные из стойки, отличной от той, на которой они

выполняются. Для конкретного запуска задания вы можете определить количество задач, которые выполнялись на каждом уровне местности, просмотрев счетчики задания.

В запросах также указываются требования к памяти и ЦП для задач. По умолчанию каждой задаче сопоставления и сокращения выделяется 1024 МБ памяти и одно виртуальное ядро. Значения настраиваются для каждого задания.

Выполнение задачи

После того как задаче были назначены ресурсы для контейнера на конкретном узле планировщиком диспетчера ресурсов, мастер приложения запускает контейнер, связываясь с диспетчером узлов (шаги 9a и 9b). Задача выполняется приложением Java, основным классом которого является YarnChild. Прежде чем он сможет запустить задачу, он локализует ресурсы, в которых она нуждается, включая конфигурацию задания и файл JAR, а также любые файлы из распределенного кеша (шаг 10). Наконец, он запускает задачу сопоставления или сокращения (шаг 11).

YarnChild работает в выделенной JVM, поэтому любые ошибки в определяемой пользователем карте и функциях сокращения (или даже в YarnChild) не влияют на диспетчер узлов - например, вызывая его сбой или зависание.

Каждая задача может выполнять действия по настройке и фиксации, которые выполняются в той же JVM, что и сама задача, и определяются OutputCommitter для задания. Для файловых заданий действие фиксации перемещает выходные данные задачи из временного местоположения в его окончательное местоположение. Протокол фиксации гарантирует, что при включении спекулятивного выполнения только одна из повторяющихся задач будет зафиксирована, а другая прервана.

Потоковая передача

Потоковая передача запускает специальную карту и сокращает задачи с целью запуска пользовательского исполняемого файла и связи с ним (рис. 34).

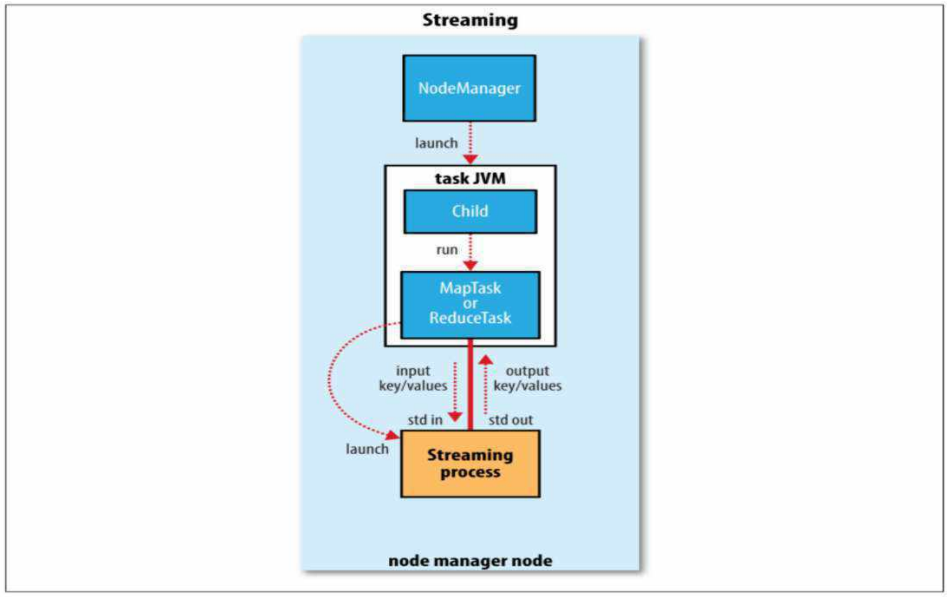


Рис. 34. Потоковая передача данных

Задача потоковой передачи взаимодействует с процессом (который может быть написан на любом языке), используя стандартные потоки ввода и вывода. Во время выполнения задачи Java-процесс передает входные пары ключ-значение внешнему процессу, который запускает его через определяемую пользователем функцию map или reduce и передает выходные пары ключ-значение обратно в Java-процесс. С точки зрения менеджера узлов, это как если бы дочерний процесс сам запускал карту или сокращал код.

Прогресс и обновления статуса

Задания MapReduce – это длительные пакетные задания, выполнение которых занимает от десятков секунд до часов. Поскольку это может быть значительный период времени, важно, чтобы пользователь получил обратную связь о том, как продвигается работа. Задание и каждая из его задач имеют статус, который включает в себя такие вещи, как состояние задания или задачи (например, выполняется, успешно завершено, не удалось), ход сопоставлений и сокращений, значения счетчиков задания и сообщение о состоянии или описание (которое может быть установлено кодом пользователя). Эти статусы меняются в процессе работы, так как же они передаются клиенту?

Когда задача выполняется, она отслеживает ее прогресс (то есть долю выполненной задачи). Для задач карты это доля обработанных входных

данных. Для задач сокращения это немного сложнее, но система все равно может оценить долю обработанных входных данных сокращения. Он делает это, разделяя общий прогресс на три части, соответствующие трем фазам перемешивания. Например, если задача запустила редуктор на половине входных данных, прогресс задачи составляет 5/6, поскольку она завершила фазы копирования и сортировки (каждая по 1/3) и находится на полпути через фазу сокращения (1/6).

Что составляет прогресс в MapReduce?

Прогресс не всегда поддается измерению, но, тем не менее, он сообщает Hadoop, что задача что-то делает. Например, задача по написанию выходных записей выполняется, даже если это не может быть выражено в процентах от общего числа, которое будет записано (поскольку последняя цифра может быть неизвестна даже задаче, производящей выходные данные). Отчеты о ходе выполнения важны, поскольку Hadoop не откажется от выполнения задачи, которая уже выполняется. Все следующие операции составляют прогресс (рис. 35):

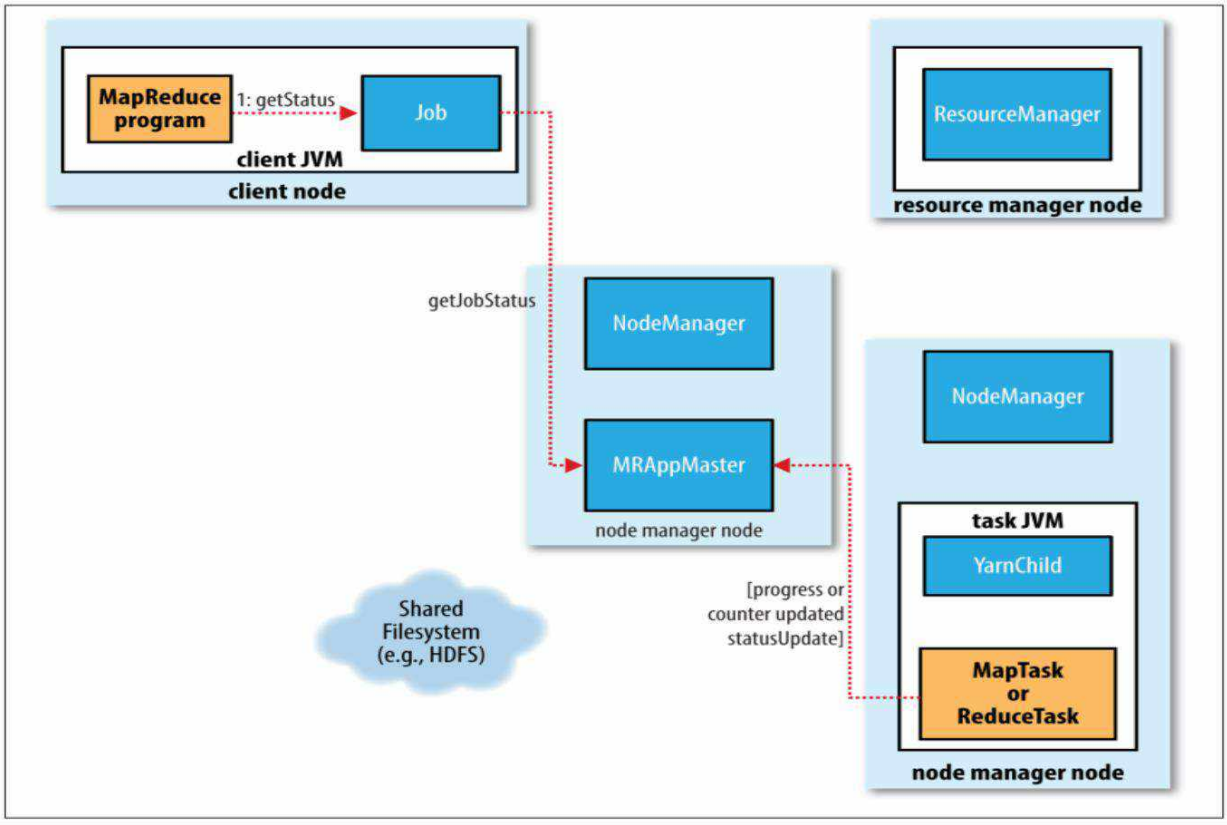


Рис. 35. Прогресс в MapReduce

• Чтение входной записи (в маппере или редукторе);

• Запись выходной записи (в маппере или редукторе);

• Установка описания статуса (с помощью метода setSta tus () Reporter или TaskAttemptContext);

• Увеличение счетчика (с помощью метода IncrCounter () Reporter или метода increment () Counter);

• Вызов метода progress () Reporter или TaskAttemptContext.

У задач также есть набор счетчиков, которые подсчитывают различные события во время выполнения задачи, которые либо встроены в структуру, например, количество записанных выходных записей карты, либо определены пользователями.

При выполнении задачи сопоставления или сокращения дочерний процесс связывается со своим родительским мастером приложения через соединительный интерфейс. Задача сообщает о своем прогрессе и состоянии (включая счетчики) своему мастеру приложения, который имеет сводное представление о задании каждые три секунды через шлангокабельный интерфейс.

Веб-интерфейс диспетчера ресурсов отображает все запущенные приложения со ссылками на веб-интерфейсы соответствующих мастеров приложений, каждый из которых отображает дополнительные сведения о задании MapReduce, включая его ход выполнения.

В ходе работы клиент получает последний статус, ежесекундно опрашивая мастера приложения. Клиенты также могут использовать метод задания getStatus () для получения экземпляра JobStatus, который содержит всю информацию о состоянии задания.

Завершение работы

Когда мастер приложения получает уведомление о завершении последней задачи задания, он меняет статус задания на «успешно». Затем, когда задание запрашивает статус, оно узнает, что задание выполнено успешно, и

распечатывает сообщение, чтобы сообщить об этом пользователю. На этом этапе на консоль выводится статистика заданий и счетчики.

Мастер приложения также отправляет уведомление о задании HTTP, если он настроен для этого. Это может быть настроено клиентами, желающими получать обратные вызовы.

Наконец, по завершении задания мастер приложения и контейнеры задач очищают свое рабочее состояние (таким образом, промежуточный вывод удаляется), и вызывается метод commit Job () класса OutputCommitter. Информация о задании архивируется сервером истории заданий, чтобы при желании пользователи могли позже опрашивать.

Сбой задачи

В реальном мире код пользователя содержит ошибки, процессы дают сбой, а машины выходят из строя. Одним из основных преимуществ использования Hadoop является его способность справляться с такими сбоями и обеспечивать успешное выполнение вашей работы. Нам необходимо учитывать отказ любой из следующих сущностей: задачи, мастера приложения, диспетчера узлов и диспетчера ресурсов.

Рассмотрим сначала случай, когда задача не выполняется. Чаще всего этот сбой возникает, когда пользовательский код в задаче сопоставления или сокращения выдает исключение времени выполнения. В этом случае JVM задачи сообщает об ошибке своему главному устройству родительского приложения перед завершением работы. В конечном итоге ошибка попадает в журналы пользователя. Мастер приложения отмечает попытку задачи как неудачную и освобождает контейнер, чтобы его ресурсы стали доступны для другой задачи.

Для задач потоковой передачи, если процесс потоковой передачи завершается с ненулевым кодом выхода, он помечается как сбойный.

Другой режим отказа - это внезапный выход из JVM задачи - возможно, есть ошибка JVM, которая заставляет JVM завершать работу при определенных обстоятельствах, обнаруженных пользовательским кодом

MapReduce. В этом случае диспетчер узлов замечает, что процесс завершился, и сообщает об этом мастеру приложения, чтобы он мог отметить попытку как неудачную.

По-разному решаются подвесные задачи. Мастер приложения замечает, что какое-то время не получает обновления о ходе выполнения, и отмечает задачу как неудачную. По истечении этого периода процесс JVM задачи будет автоматически завершен. Период ожидания, по истечении которого задачи считаются невыполненными, обычно составляет 10 минут и может быть настроен для каждого задания (или кластера), задав для свойства mapreduce.task.timeout значение в миллисекундах.

Установка нулевого значения тайм-аута отключает тайм-аут, поэтому длительные задачи никогда не помечаются как неудачные. В этом случае зависшая задача никогда не освободит свой контейнер, что со временем может привести к замедлению работы кластера. Поэтому этого подхода следует избегать, и достаточно убедиться, что задача периодически сообщает о ходе выполнения.

Когда мастер приложения получает уведомление о неудачной попытке выполнения задачи, он перепланирует выполнение задачи. Мастер приложения попытается избежать перепланирования задачи в диспетчере узлов, где она ранее не удалась. Более того, если задача не удалась четыре раза, она не будет повторяться снова. Это значение можно настроить. Максимальное количество попыток запуска задачи контролируется для уменьшения количества задач. По умолчанию, если какая-либо задача терпит неудачу четыре раза (или независимо от того, какое максимальное количество попыток настроено), все задание терпит неудачу.

Для некоторых приложений нежелательно прерывать задание, если несколько задач не удается, так как результаты задания могут быть использованы, несмотря на некоторые сбои. В этом случае для задания может быть установлен максимальный процент задач, которые могут завершиться неудачно, не вызывая сбоя задания.

Попытка задачи также может быть прервана, что отличается от неудачной попытки. Попытка задачи может быть прекращена из-за того, что она является предполагаемой дубликатом, или из-за того, что диспетчер узлов, на котором она выполнялась, завершился неудачно, а мастер приложения пометил все попытки задачи, выполнявшиеся на нем, как убитые. Успешные попытки выполнения задачи не учитываются при подсчете количества попыток запуска задачи, потому что попытка была прервана не по вине задачи. Пользователи также могут прекращать попытки выполнения задач или отказываться от них с помощью веб-интерфейса или командной строки (введите mapred job, чтобы увидеть параметры). Рабочие места могут быть убиты одними и теми же механизмами.

Ошибка мастера приложений

Подобно тому, как задачам MapReduce дается несколько попыток для успешного выполнения (перед лицом сбоев оборудования или сети), приложения в YARN повторяются в случае сбоя. Максимальное количество попыток запустить мастер приложения MapReduce контролируется специальным свойством. Значение по умолчанию - 2, поэтому, если мастер приложения MapReduce выйдет из строя дважды, повторная попытка не будет выполнена, и задание завершится ошибкой.

YARN налагает ограничение на максимальное количество попыток для любого мастера приложения YARN, работающего в кластере, и отдельные приложения не могут превышать этот предел. Предел устанавливается yarn.resourcemanager.am.max-plays и по умолчанию равен 2, поэтому, если вы хотите увеличить количество попыток главного приложения MapReduce, вам также придется увеличить настройку YARN в кластере.

Восстановление работает следующим образом. Мастер приложения отправляет периодические контрольные сигналы диспетчеру ресурсов, и в случае сбоя главного приложения диспетчер ресурсов обнаружит сбой и запустит новый экземпляр мастера, работающего в новом контейнере (управляемом диспетчером узлов). В случае мастера приложения MapReduce

он будет использовать историю заданий для восстановления состояния задач, которые уже были запущены (неудачным) приложением, чтобы их не нужно было запускать повторно. Восстановление включено по умолчанию, но его можно отключить, установив значение false.

Клиент MapReduce опрашивает мастер приложения на предмет отчетов о ходе выполнения, но если его мастер приложения выходит из строя, клиенту необходимо найти новый экземпляр. Во время инициализации задания клиент запрашивает у диспетчера ресурсов адрес мастера приложения, а затем кэширует его, чтобы не перегружать диспетчер ресурсов запросом каждый раз, когда ему нужно опрашивать мастер приложения. Однако, если мастер приложения выходит из строя, у клиента возникает тайм-аут, когда он выдает обновление статуса, после чего клиент вернется к диспетчеру ресурсов, чтобы запросить адрес нового мастера приложения. Этот процесс прозрачен для пользователя.

Ошибка менеджера узла

Если диспетчер узлов выйдет из строя из-за сбоя или очень медленной работы, он перестанет отправлять контрольные сигналы диспетчеру ресурсов (или отправлять их очень редко). Диспетчер ресурсов заметит, что диспетчер узлов прекратил посылать контрольные сигналы, если он не получил их в течение 10 минут, и удалит его из своего пула узлов, чтобы запланировать включение контейнеров.

Любой мастер задачи или приложения, запущенный на отказавшем диспетчере узлов, будет восстановлен с использованием механизмов, описанных в предыдущих двух разделах. Кроме того, мастер приложения организует повторный запуск задач сопоставления, которые были успешно запущены и успешно завершены на отказавшем диспетчере узлов, если они относятся к незавершенным заданиям, поскольку их промежуточный вывод, находящийся в локальной файловой системе отказавшего диспетчера узлов, может быть недоступен для задача уменьшения.

Менеджеры узлов могут быть занесены в черный список, если количество отказов для приложения велико, даже если сам менеджер узлов не отказал. Внесение в черный список выполняется мастером приложения, а для MapReduce мастер приложения будет пытаться перепланировать задачи на разных узлах, если на диспетчере узлов не удается выполнить более трех задач. Пользователь может установить порог с помощью свойства задания.

Обратите внимание, что диспетчер ресурсов не вносит в черный список приложений (на момент написания), поэтому задачи из новых заданий могут быть запланированы на неисправных узлах, даже если они были внесены в черный список мастером приложения, выполняющим более раннее задание.

Ошибка диспетчера ресурсов

Сбой менеджера ресурсов - серьезная проблема, потому что без него невозможно запустить ни задания, ни контейнеры задач. В конфигурации по умолчанию диспетчер ресурсов представляет собой единую точку отказа, поскольку в (маловероятном) случае сбоя машины все выполняемые задания завершаются сбоем и не могут быть восстановлены.

Для достижения высокой доступности (HA) необходимо запустить пару менеджеров ресурсов в конфигурации «активный-резервный». Если активный менеджер ресурсов выходит из строя, то резервный может взять на себя управление без значительного прерывания работы клиента.

Информация обо всех запущенных приложениях хранится в высокодоступном хранилище состояний (поддерживаемом ZooKeeper или HDFS), так что резервный может восстановить состояние ядра отказавшего активного диспетчера ресурсов. Информация о менеджере узлов не сохраняется в хранилище состояний, поскольку она может быть относительно быстро восстановлена новым менеджером ресурсов, когда менеджеры узлов отправляют свои первые контрольные сигналы. (Обратите также внимание на то, что задачи не являются частью состояния диспетчера ресурсов, поскольку ими управляет мастер приложения. Таким образом, объем сохраняемого состояния гораздо более управляем, чем у трекера заданий в MapReduce 1.)

Когда запускается новый диспетчер ресурсов, он считывает информацию о приложении из хранилища состояний, а затем перезапускает мастера приложений для всех приложений, работающих в кластере. Это не считается неудачной попыткой приложения, поскольку приложение не завершилось неудачно из-за ошибки в коде приложения, а было принудительно уничтожено системой. На практике перезапуск главного приложения не является проблемой для приложений MapReduce, поскольку они восстанавливают работу, выполненную выполненными задачами.

Переход диспетчера ресурсов из режима ожидания в активный выполняется контроллером аварийного переключения. Контроллер аварийного переключения по умолчанию является автоматическим, который использует выбор лидера ZooKeeper, чтобы гарантировать, что одновременно будет только один активный менеджер ресурсов. В отличие от HDFS HA, контроллер аварийного переключения не обязательно должен быть автономным процессом и по умолчанию встроен в диспетчер ресурсов для простоты настройки. Также можно настроить переключение вручную, но это не рекомендуется.

Клиенты и диспетчеры узлов должны быть настроены для обработки аварийного переключения диспетчера ресурсов, поскольку теперь есть два возможных диспетчера ресурсов для связи. Они пытаются подключиться к каждому менеджеру ресурсов по круговой схеме, пока не найдут активный. Если активный выйдет из строя, они будут повторять попытки, пока резервный не станет активным.

Перемешивание и сортировка (Shuffle and Sort)

MapReduce гарантирует, что входные данные для каждого редуктора отсортированы по ключу. Процесс, с помощью которого система выполняет сортировку и передает выходные данные карты редукторам в качестве входных данных, известен как перемешивание. В этом разделе мы рассмотрим, как работает перемешивание, поскольку базовое понимание будет полезно, если вам понадобится оптимизировать программу MapReduce.

Перестановка - это область кодовой базы, в которой постоянно вносятся уточнения и улучшения, поэтому следующее описание обязательно скрывает многие детали. Во многих отношениях перемешивание - это сердце MapReduce, и именно здесь происходит «волшебство» (рис. 36).

Сторона свертки (map)

Когда функция карты начинает производить вывод, он не просто записывается на диск. Этот процесс более сложен и использует преимущества буферизации записи в памяти и выполнения некоторой предварительной сортировки по соображениям эффективности. На этом рисунке показано, что происходит.

Каждая задача карты имеет кольцевой буфер памяти, в который записываются выходные данные. По умолчанию размер буфера составляет 100 МБ. Когда содержимое буфера достигает определенного порогового размера, который имеет значение по умолчанию 80%, фоновый поток начинает передавать содержимое на диск. Выходные данные карты будут продолжать записываться в буфер, пока происходит разлив, но если буфер заполняется в течение этого времени, карта будет блокироваться до тех пор, пока разлив не будет завершен. Разливы записываются в циклическом порядке в определенные каталоги в подкаталоге, относящемся к заданию.

Перед записью на диск поток сначала разделяет данные на разделы, соответствующие редукторам, на которые они в конечном итоге будут отправлены. Внутри каждого раздела фоновый поток выполняет сортировку в памяти по ключу, и, если есть функция объединения, он запускается на выходе сортировки. Выполнение функции объединителя делает вывод карты более компактным, поэтому меньше данных для записи на локальный диск и передачи в редуктор.

Каждый раз, когда буфер памяти достигает порога сброса, создается новый файл сброса, поэтому после того, как задача карты записала свою последнюю выходную запись, может быть несколько файлов сброса. Перед завершением задачи файлы утечки объединяются в один разделенный и

отсортированный выходной файл. Свойство конфигурации контролирует максимальное количество потоков для одновременного слияния; по умолчанию - 10.

Если имеется по крайней мере три файла сброса, объединитель запускается снова перед записью выходного файла. Напомним, что комбайнеры могут многократно запускаться по входу, не влияя на конечный результат. Если есть только один или два сброса, потенциальное уменьшение размера вывода карты не стоит накладных расходов при вызове комбайнера, поэтому он не запускается снова для этого вывода карты.

Часто рекомендуется сжимать выходные данные карты по мере их записи на диск, потому что это ускоряет запись на диск, экономит дисковое пространство и уменьшает объем данных, передаваемых редуктору. По умолчанию вывод не сжимается, но это легко включить, установив для mapreduce.map.output.compress значение true. Используемая библиотека сжатия указывается в mapreduce.map.output.compress.codec.

Разделы выходного файла становятся доступными для редукторов через HTTP. Максимальное количество рабочих потоков, используемых для обслуживания разделов файлов, контролируется свойством shuffle.max; этот параметр относится к диспетчеру узлов, а не к задаче карты. Значение по умолчанию 0 устанавливает максимальное количество потоков, равное удвоенному количеству процессоров на машине.

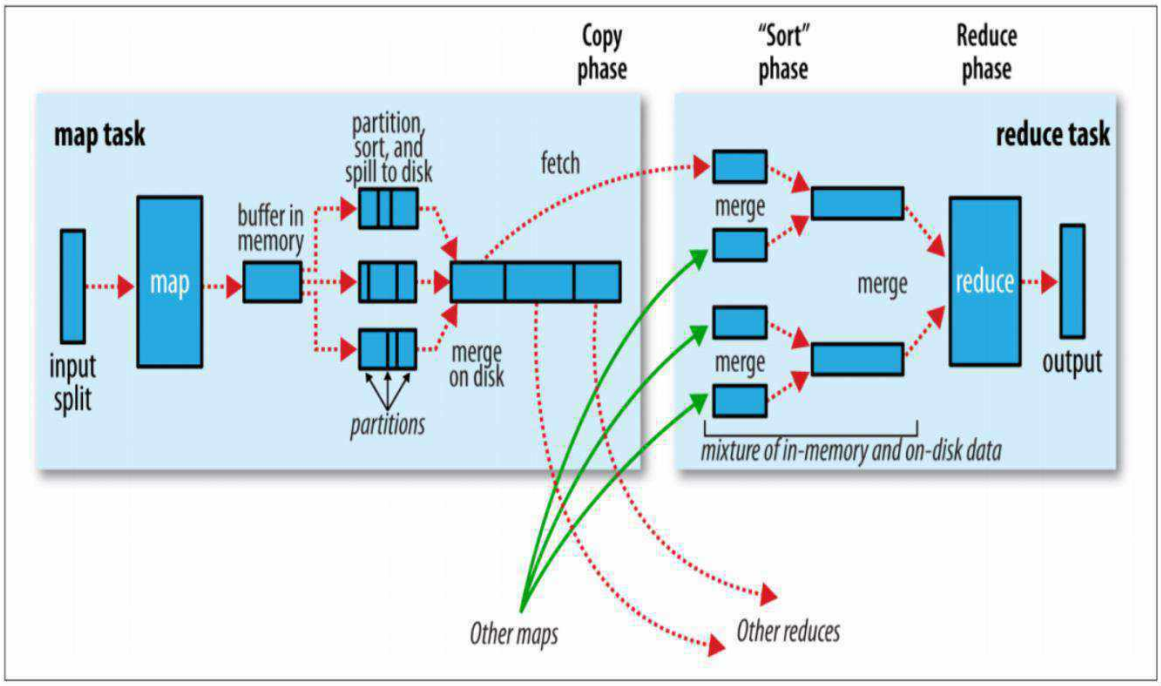


Рис. 36. Перемешивание, сортировка и свёртка

Сторона уменьшения

Теперь перейдем к сокращающей части процесса. Выходной файл карты находится на локальном диске машины, на которой выполнялась задача карты (обратите внимание, что, хотя выходные данные карты всегда записываются на локальный диск, сокращение выходных данных может не выполняться), но теперь он необходим машине, которая собирается запустите задачу сокращения для раздела. Более того, для задачи сокращения требуется вывод карты для ее конкретного раздела из нескольких задач карты в кластере. Задачи карты могут завершаться в разное время, поэтому задача уменьшения начинает копировать свои выходные данные, как только каждая из них завершается. Это называется этапом копирования задачи сокращения. Задача уменьшения имеет небольшое количество потоков копировального устройства, поэтому она может получать выходные данные карты параллельно.

Как редукторы узнают, с каких машин получить вывод карты?

После успешного выполнения задач карты они уведомляют своего хозяина приложения с помощью механизма тактового сигнала. Следовательно, для данного задания мастер приложения знает соответствие

между выходными данными карты и хостами. Поток в редукторе периодически запрашивает у мастера выходные узлы карты, пока он не получит их все.

Хосты не удаляют выходные данные карты с диска, как только первый редуктор их получил, так как редуктор может впоследствии выйти из строя. Вместо этого они ждут, пока мастер приложения не скажет им удалить их, то есть после завершения задания.

Выходные данные карты копируются в память JVM задачи уменьшения, если они достаточно малы (размер буфера контролируется пропорционально кучи, используемой для этой цели); в противном случае они копируются на диск. Когда буфер в памяти достигает порогового размера, он объединяется и переносится на диск. Если указан комбайнер, он будет запущен во время слияния, чтобы уменьшить объем данных, записываемых на диск.

По мере накопления копий на диске фоновый поток объединяет их в более крупные отсортированные файлы. Это сэкономит время на последующем слиянии. Обратите внимание, что любые выходные данные карты, которые были сжаты (задачей карты), должны быть распакованы в памяти, чтобы выполнить их слияние.

Когда все выходные данные карты скопированы, задача уменьшения переходит в фазу сортировки (которую следует правильно называть фазой слияния, поскольку сортировка выполнялась на стороне карты), которая объединяет выходы карты, сохраняя порядок их сортировки. Это делается раундами. Например, если было 50 выходных карт и коэффициент слияния был равен 10 (по умолчанию, управляемый свойством mapreduce.task.sort.factor, как и при слиянии карты), будет пять раундов. В каждом раунде 10 файлов объединяются в 1, поэтому в конце будет 5 промежуточных файлов.

Вместо того, чтобы иметь последний раунд, который объединяет эти пять файлов в один отсортированный файл, слияние экономит поездку на диск, напрямую загружая функцию сокращения на последней фазе: фазе

сокращения. Это окончательное слияние может происходить из сочетания сегментов в памяти и на диске.

Источник: *Hadoop: The Definitive Guide, Fourth Edition by Tom White Copyright © 2015 Tom White. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.*