Министерство образования и науки Российской Федерации

ГБПОУ «Дзержинский педагогический колледж»

**Методические рекомендации**

**для обучающихся по проведению лекций и практических занятий**

**по МДК 05.01 Проектирование и дизайн информационных систем**

**Дзержинск - 20**

|  |  |
| --- | --- |
| Одобрено на заседании ПЦК преподавателей спец. Информатика  Протокол \_\_\_\_\_\_\_от\_\_\_\_\_\_\_\_  Председатель ПЦК \_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_/  Составитель:Руденко Н.А. | Методические разработки практических занятий составлены в соответствии с требованиями федеральных государственных образовательных стандартов среднего профессионального образования по спец. 09.02.07. Информационные системы и программирование |

Методические разработки по проведению лекций и практических занятий по МДК 05.01 Проектирование и дизайн информационных систем специальность 09.02.07. Информационные системы в программировании содержат план практического занятия с указанием вида практической работы в соответствии с рабочей программой учебной дисциплины. В них имеется необходимый теоретический материал, включены задачи для самостоятельной работы и вопросы для самоконтроля.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Тема №1. Основные понятия и определения ИС. Жизненный цикл информационных систем. Основные процессы жизненного цикла | 4 |
| Тема №2. Вспомогательные и организационные процессы жизненного цикла ПО | **7** |
| Тема №3. Организация и методы сбора информации. Анализ предметной области.  Основные понятия системного и структурного анализа | 9 |
| Лекция №4. Основные модели построения информационных систем, их структура, особенности и область применения. Классификация информационных систем | 20 |
| Тема №5. Принципы построения модели IDEF0: контекстная диаграмма, субъект | 25 |
| Тема №6. Построение диаграммы переходов состояний | 29 |
| Тема №7. Построение диаграммы потоков данных | 32 |
| Тема №8. Построение структурной и функциональной схем | 37 |
| Тема №9. Диаграммы вариантов использования | 40 |
| Тема №10. Диаграммы деятельности | 46 |
| Тема №11. Диаграммы последовательности | 49 |
| Тема№12. Диаграмма «Сущность-связь» | 51 |
| Тема№13. Диаграммы классов | 55 |
| Тема №14. Основные сведения о прототипах. Виды прототипов | 59 |
| Тема №15. Основные правила и принципы создания интерфейса | 61 |
| Тема №16. Оценка эффективности функционирования информационных систем | 63 |

**Тема №1**

**Основные понятия и определения ИС. Жизненный цикл информационных систем.**

**Основные процессы жизненного цикла**

**Цель:** ознакомиться с основными понятиями жизненного цикла

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

***Основные понятия и определения ИС***

**Технология программирования** — совокупность методов и средств, применяемых в процессе разработки программного обеспечения.

**Программа (program, routine)** — упорядоченная последовательность команд (инструкций) компьютера для решения задачи.

**Программное обеспечение (software)** — совокупность программ обработки данных и необходимых для их эксплуатации документов.

**Задача (problem, task)** — проблема, подлежащая решению.

**Приложение (application)** — программная реализация на компьютере решения задачи-

Термин «задача» в программировании означает единицу работы вычислительной системы, требующую выделения вычислительных ресурсов (процессорного времени, памяти).

Процесс создания программ можно представить как последовательность следующих действий:

1) постановка задачи;

2) алгоритмизация решения задачи;

3) программирование.

**Постановка задачи (problem definition)** — это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

**Алгоритм** — система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (выходной информации) в желаемый результат (выходную информацию) за конечное число шагов.

**Программирование (programming)** — теоретическая и практическая деятельность, связанная с созданием программ.

По отношению к ПО компьютерные пользователи делятся на следующие группы:

1) системные программисты. Занимаются разработкой, эксплуатацией и сопровождением системного программного обеспечения;

2) прикладные программисты. Осуществляют разработку и отладку программ для решения различных прикладных задач;

3) конечные пользователи. Имеют элементарные навыки работы с компьютером и используемыми ими прикладными программами;

4) администраторы сети. Отвечают за работу вычислительных сетей;

5) администраторы баз данных. Обеспечивают организационную поддержку базы данных.

**Сопровождение программы** — поддержка работоспособности программы, переход на ее новые версии, внесения изменений, исправление ошибок и т. д.

**Основные характеристики программ:**

1) алгоритмическая сложность;

2) состав функций обработки информации;

3) объем файлов, используемых программой;

4) требования к операционной системе (ОС) и техническим средствам обработки, в том числе объем дисковой памяти, размер оперативной памяти для запуска программы, тип процессора, версия ОС, наличие вычислительной сети и т. д.

**Показатели качества программы:**

1) мобильность (многоплатформенность) — независимость от технического комплекса системы обработки данных, ОС, сетевых возможностей, специфики предметной области задачи и т. д.;

2) надежность — устойчивость, точность выполнения предписанных функций обработки, возможность диагностики возникающих ошибок в работе программы;

3) эффективность как с точки зрения требований пользователя, так и расхода вычислительных ресурсов;

4) учет человеческого фактора — дружественный интерфейс, контекстно-зависимая подсказка, хорошая документация;

5) модифицируемость — способность к внесению изменений, например, расширение функций обработки, переход на другую техническую базу обработки и т. п.

6) коммуникативность — максимально возможная интеграция с другими программами, обеспечение обмена данными между программами.

Все программы по характеру использования и категориям пользователей можно разделить на два класса — утилитарные программы и программные продукты.

**Утилитарные программы** («программы для себя») предназначены для удовлетворения нужд их разработчиков. Чаще всего такие программы выполняют роль отладочных приложений, являются программами решения задач, не предназначенных для широкого распространения.

**Программные продукты (изделия)** используются для удовлетворения потребностей пользователей, широкого распространения и продажи.

В настоящее время существуют и другие варианты легального распространения программных продуктов, которые появились с использованием глобальных телекоммуникаций:

* freeware — бесплатные программы, свободно распространяемые, поддерживаются самим пользователем, который правомочен вносить в них необходимые изменения;
* shareware некоммерческие (условно-бесплатные) программы, которые могут использоваться, как правило, бесплатно. Ряд производителей использует OEM-программы (Original Equipment Manufacturer), т. е. встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с компьютерами.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации (отлажен), иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак изготовителя, а также наличие кода государственной регистрации.

***Жизненный цикл информационных систем.***

В современном мире всеобщей компьютеризации и информатизации требования, предъявляемые к программному обеспечению (ПО) вообще и к программным продуктам (ПП), программным средствам (ПС) и программам в частности, весьма высоки. В связи с этим обеспечение удовлетворяющих пользователя потребительских качеств программы, таких как надежность, быстродействие, соответствие заявленным возможностям, полнота документации, возможности расширения, развития и т. д., без строгого соблюдения определенной технологии практически невозможно.

В основе разработки и дальнейшего применения программного обеспечения пользователем лежит понятие жизненного цикла, который, в сущности, является моделью его создания и использования, отражающей различные состояния, начиная с момента осознания необходимости появления данного ПО и заканчивая моментом его полного выхода из употребления [5].

Структуру жизненного цикла ПП, состав процессов, действия и задачи, которые должны быть выполнены во время создания ПП, определяет и регламентирует международный стандарт ISO/IEC 12207 Standard for Information Technoiogy — Software Life Cycle Processes (Стандарт информационных технологий – процессы жизненного цикла программ).

Этот международный стандарт определяет действия, которые могут быть выполнены на протяжении жизненного цикла программного обеспечения.

В России, начиная с 1970-х годов, создание ПП регламентировалось стандартами ЕСПД (Единая система программной документации, серия ГОСТ 19.ХХХ), которые были ориентированы на класс относительно простых программ, небольшого объема, создаваемых отдельными программистами. В настоящее время указанные стандарты устарели концептуально и по форме, их сроки действия закончились и дальнейшее использование нецелесообразно. Поэтому в отечественных разработках следует использовать современные международные стандарты.

В соответствии с ISO/IEC 12207 выделяют 5 основных процессов жизненного цикла программных продуктов, 8 вспомогательных процессов и 4 организационных процесса.

***Основные процессы жизненного цикла.***

**1).** **Процесс приобретения.** Определяет действия предприятия-покупателя, которое приобретает систему, программный продукт или сервис программного обеспечения. Этот процесс состоит из следующих действий:

1. инициирование: определение заказчиком своих потребностей, анализ требований к системе, принятие решения о приобретении, разработке или усовершенствовании существующего ПП, проверка наличия необходимой документации, подготовка и утверждение плана приобретения, включающего требования к системе, тип договора, ответственность сторон и т.д.;
2. заявка на подготовку предложения, содержащая требования к разрабатываемой или покупаемой системе, перечень необходимых ПП, технические ограничения;
3. подготовка контракта и модернизация: выбор поставщика, заключение договора;
4. текущий контроль (мониторинг) поставщика;
5. принятие и завершение.

**2). Процесс поставки.** Определяет действия предприятия-поставщика, которое снабжает покупателя системой, программным продуктом или сервисом программного обеспечения. Этот процесс состоит из следующих действий:

1. Инициирование (начало): рассмотрение, дополнение и утверждение поставщиком заявочных предложений;
2. Подготовка ответа;
3. Заключение контракта;
4. Планирование;
5. Выполнение и контроль;
6. Оценка и проверка;
7. Поставка и завершение.

**3). Процесс разработки**. Определяет действия предприятия-разработчика, которое разрабатывает принцип построения программного изделия и программный продукт. Этот процесс состоит из действий:

1. Реализация процесса.
2. Анализ системных требований.
3. Проектирование архитектуры системы.
4. Анализ требований программного обеспечения.
5. Архитектура программного обеспечения.
6. Детальное конструирование программного обеспечения.
7. Кодирование и тестирование программного обеспечения.
8. Интеграция программного обеспечения.
9. Квалификационные испытания программного обеспечения.
10. Интеграция системы.
11. Квалификационные испытания системы.
12. Установка программного обеспечения.
13. Поддержка принятия программного обеспечения.

**4). Процесс эксплуатации**. Определяет действия персонала эксплуатации, который обеспечивает обслуживание вычислительной системы в процессе ее функционирования в интересах пользователей. Этот процесс содержит следующие действия:

1. Подготовительная работа: планирование работ, выполняемых в процессе эксплуатации, установка эксплуатационных стандартов, определение процедур локализации и решения проблем, возникающих в процессе эксплуатации.
2. Операционное (эксплуатационное) тестирование.
3. Функционирование системы.
4. Поддержка пользователя.

**5). Процесс сопровождения.** Представляет собой управление модификациями программно продукта, поддержку его текущего состояния и функциональную пригодность и включает в себя инсталляцию и удаление программного изделия на вычислительной системе. Этот процесс активизируется, когда программный продукт подвергается модификациям для кодирования и связывания (совмещения документации), вызванными проблемой или потребностью улучшения и адаптации. Цель - изменить существующий программный продукт, сохраняя его целостность. Этот процесс включает инсталляцию и удаление программного продукта. Процесс заканчивается удалением программного продукта.

**Тема №2**

**Вспомогательные и организационные процессы жизненного цикла ПО**

**Цель:** изучить процессы жизненного цикла

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

***Вспомогательные процессы жизненного цикла***

Вспомогательный процесс поддерживает реализацию другого процесса, будучи неотъемлемой частью всего жизненного цикла программного изделия, с определенной целью и обеспечивает должное качество проекта программного обеспечения. Вспомогательный процесс используется и выполняется по мере необходимости и инициируется другим процессом.

**1). Процесс документирования.** Это процесс для записи информации, произведенной процессом жизненного цикла или деятельностью. Процесс содержит набор действий, которые планируют, проектируют, разрабатывают, производят, редактируют, распределяют и сопровождают те документы, в которых нуждаются все заинтересованные лица, такие как менеджеры, инженеры и пользователи системы или программного продукта. Включает следующие действия: подготовительная работа, проектирование и разработка документации, выпуск документации, сопровождение (обновление) документации.

**2). Процесс управления конфигурацией.** Конфигурация – совокупность функциональных и физических характеристик, установленных в технической документации и реализованных в ПП. Процесс управления конфигурацией - это процесс применения административных и технических процедур на всем протяжении жизненного цикла программного обеспечения для:

* идентификации, определения и базирования единиц программного обеспечения в системе;
* управления модификацией и выпуском изделий;
* записи и сообщения о состоянии изделия;
* управление и контролирования хранения, обращения (ухода) и поставок изделий.

**3). Процесс обеспечения качества**. Определяет действия для объективной гарантии, что программные продукты и процессы соответствуют определенным требованиям к ним и придерживаются установленным замыслам. Чтобы быть беспристрастной, гарантия качества должна иметь организационную свободу и власть (полномочия) от людей, непосредственно ответственных за разработку программного продукта или выполнение процесса в проекте. Гарантия качества может иметь внутреннюю или внешнюю зависимость от того, демонстрируются ли доказательства качества продукта или процесса к управлению поставщика или покупателя.

Гарантия качества может использовать результаты других обеспечивающих процессов, таких как Верификация, Аттестация, Совместный Обзор, Проверки и Разрешение проблем.

**4). Процесс верификации**. Это процесс для определения, выполняют ли программные продукты действия требования или условия, наложенные на них в предыдущих действиях. Для эффективности затрат и выполнения, верификация должна быть интегрирована как можно раньше с процессом (таким как поставка, разработка, финансирование или сопровождение), который использует ее. Этот процесс может включать анализ, обзор и испытание. Этот процесс может быть выполнен с различными степенями независимости. Степень независимости может колебаться от одного и тогоже лица или другого лица в одной и той же организации до лица в другой организации с различными степенями разделения. В случае когда процесс выполнен администрацией, независящей от поставщика, оператора, разработчика или персонала сопровождения, он называется Независимый процесс верификации.

**5). Процесс аттестации**. Представляет собой процесс определения соответствия требований и конечной технической системы или программного продукта их функциональному использованию. Аттестация проводится на ранних стадиях. Этот процесс может проходить как часть Обеспечения Принятия Программного Обеспечения. Этот процесс может выполняться с различной степенью независимости. В случае, когда процесс выполняется организацией, независимой от поставщика, разработчика, персонала эксплуатации или персонала сопровождения, он называется Независимым Процессом Аттестации.

**6). Процесс совместной оценки.** Определяет действия для оценки состояния и результатов какого либо действия. Совместные оценки применяются в управлении проектом , в технических уровнях и проводятся в срок действительности контракта. Этот процесс может быть использован любыми двумя сторонами, где одна сторона (проверяющая, рецензирующая) проверяет (рецензирует) другую сторону (проверяемую) на совместном форуме.

**7). Процесс проверки (аудита).** Определяет деятельность для определения соответствия с требованиями, замыслами и контрактом. Этот процесс может быть использован любыми двумя сторонами, где одна сторона (проверяющая) проверяет программные продукты или деятельность другой стороны (проверяемой). Аудит служит для установления соответствия реальных работ и отчетов требованиям, планам и контракту. Аудиторы (ревизоры) не должны иметь прямой зависимости от разработчиков ПП. Они оценивают состояние работ, использование ресурсов, соответствие документации спецификациям и стандартам, корректность проводимого тестирования.

**8). Процесс решения проблем.** Определяет процесс анализа и устранения проблем (включая несоответствия), какова бы ни была их природа или источник, которые были обнаружены на протяжении разработки, эксплуатации, сопровождения или других процессов.

***Организационные процессы жизненного цикла***

Организационные процессы жизненного цикла выполняются какой-либо организацией с целью создания и обеспечения деятельности какой-либо нижестоящей структуры, включающей в себя связанные процессы жизненного цикла и персонал и совершенствования структуры и процессов. Они, как правило, инвариантны относительно конкретных проектов и контрактов, однако, уроки, извлеченные из таких проектов и контрактов, способствуют совершенствованию организации. Организационные процессы включают в себя:

1). Процесс управления. Определяет основную деятельность управления, включая проектный менеджмент: управление выпуском продукта, проектом и задачами соответствующих процессов, таких как приобретение, поставка, разработка, эксплуатация, сопровождение и др.

2). Процесс создания инфраструктуры. Охватывает выбор и поддержку технологии, стандартов и инструментальных средств, выбор и установку программных и аппаратных средств, используемых для разработки, эксплуатации, сопровождения ПП. Инфраструктура должна модифицироваться и сопровождаться в соответствии с изменениями требований к соответствующим процессам. Инфраструктура в свою очередь, является одним из объектов управления конфигурацией.

3). Процесс усовершенствования. Определяет основные действия, которые организация (покупатель, поставщик, разработчик, оператор или менеджер другого процесса) выполняет для создания, оценки, управления и совершенствования их процесса жизненного цикла. Усовершенствование направлено на повышение производительности труда за счет совершенствования используемой технологии, методов управления, выбора инструментальных средств и обучения персонала. Усовершенствование основано на анализе достоинств и недостатков каждого процесса. Такому анализу в большой степени способствует накопление в организации исторической, технической и иной информации по реализованным проектам.

4). Процесс обучения. Охватывает первоначальное обучение и последующее повышение квалификации персонала. Содержание процесса обучения определяется требованиями к проекту. Разработка учебных материалов является неотъемлемой частью процесса обучения, так как позволяет существенно повысить его эффективность и качество.

**Тема №3**

**Организация и методы сбора информации. Анализ предметной области.**

**Основные понятия системного и структурного анализа**

**Цель:** научиться анализировать предметную область

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**Организация и методы сбора информации**

После определения типа необходимой информации следует выбрать методы сбора данных. Первичные данные могут быть получены путем наблюдений, опросов фокусных групп, сбора статистических данных и экспериментов. По сравнению с первичными данными вторичные собирать значительно легче. Если разговор идет о сборе данных внутри организации, то их берут из отчетов компании, из бесед с сотрудниками отдела сбыта и другими руководителями и сотрудниками, из маркетинговой информационной системы. Эти результаты сводятся в таблицы, по ним строят графики, и уже на их основе делают соответствующие выводы.

Главными методами сбора первичных данных являются наблюдение и обследование, а также другие методы, перечисленные ниже.

*Наблюдение –* сбор первичной информации путем наблюдения за выбранными группами людей, действиями и ситуациями, например за поведением покупателей в магазине. Данные могут быть получены при наблюдении за вовлеченными в изучаемую ситуацию лицами и анализе сопутствующих обстоятельств. Служащие концерна «Калина» могут обосноваться в магазинах, на выставках косметики, на рынке, слушая, что говорят люди о разных косметических фирмах, и следя за выполнением персоналом компании своих обязанностей.

*Обследование* заключается в сборе первичной информации путем прямого опроса людей относительно уровня их знаний, отношений к продукту, предпочтений и покупательского поведения. Обследование может носить структуризированный и неструктуризированный характер. В первом случае все опрашиваемые отвечают на одни и те же вопросы, во втором – интервьюер задает вопросы в зависимости от полученных ответов.

*Опрос* – это, как правило, полезный ознакомительный шаг перед началом широкомасштабных исследований. Например, опрос, проведенный в Великобритании, выявил, что наиболее часто при проведении маркетинговых исследований используются следующие методы и источники получения маркетинговой информации: отчеты компании – 70%, вторичная информация – 56,6%, результаты опроса потребителей – 27,9%, качественные исследования – 25,6%, полевые эксперименты – 18,4%, результаты опроса дистрибьюторов – 17,2%, лабораторные эксперименты – 13,1%.

*Опрос фокус-групп. Фокус-группа* состоит из шести-десяти человек, которых просят провести несколько часов в обществе опытного ведущего для обсуждения товара, услуги, компании или какого-либо другого объекта маркетингового исследования. Ведущий должен быть объективным, хорошо осведомленным о предмете беседы, но главное — специалистом, разбирающимся в динамике групп и поведении потребителей. За участие в работе фокусной группы, как правило, выплачивается небольшое денежное вознаграждение. Обсуждение обычно проводится в комфортной обстановке, предлагается легкая закуска.

При этом ведущий должен поощрять свободные и непринужденные высказывания, чтобы динамика группы, устанавливающиеся в ней взаимоотношения, позволили выявить глубинные чувства и мысли участников. Одновременно ведущий стремится «сфокусировать» обсуждение. Ход дискуссии записывается на аудио- или видеомагнитофон и впоследствии внимательно изучается.

Фирмы, торгующие потребительскими товарами, на протяжении многих лет используют данные, полученные в фокус-группах; в достоинствах этого метода все больше убеждаются газеты, юридические конторы, больницы и предприятия сферы услуг. Тем не менее, исследователь должен остерегаться переносить выводы, сделанные на основе анализа мнения фокус-группы, на весь рынок, поскольку ее члены представляют собой очень маленькую выборку.

Сбор первичной информации может также осуществляться на основе использования панельного метода опроса, в основе которого лежит понятие панели.

*Панель –* выборочная совокупность опрашиваемых единиц, подвергаемых повторяющимся исследованиям. Членами панели могут быть отдельные потребители, семьи, организации торговли и промышленности, эксперты. Панельный метод опроса имеет преимущества по сравнению с обычными одноразовыми опросами: он дает возможность сравнивать результаты последующих опросов с итогами предыдущих и устанавливать тенденции и закономерности развития изучаемых явлений; обеспечивает более высокую репрезентативность выборки по отношению к генеральной совокупности.

Организация и проведение сбора информации представляет собой очень ответственную и трудоемкую процедуру, особенно если в качестве метода был выбран личный опрос. Ответственность обусловлена, прежде всего, опасностью получения в результате опроса недостоверной информации, причинами которой являются труднодоступность некоторых категорий опрашиваемых, отказы опрашиваемых участвовать в опросе; неискренность или пристрастность опрашиваемых; неискренность или пристрастность интервьюера.

Находящиеся в распоряжении маркетологов инструменты сбора первичных данных могут быть разбиты на два основных типа: анкеты и автоматические устройства, например, счетчики количества людей, кино-видеокамеры.

*Анкета –* опросный лист для получения сведений, состоящий обычно из трех частей: введения, реквизитной части и основной части.

Главная задача введения – убедить респондента принять участие в опросе. Оно должно содержать цель проводимого опроса и показать, какую пользу получит респондент, приняв участие в опросе. В реквизитной части приводится информация, касающаяся респондентов: возраст, пол, принадлежность к определенному классу, род занятий, семейное положение, имя и адрес для частных лиц и для организаций: размер, месторасположение, направление производственно-хозяйственной деятельности, положение респондента в организации, его имя. При разработке основной части вопросника следует обратить внимание: на содержание вопросов, их тип и число, последовательность представления вопросов в анкете, наличие контрольных вопросов.

*Открытый вопрос –* вопрос анкеты, с помощью которой собирается первичная маркетинговая информация; он позволяет опрашиваемому на вопрос отвечать своими словами, что дает ему возможность чувствовать себя при ответе на вопрос достаточно свободно, приводить примеры, иллюстрации. Примером открытого вопроса является вопрос: "Как Вы относитесь к....?" В таких вопросах отсутствует предвзятость, стремление навязать определенный ответ. Однако ответы на открытые вопросы требуют достаточно больших затрат времени. Кроме того, полученные ответы могут быть интерпретированы по-разному.

*Закрытый вопрос –* вопрос анкеты, который включает все возможные варианты ответов, из которых опрашиваемый делает свой выбор. Существует два варианта таких вопросов: а) дихотомический, типа: "Считаете ли Вы, что лечебная косметика необходима мужчинам?" В данном случае существует только два варианта ответа: да, нет; б) многовариантного выбора: "Насколько важна для Вас цена при покупке косметики?", где существует несколько вариантов ответов. Основным недостатком вопросов с многовариантным выбором являются трудности формулирования всех возможных вариантов ответов, характеристик или факторов.

Что касается последовательности представления вопросов в анкете, то не рекомендуется начинать вопросник с трудных или персональных вопросов, вопросов, не интересных для респондентов; такие вопросы рекомендуется ставить в середине или в конце анкеты. Желательно, чтобы вопросы излагались в определенной логической последовательности, позволяющей как можно полнее рассматривать отдельные темы. Анкета не должна содержать вопросы, на которые респонденты не захотят отвечать или на которые невозможно ответить, или которые не требуют ответа.

Контрольные вопросы используются, прежде всего, для определения уровня компетентности респондентов в области проводимого исследования.

Во избежание ошибок можно апробировать анкету на единичных респондентах, а после провести пилотное испытание анкеты на малом числе респондентов (пять-десять человек). При этом используются те же процедуры отбора респондентов и их опроса, которые предполагается использовать при проведении полномасштабного анкетирования. Анкета является наиболее распространенным средством сбора первичных данных.

При сборе данных могут иметь место многие погрешности. Эти ошибки включают выбор неверных элементов выборки для взятия интервью, неучет мнения тех, кто отказался давать интервью или не оказался дома, ложные оценки, даваемые интервьюируемыми преднамеренно. Возможна фальсификация полученных данных со стороны интервьюера. Ошибки могут совершать и добросовестные интервьюеры при переписывании собранной информации из анкет. Эти ошибки не могут быть измерены. Поэтому важным является заранее выяснить их возможные причины и предпринять соответствующие меры по их предотвращению.

*Автоматические устройства* применяются в маркетинговых исследованиях не так часто. Гальванометры (приборы для измерения слабых токов) фиксируют малейшие изменения физиологических параметров, сопровождающие возникновение интереса или эмоционального возбуждения у испытуемого, например под воздействием той или иной рекламы или конкретного изображения. Тахистоскоп позволяет варьировать время демонстрации рекламного объявления от сотой доли секунды до нескольких секунд. После каждой такой демонстрации испытуемый описывает все, что ему удалось увидеть и понять. Специальные устройства фиксируют движения глаз при восприятии участником эксперимента того или иного объекта, что примерно характеризует очередность и длительность концентрации внимания потребителя. Электронное устройство под названием аудиометр, подключаемое к телевизору в домах участников эксперимента, фиксирует сведения обо всех его включениях и каналах, на которые он при этом настроен.

*Сбор статистических данных.* Если наблюдение и опрос фокус-групп прекрасно отвечают целям ознакомительного исследования, при описательном исследовании наиболее уместен сбор статистических данных. Для того чтобы составить портрет покупателя, компания проводит статистический обзор знаний, убеждений, привычек и желаний потребителей. Для концерна «Калина» имеет значение информация о том, сколько примерно человек слышали о компании, пользовались ее косметикой, отдают ей предпочтение и т. п.

Наиболее строгим с научной точки зрения является *экспериментальное исследование,* имеющее целью установление причинно-следственных связей. Объекты эксперимента должны быть специально отобраны и подвергнуты запланированным воздействиям в условиях контроля за внешним окружением, чтобы выявить статистически значимые различия в их реакции. В той же мере, в какой исследователям удается «отсечь» или взять под контроль не относящиеся к делу внешние факторы, наблюдаемые эффекты могут быть соотнесены с воздействиями экспериментаторов на объект. Устанавливаемые таким образом связи между событиями послеих критического анализа могут считаться причинно-следственными, а цели эксперимента — достигнутыми. Эксперимент может быть проведен в полевых (люди не знают, что за ними ведется наблюдение) или лабораторных (моделируется ситуация, максимально приближенная к реальности, однако люди сознательно идут на эксперимент) условиях. Эксперимент в полевых условиях провести труднее из-за того, что на покупательское поведение оказывают влияние многие факторы и варьировать лишь один из них, оставляя другие постоянными, не представляется возможным. Однако результаты наиболее точно отражают реальную ситуацию. Для эксперимента в лабораторных условиях эти достоинства и недостатки меняются местами.

Примером эксперимента может служить следующее: одну неделю концерн «Калина» мог бы предложить каждой покупательнице бесплатный журнал о новинках косметики, а на другой неделе журнал не предлагать. При этом остальные параметры (цвет упаковки, расположение товара на полках и т. д.) должны оставаться постоянными. И если количество покупательниц и стоимость покупок за эти дни изменились, то это изменение относят на счет предложения журнала. Точность эксперимента возрастет, если попробовать повторить его. В той степени, в какой при разработке и проведении эксперимента удалось исключить альтернативные гипотезы, организаторы маркетингового исследования могут быть уверены в достоверности своих выводов.

Существуют компьютеризированные программы-вопросники, которые позволяют потребителям "бродить" по супермаркету и выбирать товары для покупки. Компьютер регистрирует их покупки и измеряет их реакцию на применение определенных элементов комплекса маркетинга (цену, цвет и форму упаковки, внутримагазинные методы продвижения товара).

*Имитация*– это новый, современный метод сбора данных, генерируемых ЭВМ с помощью заранее разработанной математической модели, адекватно воспроизводящей поведение объекта исследования. Достоинство этого метода состоит в возможности оперативного анализа множества вариантов маркетинговых действий и выбора на этой основе наилучшего. К недостаткам относится прежде всего сложность и трудоемкость создания самой модели.

На практике создание такой модели нередко не представляется возможным. Даже если возможности фирмы позволяют создать такую модель, то ее разработка потребует значительных затрат.

После принятия решения о методах сбора данных и используемых инструментах наступает черед определения способа формирования выборки, на основе которой будет проводиться исследование, т. е. тех конкретных людей, с которыми будут взаимодействовать исследователи. Составленный план должен ответить на три вопроса:

*1. Состав выборки: кто станет участником исследования?* Для проведения маркетингового исследования должны быть определены целевые группы населения, которые примут в нем участие. Например, в случае с покупателями косметики необходимо решить, составлять ли выборку только из женщин или из женщин и мужчин одновременно? Включать ли в выборку подростков? Когда ответ на подобные вопросы найден, необходимо обеспечить равенство всех членов намеченных целевых групп, т. е. не допускать преобладания в выборке каких-либо специфических подгрупп.

*2. Размер выборки: сколько людей будет охвачено исследованием?* Чем больше размер выборки, тем достовернее полученные результаты. Тем не менее, нет необходимости включать в выборку всю целевую группу потребителей или ее значительную часть. Обычно при строгом соблюдении правил их формирования достоверные выводы позволяют сделать выборки, охватывающие менее 1 *%* населения.

3. *Процедура выборки: как будут отобраны участники исследования?* Для получения представительной выборки должен использоваться метод случайного отбора, который позволяет рассчитывать погрешность полученных результатов. Вывод, сделанный по результатам анализа такой выборки, может звучать, например, так: «60% женщин г. Омска приобретают косметику на сумму в интервале от 70 до 200 рублей ежемесячно».

После разработки плана составления выборки исследователь должен остановиться на конкретном способе общения с аудиторией, которое может осуществляться по телефону, почтой (в том числе электронной) и личным обращением.

*Анкета, рассылаемая по почте, —* самый подходящий способ связаться с людьми, которые либо не согласны на личную беседу, либо в силу черт личности склонны под влиянием интервьюера искажать ответы. Почтовая анкета требует простых и четко сформулированных вопросов, время получения ответов на которые может быть значительным, а процент ответивших обычно невелик. Обязательно вкладывать конверт с маркой и обратным адресом, чтобы респондент не оплачивал почтовые услуги. Для российской действительности этот метод, по нашему мнению, не является удачным, т. к. такие анкеты россияне вообще не отсылают обратно компании. Эту ситуацию можно изменить, предложив ответившему небольшое вознаграждение.

*Телефонное интервью —* наилучший путь быстрого сбора информации, к тому же интервьюер имеет возможность пояснять вопросы, непонятные для респондента. Количество ответивших, как правило, выше, чем в случае опроса по почте. Основная трудность, с которой сталкиваются исследователи, заключается в необходимости не затягивать разговор и не затрагивать личные темы. Однако такой метод исключает тех, у кого нет телефона.

*Личное собеседование —* самый универсальный метод. Интервьюер не только задает респонденту больше вопросов, но и дополняет результаты беседы личными наблюдениями по поводу, например, внешнего вида или манер собеседника. В то же время этот метод требует тщательного планирования, контроля и больших затрат. Кроме того, на его результаты влияют взгляды и предубеждения интервьюера.

Обычно различают две формы личного собеседования: назначенное и уличное. В первом случае проводится случайный выбор респондентов, с которыми договариваются об интервью либо по телефону, либо при посещении их дома или на работе. Нередко в качестве компенсации за потраченное время предлагается небольшое денежное вознаграждение или подарок. Во втором случае интервью берут у прохожих в каком-либо оживленном местеили у посетителей крупного магазина. Здесь опрашиваемый имеет меньше времени для ответов, кроме того, такую выборку нельзя считать случайной.

В наши дни интервьюером может быть не только человек. Некоторые компании используют для сбора маркетинговой информации бесплатные для клиентов телефонные номера. Например, в 1993 г. компания *Pepsi-Cola* разослала почтовые карточки, адресованные миллиону американских семей, покупателей *«Diet Coкe*», предложив им возможность позвонить по телефонному номеру 800 и «поговорить» в интерактивном режиме с великим джазовым музыкантом Рэем Чарльзом, а если повезет — получить приз. Но прежде им надо было ответить на несколько вопросов, разработанных для маркетингового опроса, проводимого в целях изучения семей, покупающих *«Diet Coke».*Чтобы услышать приветствие Рэя Чарльза и *Uh-Huh Girls,* в компанию позвонили более 500 тыс. человек. Между вопросами звонящий слышал отрывки джазовых мелодий, чтобы не возникало той «телезанудности», которая способна погубить любой разговор в интерактивном режиме. В конце 3,5-минутного опроса респондент узнавал, достался ли ему выигрыш в мгновенную лотерею (годичная поставка *Diet Pepsi* или миниатюрный автомат по продаже этого напитка). Но самый большой приз получила компания *Pepsi-Cola,* которой удалось получить информацию о покупательских предпочтениях, уровне потребления различных напитков и некоторых увлечениях более чем 500 тыс. потребителей.

Это облегчает ознакомление с запросами потребителей, их восприятием, мотивами и их отношением к изучаемому продукту, его марке, методам продвижения.

Сбор данных, как правило, требует самых больших расходов и служит источником максимального числа ошибок. В случае сбора статистических данных возникают четыре основные проблемы. Некоторых респондентов не оказывается в условленном месте, с ними придется связываться повторно или заменять. Другие отказываются от сотрудничества. Третьи дают предвзятые или заведомо ложные ответы. Наконец, люди, проводящие опрос, тоже могут вести себя предвзято или нечестно.

Как уже было сказано, благодаря современным вычислительным и телекоммуникационным технологиям методы сбора данных быстро развиваются. Профессиональные интервьюеры сидят в кабинках, набирают случайные телефонные номера и просят поднявшего трубку ответить на несколько вопросов, зачитывая их с экрана монитора компьютера и набирая ответы респондентов на клавиатуре. Такой метод исключает необходимость в оформлении и кодировке данных, уменьшает число ошибок, сберегает время и позволяет получать всю требуемую статистику. Другие исследовательские фирмы устанавливают интерактивные терминалы в торговых центрах. Любой желающий ответить на вопросы может сесть за терминал, считать вопросы с экрана монитора и набрать свои ответы на клавиатуре.

## ***Анализ предметной области***

Деятельность, направленная на **выявление реальных потребностей заказчика, а также на выяснения смысла высказанных требований, называется анализом предметной области** (бизнес-моделированием, если речь идет о потребностях коммерческой организации). Анализ предметной области – **это первый шаг этапа системного анализа**, с которого начинается разработка программной системы. Разработчики должны научиться

1. понимать язык, на котором говорят заказчики;
2. выявить цели их деятельности;
3. определить набор решаемых ими задач;
4. определить набор сущностей, с которыми приходится иметь дело при решении этих задач.

## ***Модели предметной области***

Анализом предметной области **занимаются системные аналитики или бизнес-аналитики**. Они передают полученные ими знания другим членам проектной команды, сформулировав их на более понятном разработчикам языке. Для передачи этих знаний обычно служит некоторый **набор моделей, в виде графических схем и текстовых документов**.

Определения=  
**Система** – совокупность взаимодействующих компонентов и взаимосвязей между ними.  
**Модель M некоторой системы S** – информационный объект, который может быть использован для получения ответов на некоторый круг вопросов относительно S.

**Цель моделирования**: получение ответов на эту совокупность вопросов.  
Цель моделирования формулируется на самом раннем этапе разработки модели.  
**Объектом моделирования** является сама система. При этом необходимо точно определить границы системы, чтобы избежать включения в модель посторонних объектов.  
**Результатом моделирования** является набор взаимоувязанных описаний, начиная с описания самого верхнего уровня системы и кончая подробным описанием деталей или операций.

Основные понятия системного анализа

**Предметная область** - раздел науки, изучающий предметные аспекты системных процессов и системные аспекты предметных процессов и явлений. Это определение можно считать системным определением *предметной области*.

**Системный анализ** - совокупность понятий, методов, процедур и технологий для изучения, описания, реализации явлений и процессов различной природы и характера, междисциплинарных проблем; это совокупность общих законов, методов, приемов исследования таких систем.

*Системный анализ* - методология исследования сложных, часто не вполне определенных проблем теории и практики.

Строго говоря, различают три ветви науки, изучающей системы:

1. системологию (теорию систем) которая изучает теоретические аспекты и использует теоретические методы (теория *информации*, теория вероятностей, теория игр и др.);
2. *системный анализ* (методологию, теорию и практику исследования систем), которая исследует методологические, а часто и практические аспекты и использует практические методы (математическая статистика, исследование операций, программирование и др.);
3. системотехнику, системотехнологику (практику и технологию проектирования и исследования систем).

За термин системотехнологика ответственность несет автор. Такое деление достаточно условно.

Общим у всех этих ветвей является системный подход, системный принцип исследования - рассмотрение изучаемой совокупности не как простой суммы составляющих (линейно взаимодействующих объектов), а как совокупности нелинейных и многоуровневых взаимодействующих объектов.

Любую *предметную область* также можно определить как системную.

Пример. **Информатика** - наука, изучающая информационно-логические и алгоритмические аспекты системных процессов, системные аспекты информационных процессов. Это определение можно считать системным определением *информатики*.

*Системный анализ* тесно связан с *синергетикой*. **Синергетика** - междисциплинарная наука, исследующая общие идеи, методы и закономерности *организации* (изменения структуры, ее пространственно-временного усложнения) различных объектов и процессов, инварианты (неизменные сущности) этих процессов. "Синергический" в переводе означает "совместный, согласованно действующий". Это теория возникновения новых качественных свойств, структур на макроскопическом уровне.

*Системный анализ* тесно связан и с философией. Философия дает общие методы содержательного анализа, а *системный анализ* - общие методы формального, межпредметного анализа *предметных областей*, выявления и описания, изучения их системных инвариантов. Можно дать и философское определение системного анализа: *системный анализ* - это прикладная диалектика.

*Системный анализ* предоставляет к использованию в различных науках, системах следующие системные методы и процедуры:

1. абстрагирование и конкретизация;
2. анализ и синтез, индукция и дедукция;
3. формализация и конкретизация;
4. композиция и декомпозиция;
5. линеаризация и выделение нелинейных составляющих;
6. структурирование и реструктурирование;
7. макетирование;
8. реинжиниринг;
9. алгоритмизация;
10. моделирование и эксперимент;
11. программное управление и регулирование;
12. распознавание и идентификация;
13. кластеризация и классификация;
14. экспертное оценивание и тестирование;
15. верификация
16. и другие методы и процедуры.

Имеются следующие основные типы ресурсов в природе и в обществе.

1. **Вещество** - наиболее хорошо изученный ресурс, который в основном представлен таблицей Д.И. Менделеева достаточно полно и пополняется не так часто. *Вещество* выступает как отражение постоянства материи в природе, как мера однородности материи.
2. *Энергия* - не полностью изученный тип ресурсов, например, мы не владеем управляемой термоядерной реакцией. **Энергия**выступает как отражение изменчивости материи, переходов из одного вида в другой, как мера необратимости материи.
3. **Информация** - мало изученный тип ресурсов. *Информация* выступает как отражение порядка, структурированности материи, как мера порядка, самоорганизации материи (и социума). Сейчас этим понятием мы обозначаем некоторые сообщения; ниже этому понятию мы посвятим более детальное обсуждение.
4. **Человек** - выступает как носитель интеллекта высшего уровня и является в экономическом, социальном, гуманитарном смысле важнейшим и уникальным ресурсом общества, рассматривается как мера разума, интеллекта и целенаправленного действия, мера социального начала, высшей формы отражения материи (сознания).
5. **Организация** (или организованность) выступает как форма ресурсов в социуме, группе, которая определяет его структуру, включая институты человеческого общества, его надстройки, применяется как мера упорядоченности ресурсов. *Организация* системы связана с наличием некоторых причинно-следственных связей в этой системе. *Организация* системы может иметь различные формы, например, биологическую, информационную, экологическую, экономическую, социальную, временную, пространственную, и она определяется причинно-следственными связями в материи и социуме.
6. **Пространство** - мера протяженности материи (события), распределения ее (его) в окружающей среде.
7. **Время** - мера обратимости (необратимости) материи, событий. *Время* неразрывно связано с изменениями действительности.

Можно говорить о различных полях, в которые "помещен" *человек*, - материальном, энергетическом, информационном, социальном, об их пространственных, ресурсных (материя, *энергия*, *информация*) и временных характеристиках.

Пример. Рассмотрим простую задачу - пойти утром на занятия в вуз. Эта часто решаемая студентом задача имеет все аспекты:

1. материальный, физический аспект - студенту необходимо переместить некоторую массу, например, учебников и тетрадей на нужное расстояние;
2. энергетический аспект - студенту необходимо иметь и затратить конкретное количество *энергии* на перемещение;
3. информационный аспект - необходима *информация* о маршруте движения и месторасположении вуза и ее нужно обрабатывать по пути своего движения;
4. человеческий аспект - перемещение, в частности, передвижение на автобусе невозможно без *человека*, например, без водителя автобуса;
5. организационный аспект - необходимы подходящие транспортные сети и маршруты, остановки и т.д.;
6. пространственный аспект - перемещение на определенное расстояние;
7. временной аспект - на данное перемещение будет затрачено *время* (за которое произойдут соответствующие необратимые изменения в среде, в отношениях, в связях).

Все типы ресурсов тесно связаны и сплетены. Более того, они невозможны друг без друга, актуализация одного из них ведет к актуализации другого.

Пример. При сжигании дров в печке выделяется тепловая *энергия*, тепловая *энергия* используется для приготовления пищи, пища используется для получения биологической *энергии* организма, биологическая *энергия* используется для получения *информации*(например, решения некоторой задачи), перемещения во *времени* и в *пространстве*. *Человек* и во *время* сна расходует свою биологическую энергию на поддержание информационных процессов в организме; более того, сон - продукт таких процессов.

Социальная *организация* и активность людей совершенствует информационные ресурсы, процессы в обществе, последние, в свою очередь, совершенствуют производственные отношения.

Если классическое естествознание объясняет мир исходя из движения, взаимопревращений *вещества* и *энергии*, то сейчас реальный мир, объективная реальность могут быть объяснены лишь с учетом сопутствующих системных, и особенно системно-информационных и синергетических процессов.

Особый тип мышления - *системный*, присущий аналитику, который хочет не только понять суть процесса, явления, но и управлять им. Иногда его отождествляют с аналитическим мышлением, но это отождествление не полное. Аналитическим может быть склад ума, а системный подход есть методология, основанная на теории систем.

**Предметное** (предметно-ориентированное) **мышление** - это метод (принцип), с помощью которого можно целенаправленно (как правило, с целью изучения) выявить и актуализировать, познать причинно-следственные связи и закономерности в ряду частных и общих событий и явлений. Часто это методика и технология исследования систем.

**Системное** (системно-ориентированное) **мышление** - это метод (принцип), с помощью которого можно целенаправленно (как правило, с целью управления) выявить и актуализировать, познать причинно-следственные связи и закономерности в ряду общих и всеобщих событий и явлений. Часто это методология исследования систем.

При *системном мышлении* совокупность событий, явлений (которые могут состоять из различных составляющих элементов) актуализируется, исследуется как целое, как одно организованное по общим правилам событие, явление, поведение которого можно предсказать, прогнозировать (как правило) без выяснения не только поведения составляющих элементов, но и качества и количества их самих. Пока не будет понятно, как функционирует или развивается система как целое, никакие знания о ее частях не дадут полной картины этого развития.

Пример. В соответствии с принципом *системного мышления* общество состоит из людей (и, разумеется, из общественных институтов). Каждый *человек* - также система (физиологическая, например). У *человека*, в свою очередь, существуют присущие ему как организму системы, например, система кровообращения. Когда люди взаимодействуют с другими людьми, образуются новые системы - семья, этнос и др. Это взаимодействие может происходить на уровне общественных институтов, отдельных людей (например, социальные взаимодействия) и даже отдельных систем кровообращения (например, при прямом переливании крови).

В соответствии с принципом системного подхода, каждая система влияет на другую систему. Весь окружающий мир - взаимодействующие системы. Цель *системного анализа* - выяснить эти взаимодействия, их потенциал и "направить их на службу *человека*".

**Предметный аналитик** (предметно-ориентированный или просто аналитик) - *человек*, профессионал, изучающий, описывающий некоторую *предметную область*, проблему в соответствии с принципами и методами, технологиями этой области. Это не означает "узкое" рассмотрение этой проблемы, хотя подобное часто встречается.

**Системный** (системно-ориентированный) **аналитик** - *человек*, профессионал высокого уровня (эксперт), изучающий, описывающий системы в соответствии с принципами системного подхода, анализа, т.е. изучающий проблему комплексно. Ему присущ особый склад ума, базирующийся на мультизнаниях, достаточно большом кругозоре и опыте, высоком уровне интуиции предвидения, умении принимать целесообразные ресурсообеспеченные решения. Его основная задача - помочь *предметному аналитику* принять правильное (сообразующееся с другими системами, не "ухудшающее" их) решение при решении предметных проблем, выявление и изучение критериев эффективности их решения.

***Понятие структурного анализа***

На этапе анализа требований к системе формализуются, документируются и уточняются требования заказчика. Список требований включает:

- совокупность условий при которых будет эксплуатироваться система;

- написание выполняемых системой функций;

- ограничение на процессы разработки - сроки завершения работ и мероприятия по защите информации.

Особенностью разработки программного обеспечения является то, что наиболее сложные работы выполняются на этапах анализа и проектирования. Последующие этапы имеют значительно меньшую сложность и трудоемкость. Язык, на котором формулируются требования к системе должен быть достаточно простым и понятным.

Системный аналитик должен уметь решать следующие задачи:

- получение исчерпывающей информации для оценки требований к системе;

- уметь выбирать только существенную информацию на предметной области;

- спецификация системы, которую составляет аналитик из-за технических терминов и значительного объема часто непонятны заказчику.

Решение этой проблемы состоит в использовании методов структурного анализа. Для метода структурного анализа характерно разбиение описания системы на уровне абстрактного представления. Метод структурного анализа состоит в том, что исследования системы начинается с общего обзора, а затем выполняется более детальное исследование результаты которого приобретают иерархическую структуру.

Основные принципы структурного анализа:

- решение трудных задач выполняется путем разбиения на множество меньших относительно независимых задач;

- принцип иерархического упорядочивания;

- принцип абстрагирования заключается в выделении наиболее существенных аспектов системы для представления проблемы в простом общем виде;

- принцип формализаций, состоит в необходимости применения строгого методического подхода для решения всех задач;

- принцип упрятывания, заключается в том, что несущественная на конкретном этапе информация скрывается;

- принцип концептуальной общности означает, что на всех этапах жизненного цикла должна использоваться единая методология;

- принцип полноты, заключается в выполнении контроля присутствия в функциях системы лишних элементов;

- принцип непротиворечивости, состоит в проверке обоснованности использования и согласованности всех элементов системы;

- принцип логической независимости, состоит в том, что проектирование выполняющееся на логическом уровне не должно определяться последующим физическим проектированием;

- принцип независимости данных, состоит в том, что модель данных должна быть спроектирована независимо от процесса и их логической обработки;

- принцип структурирования данных;

- принцип доступа конечного пользователя, означает что пользователь должен иметь возможность без программирования изменять значения данных в базе данных.

Средства структурного анализа.

Существует три группы средств структурного анализа которые иллюстрируют:

- функции, которые система должна выполнять - диаграммы потока данных;

- отношение между данными - диаграммы сущности связи;

- поведение системы зависящее от времени - диаграммы переходов состояний.

Необходимые атрибуты *системного анализа* как научного знания:

1. наличие предметной сферы - системы и системные процедуры;
2. выявление, систематизация, описание общих свойств и атрибутов систем;
3. выявление и описание закономерностей и инвариантов в этих системах;
4. актуализация закономерностей для изучения систем, их поведения и связей с окружающей средой;
5. накопление, хранение, актуализация знаний о системах (коммуникативная функция).

*Системный анализ* базируется на ряде общих принципов, среди которых:

1. **принцип дедуктивной последовательности** - последовательного рассмотрения системы по этапам: от окружения и связей с целым до связей частей целого (см. этапы *системного анализа* подробнее ниже);
2. **принцип интегрированного рассмотрения** - каждая система должна быть неразъемна как целое даже при рассмотрении лишь отдельных подсистем системы;
3. *принцип согласования ресурсов и целей* рассмотрения, актуализации системы;
4. **принцип бесконфликтности** - отсутствия конфликтов между частями целого, приводящих к конфликту целей целого и части.

Системно в мире все: практика и практические действия, знание и процесс познания, окружающая среда и связи с ней (в ней). *Системный анализ* как методология научного познания структурирует все это, позволяя исследовать и выявлять инварианты (особенно скрытые) объектов, явлений и процессов различной природы, рассматривая их общее и различное, сложное и простое, целое и части.

Любая человеческая интеллектуальная деятельность обязана быть по своей сути системной деятельностью, предусматривающей использование совокупности взаимосвязанных системных процедур на пути от постановки задачи, целей, планирования ресурсов к нахождению и использованию решений.

Пример. Любое экономическое решение должно базироваться на фундаментальных принципах *системного анализа*, экономики,*информатики*, управления и учитывать поведение *человека* в социально-экономической среде, т.е. должно базироваться на рациональных, социально и экономически обоснованных нормах поведения в этой среде.

Неиспользование *системного анализа* не позволяет знаниям (закладываемым традиционным образованием) превращаться в умения и навыки их применения, в навыки ведения системной деятельности (построения и реализации целенаправленных, структурированных, обеспеченных ресурсами конструктивных процедур решения проблем). Системно мыслящий и действующий *человек*, как правило, прогнозирует и считается с результатами своей деятельности, соизмеряет свои желания (цели) и свои возможности (ресурсы) учитывает интересы окружающей среды, развивает интеллект, вырабатывает верное мировоззрение и правильное поведение в человеческих коллективах.

Окружающий нас мир бесконечен в *пространстве* и во *времени*; *человек* существует конечное *время*, располагая при реализации цели конечными ресурсами (материальными, энергетическими, информационными, людскими, организационными, пространственными и временными).

Противоречия между неограниченностью желания *человека* познать мир и ограниченной (ресурсами, неопределенностью) возможностью сделать это, между бесконечностью природы и конечностью ресурсов человечества, имеют много важных последствий, в том числе - и для самого процесса познания *человеком* окружающего мира. Одна из таких особенностей познания, которая позволяет постепенно, поэтапно разрешать эти противоречия: использование аналитического и синтетического образа мышления, т.е. разделения целого на части и представления сложного в виде совокупности более простых компонент, и наоборот, соединения простых и построения, таким образом, сложного. Это также относится и к индивидуальному мышлению, и к общественному сознанию, и ко всему знанию людей, и к самому процессу познания.

Пример. Аналитичность человеческого знания проявляется и в существовании различных наук, и в дифференциации наук, и в более глубоком изучении все более узких вопросов, каждый из которых сам по себе и интересен, и важен, и необходим. Вместе с тем, столь же необходим и обратный процесс синтеза знаний. Так возникают "пограничные" науки - бионика, биохимия, *синергетика* и другие. Однако это лишь одна из форм синтеза. Другая, более высокая форма синтетических знаний реализуется в науках о самых общих свойствах природы. Философия выявляет и описывает общие свойства всех форм материи; математика изучает некоторые, но также всеобщие отношения. К числу синтетических наук относятся *системный анализ*, *информатика*, кибернетика и др., соединяющие формальные, технические, гуманитарные и прочие знания.

Итак, расчлененность мышления на анализ, синтез и взаимосвязь этих частей является очевидным признаком системности познания.

Процесс познания структурирует системы, окружающий нас мир. Все, что не познано в данный момент *времени*, образует "хаос в системе", который, будучи необъясним в рамках рассматриваемой теории, заставляет искать новые структуры, новую *информацию*, новые формы представления и описания знаний, приводит к появлению новых ветвей знания; этот хаос также дает стимул и для развития умений и навыков исследователя.

Системный подход к исследованию проблем, *системный анализ* - следствие научно-технической революции, а также необходимости решения ее проблем с помощью одинаковых подходов, методов, технологий. Такие проблемы возникают и в экономике, и в *информатике*, и в биологии, и в политике и т.д.

***Задание***

На основании индивидуального задания применить системный и структурный анализы

*Контрольные вопросы*

1. Что представляет собой маркетинговая информационная система?
2. В чем состоят различия между внутренней и внешней маркетинговой информацией?
3. Назвать основные источники получения вторичной информации.
4. Охарактеризовать основные методы сбора первичной информации.
5. Каковы основные правила составления маркетинговых опросников и анкет?
6. Охарактеризовать основные типы вопросов используемых при составлении анкет?
7. Каковы основные преимущества и недостатки информации полученной в ходе эксперимента.
8. Охарактеризовать основные методы, используемые в процессе обработки и анализа данных.

**Тема №4**

**Основные модели построения информационных систем, их структура, особенности и область применения. Классификация информационных систем**

**Цель:** научиться строить модели информационных систем

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Проектирование концептуальной модели основано на анализе решаемых в данной организации задач по обработке данных. Концептуальная модель включает в себя описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных. Здесь имеются в виду данные, используемые как в уже разработанных прикладных программах, так и в тех, которые только будут реализованы.

Проектирование информационной системы начинается отнюдь не с написания программы, а с анализа перевозочного процесса и технологии работы участвующих в этом процессе организаций. При анализе бизнес-процесса организации полезно ответить на шесть вопросов: что, как, где, кто, когда и почему делает.

При ответе на вопрос «что лежит в основе бизнеса данной организации?», как правило, выявляются наиболее важные для данного бизнеса или производственного процесса компоненты.

Ответы на второй вопрос «как это делается?» позволяют получить список основных бизнес-процессов, происходящих в организации.

Вопрос «где происходят данные процессы?» больше относится к проблемам телекоммуникаций и организации совместной работы персонала. Ведь в случае, например, большого объема операций, которые выполняются вне территории организации торговыми агентами, придется учитывать проблемы синхронизации данных. При наличии филиалов весьма непростой проблемой является оптимальный выбор системы распределенных данных. Можно централизовать всю обработку данных, и филиалы будут выполнять свои операции, пользуясь возможностями телекоммуникаций. Работа с данными в этом случае упрощается, но могут возникнуть перебои из-за потери связи.

Ответ на вопрос «кто выполняет эти процессы?» даст организационная структура организации.

Важно получить и ответ на вопрос «когда выполняется то или иное действие?». Это внесет ясность в частоту осуществляемых бизнес-процессов и позволит правильно расставить акцепты в будущей прикладной программе.

Последний вопрос «почему эти действия выполняются?» позволяет определить основные мотивации производственной деятельности организации. Типичные бизнес-задачи могут быть представлены следующим списком: достижение наилучшего соотношения: «затраты — удобство для клиентов»; обеспечение условий для успешной деятельности персонала; получение приемлемой прибыли; повышение доходов при автоматизации обработки данных.

Ответы на шесть перечисленных вопросов позволяют подойти к главному моменту в постановке задачи - получению информационной модели организации. В простейшем виде такая модель может быть отражена в виде взаимосвязей между бизнес-компонентами и бизнес-процессами. В практике проектирования информационных систем такие схемы получили название ER-диаграмм *{Entity-relationship diagram —* ERD) — диаграммы «сущность — связь». ER-диаграммы хорошо вписываются в методологию структурного анализа и проектирования информационных систем. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, давая возможность получить различную степень детализации объекта с разным количеством уровней.

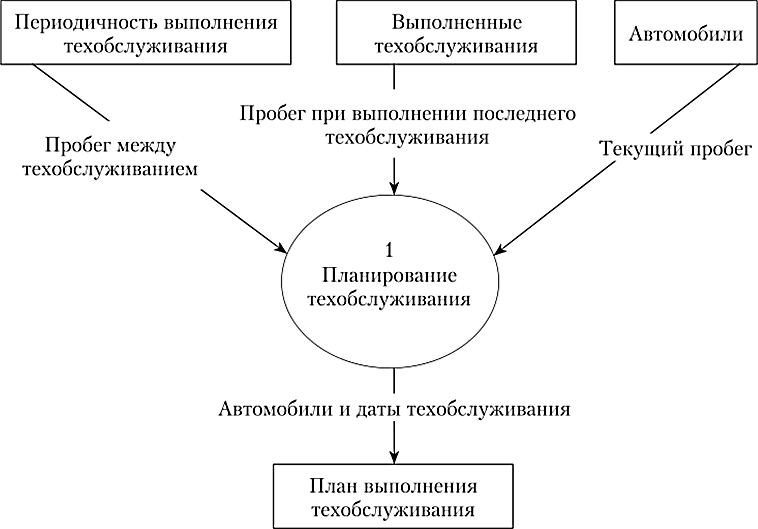
Для проектирования сложных информационных моделей с помощью ER-диаграмм используются CASE-системы (*Computer-aided Software Engineering).* Эти системы позволяют проектировать информационные системы на основании построения в виде диаграмм многоуровневых потоков данных, создавать на их основе логическую и физическую модель БД.

В соответствии с CASE-методологией модель системы определяется как иерархия диаграмм потоков данных (ДПД, или DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы информационной системы (ИС) с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям — потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

* • внешняя сущность — материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчики, персонал, поставщики, клиенты, склад;
* • система/подсистема; при построении модели сложной информационной системы она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого либо может быть декомпозирована на ряд подсистем;
* • процесс, представляющий собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом; физически процесс может быть реализован различными способами: это могут быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно-реализованное логическое устройство и т.д.;
* • накопитель данных, являющийся абстрактным устройством для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми;
* • поток данных, определяющий информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой, и т.д.

Пример построения модели потока данных при планировании технического обслуживания (ТО) автомобилей приведен на рис. Этот бизнес- процесс отражается в виде окружности и отнесен к первому уровню. Внешние по отношению к этому процессу сущности с необходимыми данными изображены в виде прямоугольников.



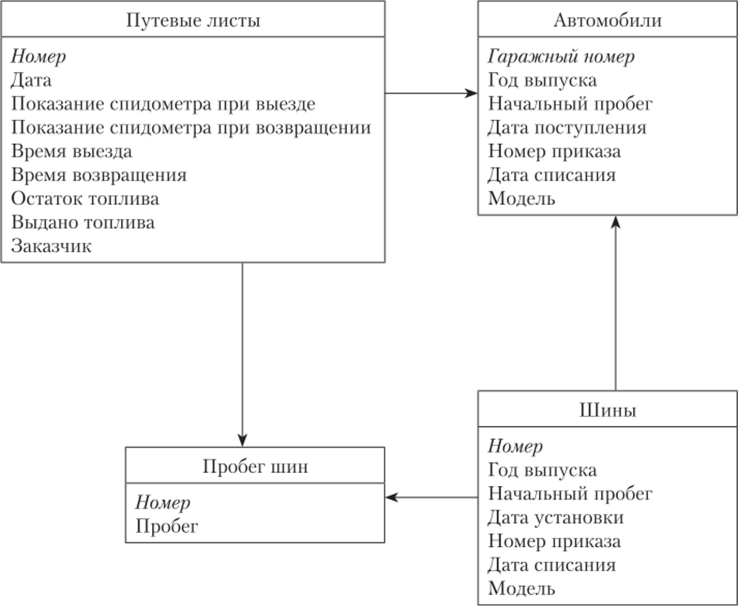
*Рис.* *Модель потока данных при планировании технического обслуживания*

Концептуальная модель транслируется затем в модель данных, совместимую с выбранной СУБД. Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели.

После завершения проектирования концептуальной модели может быть начата **разработка логической модели.**Логическая модель данных — это версия концептуальной модели, которая может быть обеспечена планируемой для реализации информационной системы СУБД. Логическая модель отражает логические связи между элементами данных безотносительно к их содержанию и среде хранения. Эта модель может быть реляционной, иерархической или сетевой.

При разработке логической модели БД прежде всего необходимо решить, какая модель данных наиболее подходит для отражения конкретной концептуальной модели предметной области. Коммерческие системы управления базами данных поддерживают одну из известных моделей данных или некоторую их комбинацию. Почти что все популярные системы для персональных компьютеров используют реляционную модель данных.

Построение на основе концептуальной модели данных реляционной модели производится относительно просто. Каждый объект концептуальной модели отражается в одном отношении, которое отражает представление пользователя в удобном для него табличном формате. Простота отражения связана с тем, что при разработке концептуальной модели, как правило, используется реляционный подход. Пример логической модели автоматизированного рабочего места (АРМ) диспетчера автотранспортной организации (АТО) приведен на рис.



*Рис.* *Логическая модель автоматизированного рабочего места диспетчера автотранспортной организации*

На этане физического проектирования информационной системы необходимо обеспечить безошибочность и точность информации, хранящейся в БД. Для этого используются имеющиеся в СУБД способы реализации ограничений и правил обработки данных, которые в совокупности обеспечивают целостность БД. Таким образом, обеспечением целостности БД называется система мер, направленных на поддержание правильности данных в БД в любой момент времени.

В системах обработки данных затраты на проверку и поддержание достоверности данных могут составлять значительную часть общих эксплуатационных расходов. Например, в транспортных организациях для контроля правильности ввода данных с путевой документации практикуется параллельный ввод одних и тех же данных несколькими операторами. Считается, что вероятность совершения одной и той же ошибки в этом случае будет крайне мала и простое сравнение результатов ввода различных операторов поможет получить безошибочные данные.

Важно

Ошибки в данных, к сожалению, остаются незамеченными до тех пор, пока не приведут к тяжелым последствиям, если только не позаботиться об их обнаружении заранее. Это достаточно убедительный довод, чтобы предусмотреть предоставляемые СУБД определенные меры для блокирования появления возможных ошибок в разрабатываемой БД. В системе управления базами данных целостность данных обеспечивается набором специальных предложений, называемых ограничениями целостности.

Ограничения целостности — это набор определенных правил, которые определяют допустимость данных и связей между ними. Ограничения целостности в большинстве случаев зависят от особенностей предметной области. Например, мощность двигателя серийного легкового автомобиля вряд ли может быть ниже 30 л.с. Ограничения целостности могут относиться к разным объектам БД: атрибутам (полям), записям, отношениям, связям между ними и т.п. Для полей, как правило, могут использоваться следующие виды ограничений:

* • тип и формат поля автоматически допускают ввод только данных определенного типа. — выбор типа ноля *Date* в формате ДД.ММ.ГГ позволит пользователю ввести только шесть чисел, при этом первая пара цифр не сможет превысить в лучшем случае значения 31, а вторая — 12;
* • задание диапазона значений, как правило, используется для числовых полей — диапазон допустимых значений может быть ограничен или с двух сторон (закрытый диапазон), или с какой-то одной: верхней или нижней (открытый диапазон);
* • недопустимость пустого поля позволяет избежать появления в БД «ничейных» записей, в которых пропущены какие-либо обязательные атрибуты;
* • задание списка значений дает возможность избежать излишнего разнообразия данных, если его можно ограничить. Например, для указания типа кузова мы можем ограничить фантазию пользователя только общепринятыми названиями: седан, кабриолет и т.д.;
* • проверка на уникальность значения какого-то ноля позволяет избежать записей-дубликатов; вряд ли будет удобно в справочнике клиентов иметь несколько записей для одного и того же лица.

Для реализации ограничений целостности, имеющих отношение к записи, таблицам или связям между ними, в СУБД существует специальный механизм, называемый *триггером.* Триггер представляет собой функцию или программу, в которой описываются правила выполнения тех или иных операций.

***Классификация информационных систем***

*Классификация по масштабу*

По масштабу информационные системы подразделяются на следующие группы:

1. одиночные;
2. групповые;
3. корпоративные.

*Классификация по сфере применения*

По сфере применения информационные системы обычно подразделяются на четыре группы:

1. системы обработки транзакций;
2. системы принятия решений;
3. информационно-справочные системы;
4. офисные информационные системы.

*Классификация по способу организации*

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы:

1. системы на основе архитектуры файл-сервер;
2. системы на основе архитектуры клиент-сервер;
3. системы на основе многоуровневой архитектуры;
4. системы на основе Интернет/интранет - технологий.

В любой информационной системе можно выделить необходимые функциональ­ные компоненты, которые помогают понять ограничения различных архитектур информационных систем.

По характеру хранимой информации БД делятся на *фактографические* и *документальные*. Если проводить аналогию с описанными выше примерами информационных хранилищ, то фактографические БД — это картотеки, а документальные — это архивы. В фактографических БД хранится краткая информация в строго определенном формате. В документальных БД — всевозможные документы. Причем это могут быть не только текстовые документы, но и графика, видео и звук (мультимедиа).

Автоматизированная система управления (АСУ) - это комплекс технических и программных средств, совместно с организационными структурами (отдельными людьми пли коллективом), обеспечивающий управление объектом (комплексом) в производственной, научной или общественной среде.

Выделяют информационные системы управления образования (Например, кадры, абитуриент, студент, библиотечные программы). Автоматизированные системы для научных исследований (АСНИ), представляющие собой программно-аппаратные комплексы, обрабатывающие данные, поступающие от различного рода экспериментальных установок и измерительных приборов, и на основе их анализа облегчающие обнаружение новых эффектов и закономерностей.Системы автоматизированного проектирования и геоинформационные системы.

Систему искусственного интеллекта, построенную на основе высококачествен­ных специальных знании о некоторой предметной области (полученных от экспер­тов - специалистов этой области), называют экспертной системой. Экспертные системы - один из немногих видов систем искусственного интеллекта - получили широкое распространение, и нашли практическое применение. Существу­ют экспертные системы по военному делу, геологии, инженерному делу, информа­тике, космической технике, математике, медицине, метеорологии, промышленности, сельскому хозяйству, управлению, физике, химии, электронике, юриспруденции и т.д. И только то, что экспертные системы остаются весьма сложными, дорогими, а главное, узкоспециализированными программами, сдерживает их еще более широ­кое распространение.

Экспертные системы (ЭС) - это компьютерные программы, созданные для выполнения тех видов деятельности, которые под силу человеку-эксперту. Они работают таким образом, что имитируют образ действий человека-эксперта, и существенно отличаются от точных, хорошо аргументированных алгоритмов и не похожи на математические процедуры большинства традиционных разработок.

**Тема №5**

# Принципы построения модели IDEF0: контекстная диаграмма, субъект

# Цель: научиться строить модель idef0

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. Никто в организации не знает, как она работает в той мере подробности, которая необходима для создания ИС. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но плохо знает, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель. Такая модель должна быть адекватна предметной области, следовательно, она должна содержать в себе знания всех участников бизнес-процессов организации.

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом (SoftTech, Inc.) и называвшийся первоначально SADT - Structured Analysis and Design Technique. (В начале 70-х годов вооруженные силы США применили подмножество SADT, касающееся моделирования процессов, для реализации проектов в рамках программы ICAM (Integrated Computer-Aided Manufacturing). В дальнейшем это подмножество SADT было принято в качестве федерального стандарта США под наименованием IDEF0. Подробные спецификации на стандарты IDEF можно найти на сайте http://www.idef.com .

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной - функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Моделируемая система рассматривается как произвольное подмножество Вселенной. Произвольное потому, что, во-первых, мы сами умозрительно определяем, будет ли некий объект компонентом системы, или мы будем его рассматривать как внешнее воздействие, и, во-вторых, оно зависит от точки зрения на систему. Система имеет границу, которая отделяет ее от остальной Вселенной. Взаимодействие системы с окружающим миром описывается как вход (нечто, что перерабатывается системой), выход (результат деятельности системы), управление (стратегии и процедуры, под управлением которых производится работа) и механизм (ресурсы, необходимые для проведения работы). Находясь под управлением, система преобразует входы в выходы, используя механизмы.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т. е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования - вопросы, на которые построенная модель должна дать ответ. Другими словами, первоначально необходимо определить область (Scope) моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента - широту и глубину. Широта подразумевает определение границ модели - мы определяем, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени - трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему; поскольку все объекты модели взаимосвязаны, внесение нового объекта может быть не просто арифметической добавкой, но в состоянии изменить существующие взаимосвязи. Внесение таких изменений в готовую модель является, как правило, очень трудоемким процессом (так называемая проблема "плавающей области").

**Цель моделирования (Purpose).**Модель не может быть построена без четко сформулированной цели. Цель должна отвечать на следующие вопросы:

Почему этот процесс должен быть замоделирован?

Что должна показывать модель?

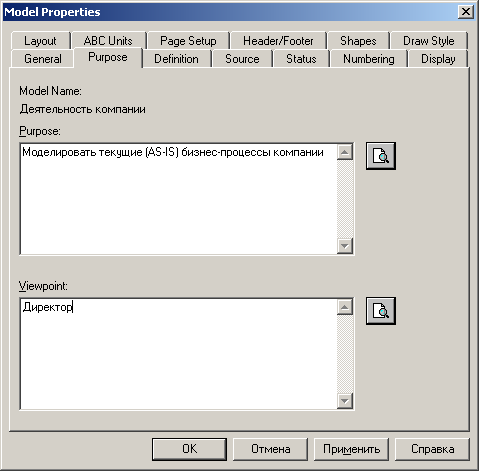
Что может получить читатель?

Формулировка цели позволяет команде аналитиков сфокусировать усилия в нужном направлении. Примерами формулирования цели могут быть следующие утверждения: "Идентифицировать и определить текущие проблемы, сделать возможным анализ потенциальных улучшений", "Идентифицировать роли и ответственность служащих для написания должностных инструкций", "Описать функциональность предприятия с целью написания спецификаций информационной системы" и т. д.

**Точка зрения (Viewpoint)**. Хотя при построении модели учитываются мнения различных людей, модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. Очевидно, что описание работы предприятия с точки зрения финансиста и технолога будет выглядеть совершенно по-разному, поэтому в течение моделирования важно оставаться на выбранной точке зрения. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Часто при выборе точки зрения на модель важно задокументировать дополнительные альтернативные точки зрения. Для этой цели обычно используют диаграммы FEO (For Exposition Only).

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует выбрать пункт меню**Edit/Model Properties**, вызывающий диалог Model Properties (рис.). В закладке**Purpose**следует внести цель и точку зрения, а в закладку**Definition**- определение модели и описание области.

В закладке **Status**того же диалога можно описать статус модели (черновой вариант, рабочий, окончательный и т. д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате). В закладке**Source**описываются источники информации для построения модели (например, "Опрос экспертов предметной области и анализ документации"). Закладка**General**служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок модели -**AS-IS**и**ТО-ВЕ**.



*Рис. Диалог задания свойств модели*

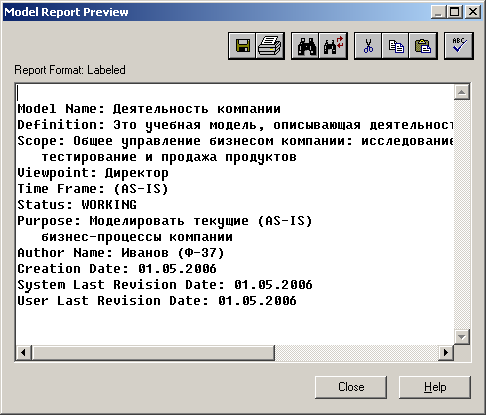
**Модели AS-IS и ТО-ВЕ**. Обычно сначала строится модель существующей организации работы - AS-IS (как есть). На основе модели AS-IS достигается консенсус между различными единицами бизнеса по тому, "кто что сделал" и что каждая единица бизнеса добавляет в процесс. Модель AS-IS позволяет выяснить, "что мы делаем сегодня" перед тем, как перепрыгнуть на то, "что мы будем делать завтра". Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Детализация бизнес-процессов позволяет выявить недостатки организации даже там, где функциональность на первый взгляд кажется очевидной. Признаками неэффективной деятельности могут быть бесполезные, неуправляемые и дублирующиеся работы, неэффективный документооборот (нужный документ не оказывается в нужном месте в нужное время), отсутствие обратных связей по управлению (на проведение работы не оказывает влияния ее результат), входу (объекты или информация используются нерационально) и т. д. Найденные в модели AS-IS недостатки можно исправить при создании модели ТО-ВЕ (как будет) - модели новой организации бизнес-процессов. Модель нужна ТО-ВЕ для анализа альтернативных/лучших путей выполнения работы и документирования того, как компания будет делать бизнес в будущем.

Следует указать на распространенную ошибку при создании модели AS-IS - это создание идеализированной модели. Примером может служить создание модели на основе знаний руководителя, а не конкретного исполнителя работ. Руководитель знаком с тем, как предполагается выполнение работы по руководствам и должностным инструкциям и часто не знает, как на самом деле подчиненные выполняют рутинные работы. В результате получается приукрашенная, искаженная модель, которая несет ложную информацию и которую невозможно в дальнейшем использовать для анализа. Такая модель называется SHOULD\_BE (как должно бы быть).

Технология проектирования ИС подразумевает сначала создание модели AS-IS, ее анализ и улучшение бизнес-процессов, т. е. создание модели ТО-ВЕ, и только на основе модели ТО-ВЕ строится модель данных, прототип и затем окончательный вариант ИС. Построение системы на основе модели AS-IS приводит к автоматизации предприятия по принципу "все оставить как есть, только чтобы компьютеры стояли", т. е. ИС автоматизирует несовершенные бизнес-процессы и дублирует, а не заменяет существующий документооборот. В результате внедрение и эксплуатация такой системы приводит лишь к дополнительным издержкам на закупку оборудования, создание программного обеспечения и сопровождение того и другого.

Иногда текущая AS-IS и будущая ТО-ВЕ модели различаются очень сильно, так что переход от начального к конечному состоянию становится неочевидным. В этом случае необходима третья модель, описывающая процесс перехода от начального к конечному состояния системы, поскольку такой переход - это тоже бизнес-процесс.

Результат описания модели можно получить в отчете **Model Report**. Диалог настройки отчета по модели вызывается из пункта меню**Report/Model Report**. В диалоге настройки следует выбрать необходимые поля, при этом автоматически отображается очередность вывода информации в отчет (рис.).



**Рис. Отчет по модели**

**Диаграммы IDEF0.**Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать четыре типа диаграмм:

контекстную диаграмму (в каждой модели может быть только одна контекстная диаграмма);

диаграммы декомпозиции;

диаграммы дерева узлов;

диаграммы только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы - эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения, либо для специальных целей.

Пример создания функционально модели.

В качестве примера рассматривается деятельность вымышленной компании «Computer Word». Компания занимается в основном сборкой и продажей настольных компьютеров и ноутбуков. Компания не производит компоненты самостоятельно, а только собирает и тестирует компьютеры.

Основные виды работ в компании таковы:

продавцы принимают заказы клиентов;

операторы группируют заказы по типам компьютеров;

операторы собирают и тестируют компьютеры;

операторы упаковывают компьютеры согласно заказам;

кладовщик отгружает клиентам заказы.

Компания использует лицензионную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

***Задание***

На основании индивидуального задания использовать контекстную диаграмму

**Тема №6**

**Построение диаграммы переходов состояний**

**Цель**: научиться строить диаграмму переходов состояний

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**Характеристика диаграмм переходов состояний (SDT).** Диаграммы SDT демонстрируют поведение разрабатываемой программной системы при получении управляющих воздействий. Под *управляющими воздействиями,* или *сигналами,* в данном случае понимают управляющую информацию, получаемую системой извне; например, управляющими воздействиями считают команды пользователя и сигналы датчиков, подключенных к компьютерной системе. Получив такое управляющее воздействие, разрабатываемая система должна выполнить определенные действия, а затем или остаться в том же состоянии, или перейти в другое состояние, зафиксировав некоторые изменения в системе.

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма переходов состояний представляет динамическое поведение сущностей на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние воздействия других систем или пользователей, иногда называют реактивными. Если такие воздействия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Несмотря на то, что диаграммы чаще всего используются для описания поведения отдельных экземпляров классов (объектов), они также могут применяться для спецификации функциональности других компонентов моделей, таких как варианты использования, действующие лица, подсистемы, операции и методы.

Для построения диаграммы переходов состояний необходимо в соответствии с теорией конечных автоматов определить основные состояния, управляющие воздействия (или условия перехода), выполняемые действия и возможные переходы разрабатываемого программного обеспечения. Условные обозначения, используемые при построении диаграмм переходов состояний, показаны на рис.

*а б в*

Состояние

Действие

*Рис. Условные обозначения диаграмм переходов состояний: а – начальное и конечное состояния; б – промежуточное состояние; в – переход.*

Диаграмма переходов состояний, по существу, представляет собой граф специального вида, который обозначает некоторый автомат. Понятие *автомат* в контексте UML обладает довольно специфической семантикой, основанной на теории автоматов. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Простейшим примером визуального представления состояний и переходов на основе формализма автоматов может служить ситуация с исправностью технического устройства, такого как компьютер. В этом случае вводятся в рассмотрение два самых общих состояния: «исправен» и «неисправен» и два перехода: «выход из строя» и «ремонт». Графически эта информация может быть представлена в виде диаграммы состояний компьютера (рис.).

*Рис. Простейший пример диаграммы переходов состояний для технического устройства типа компьютер.*

Основными понятиями, входящими в формализм автомата, являются *состояние* и *переход.* Главное различие между ними в том, что время нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что в пределе время перехода из одного состояния в другое равно нулю (если дополнительно ничего не сказано). Другими словами, переход объекта из состояния в состояние происходит мгновенно.

Для понимания семантики конкретной диаграммы состояний необходимо представить не только особенности поведения моделируемой сущности, но и знать общие сведения по теории автоматов.

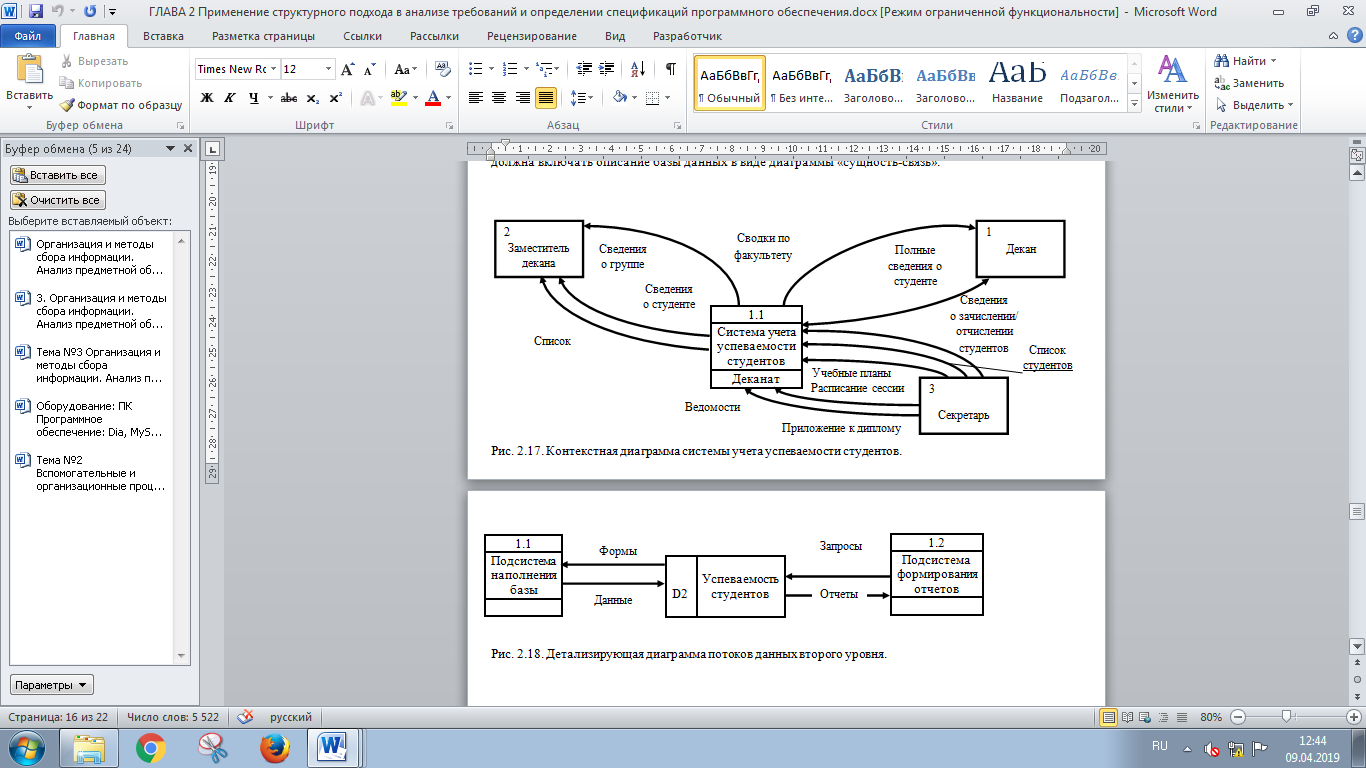
*Сторожевое условие* (guard condition), если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское выражение. Напомним, что булевское выражение должно принимать одно из двух взаимно исключающих значений: «истина» или «ложь». Из контекста диаграммы состояний должна явно следовать семантика этого выражения. Если же сторожевое условие принимает значение «ложь», то переход не может сработать при отсутствии других переходов объект не может перейти в целевое состояние по этому переходу. Однако вычисление истинности сторожевого условия происходит только после возникновения связанного с ним события.

Для интерактивного программного обеспечения с развитым пользовательским интерфейсом основными управляющими воздействиями выступают команды пользователя, для программного обеспечения реального времени – сигналы от датчиков и (или) оператора производственного процесса. Общим для этих типов программного обеспечения является наличие состояния ожидания, когда система приостанавливает работу до получения очередного управляющего воздействия. Для интерактивного программного обеспечения наиболее характерно получение команд различных типов, а для программного обеспечения реального времени – однотипных сигналов либо от многих датчиков, либо требующих продолжительной обработки.

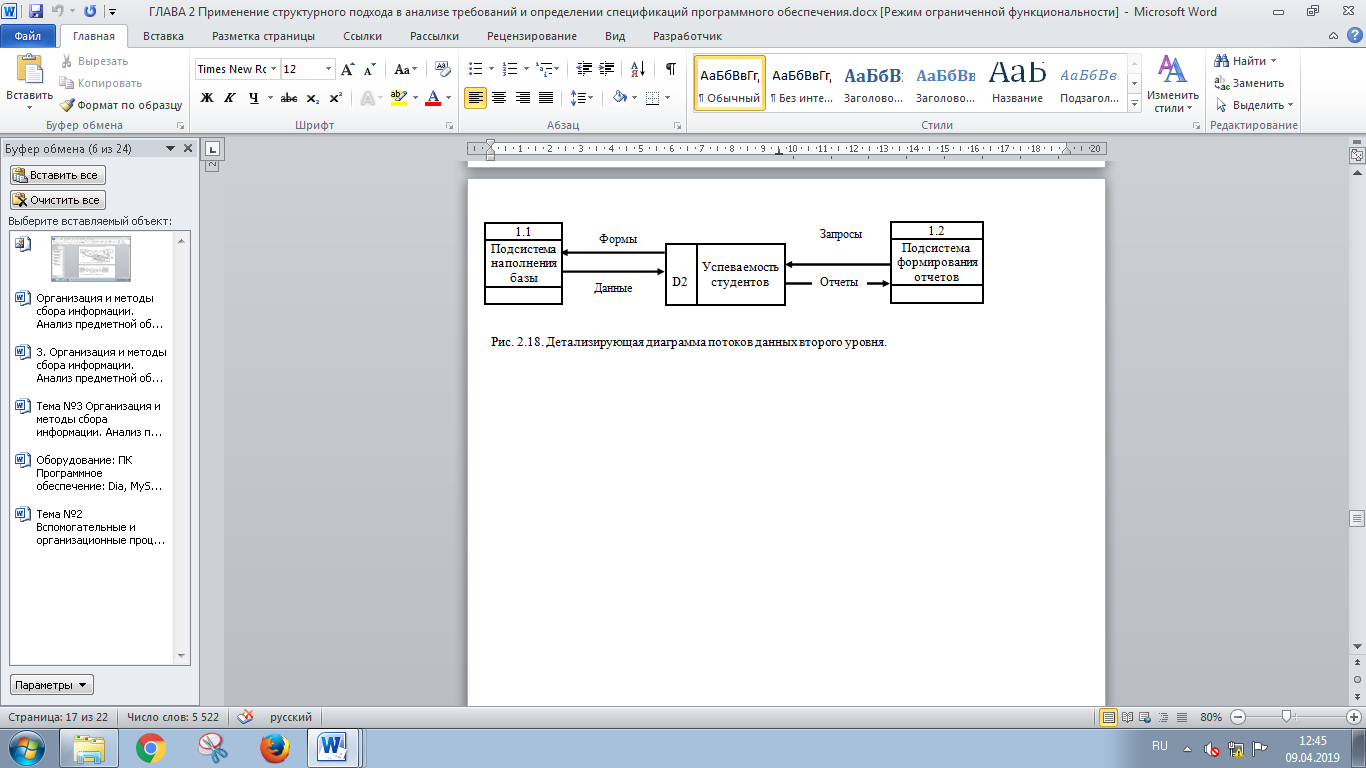
В отличии от интерактивных систем для систем реального времени обычно установлено более жесткое ограничение на время обработки полученного сигнала. Такое ограничение часто требует выполнения дополнительных исследований поведения системы во времени.

К программному обеспечению, при разработке которого требуется уточнение особенностей поведение посредством поступления диаграммы переходов состояний, относится и программное обеспечение, ориентированное на работу а сети. При этом обычно отдельно строят модели поведения сервера и клиента, представляя сообщения, передаваемые между ними в виде управляющих воздействий.

**Пример построения диаграммы переходов состояний.** На предлагаемой диаграмме (рис.) описываются возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение объекта «Заказ» автоматизированной информационной системы «Склад оптовой торговли» в течение его существования (поступление, обработка, формирование поставки). На ней отображаются функции, которые выполняются объектом «Заказ» в определенном состоянии. Метки деятельности имеют следующий синтаксис: *выполнить / <деятельность>* (например, выполнить/проверить строку). Переходы имеют метки, которые синтаксически состоят из трех необязательных частей: *<Событие> <[Условие]> </Действие>* (например, [не все строки проведены]/получить следующую строку).



*Рис. Диаграмма переходов состояний объекта «Заказ»*



*Рис. Детализирующая диаграмма потоков данных второго уровня.*

***Залдание***

На основании индивидуального задания использовать диаграммы переходов состояний.

**Тема №7**

**Построение диаграммы потоков данных**

**Цель:** научиться строить диаграмму потоков данных

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**Характеристика диаграмм потоков данных (DFD).** Диаграмма DFD состоит из узлов обработки данных, средств их хранения и внешних по отношению к используемой диаграмме источников или потребителей данных.

Диаграмма потоков данных – основное средство моделирования функциональных требований к системе, проектируемой или реально существующей. В основе модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных потока данных. Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам; те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации.

Для изображения диаграмм потоков данных традиционно используют два вида нотаций – Йордана и Гейна-Сарсона, которые представлены в табл.

|  |  |  |
| --- | --- | --- |
| Таблица Виды нотаций | | |
| Понятие | Нотация Йордана | Нотация Гейна-Сарсона |
| Внешняя сущность | Наименование | Номер Наименование |
| Процесс, система, подсистема |  | |  | | --- | | Номер | | Наименование | | Механизм | |
| Накопитель данных | |  | | --- | | Наименование | | |  |  | | --- | --- | | № | Наименование | |
| Поток данных | Наименование | Наименование |

*Внешняя сущность –* это материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчик, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешний сущности указывает на то, что этот объект или эта система находятся за пределами границ анализируемой информационной системы. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой информационной системы, если это необходимо, или, наоборот, часть процессов информационной системы может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

При построении модели сложной информационной системы она может быть представлена в самом общем виде на так называемой *контекстной диаграмме*. На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с потребителями и источниками информации, т.е. описывают интерфейс системы с внешним миром. Система может быть представлена как единое целое либо может быть декомпозирована на ряд подсистем. Номер подсистемы служит для ее идентификации. В поле имени вводится наименование подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

*Процесс* представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д. Номер процесса служит для его идентификации. В поле имени вводится наименование процесса, при этом использование таких глаголов, как обработать, модернизировать или отредактировать означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа. Информация в поле физической реализации показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

*Накопитель данных* представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь; причем способы помещения и извлечения могут быть любыми. Накопитель данных может быть реализован физически в виде ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носители и т.д.

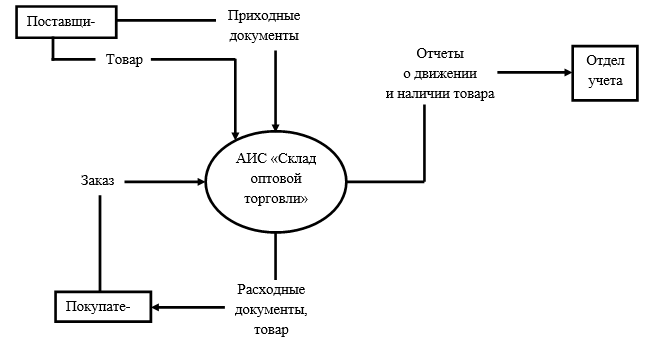
В нотации Гейна-Сарсона накопитель данных идентифицируется буквой D и произвольным числом. Имя накопителя выбирается из соображения наибольшей информативности для проектировщика.

Накопитель данных в общем случае выступает прообразом будущей базы данных, и описание хранящихся в нем данных должно быть увязано с информационной моделью.

*Поток данных* определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т.д. Каждый поток данных имеет имя, отражающее его содержание.

Построение иерархии диаграмм потоков данных начинают с контекстной диаграммы, которая определяет наиболее общий вид системы. Обычно начальная контекстная диаграмма имеет форму звезды. Если проектируемая система содержит большое количество внешних сущностей (более 10), имеет распределение природу или включает уже существующие подсистемы, то строят иерархии контекстных диаграмм. В процессе детализации соблюдают правило балансировки – при детализации подсистемы могут использоваться компоненты только тех подсистем, с которыми у разрабатываемой подсистемы существует информационная связь (т.е. с которыми она связана потоками данных). На недетализируемые процессы составляют спецификации, которые должны содержать описание функций данного процесса. Такое описание может выполняться на естественном языке, с применением структурного языка (псевдокодов), с применением таблиц и деревьев решений, в виде схем алгоритмов.

**Пример построения диаграмм потоков данных АИС «Склад оптовой торговли».** Построение иерархии диаграмм потоков данных начнем с контекстной диаграммы, которая определяет наиболее общий вид системы. Таким образом, определим, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации (рис.).

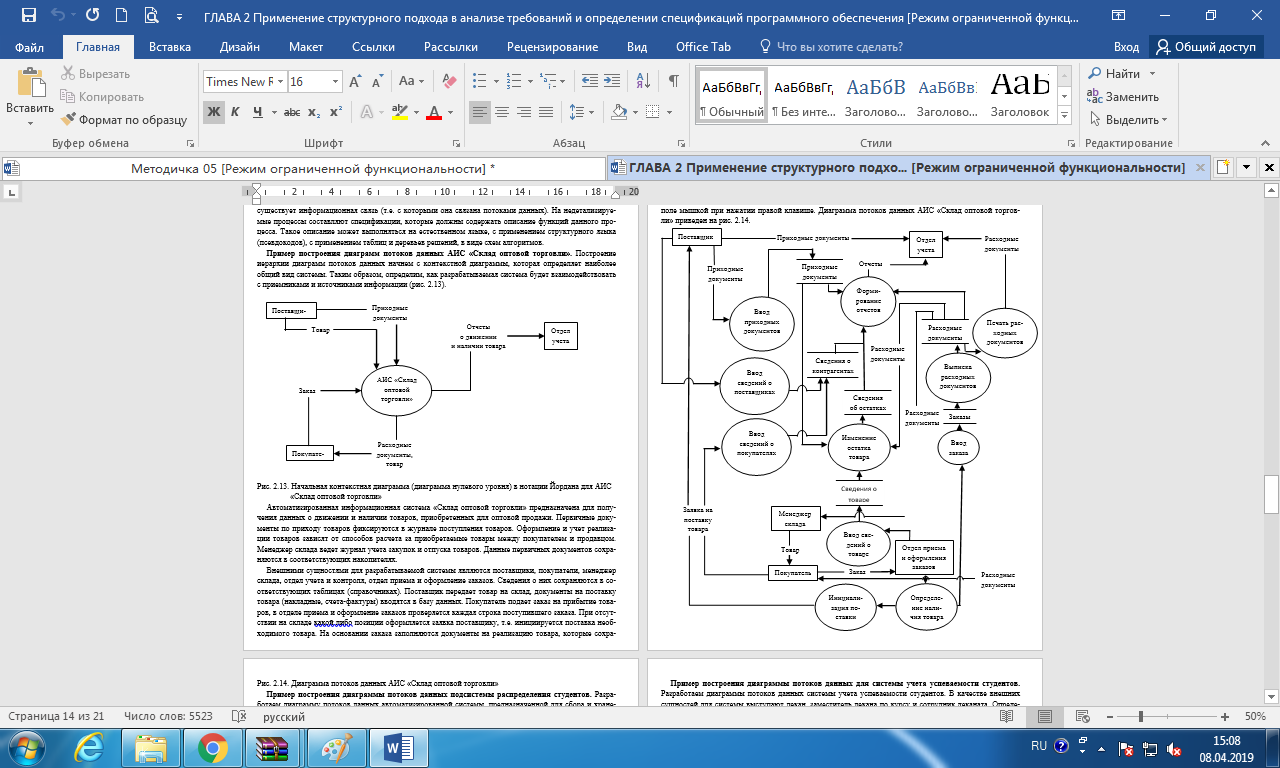


*Рис. Начальная контекстная диаграмма (диаграмма нулевого уровня) в нотации Йордана для АИС «Склад оптовой торговли»*

Автоматизированная информационная система «Склад оптовой торговли» предназначена для получения данных о движении и наличии товаров, приобретенных для оптовой продажи. Первичные документы по приходу товаров фиксируются в журнале поступления товаров. Оформление и учет реализации товаров зависят от способов расчета за приобретаемые товары между покупателем и продавцом. Менеджер склада ведет журнал учета закупок и отпуска товаров. Данные первичных документов сохраняются в соответствующих накопителях.

Внешними сущностями для разрабатываемой системы являются поставщики, покупатели, менеджер склада, отдел учета и контроля, отдел приема и оформление заказов. Сведения о них сохраняются в соответствующих таблицах (справочниках). Поставщик передает товар на склад, документы на поставку товара (накладные, счета-фактуры) вводятся в базу данных. Покупатель подает заказ на прибытие товаров, в отделе приема и оформление заказов проверяется каждая строка поступившего заказа. При отсутствии на складе какой либо позиции оформляется заявка поставщику, т.е. инициируется поставка необходимого товара. На основании заказа заполняются документы на реализацию товара, которые сохраняются в базе данных, распечатываются и выдаются покупателю. В конце каждого дня все первичные документы передают менеджеру отдела учета (бухгалтеру). На основании сведений о приходе и реализации товара менеджеры склада и работники отдела учета формируют отчеты по оборотам и остатком товара на складе.

Для построения диаграммы потоков данных нужно запустить MS Visio. Появится окно в котором необходимо выбрать папку Shapes/Business Process. В открывшемся списке форм (Shapes) для построения диаграммы потоков данных следует выбрать пункт Data Flow Diagram Shapes. В результате проделанных действий на экране появится окно, в левой часть которого будет отображен набор графических символов, а в правой части – лист для рисования диаграммы. Выбрав нужный компонент можно щелчком мыши по соответствующей пиктограмме, после чего графический символ переносится на рабочее поле мышкой при нажатии правой клавише. Диаграмма потоков данных АИС «Склад оптовой торговли» приведен на рис.



*Рис. Диаграмма потоков данных АИС «Склад оптовой торговли»*

**Пример построения диаграммы потоков данных подсистемы распределения студентов.** Разработаем диаграмму потоков данных автоматизированной системы, предназначенной для сбора и хранения информации о студентах, обучающихся на старших курсах учебного заведения. Система предназначена для использования в образовательных учреждениях, с целью предоставления через Интернет потенциальным работодателям информации о выпускниках. Начальная контекстная диаграмма потоков данных в нотации Гейна-Сарсона приведена на рис. Внешними сущностями в ней являются работодатель, администратор и студент.

Работодатель регистрируется в системе, вводит данные для поиска кандидатов и получает от системы результат поиска. Администратор вводит информацию о студентах, которые сохраняются в базе данных. Студент может изменить свою контактную информацию, изменения так же сохраняются в базе данных. Выбранным студентам работодатель посылает сообщение с предложением о работе. Диаграмма потоков данных системы распределения студентов приведена на рис.

1

Работодатель

2

Студент

Требования к потенциальным сотрудникам

Сведения

о возможных кандидатах

Сведения о вакансиях

Резюме

Подсистема распределения студентов

1.1

Администратор

*Рис.Начальная контекстная диаграмма в нотации Гйна-Сарсона для системы*

*распределения студентов.*

**Пример построения диаграммы потоков данных для системы учета успеваемости студентов.** Разработаем диаграммы потоков данных системы учета успеваемости студентов. В качестве внешних сущностей для системы выступают декан, заместитель декана по курсу и сотрудник деканата. Определим потоки данных между этими сущностями и системой.

Декан должен получать:

* Сводку успеваемости по факультету (процент успеваемости групп, курсов и в целом по факультету) на текущий или указанный момент времени;
* Полные сведения об учебе конкретного студента (успеваемость по всем изученным предметам всех завершенных семестров обучения с учетом пересдач).

Заместитель декана по курсу должен получать:

* Сводку успеваемости по курсу (процент успеваемости по группам) на текущий или указанный момент;
* Сведения о сдаче экзаменов и зачетов указанной группой;
* Текущие сведения об успеваемости конкретного студента;
* Полные сведения об учебе конкретного студента (успеваемость по всем изученным предметам всех завершенных семестров обучения с учетом пересдач);
* Список задолжников по факультету и группам с указанием несданных предметов.

Сотрудник деканата должен обеспечивать:

* Ввод списков студентов, зачисленных на первый курс;
* Корректировку списков студентов в соответствии с приказом о зачислении, отчислении, переводе и т.п.;
* Ввод учебных планов кафедр;
* Ввод расписание сессии;
* Ввод результатов сдачи зачетов и экзаменов на основании ведомостей и направлений.

Кроме того, сотрудник деканата должен иметь возможность получать :

* Справку о прослушанных студентом предметах с указанием часов и итоговых оценок;
* Приложение к диплому выпускника также с указанием часов и итоговых оценок.

В результате получаем конкретную диаграмму в нотации Гейна-Сарсона (рис. 2.17). Далее детализируем процессы в системе (рис. 2.18). На детализирующей диаграмме потоков данных выделены две подсистемы: подсистема наполнения базы и подсистема формирования отчетов, а также хранилище данных, которое может быть реализовано как с помощью средств СУБД, так и без них. Дальнейшая детализация процессов очевидна, однако становится ясно, что полная спецификация данной разработки должна включать описание базы данных в виде диаграммы «сущность-связь».

2

Заместитель декана

1

Декан

3

Секретарь

Сводки по факультету

Полные сведения о студенте

Учебные планы

Ведомости

Приложение к диплому

Расписание сессии

Сведения

о зачислении/

отчислении

студентов

Список студентов

Список

задолжников

Сведения о группе

Сведения о студенте

|  |
| --- |
| 1.1 |
| Система учета успеваемости студентов |
| Деканат |

Рис. 2.17. Контекстная диаграмма системы учета успеваемости студентов.

|  |
| --- |
| 1.1  Формы  Данные  Запросы  Отчеты |
| Подсистема наполнения базы |
|  |

|  |  |
| --- | --- |
| D2 | Успеваемость студентов |

|  |
| --- |
| 1.2 |
| Подсистема  формирования  отчетов |
|  |

Рис. 2.18. Детализирующая диаграмма потоков данных второго уровня.

***Задание***

На основании индивидуального задания использовать диаграммы потоков данных.

**Тема №8**

**Построение структурной и функциональной схем**

**Цель:** научиться строить структурную и функциональную схемы

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**СТРУКТУРНАЯ СХЕМА**

Процесс проектирования программного обеспечения включает в себя определение структурных компонентов программной системы и связей между ними. Результат уточнения структуры может быть представлен в виде структурной схемы, которая дает достаточно полное представление о проектируемом программном обеспечении.

На рис. представлена структурная схема программного обеспечения автоматизированной информационной системы «Склад оптовой торговли».

Рис. Структурная схема программного обеспечения АИС «Склад оптовой торговли».

**ФУНКЦИОНАЛЬНАЯ СХЕМА**

*Функциональная схема* – это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состав данных в потоках и указанием используемых файлов и устройств.

Схемы могут использоваться на различных уровнях детализации, при этом число уровня зависит от размеров и сложности задачи обработки данных. Уровень детализации должен быть таким, чтобы различные части и взаимосвязь между ними были понятны в целом.

*Схемы данных* отображают путь данных при решении задач и определяют этапы обработки, а также различные применяемые носители данных. Схема данных состоит из следующих символов:

* Символы данных (символы данных могут также указывать вид носителя данных);
* Символы процесса, производимого с данными (символы процесса могут также указывать функции, выполняемые вычислительной машиной);
* Символов линий, указывающих потоки данных между процессами и (или) носителями данных;
* Специальные символы, используемые для облегчения написания и чтения схемы.

*Схемы программ* отображают последовательность операций в программе. Схема программ состоит из следующих символов:

* Символы процесса, указывающие фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);
* Линейные символы, указывающие поток управления;
* Специальные символы, используемые для облегчения написания и чтения схемы.

*Схемы работы системы* отображают управление операциями и поток данных в системе. Схема работы системы состоит из следующих символов:

* Символы данных, указывающие на наличие данных (символы данных могут также указывать вид носителя данных);
* Символы процесса, указывающие операции, которые следует выполнять над данными, а также определяющие логический путь, которого следует придерживаться;
* Линейные символы, указывающие потоки данных между процессами и (или) носителями данных, а также поток управления и чтения блок-схемы.

Для изображения функциональных схем используют специальные обозначения, установленные ГОСТ 19.701–90 (табл.).

Функциональные схемы более информативны, чем структурные. На рис. приведена функциональная схема программной системы, реализующей операцию продажи товара автоматизированной информационной системы «Склад оптовой торговли».

|  |  |  |
| --- | --- | --- |
| Таблица Графические обозначения основных блоков алгоритмов | | |
| Наименование | Графическое обозначение | Назначение |
| Терминатор |  | Символ отображает выход во внешнюю среду и из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных) |
| Данные |  | Символ отображает данные, носитель данных не определен |
| Запоминающее устройство с прямым доступом |  | Символ отображает данные, хранящиеся в запоминающем устройстве с прямым доступом (магнитный диск, магнитный барабан, гибкий магнитный диск) |
| Документ |  | Символ отображает данные, представленные на носителе в удобочитаемой форме (машинограмма, документ для оптического или магнитного считывания, микрофильм, рулон ленты с итоговыми данными, бланк ввода данных) |
| Ручной ввод |  | Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом) |
| Дисплей |  | Символ отображает данные, представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран, индикаторы ввода информации) |
| Процесс |  | Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или к определению, по которому из нескольких направлений потока следует двигаться) |
| Предопределенный процесс |  | Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле) |

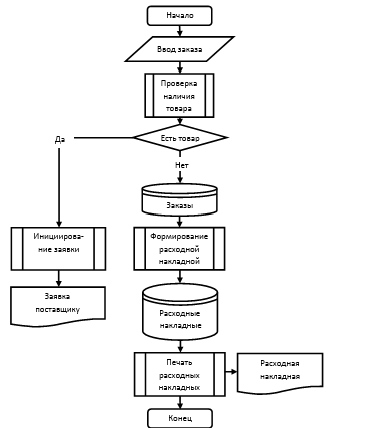


Рис. Пример функциональной схемы программного обеспечения АИС «Склад оптовой торговли»

***Задание***

На основании технического задания и анализа требований к программному обеспечению в предыдущих работах разработать структурные и функциональные схемы программного обеспечения по своему варианту задания.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Приведите пример структурной схемы ПО.
2. Для чего используются функциональные схемы?
3. Опишите основные элементы функциональных схем ПО.
4. В чем заключается достоинства и недостатки использования функциональных схем?

**Тема №9**

**Диаграммы вариантов использования**

**Цель:** научиться строить диаграмму вариантов использования

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

В настоящее время существуют десятки приемов, методик, визуальных представлений, позволяющих моделировать требования к программному обеспечению. Необходимо определить целесообразность использования тех или иных приемов. Анализ требований должен соответствовать тому, что делает система, абстрагируясь от деталей реализации, т. е. от того, как она это делает.

Язык UML предоставляет в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, предназначенный для разработки и документирования сложных моделей систем различного целевого назначения.

**Определение вариантов использования.** Разработку спецификаций программного обеспечения начинают с анализа требований к функциональности, указанных в техническом задании. В процессе анализа выявляют внешних пользователей разрабатываемого программного обеспечения и перечень отдельных аспектов его поведения в процессе взаимодействия с конкретными пользователями. Аспекты поведения программного обеспечения были названы вариантами использования, или прецедентами (use cases).

Не следует путать вариант использования с конкретными операциями будущей системы. Каждый вариант использования связан с некоторой целью, имеющей самостоятельное значение; например, для текстового редактора Формирование оглавления — это вариант использования, а Связывание заголовков со специальными стилями — операция, которую необходимо выполнить, чтобы стало возможно автоматическое построение оглавления.

В зависимости от цели выполнения процедуры различают следующие варианты использования:

* основные (базовые) — обеспечивают требуемую функциональность разрабатываемого ПО;
* вспомогательные — обеспечивают выполнение необходимых настроек системы и ее обслуживание (например, архивирование информации и т.п.);
* дополнительные — обеспечивают дополнительные удобства для пользователя (как правило, реализуются, если не требуют серьезных затрат каких-либо ресурсов ни при разработке, ни при эксплуатации).

Графических средств языка UML на практике оказывается недостаточно для спецификации функциональных требований к программной системе. Следует отметить, что одним из требований языка UML является самодостаточность диаграмм для представления информации о моделях проектируемых систем. Однако изобразительных средств языка UML явно не хватает для того, чтобы учесть на диаграммах вариантов использования особенности функционального поведения сложной системы. С этой целью рекомендуется дополнять этот тип диаграмм текстовыми сценариями, которые уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее вариантов использования.

В контексте языка UML сценарий используется для дополнительной иллюстрации взаимодействия актеров и вариантов использования. Предлагаются различные способы представления или написания подобных сценариев. Один из таких шаблонов представлен в табл. и может быть рекомендован для применения на начальных этапах концептуального моделирования. В зависимости от уровня абстракции вариант использования может описываться кратко или более подробно. Краткая форма описания содержит: имя варианта использования, перечень действующих лиц (актеров) и тип варианта использования (основной, вспомогательный или дополнительный) и его краткое описание.

При написании *сценариев* вариантов использования важно понимать, что текст *сценария* должен дополнять или уточнять диаграмму вариантов использования, но не заменять ее полностью. В противном случае будут потеряны достоинства визуального представления моделей.

Таблица Шаблон для написания сценария отдельного варианта использования

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Типичный ход событий, приводищий к успешному выполнению варианта использования | Исключение 1 | Примечание 1 |
| Действующие лица | Исключение 2 | Примечание 2 |
| Цель | Исключение 3 | Примечание 3 |
| Тип | ... | ... |
| Краткое описание |  |  |
| Ссылки на другие варианты использования | Исключение n | Примечание n |

**Основные понятия.** Диаграммы вариантов использования. Диаграммы вариантов использования позволяют наглядно представить ожидаемое поведение программной системы. Основными понятиями диаграмм вариантов использования являются: действующее лицо, вариант использования и связь.

На рис. приведены условные обозначения, которые применяют при изображении диаграмм вариантов использования.



Рис. Основные условия обозначения диаграмм вариантов использования:

*а —* действующее лицо; *б —* вариант использования; *в -* связь

*Действующее лицо* — внешняя по отношению к разрабатываемому программному обеспечению сущность, которая взаимодействует с ним в целях получения или предоставления какой-либо информации. Как уже упоминалось выше, действующими лицами могут быть пользователи, другое программное обеспечение или какие-либо технические средства.

*Вариант использования* — некоторая очевидная для действующего лица процедура, решающая его конкретную задачу. Все варианты использования так или иначе связаны с требованиями к функциональности разрабатываемой системы и могут сильно различаться по объему выполняемой работы (рис.).

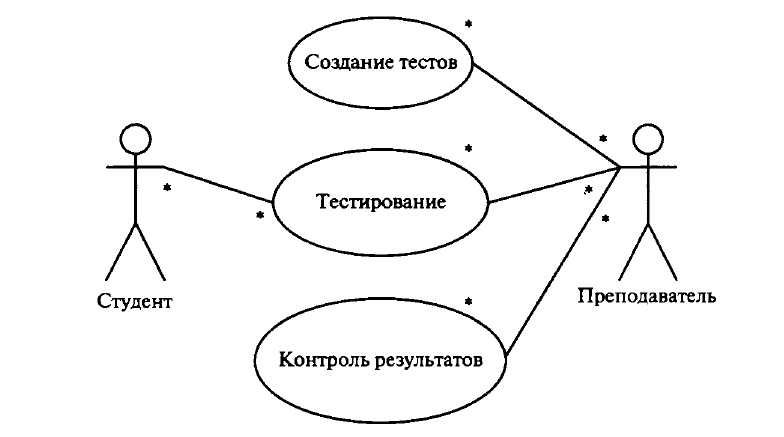


Рис. Пример диаграммы вариантов использования для тестовой системы

*Связь* — взаимодействие действующих лиц и соответствующих вариантов использования.

Варианты использования также могут быть связаны между собой. При этом фиксируют связи использования и расширения.

*Использование (uses (include))* подразумевает, что существует некоторый фрагмент поведения разрабатываемого ПО, который повторяется в нескольких вариантах использования. Этот фрагмент оформляют как отдельный вариант и указывают связь с ним типа «использование».

*Расширение (extends)* применяют, если имеется два подобных варианта использования, различающиеся наличием в одном из них некоторых дополнительных действий. В этом случае дополнительные действия определяют как отдельный вариант использования, который связан с основным вариантом связью типа «расширение».

Главное назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе и возможности согласования полученной модели с заказчиком на ранней стадии проектирования. Любой из вариантов использования может быть подвергнут дальнейшей декомпозиции на множество подвариантов использования отдельных элементов, которые образуют исходную сущность.

**Пример разработки диаграммы вариантов использования.** Для иллюстрации особенностей спецификации функциональных требований на диаграмме вариантов использования можно рассмотреть модель системы «Склад оптовой торговли» (рис.). Для первоначального понимания структуры программной системы выявляются действующие лица (люди-актеры или системы, между которыми происходит взаимодействие). Рассматриваемая система имеет пять актеров, двое из которых выступают контрагентами, а другие — менеджерами склада, осуществляющими выполнение всех операций. Каждый из этих актеров взаимодействует с системой, хотя главными актерами, пожалуй, являются поставщики и покупатели (контрагенты), поскольку именно они инициируют функциональность системы. Далее формулируются варианты использования, т. е. действия, выполняемые системой для реализации общения действующих лиц (актеров).

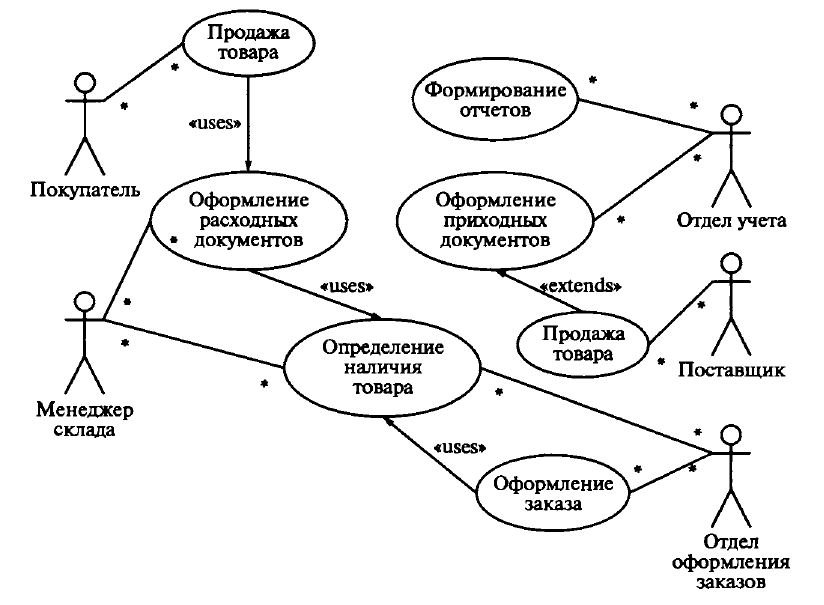


Рис. Диаграмма вариантов использования для проектирования программного обеспечения

АИС «Склад оптовой торговли»

Каждый из действующих лиц преследует определенные цели по отношению к системе:

поставщик — сдать товар на склад, покупатель — приобрести товар, менеджер склада — принять и отпустить товар, менеджер учетного отдела — определить объемы поступления и продаж и проанализировать товарный запас. На основании этих целей можно сформулировать базовые варианты использования и проанализировать взаимосвязи между ними. На самом деле вариантов использования может быть гораздо больше. Например, проверить платежеспособность клиента, получить информацию о товаре, оценить запасы товара на складе, получить оплату и т. д. Однако эта диаграмма дает понять, что будет делать система, как она будет функционировать.

На следующем этапе разработки модели вариантов использования для рассматриваемой системы следует дополнить данную диаграмму текстовым сценарием, написанным на основе предложенного ранее шаблона. Этот сценарий будет дополнять диаграмму, раскрывая содержание и логическую последовательность отдельных действий, которые выполняются системой и актерами в процессе поступления и реализации товаров. В этом случае сценарий удобно представить в виде таблиц, каждая из которых описывает отдельный раздел шаблона.

В г л а в н о м р а з д е л е сценария указывается имя рассматриваемого варианта использования, имена взаимосвязанных с ним актеров, цель выполнения варианта, условный тип и ссылки на другие варианты использования (табл. 4.2).

В с л е д у ю щ е м р а з д е л е сценария описывается последовательность действий, приводящая к успешному выполнению рассматриваемого варианта использования. При этом инициатором действий должен выступать актер Покупатель. Для удобства последующих ссылок каждое действие актеров помечается порядковым номером в последовательности действий (табл.).

Таблица Сценарий варианта использования

|  |  |
| --- | --- |
| Вариант использования | Продажа товара |
| Актеры | Покупатель, Менеджер отдела оформления заказов, Менеджер склада |
| Краткое описание | Покупатель запрашивает товар. Менеджер отдела оформления заказов резервирует товар, оформляет заказ, передает заказ Менеджеру склада. Покупатель оплачивает товар, получает товар на складе |
| Цель | Получение необходимого товара |
| Тип | Базовый |
| Ссылки на другие варианты использования | Включает в себя варианты использования:  определить наличие товара;  оформить заказ |

Таблица Последовательность действий актеров

|  |  |
| --- | --- |
| Действия актеров | Отклик системы |
| 1. Покупатель запрашивает товар  Исключение 1. На складе нет необходимого количества запрашиваемого товара | 2. Менеджер отдела оформления заказов проверяет наличие необходимого товара на складе  3. Менеджер отдела оформления заказов резервирует нужный товар |
| 4. Покупатель оплачивает товар  Исключение 2. Покупатель не оплатил товар | 5. Менеджер отдела оформления заказов выдает разрешение на получение товара  6. Менеджер отдела оформления заказов передает заказ на склад  7. Менеджер склада выдает товар и расходную накладную покупателю  8. Менеджер оформления заказов блокирует получение товара покупателем |

В т р е т ь е м р а з д е л е сценария описывается последовательность действий, выполняемых при возникновении исключительных ситуаций, или исключений (табл.).

Таблица Последовательность действий актеров при возникновении исключительных ситуаций

|  |  |
| --- | --- |
| Действия актеров | Отклик системы |
| Исключение 1. На складе нет необходимого количества запрашиваемого товара | |
| 4. Покупатель оплачивает товар | 3. Менеджер отдела оформления заказов инициирует поставку нужного товара |
| Исключение 2. Покупатель не оплатил товар | |
|  | 8. Менеджер оформления заказов блокирует получение товара покупателем |

Можно дополнить данный сценарий, аналогичным образом описав не только варианты использования «Оформление заказа» и «Определение наличия товара», но и рассмотрев другие исключения, например оформление скидок постоянным покупателям и т. п. При этом полнота сценариев и модели вариантов использования будут определяться теми функциональными требованиями, которые сформулированы в рамках конкретного проекта.

Отдельные небольшие по своему объему сценарии могут быть размещены на диаграмме в форме примечаний note, предназначенных для включения в модель произвольной текстовой информации, которая имеет непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения. Применительно к диаграммам вариантов использования примечание может иметь уточняющую информацию, относящуюся к контексту тех или иных вариантов использования.

Графически примечания на всех типах диаграмм обозначаются прямоугольником с «загнутым» верхним правым уголком (рис.).

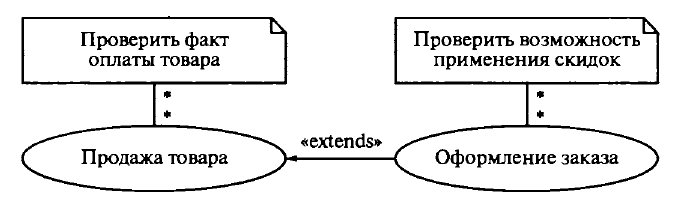


Рис. Пример примечаний на диаграммах вариантов использования

Собственно текст примечания размещается внутри этого прямоугольника. Примечание может относиться к любому элементу диаграммы, в этом случае их соединяет пунктирная линия. Если примечание относится к нескольким элементам, то от него проводятся, соответственно, несколько линий. Как уже отмечалось, примечания могут присутствовать не только на диаграмме вариантов использования, но и на других канонических диаграммах.

**Рекомендации к разработке диаграмм вариантов использования.** Как было отмечено ранее, одно из главных назначений диаграммы вариантов использования заключается в формализации функциональных требований к системе. Диаграмма вариантов использования может служить основой для согласования с заказчиком функциональных требований к системе на ранней стадии проектирования. Любой из базовых вариантов использования в последующем может быть подвергнут декомпозиции на частные варианты использования. При этом рекомендуется, чтобы общее количество актеров в модели не превышало 20, а вариантов использования — 50. В противном случае модель теряет свою наглядность и, возможно, заменяет собой одну из некоторых других диаграмм.

Для разработки диаграммы вариантов использования рекомендуется соблюдать некоторую последовательность действий:

* определить главных, или первичных, и второстепенных актеров;
* определить цели главных актеров по отношению к системе;
* сформулировать основные варианты использования, которые специфицируют функциональные требования к системе;
* упорядочить варианты использования по степени убывания риска их реализации;
* рассмотреть все базовые варианты использования в порядке убывания их степени риска;
* выделить участников, интересы, предусловия и постусловия выполнения выбранного варианта использования;
* написать успешный сценарий реализации выбранного варианта использования;
* определить исключения или неуспех в выполнении сценария варианта использования;
* написать сценарии для всех исключений;
* выделить общие варианты использования и изобразить их взаимосвязи с базовыми со стереотипом uses (include);
* выделить варианты использования для исключений и изобразить их взаимосвязи с базовыми со стереотипом extend;
* проверить диаграмму на отсутствие дублирования вариантов использования и актеров.

Семантика построения диаграммы вариантов использования должна определяться следующими особенностями рассмотренных выше элементов модели. Отдельный экземпляр варианта использования по своему содержанию является выполнением последовательности действий, которая инициализируется посредством экземпляра сообщения от экземпляра актера. В качестве отклика или ответной реакции на сообщение актера выполняется последовательность действий, установленная для данного варианта использования. При этом актеры могут генерировать новые сообщения для инициирования вариантов использования.

Подобное взаимодействие будет продолжаться до тех пор, пока не закончится выполнение требуемой последовательности действий экземпляром варианта использования и указанный в модели экземпляр актера не получит требуемый экземпляр сервиса. Окончание взаимодействия означает отсутствие инициализации сообщений от актеров для базовых вариантов использования.

Варианты использования могут быть дополнительно специфицированы примечаниями с текстом, которые в последующем могут стать прототипами операций и методов совместно с атрибутами. Дальнейшая разработка моделей связана с реализацией вариантов использования в виде графа деятельности посредством конечного автомата или любого другого механизма логического представления поведения, включающего предусловия и постусловия. Взаимодействие между вариантами использования и актерами может уточняться на диаграмме кооперации, когда описываются взаимосвязи между системой, содержащей эти варианты использования, и окружением или внешней средой этой системы.

***Задание***

На основании индивидуального задания использовать диаграммы вариантов использования.

**Тема №10**

**Диаграммы деятельности**

**Цель:** научиться строить диаграмму деятельности

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**Характеристика диаграмм деятельности.** Если диаграмма вариантов использования дает «вид сверху» на функциональность системы, диаграмма деятельности, напротив, позволяет подробно иллюстрировать отдельный вариант использования и его сценарии. В зависимости от степени детализации диаграммы деятельностей могут использоваться на разных этапах разработки. На этапе анализа требований и уточнения спецификаций диаграммы деятельностей позволяют конкретизировать основные функции разрабатываемого программного обеспечения.

Под деятельностью в данном случае понимают задачу (операцию), которую необходимо выполнить вручную или с помощью средств автоматизации. Каждому варианту использования соответствует своя последовательность задач. В теоретическом плане диаграмма деятельности — это обобщенное представление алгоритма, реализующего анализируемый вариант использования. На диаграмме деятельность обозначается прямоугольником с закругленными углами (рис. *а*). Диаграммы деятельности позволяют описывать альтернативные и параллельные процессы.

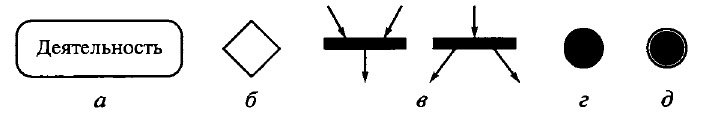
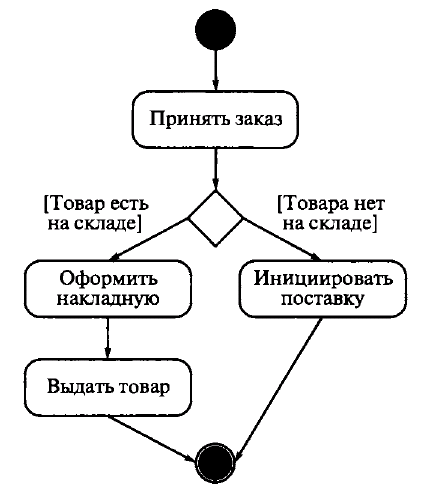
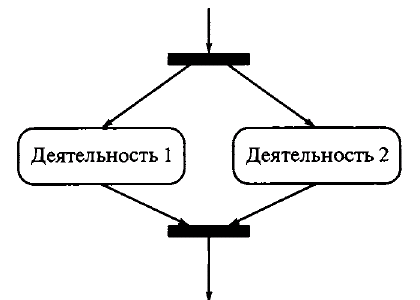


Рис. Условные обозначения диаграммы деятельностей:

*а* — деятельность; *б* — выбор; *в* — линейки синхронизации; *г* — начало; *д* — конец

Для обозначения *альтернативных процессов* используют ромб (рис.*б*), условие указывают рядом, а альтернативы «да», «нет» — рядом с соответствующими выходами. С помощью этого же блока можно построить циклический процесс. Множественность активации деятельности обозначают символом «\*», помещенным рядом со стрелкой активации деятельности, и при необходимости уточняют надписью вида «для каждой строки».

Для обозначения *параллельных процессов* используют линейки синхронизации (рис.*в*). Условные обозначения начала и окончания диаграммы деятельности даны на рис. *г* и *д* . Пример диаграммы деятельности с указанием параллельности процессов приведен на рис. Условие синхронизации можно уточнить, указав его на диаграмме (рис.).



|  |  |
| --- | --- |
| Рис. Пример диаграммы деятельности с указанием параллельности процессов | Рис. Пример диаграммы деятельности |

**Пример построения диаграммы деятельности.** В предыдущем примере для конкретизации описания вариантов использования в автоматизированной информационной системе «Склад оптовой торговли» был разработан текстовый сценарий. Этот сценарий дополняет диаграмму, раскрывая содержание отдельных действий, выполняемых системой и актерами. Однако вместо описания вариантов использования или в дополнение к ним можно использовать диаграммы деятельностей. Диаграмма деятельности позволяет проиллюстрировать вариант использования с требуемой степенью подробности. Имеет смысл уточнять только те варианты использования, краткое описание которых недостаточно для понимания сущности решаемых проблем.

В рамках разрабатываемой модели построим диаграммы деятельности для реализации вариантов использования «Поставка товара» (рис. ) и «Продажа товара» (рис. ). Диаграмма деятельности позволяет проиллюстрировать вариант использования с различной степенью подробности. Полная модель системы может содержать несколько диаграмм деятельности, каждая из которых описывает последовательность реализации либо наиболее важных вариантов использования (типичный ход событий и все исключения), либо нетривиальных операций классов.

Помимо стандартного формата описания, UML предлагает вариант с «плавательными дорожками». Этот формат удобен для описания случая, когда в варианте использования участвуют несколько действующих лиц. При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями (рис. ).

Применение дорожек открывает дополнительные возможности для наглядного представления бизнес-процессов, позволяя специфицировать деятельность подразделений предприятий (рис. ). В случае типового проекта большинство деталей реализации действий могут быть известны заранее на основе анализа существующих систем или предшествующего опыта разработки систем-прототипов. Использование типовых решений может существенно сократить время разработки и избежать возможных ошибок при реализации проекта.

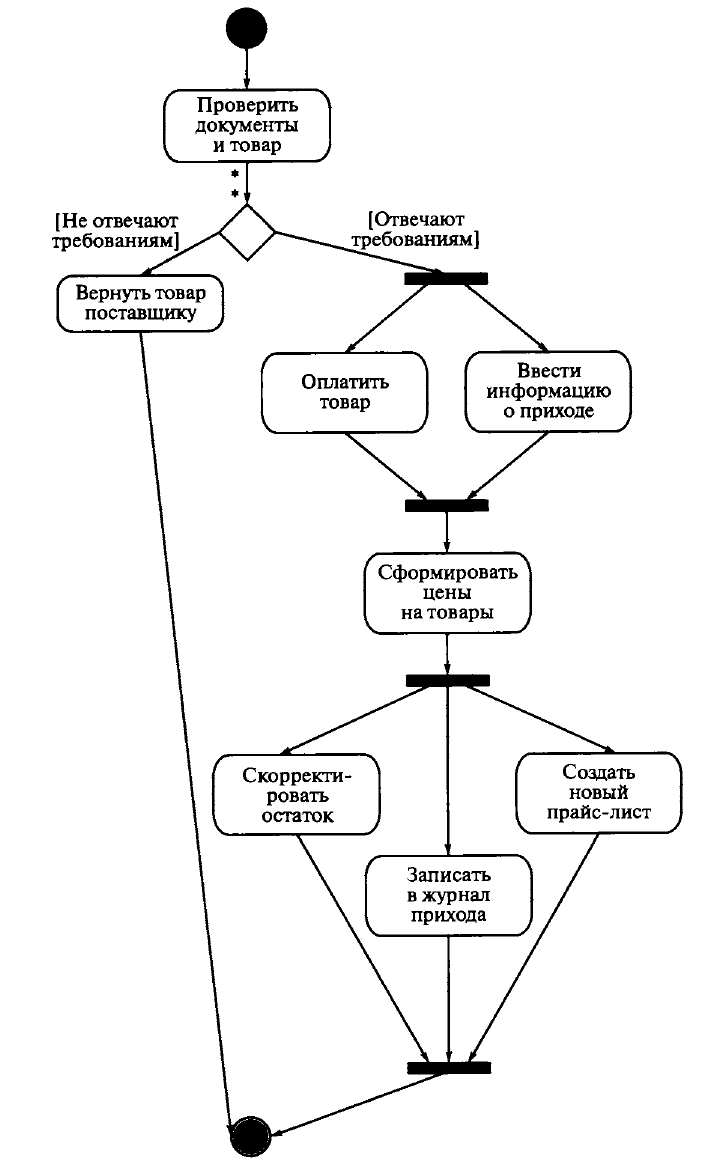
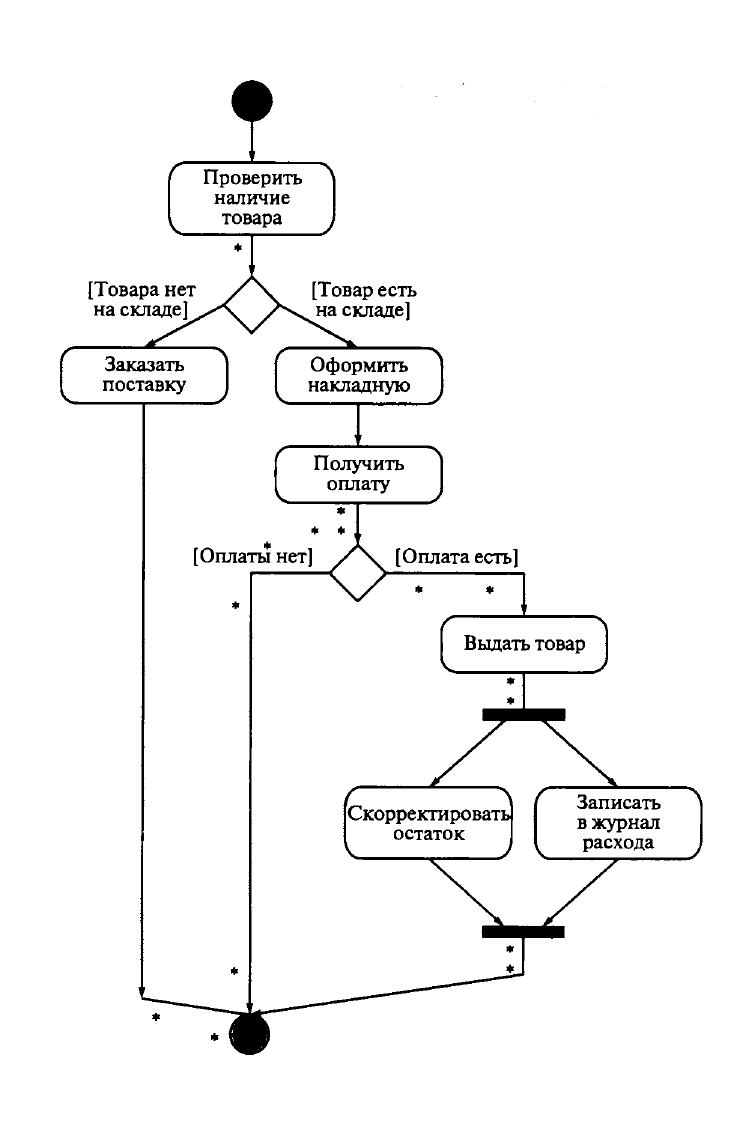


Рис. Диаграмма деятельности для варианта использования «Поставка товара»

Рис. Диаграмма деятельности для варианта использования «Продажа товара»

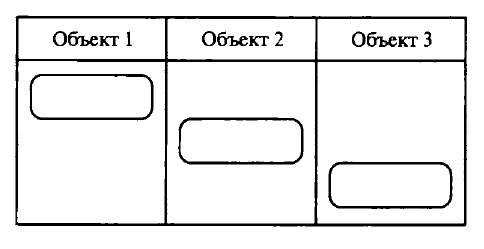


Рис. Вариант диаграммы деятельности с дорожками

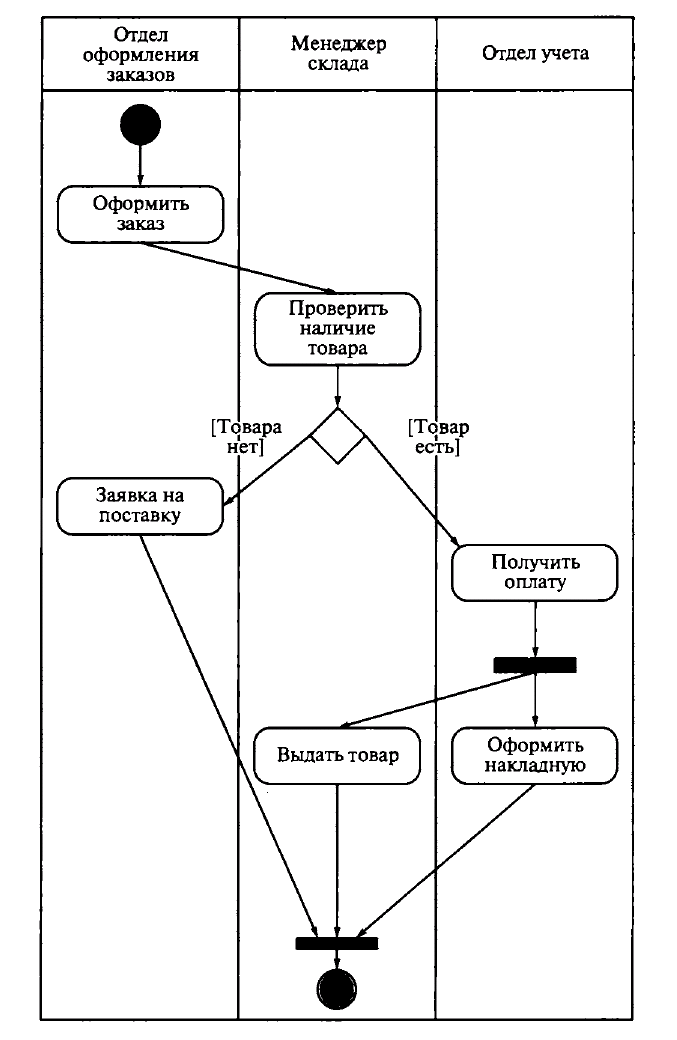


Рис. Пример применения диаграммы деятельности с дорожками для варианта использования

«Оформление заказа»

***Задание***

На основании индивидуального задания использовать диаграмму деятельности.

**Тема №11**

**Диаграммы последовательности**

**Цель:** научиться строить диаграмму последовательности

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Характеристика диаграмм последовательности. Рассмотренные диаграммы деятельности используются для спецификации динамики поведения программных систем, время в явном виде в них не присутствует. Однако временной аспект поведения может иметь существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели в языке UML используются диаграммы последовательности.

Диаграмма последовательности системы — графическая модель, которая для определенного сценария варианта использования показывает динамику взаимодействия объектов во времени. Для построения диаграммы последовательности системы необходимо:

* идентифицировать каждое действующее лицо (объект) и изобразить для него линию жизни;
* из описания варианта использования определить множество системных событий и их последовательность;
* изобразить системные события в виде линий со стрелкой на конце между линиями жизни действующих лиц и системы, а также указать имена событий и списки передаваемых значений.

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения.

Одно измерение — слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни. Внутри прямоугольника записываются имя объекта и имя класса, разделенные двоеточием. При этом вся запись подчеркивается линией, что является признаком объекта, представляющим собой экземпляр класса (рис.).

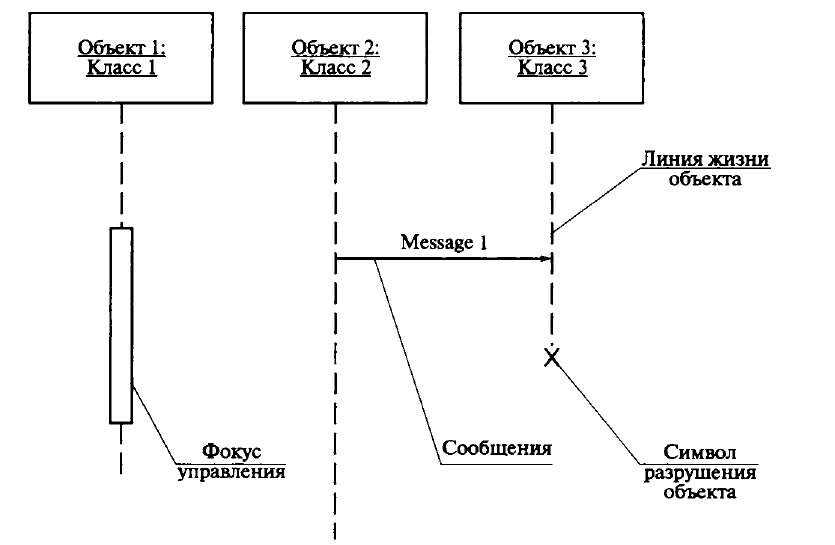


Рис. Различные графические примитивы диаграммы последовательности

Не исключается ситуация, когда имя объекта может отсутствовать на диаграмме последовательности. В этом случае указывается только имя класса, а сам объект считается анонимным. Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия (объект 1 на рис. ). Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Второе измерение диаграммы последовательности — вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и также образуют порядок по времени своего возникновения. Другими словами, сообщения, расположенные на диаграмме последовательности выше, инициируются раньше тех, которые расположены ниже. При этом масштаб на оси времени не указывается, поскольку диаграмма последовательности моделирует лишь временную упорядоченность взаимодействий типа «раньше-позже».

Линия жизни объекта служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. На диаграмме линия жизни изображается пунктирной вертикальной линией, ассоциированной с единственным объектом. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности.

Построение диаграммы последовательности целесообразно начинать с выделения из всей совокупности тех и только тех классов, объекты которых участвуют в моделируемом взаимодействии. После этого все объекты наносятся на диаграмму с соблюдением некоторого порядка инициализации сообщений. Здесь необходимо установить, какие объекты будут существовать постоянно, а какие временно — только на период выполнения ими требуемых действий. Когда объекты определены, можно приступать к спецификации сообщений. При этом следует учитывать роли, которые сообщения играют в системе.

**Пример построения диаграммы последовательности.** Для того чтобы более точно учитывать временной аспект поведения системы при моделировании процессов, описывающих взаимодействие, можно построить диаграммы последовательности. В качестве примера построим диаграмму последовательности для реализации варианта использования «Продажа товара» в информационной системе «Склад оптовой торговли» (рис. ).

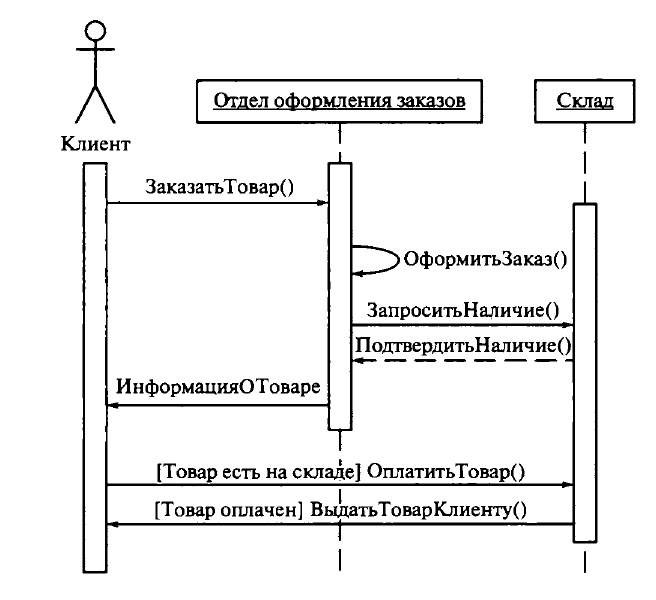


Рис. . Вариант диаграммы последовательности для моделирования операции продажи товара

Данная диаграмма содержит два объекта и одного актера. Объекты не являются постоянно активными, что показано с помощью соответствующих фокусов управления. В качестве имен сообщений указаны имена операций, которые специфицированы у соответствующих классов. Предложения—условия некоторых сообщений записаны обычным текстом в квадратных скобках. Эти условия отражают возможность ветвления процесса продажи и выполнения исключительного сценария соответствующего варианта использования, однако другие варианты использования на данной диаграмме не показаны.

*Задание*

На основании индивидуального задания использовать диаграмму последовательности

**Тема№12**

**Диаграмма «Сущность-связь»**

**Цель:** научиться строить диаграмму «Сущность-связь»

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

**Характеристика диаграмм «сущность-связь».** Данная диаграмма – (ER-модель данных) обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Диаграммы «сущность-связь» в отличие от функциональных диаграмм определяют спецификации структур данных программного обеспечения. Первый вариант модели «сущность-связь» был предложен П. Ченом. В дальнейшем многими авторами были разработаны свои варианты подобных моделей. Все варианты диаграмм «сущность –связь» исходят из одной идеи – графическое изображение нагляднее текстового описания. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями. Наиболее распространенной является нотация Баркера, ее и будем придерживаться.

Базовыми понятиями ER-модели данных (ER – Entity – Relationship) являются сущность, атрибут и связь.

*Сущность* – это класс однотипных реальных или абстрактных объектов (людей, событий, состояний, предметов и т.п.), информация о которых имеет существенное значение для рассматриваемой предметной области. Структурой данных называют совокупность правил и ограничений, которые отражают связи, существующие между отдельными частями (элементами) данных.

Каждая сущность должна иметь:

* Уникальное имя;
* Один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;
* Один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности.

Экземпляр сущности – это конкретный представитель данной сущности. Имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (студент, а не Иванов).

На диаграмме в нотации Баркета сущности изображается прямоугольником, иногда с закругленными углами (рис. *а*). Каждая сущность обладает одним или несколькими атрибутами.

*Атрибут* – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности (рис. *б*). Атрибут, таким образом, представляет собой некоторый тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов. Экземпляр атрибутов – определения характеристик конкретного экземпляра сущности.

Атрибуты делятся на ключевые, т.е. входящие в состав уникального идентификатора ключа, и описательные – прочие.

Первый ключ – это атрибут или совокупность атрибутов и (или) связей, предназначенная для уникальной идентификации каждого экземпляра сущности (совокупность признаков, позволяющих идентифицировать объект). Ключевые атрибуты помещают в начало списка и помечают символом «#» (рис.*в*).

Описательные атрибуты могут быть обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретные значение, необязательные могут быть не определены. Обязательные и необязательные описательные атрибуты помечают символами «\*» и «°» соответственно.

|  |
| --- |
| Имя  сущности |
| Атрибут 1  Атрибут 2  Атрибут 3 |

|  |
| --- |
| Имя  сущности |
| #Атрибут 1  \*Атрибут 2  °Атрибут 3 |

|  |
| --- |
| Имя  сущности |

*а б в*

Рис. Обозначения сущности в нотации Баркера: *а* – без атрибутов; *б* – с указанием атрибутов; *в* – с уточнением атрибутов и их типов (# - ключевой, \* - обязательный, ° - необязательный).

Сущность 1

Сущность 2

Сущность 1

Сущность 2

*а б*

Рис. Модальность связи: *а* – обязательная; *б* – необязательная (пунктир до середины линии).

*Связь* – это отношение одной сущности к другой к самой себе. Каждая связь может иметь одну из двух модальностей связей. Если любой экземпляр одной сущности связан хотя бы с одним экземпляром другой сущности, то связь является обязательной (рис. *а*). Необязательная связь представляет собой условное отношение между сущностями (рис. *б*). Связь может иметь разную модальность с разных концов. Каждая связь может быть прочитана как слева на право, так и справа налево. Каждая сущность может обладать любым количеством связей с другими сущностями модели. Различают три типа отношений (рис. ): «однин-к-одному»; «один-ко-многим»; «многие-ко-многим».

Сущность 1

Сущность 2

Сущность 1

Сущность 2

Сущность 1

Сущность 2

*а б в*

Рис. Обозначение отношений в нотации Баркера: *а* - «однин-к-одному»; *б* - «один-ко-многим»; *в* - «многие-ко-многим».

Связь «один-к-одному» означает, что один экземпляр первой сущности связан только с одним экземпляром второй сущности. Такая связь, скорее всего, свидетельствует о том, что была неверно разделена одна сущность на две (хотя иногда к такому типу связи прибегают в случае если есть необходимость «засекретить» часть данных). При связи «один-ко-многим» каждый экземпляр первой сущности связан с несколькими экземплярами второй сущности. Связь «многие-ко-многим» показывает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности и наоборот. Такой тип связи является временным. Он допустим на ранних этапах разработки модели. В дальнейшем такую связь необходимо заменить двумя связями «один-ко-многим» путем создания промежуточной сущности.

*Независимая сущность* представляет независимые данные, которые всегда присутствуют в системе. Они могут быть как связаны с другими сущностями, так и нет.

*Зависимая сущность* представляет данные, зависящие от других сущностей системы, поэтому она всегда должна быть связана с другими сущностями.

*Ассоциированная сущность* представляет данные, которые ассоциируются с отношениями между двумя и более сущностями. Обычно данный вид сущностей появляется в модели для разрешения отношения «многие-ко-многим» (рис.).

Сущность 1

Сущность 2

Ассоциированная сущность

Рис. Обозначение ассоциированной сущности в нотации Баркера.

Если экземпляр полностью идентифицируется своими ключевыми атрибутами, то говорят о полной идентификации сущности. В противном случае идентификация сущности осуществляется с использованием атрибутов связной сущности, что указывается черточкой на линии связи (рис.).

Идентифицирующая сущность

Идентифицируемая сущность

Рис. Обозначение идентификации посредством другой сущности в нотации Баркера.

**Пример разработки диаграммы «сущность-связь» АИС «Склад оптовой торговли».** Основными сущностями для решения указанной задачи являются: поставщик, покупатель, товар.

Сразу возникает очевидная связь между сущностями – «покупатели могут приобретать много товаров», «товары могут приобретаться многими покупателями». Отношение между ними относится к типу «многие-ко-многим» (рис. *а*). Для разрешения этого отношения введем ассоциированную сущность «Накладная», которая отражает приобретение (продажу) товара покупателем (поставщиком) (рис. *б*).

Проанализируем атрибуты сущностей. Каждый поставщик и покупатель является юридическим лицом и имеет наименование, адрес, банковские реквизиты. Каждый товар имеет наименование, цену, характеризуется единицей измерения. Каждая накладная имеет уникальный номер, дату выписи, список товаров с количеством и ценами, а также общую сумму накладной. Покупатели покупают товары, получая при этом расходные накладные, в которые внесены данные о количестве и цене приобретенного товара. Каждый покупатель может получить несколько накладных. Каждая накладная должна выписываться на одного покупателя. Каждая накладная должна содержать не менее одного товара (не может быть «пустой» накладной). Каждый товар, в свою очередь, может быть продан нескольким покупателям по нескольким накладным. Аналогичную цепь рассуждений можно выстроить для определения связей между сущностями «Товар» и «Поставщик». Покупатель может быть одновременно и поставщиком, поэтому эти две сущности объединены в одну – «Контрагент». Теперь можно все это внести в диаграмму, как показано на рис. *в*.

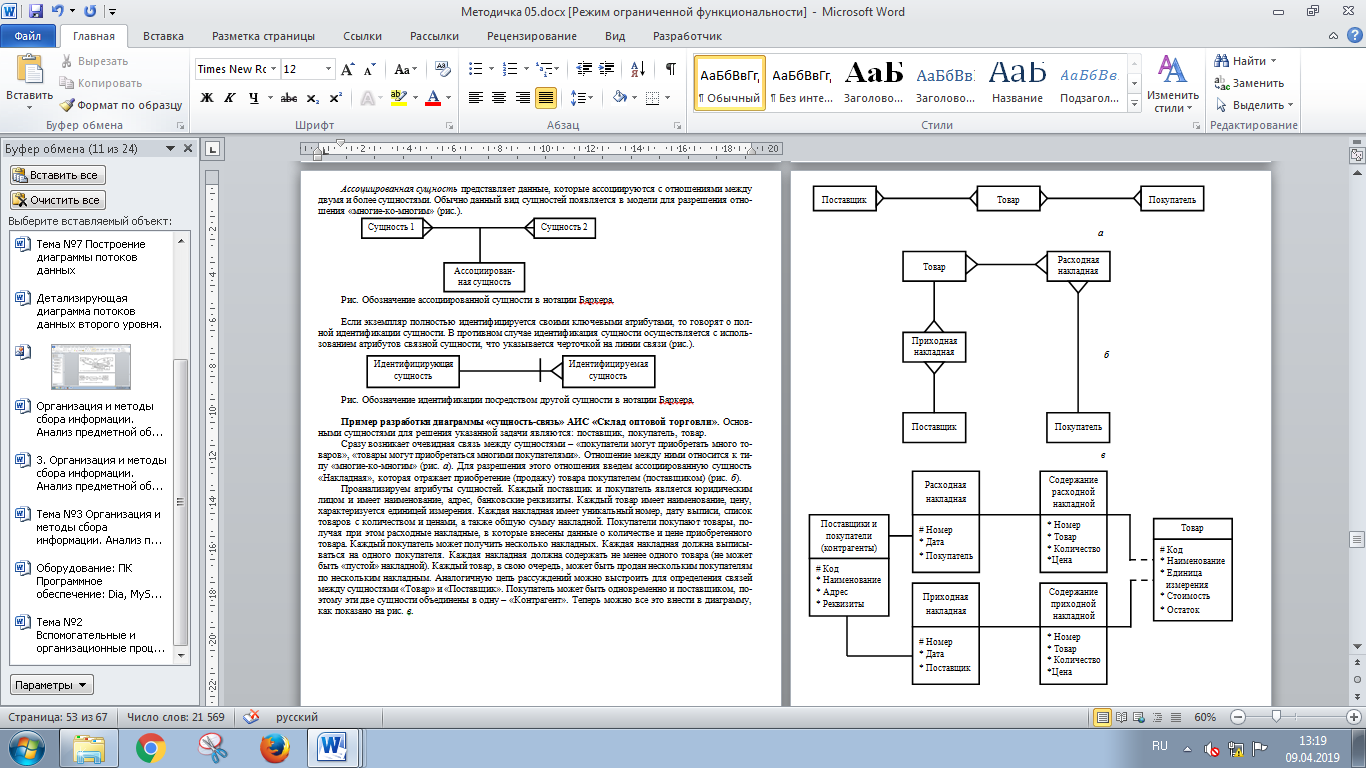


Рис. Варианты ER-диаграммы: *а* – первый; *б* – промежуточный; *в* – окончательный.

Уточненная диаграмма должна быть проверена с точки зрения возможности получения всех выходных данных (отчетов), указанных в техническом задании или показанных на диаграмме потоков данных разрабатываемой системы.

***Задание***

На основании технического задания из практической работы 1 (см. Приложение) выполнить структурный анализ функциональных требований к программному обеспечению и разработать необходимые диаграммы. Оформить результаты, используя MS Visio.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Перечислите принципы создания диаграмм. Объясните назначение каждого вида диаграммы.
2. В чем заключаются основные принципы структурного подхода?
3. Что общего и в чем различие между функциональной моделью SADT и диаграммой потоков данных?
4. Назовите достоинства и недостатки структурного подхода.

**Тема№13**

**Диаграммы классов**

**Цель:** научиться строить диаграмму классов

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Диаграммой UML, поясняющей внутреннее устройство системы, является диаграмма классов, описывающая создаваемую программную систему через ее компоненты (классы, объекты), отношения и взаимодействия между ними. В основном диаграммы классов в этих методах применяют на этапе проектирования, для того чтобы показать особенности построения конкретных классов. UML предлагает использовать три уровня диаграмм классов в зависимости от степени их детализации:

* концептуальный уровень, на котором диаграммы классов, называемые в этом случае контекстными, демонстрируют связи между основными понятиями предметной области;
* уровень спецификаций, на котором диаграммы классов отображают интерфейсы классов предметной области, т. е. связи объектов этих классов;
* уровень реализации, на котором диаграммы классов непосредственно показывают поля и методы конкретных классов.

Это три разные модели, связь между которыми неоднозначна. Так, если концептуальная модель определяет некоторое понятие предметной области как класс, то это не означает, что для реализации этого понятия будет использован отдельный класс. Однако во всех трех моделях нас интересуют типы объектов (классы) и их статические отношения, что позволяет использовать единую нотацию.

Каждая из перечисленных моделей используется на конкретном этапе разработки программного обеспечения:

* концептуальная модель — на этапе анализа;
* диаграммы классов уровня спецификации — на этапе проектирования;
* диаграммы классов уровня реализации — на этапе реализации.

Концептуальные модели в соответствии с определением оперируют понятиями предметной области, атрибутами этих понятий и отношениями между ними. Класс при этом традиционно понимают как совокупность общих признаков некоторой группы объектов предметной области. В соответствии с этим определением на диаграмме классов каждому классу соответствует группа объектов, общие признаки которых и фиксирует класс.

На диаграммах класс изображается в виде прямоугольника, внутри которого указано имя класса (рис.). При необходимости допускается указывать характеристики класса, например, атрибуты, используя специальные секции условного обозначения (рис.), а также операции класса (рис.).

В качестве атрибутов представляют некоторые, существенные с точки зрения решаемой задачи характеристики объектов, например, идентифицирующие значения (имя, номер). Для конкретного объекта атрибут всегда имеет конкретное значение (рис.).

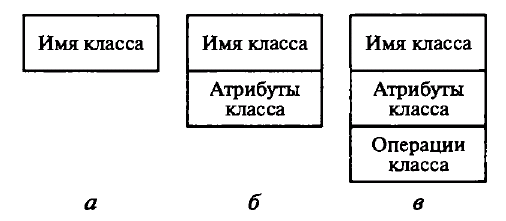


Рис. Обозначение класса на концептуальной диаграмме классов:

а — без уточнения характеристик; б — с уточнением атрибутов;

в — с указанием операций

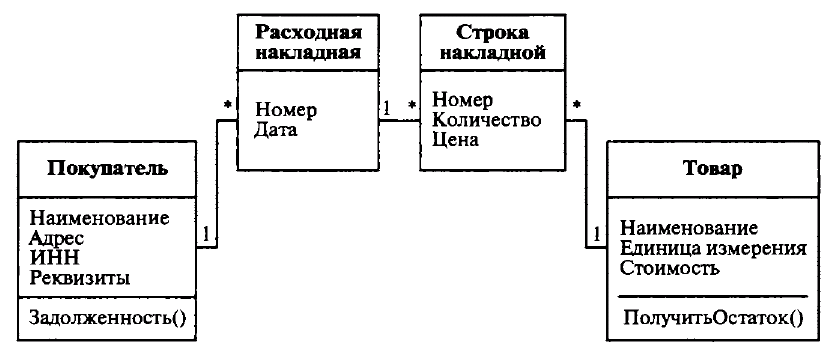


Рис. Пример графического изображения объектов на диаграммах языка UML

Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Оно указывается в первой верхней секции прямоугольника. В дополнение к общему правилу наименования элементов языка UML имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы. Рекомендуется в качестве имен классов использовать существительные, записанные по практическим соображениям без пробелов. Необходимо помнить, что именно имена классов образуют словарь предметной области при объектно-ориентированном анализе и проектировании.

В первой секции обозначения класса могут находиться ссылки на стандартные шаблоны или абстрактные классы, от которых образован данный класс и соответственно от которых он наследует свойства и методы. В этой секции может приводиться информация о разработчике данного класса и статус состояния разработки, а также могут записываться и другие общие свойства этого класса, имеющие отношение к другим классам диаграммы или стандартным элементам языка UML.

Примерами имен классов могут быть такие существительные, как «Сотрудник», «Компания», «Руководитель», «Клиент», «Продавец», «Менеджер», «Офис» и многие другие, имеющие непосредственное отношение к моделируемой предметной области и функциональному назначению проектируемой системы.

Класс может не иметь экземпляров или объектов. В этом случае он называется абстрактным классом, а для обозначения его имени используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом.

В некоторых случаях необходимо явно указать, к какому пакету относится тот или иной класс. Для этой цели используется специальный символ разделитель — двойное двоеточие «∷». Синтаксис строки имени класса в этом случае будет следующий:

<Имя\_пакета>::<Имя\_класса>.

Другими словами, перед именем класса должно быть явно указано имя пакета, к которому его следует отнести. Например, если определен пакет с именем «Банк», то класс «Счет» в этом банке может быть записан в виде: «Банк::Счет».

Во второй сверху секции прямоугольника класса записываются его атрибуты (attributes) или свойства. В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения:

<квантор видимости><имя атрибута> [кратность] :

<тип атрибута> = <исходное значение>{строка-свойство}

Квантор видимости принимает одно из трех возможных значений и отображается при помощи специальных символов:

* «+» — атрибут с областью видимости типа общедоступный (public). Атрибут доступен или виден из любого другого класса пакета, в котором определена диаграмма;
* «#» — атрибут с областью видимости типа защищенный (protected). Атрибут недоступен или невиден для всех классов за исключением подклассов данного класса;
* «-» — атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

Квантор видимости может быть опущен, в этом случае его отсутствие просто означает, что видимость атрибута не указывается. Эта ситуация отличается от принятых по умолчанию соглашений в традиционных языках программирования, когда отсутствие квантора видимости трактуется как public или private. Однако вместо условных графических обозначений можно записывать соответствующее ключевое слово: public, protected, private.

*Имя атрибута* представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и потому должна быть уникальной в пределах данного класса. Имя атрибута является единственным обязательным элементом синтаксического обозначения атрибута.

*Кратность атрибута* характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса. В общем случае кратность записывается в форме строки текста в квадратных скобках после имени соответствующего атрибута. В качестве примера рассмотрим следующие варианты задания кратности атрибутов:

[0..1] означает, что кратность атрибута может принимать значение 0 или 1. При этом 0 означает отсутствие значения для данного атрибута;

[0..\*] означает, что кратность атрибута может принимать любое положительное целое значение большее или равное 0. Эта кратность может быть записана короче в виде простого символа — [\*];

[1.:\*] означает, что кратность атрибута может принимать любое положительное целое значение большее или равное 1;

[1..5] означает, что кратность атрибута может принимать любое значение из чисел: 1, 2, 3, 4, 5;

[ 1..3,5,7] означает, что кратность атрибута может принимать любое значение из чисел: 1, 2, 3, 5, 7;

[1..3,7.. 10] означает, что кратность атрибута может принимать любое значение из чисел: 1, 2, 3, 7, 8, 9, 10;

[ 1..3,7..\*] означает, что кратность атрибута может принимать любое значение из чисел: 1, 2, 3, а также любое положительное целое значение большее или равное 7.

Если кратность атрибута не указана, то по умолчанию принимается ее значение равное 1..1, т. е. в точности 1.

*Тип атрибута* представляет собой выражение, семантика которого определяется языком спецификации соответствующей модели. В нотации UML тип атрибута иногда определяется в зависимости от языка программирования, который предполагается использовать для реализации данной модели. В простейшем случае тип атрибута указывается строкой текста, имеющей осмысленное значение в пределах пакета или модели, к которым относится рассматриваемый класс.

При задании атрибутов могут быть использованы две дополнительные синтаксические конструкции — это подчеркивание строки атрибута и пояснительный текст в фигурных скобках.

Подчеркивание строки атрибута означает, что соответствующий атрибут может принимать подмножество значений из некоторой области значений атрибута, определяемой его типом. Эти значения можно рассматривать как набор однотипных записей или массив, которые в совокупности характеризуют каждый объект класса.

В третьей сверху секции прямоугольника записываются операции, или методы класса. *Операция* (operation) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию. Совокупность операций характеризует функциональный аспект поведения класса. Запись операций класса в языке UML также стандартизована и подчиняется определенным синтаксическим правилам. При этом каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени операции, выражения типа возвращаемого операцией значения и, возможно, строка—свойство данной операции:

<квантор видимости><имя операции> (список параметров) :

<выражение типа возвращаемого значения>{строка-свойство}

Квантор видимости, как и в случае атрибутов класса, может принимать одно из трех возможных значений и соответственно отображается при помощи специального символа. Символ «+» обозначает операцию с областью видимости типа общедоступный (public). Символ «#» обозначает операцию с областью видимости типа защищенный (protected). И наконец, символ «-» используется для обозначения операции с областью видимости типа закрытый (private).

Квантор видимости для операции может быть опущен. В этом случае его отсутствие просто означает, что видимость операции не указывается. Вместо условных графических обозначений также можно записывать соответствующее ключевое слово: public, protected, private. Кроме внутреннего устройства или структуры классов, важную роль при разработке программной системы играют различные отношения между классами, которые также могут быть изображены на диаграмме классов:

* отношение ассоциации — наличие произвольной взаимосвязи между классами;
* отношение обобщения — отношение между более общим элементом и более частным элементом (родителем и потомком);
* отношение композиции (рис. *а*) — частный случай отношения агрегации, когда части не могут выступать в отрыве от целого и с уничтожением целого уничтожаются и все его составные части;
* отношение агрегации (рис. *б*) — один из классов представляет собой некоторую сущность, которая включает в себя в качестве составных частей некоторые другие сущности.

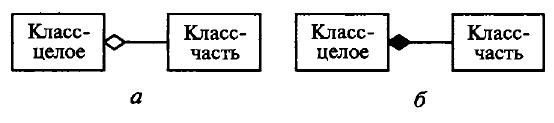


Рис. Условные обозначения специальных видов ассоциации: а — композиция; б — агрегация

В контексте языка UML существует специальный класс — *интерфейс*, у которого имеются только операции и отсутствуют атрибуты. Интерфейсы на диаграмме служат для спецификации таких элементов модели, которые видимы извне, но их внутренняя структура остается скрытой от клиентов.

Процесс разработки диаграммы классов занимает центральное место при разработке проектов сложных систем. От умения правильно выбрать классы и установить между ними взаимосвязи зависит не только успех процессов проектирования, но и производительность программы.

**Задание**

Используя техническое задание провести анализ требований к программному обеспечению в соответствии с вариантом задания, применяя объектно-ориентированный подход и язык визуального моделирования UML. Результаты оформить в MS Visio.

**Тема №14**

**Основные сведения о прототипах. Виды прототипов**

**Цель:** изучить основные понятия о прототипах, их виды

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Для начала определим, для чего используются прототипы программного обеспечения.

Во-первых, пользователю часто бывает трудно сформулировать требования к тому, что он ожидает от системы. В этом случае прототип интерфейса пользователя (User Interface — UI), оперативно созданный по результатам собеседования, дает ему возможность увидеть схематичную реализацию того, как разработчик представляет соответствующую часть системы.

При использовании прототипов важен любой результат. Если программист понял требования заказчика правильно, то очевидно, что разработка программной системы продвигается в верном направлении. Если пользователь не совсем доволен предложенным ему вариантом реализации системы, польза заключается в том, что он может указать, в чем заключается непонимание задачи программиста.

Во-вторых, прототипирование дает возможность выбрать одно из альтернативных концептуальных решений. Любую техническую задачу можно решить различными способами. Это касается как задачи формулировки требований к программной системе, так и реализации ее пользовательского интерфейса.

В-третьих, часто бывает так, что комбинация функциональных и нефункциональных требований такова, что возникает риск невозможности их реализации. Как правило, такой риск связан с требованиями к быстродействию системы при известных ограничениях на среду ее реализации. В этом случае создаются прототипы (не обязательно связанные с пользовательским интерфейсом), реализующие соответствующую часть системы, имитирующие потоки данных, поступающие на ее вход, и обработку этих данных.

**ВИДЫ ПРОТОТИПОВ**

Прототипы могут быть горизонтальные и вертикальные, одноразовые и эволюционные, бумажные и электронные.

*Горизонтальный* (поведенческий) прототип моделирует интерфейс пользователя приложения, не затрагивая логики обработки и структур данных. Горизонтальные прототипы следует использовать, когда необходимо прояснить нечеткие требования, имеющие многоальтернативную реализацию. Для горизонтальных прототипов не обязательно использовать программную среду реализации, в которой будет разрабатываться конечный вариант системы. Если в разработанном интерфейсе используются базы данных, то они имитируются в программном коде; при этом тексты, отображаемые на экране, должны отражать реальную специфику проблемной области, иначе пользователю будет трудно сосредоточиться. При создании прототипа имитируются результаты расчетов и запросов, например, запросы во внешние системы. Желательно реализовать ту часть кода, которая отвечает за перемещение между экранами в процессе исполнения вариантов использования, чтобы пользователь смог понять, как будет действовать система в ответ на его действия. Прежде чем создавать горизонтальный прототип, необходимо определить, какие основные экраны будут присутствовать, какие окна будут открываться, какие правила перехода между ними будут поддерживаться. Для этого хорошо использовать модель диаграммы состояний, где с разными экранами (окнами) сопоставляются состояния, а с активными элементами управления, вызывающими закрытие одних интерфейсных элементов и открытие других, — переходы.

*Вертикальный* (структурный) прототип направлен не столько на проектирование интерфейса пользователя, сколько на реализацию вертикального «среза» системы, затрагивая все уровни ее реализации. При создании такого рода прототипов рекомендуется использовать те же языки и среды реализации, что и при изготовлении целевой системы. Такого рода прототипы используются для анализа применимости, проверки архитектурных концепций.

*Одноразовый* (исследовательский) прототип создается, когда нужно быстро получить макет разрабатываемой программной системы, те или иные ее аспекты и компоненты. Целям создания исследовательских прототипов служит технология RAD (rapid application development) — быстрая разработка приложений. Одноразовый прототип должен создаваться быстро, при его разработке не следует уделять внимание вопросам повторного использования кода, качества, быстродействия, технологичности и т. п. В результате получается «сырой» код, который может содержать значительное число дефектов. Необходимо принять меры к тому, чтобы фрагменты кода, реализующие такого рода прототипы, не стали частью целевой системы.

*Эволюционный* прототип создается как первое приближение системы, призванное стать впоследствии самой системой. Программный код эволюционного прототипа должен последовательно перерасти в код целевого приложения. Поэтому данный вид прототипов требует всего того, от чего следует отказаться при создании одноразовых прототипов: тщательной разработки, применения технологических методов и приемов, тестирования результатов и т.п.

*Бумажный* прототип — наброски интерфейсов на бумаге. Они, конечно, не заменят интерфейса, созданного в среде разработки. Однако при всех недостатках у таких прототипов есть два существенных достоинства. Заказчик не станет акцентировать внимание на цветовом решении, форме кнопок и не будет отвлекаться от анализа функциональности.

Промежуточным решением между *электронным* и бумажным вариантами прототипов UI класса является презентация, созданная при помощи средств электронного офиса (например, комбинации Microsoft Visio и Microsoft PowerPoint).

Представим иллюстрированные сценарии прецедентов. Для того чтобы разработчику лучше понять специфику проблемной области и лучше отразить ее в интерфейсе пользователя, в текст описания сценария варианта использования включают информацию, конкретизирующую те или иные его особенности. Так, информация об объемах используемых данных позволит оптимально построить пользовательский интерфейс и оценить на ранних стадиях проекта «узкие места» в обработке данных, которые могут повлиять на производительность системы. Например, в диалоге с системой при выборе из малого числа вариантов значений лучше подойдут индикаторы (checkbox) или радиокнопки (radiobutton). При выборе, ограниченном 2 — 3 десятками позиций, удобен выпадающий список. В ситуациях, когда приходится выбирать из сотен или тысяч вариантов, потребуется дополнительное окно для фильтрации и поиска.

Также при разработке системы будет полезной информация об интенсивности выполнения тех или иных операций. Эта информация позволит разработать удобный интерфейс, обеспечивающий высокое быстродействие системы за счет минимизации действий при выполнении различных операций, в том числе структуризации подачи информации, удаления из главных интерфейсов редко используемых опций и т. п.

**Тема №15**

**Основные правила и принципы создания интерфейса**

**Цель:** изучить правила и принципы создания интерфейса

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Специалисты обычно формулируют некоторый набор принципов и правил, позволяющих как оценивать удобство интерфейса, так и предлагать решения, повышающие его удобство. Приведем эти правила.

**Правило доступности.** Система должна быть настолько понятной, чтобы пользователь, никогда раньше не видевший ее, но хорошо разбирающийся в предметной области, мог без всякого обучения начать ее использовать. Это правило служит некоторым идеалом, к которому надо стремиться, поскольку на практике достичь такой степени доступности почти никогда не удается. Тем не менее, специалисты продолжают делать в этом направлении все возможное.

**Правило эффективности.** Система не должна препятствовать эффективной работе опытных пользователей, работающих с ней долгое время. Очевидным примером нарушения этого правила является нацеленность системы только на новичков, использование средств, которые хорошо подходят для неопытного пользователя, ограничивая его в возможности сделать что-то не так, но неэффективны для эксперта, который и так знает, что и где ему нужно сделать.

**Правило непрерывного развития.** Система должна способствовать непрерывному росту знаний, умений и навыков пользователя и приспосабливаться к его меняющемуся опыту. Плохие результаты приносит предоставление только базовых возможностей или оставление начинающего пользователя наедине со сложным интерфейсом, которым уверенно пользуются эксперты. Нарушение непрерывности при переходе от одного набора возможностей к другому также приносит неудобства, поскольку пользователь вынужден разбираться с добавленными возможностями в новом контексте.

Большинство пользователей можно поместить в три группы: новички, опытные и средние, которые уже знают больше, чем новички, и не делают столько ошибок, но еще не приобрели автоматизма при выполнении большинства операций и иногда путаются в интерфейсе. Новичкам необходима помощь в освоении новой для них системы и контроль за их действиями, опытным пользователям — высокая эффективность выполнения часто требующихся действий и возможность гибкого управления системой в таких ситуациях, которые встречаются реже, но способны вызвать проблемы при их неадекватной поддержке. Про средних же пользователей часто забывают, хотя подавляющее большинство пользователей программного обеспечения относится именно к этой категории. Им нужны достаточно высокие эффективность и гибкость вместе с возможностью быстро получать адекватную помощь по возникающим время от времени разнообразным вопросам.

**Правило соблюдения контекста.** Система должна быть согласована с контекстом, в котором ей предстоит работать. Это правило требует от системы быть работоспособной не «вообще», а именно в том окружении, в котором ею будут пользоваться.В контекст могут входить специфика и объемы входных и выходных данных, тип и цели организаций, в которых система должна работать, уровень пользователей, зашумленность помещений и пр.

Представленные выше правила определяют общие требования, которым должен удовлетворять удобный интерфейс. Следующие принципы позволяют находить решения, повышающие удобство пользовательского интерфейса.

**Принцип структуризации.** Пользовательский интерфейс должен быть целесообразно структурирован. Близкие по смыслу, родственные его части должны быть связаны видимым образом, а независимые — разделены; похожие элементы должны выглядеть похоже, а непохожие — различаться.

**Принцип простоты.** Наиболее распространенные операции должны выполняться максимально просто. При этом должны быть видимые ссылки на более сложные процедуры.

**Принцип видимости.** Все функции и данные, необходимые для решения определенной задачи, должны быть видны, когда пользователь пытается ее решить.

**Принцип обратной связи.** Пользователь должен получать сообщения о действиях системы и о важных событиях внутри нее. Сообщения должны быть информативными, краткими, однозначными и написанными на языке, понятном пользователю.

**Принцип толерантности.** Интерфейс должен быть гибким и терпимым к ошибкам пользователя. Ущерб от ошибок должен снижаться за счет возможности отмены и повтора действий и за счет разумной интерпретации любых разумных действий пользователя и введенных им данных. По возможности следует избегать обязывающего взаимодействия (модальных диалогов), основанного на ограничении свободы пользователя.

**Принцип повторного использования.** Следует стараться многократно использовать внутренние и внешние компоненты, обеспечивая тем самым унифицированность интерфейса и сходство между его похожими элементами.

Рассмотрим некоторые вопросы разработки эргономичного интерфейса пользователя.

**Тема №16**

**Оценка эффективности функционирования информационных систем**

**Цель:** научиться оценивать эффективность функционирования информационных систем

**Оборудование:** ПК

**Программное обеспечение:** Dia, MySQL, NetBeans, AndroidStudio

Оценка экономического эффекта от внедрения автоматизированной информационной системы требует учета ряда особенностей. Во-первых, нежно оценить экономический эффект от внедрения новой системы или от модернизации уже существующей системы. Во-вторых, работа информационной системы во многом зависит от используемых вычислительных средств и человеческого фактора. Данное положение требует ввода критериев, позволяющих оценить эффективность работы вычислительной подсистемы и качества работы оператора – пользователя системы.

**Экономическая эффективность.** Внедрение или использование информационной системы требует оценки ее работы с точки зрения либо абсолютной, либо относительной эффективности.

*Абсолютная экономическая эффективность* рассчитывается для вновь созданной системы, так как заказчик получает определенную систему, созданную по его требованиям.

В общем виде расчет абсолютной эффективности сводится к вычислениям по формуле

Э = П2 – П1 – З,

где П1 – прибыль, получаемая организацией без использования информационной системы; П2 - прибыль, полученная при использовании информационной системы; З – затраты, которые несет организация для внедрения системы.

Затраты на внедрение информационной системы включают в себя средства, необходимые на приобретение вычислительной техники, развертывание сети персональных ЭВМ, обучение персонала. Сюда же следует включить средства, необходимые для покупки программного обеспечения, основным компонентом которого является операционная система. В расходы следует включать также затраты на электроэнергию.

Разница определяет полученный эффект от основной деятельности предприятия после проведения автоматизации:

Пд = П2 – П1.

Сделать вывод о целесообразности внедрения информационной системы можно только в том случае, если Э>0.

Для оценки степени эффективности от использования системы нужно рассчитать *относительную экономическую эффективность* по формуле

Показатель определяет экономический эффект на 1 р. Вложенных приведенных затрат. Рекомендуется, чтобы данный коэффициент лежал в пределах от 2,0 до 2,5. Значение в пределах от 1,0 до 2,0 говорит об успехе, достигнутом при локальной автоматизации деятельности организации. Неудовлетворительный результат автоматизации признается при значениях от 0 до 1,0.

В случае если организация производит модернизация уже существующей информационной системой, то при расчете эффективности учитывают текущие затраты на ее эксплуатацию

Э = (П2 – П1) – (З2 – З1),

где З2 – затраты на эксплуатацию новой системы; З1 – затраты на эксплуатацию старой системы.

Для более точной оценки полученного эффекта от автоматизации используются приведенные оценки. Расчеты экономической эффективности проводят на конкретной период времени. Для учета процесса инфляции в расчетах применяют дисконтные коэффициенты. Если норма дисконтирования постоянна в течении срока реализации поставки информационной системы, то поправочный коэффициент для расчета эффективности примет вид

К = (1 + n)t,

Где n – коэффициент дисконтирования; t – период времени, для которого производится расчет.

Затраты и полученные предварительные результаты должны быть поделены на этот коэффициент.

Для учета фактора неопределенности при расчете экономический эффективности используют принцип минимакса и метод интервальной оценки результатов.

Экономическую эффективность от внедрения информационной системы определяют при максимальных затратах, получая минимальный результат. При этом задается конкретный интервал, в котором должна попасть расчетная величина эффективности. Если результат отрицательный, то либо интервал выбран неверно, либо использование информационной системы в организации нерентабельно.

При сравнении альтернативных вариантов автоматизации в случае фиксированного бюджета на заказ информационной системы из нескольких вариантов автоматизации выбирается тот, который отвечает принципу максимина. Выбирается тот вариант, который позволяет получить максимальный эффект при минимальных затратах или минимальной степени риска.

При возможности выделить на автоматизацию нефиксированные средства, нужно выбрать определенного подрядчика и поставщика оборудования. Для выбора оптимального проекта анализируют разницу между затратами и значением относительной экономичной эффективности.

**Производительность вычислительной системы.** Для оценки эффективности функционирования информационной системы с точки зрения используемых в ней вычислительных ресурсов применяют методику оценки производительности, основанную на теории операций.

При оценки системы вводят следующие параметры: Т – период наблюдения за вычислительной подсистемой; Т3 – время обработки подсистемы одной команды оператора; Коб – число обработанных команд.

После измерения и фиксирования этих параметров определяют следующие показатели:

* Коэффициент использования системы
* Среднее время обработки одной команды оператора
* Интенсивность выходного потока заданий

В случае если система обладает вычислительной подсистемой из нескольких процессоров, то оценивают показатели для каждого процессора с учетом кооперации работы процессоров при выполнении команд оператора.

Число обработанных заданий может определяться как для одного процессора, так и для нескольких процессоров, работающих в кооперации. Соответственно условные обозначения параметров наблюдения примут следующий вид Коб(i) – для i-го процессора Коб(I,j) – число обработанных заданий для пары процессоров, работающих в кооперации. Время отработки может фиксироваться как для одного процессора, так и для нескольких процессоров, обрабатывающих одну команду.

**Оценка качества работы пользователя системы.** Чтобы оценить эффективность работы пользователя с системой, используют следующие показатели качества [16]: быстродействие; напряженность функционирования оператора, экономическая эффективность, надежность оператора.

Для оценки реакции оператора – *быстродействия* при работе с системой вводят следующую зависимость:

Где Тц – время рабочего цикла; Топ – время выполнения операции в составе работы; Т3 – время запаздывания, т.е. время между окончание текущей операции и началом следующей.

Время выполнения операции складывается из следующих составляющих:

Топ = Тож + Тр

Где Тож – время ожидания выполнения действия; Тр – рабочее время выполнения операции.

Для определения рабочего времени выполнения операции используют следующую зависимость:

Коэффициент *а* принимается равным 0,2 с; он характеризует запаздывание при выполнении операции в системе. Коэффициент *R*0 показывает производительность оператора системы; он принимается равным 0,15 … 0,35 с/бит.

Оценку *напряженности функционирования оператора* выполняют по формуле:

где yi, yimax – физиологические количественные показатели напряженности работы оператора в реальных и экстремальных условиях при выполнении i-й операции.

*Экономическая эффективность* при использовании человека в качестве оператора системы определяется как отношение количественного результата его работы к затратам организации на обучение оператора и повышение его квалификации.

*Надежность оператора* – это такое его свойство, которое позволяет оценивать возможность выполнения им его функций в течении определенного промежутка времени.

В процессе работы оператора возникают отказы, т.е. нарушения в его работе. Отказы в общем случае носят случайный характер и подразделяются на устойчивые и временные. *Устойчивые отказы -* биологические, *временные –* психологические. Биологические отказы вызваны переутомлением, заболеванием оператора, психологические отказы носят случайный характер. Психологические отказы могут быть вызваны невнимательностью оператора, отвлечением от выполнения своей основной функции в системе.

Для определения вероятности безотказной работы оператора используют следующие зависимости:

Pоп(t) = KопPб(t) + Pп(t)

Kоп = 1- To/T

где Kоп – коэффициент готовности оператора; Pб – вероятность биологической надежности; Pп – вероятность психологической надежности; Tо – время отсутствия оператора на рабочем месте; T – общее время работы оператора.

Вероятность безотказной работы АИС с учетом технических средств и человеческого фактора определяются по формуле

P(t) = Pтехн(t)Pоп(t)Pсв(t)

где Pтехн – вероятность безотказной работы технических средств системы (ЭВМ и периферийного оборудования); Pсв – вероятность своевременного выполнения работ.

Анализ данных показателей позволяет принять решения, необходимые для обеспечения определенного уровня подготовки операторов и пользователей системы: подбора аппаратного обеспечения информационной системы в соответствии в требованиями инженерной психологии; контроля действий операторов системы и ее пользователей; организации отдыха операторов системы и профилактики нарушений в их работе; снятия или сведения к минимуму информационных перегрузок операторов.