Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

УТВЕРЖДАЮ
Заведующий кафедрой ИСПИ
_____ И.Е. Жигалов
«20» марта 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ МЕЖДИСЦИПЛИНАРНОГО КУРСА

«ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ РЕСУРСОВ»

В РАМКАХ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

«ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ РЕСУРСОВ»

09.02.09 Веб-разработка
Разработчик веб приложений

Владимир, 2025

Методические указания к лабораторным работам междисциплинарного курса «Проектирование информационных ресурсов» разработал старший преподаватель кафедры ИСПИ Данилов В.В.

Методические указания к лабораторным работам рассмотрены и одобрены на заседании УМК специальности 09.02.09 Веб-разработка
протокол № 1 от «10» марта 2025 г.

Председатель УМК специальности _____ И.Е. Жигалов

Методические указания к лабораторным работам рассмотрены и одобрены на заседании кафедры ИСПИ
протокол № 7а от «12» марта 2025 г.

Рецензент от работодателя:
руководитель группы обеспечения
качества программного обеспечения
ООО «БСЦ МСК» _____ С.С. Смирнова

Оглавление

Лабораторная работа №1

Анализ предметной области

Цель работы: изучить процесс исследования предметной области и получить практические навыки анализа предметной области и постановки задачи для разработки информационного ресурса.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Предметная область

Множество всех предметов или объектов некоторой части реального мира, свойства которых и отношения, между которыми рассматриваются в научной теории или в практической деятельности человека. Для сбора, хранения, поиска и выдачи информации о предметной области и ее объектов в настоящее время в информационных системах широко используются базы данных.

Анализ предметной области

Анализ предметной области начинается с выделения сущностей и определения их свойств или атрибутов. Видимые сущности представляют собой объекты предметной области, которые может распознать человек. Поддерживаемые сущности или абстрактные сущности разрабатываются проектировщиком базы данных для физической поддержки общей логической модели.

Описание предметной области.

Определение гласит: «Под предметной областью (application domain) принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи». Следовательно, разработчикам необходимо выделить основные объекты (компоненты), участвующие в функционировании системы, определить их наиболее существенные характеристики, взаимосвязи в рамках решаемой задачи, а также определить основные информационные потоки в системе. При этом отдельные компоненты выбираются таким образом, чтобы при последующей разработке их было удобно представить в форме классов и объектов. В этом случае немаловажное значение

приобретает и сам язык представления информации о концептуальной схеме предметной области.

Сложность предметной области определяет количество объектов и связей между ними, поэтому описание должно включать в себя базовые термины и определения, сопровождаться различными примерами, в нем могут приводиться различного рода классификации, поясняющие различные свойства описываемых объектов. Если в системе используются математические модели, то они также должны быть описаны с учетом специфики применения.

При необходимости приводятся различного рода классификации, используемые математические модели, базовые термины и определения, делается обзор существующих систем-аналогов с указанием их отличительных особенностей, достоинств и недостатков (приводятся экранные формы этих систем). Описание предметной области позволяет разработчику определить основные концепции, которые необходимо будет реализовать в системе. Результатом данного этапа является диаграмма объектов предметной области и краткое описание их свойств и функций.

Пример анализа предметной области

Далее приведен пример анализа предметной области, который был проведен при разработки системы технической поддержки для кафедры Информационные системы и программная инженерия.

Кафедра Информационные системы и программная инженерия – это подразделение Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых, отвечающее за обучение студентов по направлениям «Информационные системы и технологии» и «Программная инженерия» в бакалавриате и магистратуре, а также за аспирантуру для выпускников этих направлений. Кафедрой руководит Заведующий кафедрой, который имеет следующие функции:

1. Управление кафедрой;
2. Преподавание дисциплин.

Профессорско-преподавательский состав имеет следующие функции:

1. Преподавание дисциплин;
2. Оформление документов для работы кафедры.

Учебно-вспомогательный персонал имеет следующие функции:

1. Ремонт аппаратного и программного обеспечения;
2. Установка новых аппаратного и программного обеспечений;

3. Оформление документов для профессорско-преподавательского состава и заведующего кафедрой.

Профессорско-преподавательскому составу во время преподавания может потребоваться ремонт аппаратного обеспечения или восстановление работоспособности программного обеспечения, а также установка новых программных продуктов и новых компьютеров в аудитории (например, на замену старых, которые пойдут на списание). Для этого они обращаются лично к техникам.

Процесс обучения кафедры зависит от расписания, поэтому необходимо также учитывать расписание для проведения процессов ремонта и установки программного обеспечения в компьютерных классах в свободное от проведения занятий время. Компьютерных аудиторий, закрепленных за кафедрой Информационные системы и программная инженерия – пять, но конкретно под обслуживанием техников кафедры – четыре. Также есть еще лекционная аудитория, где находится один компьютер преподавателя, и три служебных помещения, в которых также имеется техническое обеспечение.

На рисунке 1.1 представлена структура аудиторий кафедры Информационные системы и программная инженерия.

Рисунок.1.1. Структура аудиторий кафедры Информационные системы и программная инженерия.

На компьютерах в аудиториях стоит программное обеспечение, необходимое для работы преподавателей и студентов. Оно разделяется на два вида: платное и бесплатное. Платное программное обеспечение закупается на средства университета и имеет свои лицензионные ключи на каждый компьютер. На компьютерах кафедры стоит следующее платное программное обеспечение: Microsoft Windows, Microsoft Office, Microsoft Visual Studio, Matlab, IBM SPSS Statistics, AnyLogic, Mathcad.

Бесплатное программное обеспечение имеет разные типы лицензий, но все они сводятся к одному – для учебных целей данные программы бесплатны, а многие и полностью бесплатные. На компьютерах кафедры стоит следующее бесплатное программное обеспечение: 1C:Enterprise 8 (training version), Android Studio, GPSS World Student Version, Lazarus 1.6, Denwer, NetBeans IDE 8.2, PascalABC.NET, Oracle VM VirtualBox 5.1.6, Wing IDE Personal 6.0.4-1, КуМир.

Лицензионные программы для подтверждения своих лицензий должны иметь доступ к сети Интернет хотя бы на момент активации. На рисунке 1.2 представлена структура сети кафедры Информационные системы и программная инженерия.

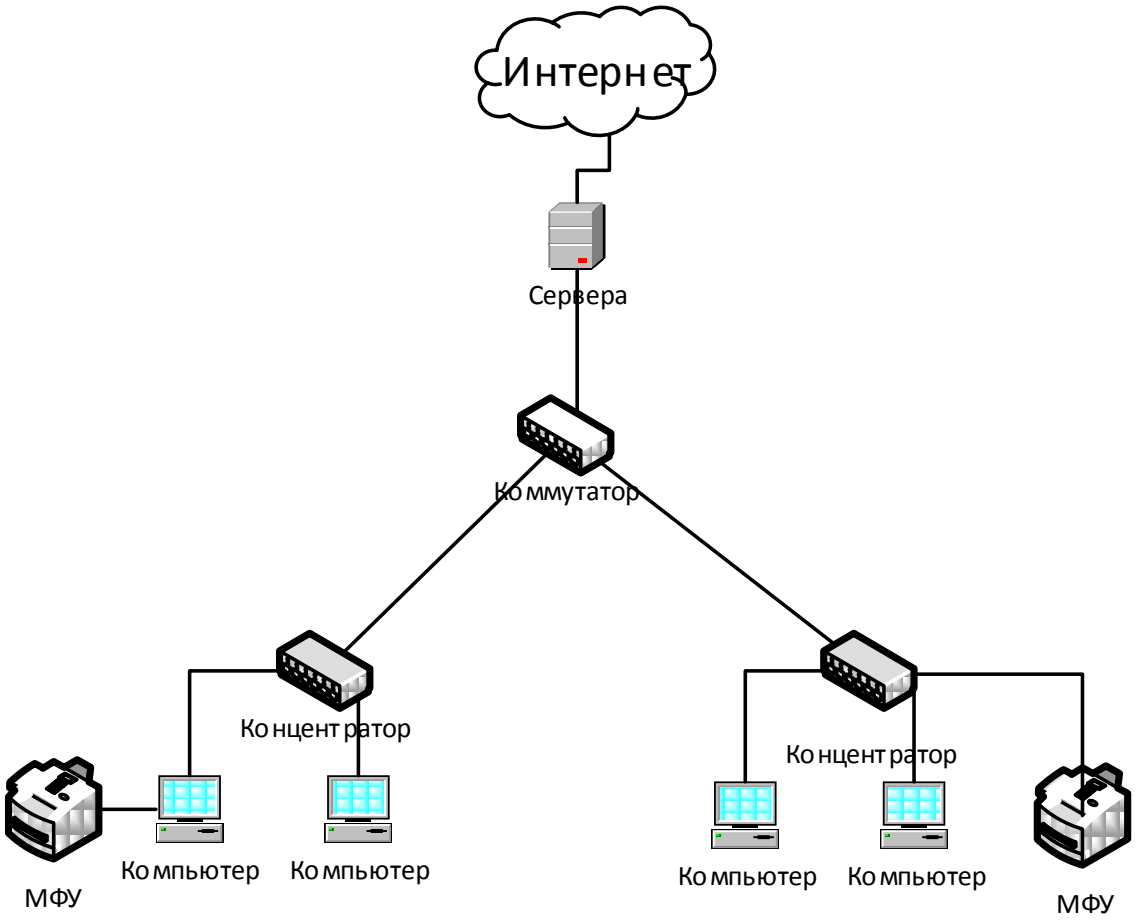


Рисунок.1.2. Структура сети кафедры Информационные системы и программная инженерия

Далее разберем как происходят процессы в настоящий момент.

Оформление заявки на неполадку – процесс важный, так как это постановка неполадки компьютера на учет инженеру кафедры. Этот процесс может занимать время, до получаса, так как не всегда получается быстро передать информацию. Также имеет

место быть человеческий фактор в виде забывания информации о неполадке, что приводит к увеличению срока устранения неполадки. (далее должны идти диаграммы и описание других процессов).

Запрос отчета о состоянии аппаратного обеспечения – процесс важный, так как от него зависит составление отчетов, требующих информации об компьютерах в аудиториях. Этот процесс занимает много времени, до нескольких часов, так как в настоящий момент информация нигде не учитывается.

Все остальные процессы примерно по таким же схемам, как и приведенные выше.

Предполагается, что при использовании разработанного информационного ресурса будет исключен из процессов человеческий фактор. А также использование разработки позволит сократить количество времени за счет более быстрого доступа к информации, которая будет храниться в централизованной базе данных, а не где-то у техников (инженеров) кафедры Информационные системы и программной инженерии

Разработаем примерный сценарий работы с предполагаемой системой. Пользователь заходит в систему под своим логином и паролем. После этого в зависимости от роли ему доступны переходы к созданию заявки (преподаватель), просмотру состояния заявки (преподаватель, инженер), просмотру аппаратного и программного обеспечения в аудиториях (преподаватель, инженер), изменению статуса заявки (инженер), работе с аппаратным и программным обеспечением (инженер). Также имеется роль администратора, которому доступны кроме вышеперечисленных действий работа с пользователями и аудиториями. Вся работа происходит через заполнение форм, содержащих при необходимости информацию.

При формулировании требований описывается также решение проблем, вызвавших необходимость системы.

Решение проблемы процесса «Оформление заявки на неполадку» заключается в использовании приложения для оформления заявки. Это ускоряет процесс до нескольких минут, так как теперь получить информацию гораздо проще, и она не забудется, так как хранится в базе данных.

Решение проблемы процесса «Запрос отчета о состоянии АО» заключается в использовании приложения для получения отчета. Это ускоряет процесс до пары минут, так как теперь получить информацию гораздо быстрее вследствие ее хранения в базе данных.

Определим требования к приложению. Они разделяются на две группы.

Пример списка функциональных требований:

1) Авторизация и аутентификация;
2) Выйти из системы;
3) Добавление пользователя;
4) Редактирование пользователя;
5) Удаление пользователя;
6) Создание аудитории в базе данных;
7) Редактирование аудитории;
8) Удаление аудитории
9) Создание корпуса в базе данных;
10) Редактирование корпуса;
11) Удаление корпуса
12) Создание заявки;
13) Удаление заявки;
14) Редактирование заявки;
15) Изменение статуса заявки;
16) Просмотр заявки;
17) Создание отчета по выполненной заявке;
18) Редактирование отчета по выполненной заявке;
19) Получение отчета о компьютерах кафедры;
20) Добавить компьютер в аудиторию;
21) Отредактировать информацию о компьютере в аудитории;
22) Удалить компьютер из аудитории;
23) Просмотреть список типов компьютеров;
24) Создать тип компьютера;
25) Редактировать типовую конфигурацию компьютера;
26) Удалить типовую конфигурацию компьютера;

Пример списка нефункциональных требований:

1) Приложение должно иметь клиент-серверную архитектуру;
2) Поддержка одновременной работы не менее 10 пользователей;
3) Отклик системы при работе каждого пользователя не должен превышать 5 с. (в случае формирования отчетов до 10 с.);
4) Уровень надежности не менее 0,95.
5) Уровень достоверности не менее 0,99

Задание для выполнения:

1. Выбрать предметную область из Приложения 1.1;
2. Обсудить и утвердить выбранную предметную область у преподавателя

2. Провести анализ предметной области с целью выявления сущностей и бизнес-процессов;

3. Продумать подробное описание работы данной предметной области с внедрением специального программного решения, решающего проблемы или улучшающего процессы.

4. Определить перечень функций, которые будут реализованы в данном программном решении.

4. Оформить отчет.

Содержание отчета:

1. Титульный лист;

2. Цель работы;

3. Ход работы (описание предметной области, список функциональных требований);

4. Выводы по работе.

Контрольные вопросы:

1) Для чего необходимо проводить анализ предметной области?

2) Для чего необходим список функциональных требований?

Приложение 1.1. Темы предметных областей

Темы предметных областей:

1) Студия анимации;

2) Склад;

3) Книжный магазин;

4) Салон красоты;

5) Ресторан;

6) Банк;

7) Автосалон;

8) Авиакомпания;

9) Развлекательный центр;

10) Библиотека;

11) Магазин электроники;

12) Зоомагазин;

13) Отель;

14) Сервисный центр автомобилей;

15) Фотоателье;

16) Служба доставки;

17) Ферма;

18) Фитнес-центр;

19) Интернет-провайдер;

20) Туристическое агентство;

21) Аптека;

22) Ветеринарная клиника;

23) Стоматология;

24) Студия танцев;

25) Автомобильная школа;

26) Интернет-магазин канцтоваров;

27) Интернет-магазин парфюмерии;

28) Интернет-магазин мебели;

29) Интернет-магазин одежды;

30) Интернет-магазин обуви;

31) Интернет-магазин бытовой техники.

Также можно выбрать свою тему, согласовав ее с преподавателем.

Лабораторная работа №2

Сравнительный анализ

Цель работы: изучить методику и получить практические навыки проведения сравнительного анализа программных продуктов.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Сравнительный анализ является хорошей практикой для понимания того, как необходимо реализовать программный продукт в определенной сфере деятельности и что ожидают пользователи (целевая аудитория разрабатываемого программного продукта) от нашего информационного ресурса после его выхода.

Для начала необходимо определить программные продукты, которые являются аналогами разрабатываемого информационного ресурса или же имеют некоторый функционал, который будет реализован. Для эффективного сбора данных во время анализа рекомендуется использовать от пяти до семи продуктов. Чем крупнее проект, тем больше программных решений необходимо для проведения сравнения.

После выбора аналогов необходимо выбрать критерии, по которым будет проводиться сравнение с конкурентом. Это могут быть как конкретные функции, так и способ их реализации. Необходимо отметить сильные и слабые стороны продукта с точки зрения UX. В будущем, при проектировании проекта, это поможет избежать ошибок ваших конкурентов, тем самым даст возможность улучшить опыт взаимодействия пользователя с разрабатываемым информационным ресурсом.

Пример сравнительного анализа.

Сначала проработаем критерии оценки программных продуктов. Они приведены в таблице 2.1.

Таблица 2.1. Критерии оценки программных продуктов

№	Признак	Описание
1	Описание КИС	Критерии, с помощью которых осуществляется оценка самой КИС
1.1	Структура	Критерии оценки структуры КИС
1.2	Функционал	Критерии оценки функциональных возможностей КИС
1.3	Принципы	Критерии оценки принципов построения КИС
1.4	Технические требования	Критерии оценки технических требований функционирования КИС

1.5	Архитектура	Критерии оценки особенностей архитектуры, заложенной при создании КИС
1.6	Стоимость	Критерии оценки стоимостных параметров КИС
1.7	Интеграция	Критерии оценки внутренней и внешней интеграции КИС
1.8	Бизнес-логика	Критерии оценки реализации бизнес-логики, заложенной в КИС
1.9	Элементы КИС	Критерии оценки элементов КИС
2	Описание фирмы-разработчика и ее партнеров	Критерии оценки фирмы-разработчика КИС и фирм-партнеров по продвижению и внедрению данной КИС
2.1	Технологии	Критерии оценки технологий, подходов, методов, применяемых фирмами-внедренцами в процессе внедрения КИС
2.2	Опыт	Критерии оценки опыта успешных и неудачных проектов фирм-внедренцев
2.3	Специалисты	Критерии оценки квалификационного уровня специалистов фирм-внедренцев
3	Принципиальность	Отношение критериев выбора КИС к соответствующим этапам выбора КИС
3.1	Принципиальные	Критерии, которые должны быть оценены в первую очередь и, которые определяют основные принципы выбираемой КИС
3.2	Не принципиальные	Критерии, которые не являются сильно принципиальными при выборе КИС
4	По степени детализации	На сколько критерий конкретно описывает исследуемый объект
4.1	Общие	Критерии, описывающие объект исследования в общем виде
4.2	Конкретные	Критерии, конкретно описывающие объект исследования
5	По сложности оценки	Доступность и достоверность информации для самостоятельной оценки
5.1	Сложный	Возможность самостоятельного получения полной и достоверной информации по данному критерию очень мала
5.2	Средней сложности	Возможно самостоятельное получение полной и достоверной информации
5.3	Легкий	Возможно самостоятельное получение полной и достоверной информации. Информация в общедоступных источниках
6	По типу значения	В зависимости от типа значения критерия: возможность количественного измерения значения либо качественный показатель
6.1	Количественный	Критерии, значения которых могут быть определены в виде

		конкретных числовых показателей
6.2	Качественный	Критерии, значения которых не могут быть определены в виде конкретных числовых показателей
7	По важности для потенциальных пользователей	Описывает степень важности того или иного критерия выбора КИС для потенциальных пользователей
7.1	Высший приоритет	Критерии, имеющие наибольшую значимость для пользователя
7.2	Средний приоритет	Критерии, имеющие среднюю значимость для пользователя
7.3	Низший приоритет	Критерии, имеющие низшую значимость для пользователя

Группы критериев оценки программных продуктов:

1. назначение и возможности пакета (область использования, степень обеспечения функций, общего назначения или специализированный);

2. отличительные признаки и свойства пакета (входной язык, структура массивов данных, способы проверки данных);

3. требования к техническим и программным средствам (объем ОП, периферийные устройства, тип ОС);

4. документация пакета (наличие руководства по использованию, руководства программиста, руководства системного программиста);

5. факторы финансового порядка (затраты на приобретение, необходимость ежегодных платежей);

6. особенности установки пакета (объем работ, время установки, требования к квалификации программистов);

7. особенности эксплуатации пакета (надежность, защита данных, возможность эксплуатации силами предприятия);

8. помощь поставщика по внедрению и поддержанию пакета (обучение персонала, внесение модификаций, обновление версий);

9. оценка качества пакета и опыт его использования (число внедрений пакета, оценки пользователей, номер версии);

10. перспективы развития пакета (совместимость версий, дополнение функциональных возможностей, развитие методов).

Пример оценки программного продукта

Для выбора программного продукта, наилучшим образом удовлетворяющего потребности Предприятия, консультанты провели анализ программных продуктов в соответствии с системой требований.

При этом использовались методики оценки, приведенные ниже.

Оценка существующей функциональности программного продукта

При анализе тиражируемых программных продуктов, предполагаемых к внедрению как основы ИСУ Предприятия, функциональные возможности программных продуктов оценивались по степени их соответствия разработанным требованиям по десятибалльной шкале.

При оценке применяется следующая шкала баллов:

0 - функция отсутствует в имеющейся конфигурации;

2 - функция реализована частично, для её реализации необходима серьезная доработка программного кола при настройке/внедрении;

4 - функция реализована частично, для её реализации необходима незначительная доработка программного кода при настройке/внедрении;

6 - функция реализована удовлетворительно, требуется адаптация под нужды Предприятия в процессе настройки/внедрения средствами ИСУ;

8 - функция реализована хорошо, однако в перспективе могут понадобиться её доработки:

10 - функция реализована полностью, удовлетворяет требованиям (в том числе - на перспективу).

Оценки "по умолчанию" выстроены по шкале четности, при заполнении теста могут применяться нечетные оценки в случае, если ответ находится на грани двух смежных четных оценок.

Оценка прочих аспектов

Оценка соответствия прочих аспектов тиражируемых программных продуктов и Поставщиков разработанным требованиям производится также по десятибалльной шкале. Количество баллов определяет степень соответствия программного продукта рассматриваемому требованию.

Система весовых коэффициентов

Для получения интегральной оценки программных продуктов и поставщиков введены весовые коэффициенты для определения значимости тех или иных критериев для Предприятия.

Используется следующая система весовых коэффициентов:

0 - функция отсутствует в имеющейся конфигурации;

2 - функция реализована частично, для её реализации необходима серьезная доработка программного кола при настройке/внедрении;

4 - функция реализована частично, для её реализации необходима незначительная доработка программного кода при настройке/внедрении;

6 - функция реализована удовлетворительно, требуется адаптация под нужды Предприятия в процессе настройки/внедрения средствами ИСУ;

8 - функция реализована хорошо, однако в перспективе могут понадобиться её доработки:

10 - функция реализована полностью, удовлетворяет требованиям (в том числе - на перспективу).

Предпочтение должно отдаваться программным продуктам, имеющим наибольший рейтинг (суммарную оценку с учетом весовых коэффициентов).

Основные выводы по результатам анализа программных продуктов

В соответствии с Техническим заданием консультантами были проанализированы следующие программные продукты:

• «Рос-1» версия 5.8;

• «Рос-2» версия 2.3.3;

• ««Рос-3»: Предприятие» версия 7.0;

• «Зап-1»;

• «Зап-2» версия 2.6.

Основные результаты анализа приведены в следующей таблице.

Таблица 2.2. Основные результаты анализа программных продуктов

№	**Наименование критерия**	**«Рос-1»**	**«Рос-2»**	**«Рос-3»**	**«Зап-1»**	**«Зап-2»**	**Макс. балл**
Общесистемные функциональные требования		**1105**	**1056**	**1128**	**951**	**1126**	**1680**
Функциональные требования по подсистемам управления		**5452**	**3431**	**1134**	**3486**	**3385**	**8310**
1.	Маркетинг	400	146	262	112	110	**610**
2.	Сбыт	328	284	262	300	300	**420**
3.	Производство	1410	684	420	810	918	**2070**
4.	Снабжение	327	183	210	372	381	**720**
5.	Управление складами	222	174	72	222	186	**300**

6.	Управление персоналом	555	402	552	90	24	**900**
7.	Управление транспортом	168	18	114	24	0	**210**
8.	Управление строительством	36	36	24	12	18	**120**
9.	Взаиморасчеты	490	340	278	406	342	**660**
10.	Финансы	524	438	286	420	396	**940**
11.	Управление себестоимостью	138	102	96	78	84	**210**
12.	Бухгалтерский и налоговый учет	854	624	758	640	626	**1150**
Прочие требования		**564**	**460**	**525**	**362**	**434**	**730**
Итого:		**7121**	**4947**	**4987**	**4799**	**4945**	**10720**

Анализ программных продуктов проведен в соответствии с описанной выше методикой.

В ходе анализа консультанты исходили из того, что наиболее важными для Предприятия являются те функции программных продуктов, которые дают Предприятию следующие возможности:

• планирование и контроль фактического выполнения работ (входит в подсистему управления "Производство" в табл. 2);

• управление движением товарно-материальных ценностей (подсистемы "Снабжение" и "Управление складами");

• планирование и контроль финансовых потоков, контроль задолженностей и взаиморасчетов (подсистемы управления "Финансы" и "Взаиморасчеты");

• бухгалтерский и налоговый учет (подсистема управления "Бухгалтерский и налоговый учет");

• управление персоналом, включая кадровый учет и расчет заработной платы (подсистема "Управление персоналом").

Анализ указанных выше программных продуктов показал следующее.

Из рассмотренных консультантами программных продуктов наилучшими функциональными характеристиками обладает система "Рос-1". Основные причины этого следующие:

• только два программных продукта ("Рос-1" и "Зап-2") обладают возможностями планирования и контроля фактического исполнения работ (подсистема "Производство");

• по подсистеме "Снабжение" наилучшей функциональностью обладают программные продукты "Рос-1", "Зап-1" и "Зап-2";

• по подсистеме "Управление складами" лучше всего удовлетворяют выдвинутым требованиям программные продукты "Рос-1" и "Зап-1";

• в подсистеме "Управление персоналом" наиболее развитую функциональность имеют программные продукты "Рос-1", "Рос-2" и "Рос- 3";

• в подсистемах "Финансы", "Взаиморасчеты" и "Бухгалтерский и налоговый учет" функциональные возможности "Рос-1" заметно шире, чем у других программных продуктов.

Таким образом, с точки зрения функциональности наиболее приемлемым программным продуктом для Предприятия является система "Рос-1".

Задание для выполнения:

1. Изучить методику сравнительного анализа;

2. Провести сравнительный анализ следующих групп программных продуктов (не менее 5 продуктов в каждой группе):

1) Браузеры;

2) Текстовые редакторы для веб-разработки;

3) Офисные пакеты;

3. Подвести итоги сравнительного анализа каждой группы;

4. Оформить отчет.

Содержание отчета:

1. Титульный лист;

2. Цель работы;

3. Ход работы (описание программных продуктов каждой группы, таблицы с оценками по каждой группе, итоги по таблицам);

4. Выводы по работе.

Контрольные вопросы:

1) Что такое сравнительный анализ и как его проводить?

2) Как выбираются критерии сравнительного анализа?

3) Как выбирается шкала оценивания и коэффициенты?

Лабораторная работа №3

Формирование технического задания на разработка информационного ресурса

Цель работы: изучить методику и получить практические навыки формирования технического задания на разработку информационного ресурса

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Техническое задание (ТЗ) - является основным документом, определяющим требования и порядок создания автоматизированной системы, в соответствии с которым проводится её разработка и приемка. ТЗ на виды работ (на дизайн, ребрендинг, модернизацию) оформляется по тем же правилам, конкретный вид работ указывается в наименовании ТЗ.

ТЗ может составлять, как правило, как заказчик, так и исполнитель. Но обсуждается и согласовывается техническое задание, безусловно, обеими сторонами, т.к. Какие-то вещи знать не может заказчик, а какие-то исполнитель. Составление правильного ТЗ просто необходимый шаг в этапах создания сайта, если что-то упустить в задании, например, дополнительный модуль, то исполнитель может отказаться от доработки (в рамках данной задачи).

Разберем типовую структуру технического задания в формате спецификации требований к веб-приложению (SRS) (шаблон будет продемонстрирован ниже, после описания):

1. Титульный лист. Он должен содержать название проекта. Опционально версию, дату документа, ФИО и должность автора.
2. Лист с информацией о документе. В нем обязательно должна содержаться информация о внесении изменений в спецификацию. Опционально может содержаться информация о допуске сотрудников к документу, утверждении документа и техническая информация о спецификации.
3. Содержание. Данный раздел необходим для улучшения навигации по документу.\
4. Введение. В данном разделе содержится обзор спецификации. Имеются следующие подразделы:

   4.1. Цель – Описывается зачем и для кого составлен данный документ (спецификация);

   4.2. Масштаб – В данном подразделе необходимо указать: название программного продукта (ов), который(ые) должен(ы) быть создан (ы) (например, СУБД для

хостинга, генератор отчетов и т.д.); объяснение, что будет делать данный программный продукт, а при необходимости и что не будет делать; описание применения указанного программного обеспечения: его преимущества задачи и предназначения. Это не должно противоречить спецификация более высокого уровня.

4.3. Определения, акронимы и аббревиатуры – в данном подразделе необходимо привести определения всех терминов, акронимов и сокращений, необходимых для правильного толкования данной спецификации. Также могут содержаться ссылки на приложения к спецификации или другие документы.

4.4. Список использованной литературы – здесь необходимо указать полный список всех документов, на которые имеются ссылки в других разделах спецификации или в отдельном указанном документе; каждый элемент списка должен содержать название, номер, если возможно дату, и организацию-издателя, а также источники.

4.5. Обзор – содержит описание остальной части спецификации и объяснение как она организована.

5. Общее описание – В этом разделе спецификации должны быть описаны общие факторы, влияющие на продукт и требования к нему. Но в этом разделе не излагаются конкретные требования, данный раздел упрощает их понимание:

5.1. Обзор продукта – в данном подразделе спецификации продукт рассматривается в сравнении с другими связанными продуктами или проектами;

5.2. Функции продукта – этот подраздел спецификации содержит краткое описание функций, которые будет выполнять программное обеспечение;

5.3. Характеристики пользователя – в данном подразделе спецификации должны быть описаны те общие характеристики потенциальных пользователей продукта, которые будут влиять на конкретные требования;

5.4. Общие ограничения – данный подраздел спецификации должен содержать общее описание любых других элементов, которые будут ограничивать возможности разработчика при проектировании программного продукт;

5.5. Допущения и зависимости – в этом подразделе спецификации следует перечислить все факторы, которые влияют на требования, изложенные в спецификации. Эти факторы не являются конструктивными ограничениями программного обеспечения, а, скорее, любыми изменениями в них, которые могут повлиять на требования спецификации (Например, может быть сделано предположение, что на оборудовании, предназначенном для программного продукта, будет доступна

определенная операционная система. Если она не будет доступна на самом деле – то необходимо внести соответствующие изменения в спецификацию.

6. Особые требования – это самый важный и большой раздел спецификации. Требования заказчика будут изложены в разделе общего описания (пункт 5), но в этом разделе будут приведены общие требования, которые используются для руководства разработкой программного обеспечения, внедрения и тестирования проекта. Каждое требование в этом разделе должно быть:
    - Правильным;
    - Прослеживаемым (как в прямом, так и в обратном направлении к предыдущим/будущим артефактам);
    - Однозначным;
    - Поддающимся проверке (тестируемым);
    - Приоритетным (с точки зрения важности и/или стабильности)
    - Полным;
    - Согласованным;
    - Однозначно идентифицируемым (обычно с помощью нумерации).

    Следует обратить внимание на тщательную организацию требований, представленных в этом разделе таким образом, чтобы к ним было легко получить доступ и понять их. Кроме того, настоящая спецификации не является документом по разработке программного обеспечения, поэтому следует избегать чрезмерных ограничений (и, следовательно, разработки) программного проекта в рамках этой спецификации.

6.1. Требования к внешнему интерфейсу

6.1.1. Пользовательские интерфейсы – в данном разделе необходимо привести требования к пользовательскому интерфейсу (текстом или макетами)

6.2. Варианты использования – в данном разделе необходимо выделить основные сценарии использования данного проекта пользователями

6.3. Функциональные требования – в данном разделе описываются специфические особенности программного проекта. При желании некоторые требования могут быть указаны в формате прецедента и перечислены в разделе Варианты использования.

6.3.1. Функциональное требование №1

6.3.1.1. Название

6.3.1.2. Описание

6.3.1.3. Назначение

6.3.1.4. Зависимость

6.4. Нефункциональные требования – для следующих свойств могут существовать нефункциональные требования. Часто эти требования должны выполняться на общесистемном уровне, а не уровне отдельного устройства. Изложите требования в следующих разделах в измеримых терминах (например, 95% транзакций должно обрабатываться менее чем за секунду, время простоя системы не должно превышать 1 минуты в день, наработка на отказ > 30 дней и т.д.)

6.4.1. Производительность

6.4.2. Надежность

6.4.3. Доступность

6.4.4. Безопасность

6.4.5. Удобство обслуживания

6.4.6. Мобильность

7. Аналитические модели – необходимо в данном разделе перечислить все аналитические модели, использованные при разработке конкретных требований, ранее приведенных в этой спецификации. Каждая модель должна включать введение и краткое описание. Кроме того, в каждой модели должны прослеживаться требования спецификации,

7.1. Диаграммы состояний и переходов

8. Процесс управления изменениями – в данном разделе определяется и описывается процесс, который будет использоваться для обновления спецификации, по мере необходимости, при изменении масштаба проекта или требований. Кто и какими средствами может вносить изменения, и как эти изменения буду утверждаться.

Пример спецификации

Пример приведен в приложении 3.1

Задание для выполнения:

1. По выбранной ранее предметной области составить техническое задание на разработку проекта в виде спецификации (SRS);

2. Оформить отчет.

Содержание отчета:

1. Титульный лист;

2. Цель работы;

3. Ход работы (техническое задание);

4. Выводы по работе.

Контрольные вопросы:

1) Что такое техническое задание?

2) Для чего необходимо техническое задание?

3) Какие пункты входят в SRS?

Мобильное приложение для стриминга музыки

Спецификация требований к программному обеспечению

Ревизия: 1.1

23.01.2024

Автор: Иванов Иван Иванович
Руководитель команды разработки

История изменений

Дата	Описание	Автор	Комментарии
19.01.2024	Первая версия	Иванов И.И.	-
23.01.2024	Исправление замечаний	Иванов И.И.	-

1. ВВЕДЕНИЕ

1.1. Цели

Основной целью является изучение процесса разработки мобильного приложения на примере конкретной программной системы, позволяющей пользователям прослушивать музыку онлайн, публиковать собственные аудиозаписи на сайт, делиться ими с пользователями и формировать собственные плейлисты.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать хранилище данных, в котором будет находиться информация о различных сущностях системы;
2. Разработать мобильное пользовательское приложение;
3. Разработать серверную часть приложения.

Во многом идея для сервиса была вдохновлена онлайн-сервисом SoundCloud

1.2. Соглашение о терминах

1) Приложение – разрабатываемый в ходе работы программный продукт.

2) Трек (песня) – аудиозапись, представляющая собой музыкальную композицию.

3) Плейлист (или альбом) – набор треков, объединённых в одну группу, создаваемую пользователем.

4) Бэкенд – серверная часть приложения.

1.3. Предполагаемая аудитория и последовательность восприятия

Целевой аудиторией приложения являются люди старше 16 лет. Ожидается, что пользователи будут опубликовывать треки, содержащие упоминание нецензурной речи. Сервис запрещает публиковать треки с призывами к противоправным действиям или нарушающие законодательство. При публикации таковых соответствующий контент и пользователи-авторы будут заблокированы.

1.4. Масштаб проекта

На этапе минимально жизнеспособного продукта (MVP) ожидается, что продуктом будет пользоваться не больше 1000 одновременных пользователей. Система будет проектироваться с учётом одновременной нагрузки на 10000 пользователей.

2. ОБЩЕЕ ОПИСАНИЕ

2.1. Видение продукта

Музыкальный стриминговый сервис представляет собой платформу, которая позволяет пользователям делиться и открывать музыку через мобильное приложение и веб-сайт.

Пользователи могут прослушивать музыку и добавлять её в свою библиотеку. Также они могут загружать собственную музыку и делиться ею с другими пользователями. Помимо плейлистов пользователи могут загружать полноценные альбомы на платформу. С точки зрения системы отличие альбома от плейлиста в работе рекомендательной системы, однако она не будет рассмотрена в рамках текущей работы.

2.2. Функциональность продукта

В продукте подразумевается следующий функционал:

- Регистрация;
- Прослушивание треков;
- Загрузка треков;
- Публикация треков;
- Формирование собственных плейлистов.

2.3. Классы и характеристики пользователей

Слушатель – категория, под которую попадает большинство ожидаемых пользователей. Имеет доступ ко всему функционалу, описанному в пункте 2.2.

Автор – пользователь, создающий музыку. С точки зрения системы на начальном этапе разработки, имеет мало отличий от обычного пользователя. Однако автор сам создаёт музыку и публикует её для слушателей.

Администратор (модератор) – пользователь, модерирующий контент на сайте и рассматривающий жалобы от пользователей. Имеет возможность блокировать пользователей и контент.

2.4. Рамки, ограничения, правила и стандарты

Приложение планируется выпустить на Android и iOS, и позднее разработать веб-версию. Единственным ограничением будет размер загружаемой аудиозаписи, аудиофайлы должны быть размером не более 15 Мб.

2.5. Документация для пользователей

Предполагается написание руководства пользователя.

3. ТРЕБОВАНИЯ К ВНЕШНИМ ИНТЕРФЕЙСАМ

3.1. Требования к интерфейсу пользователя

Первая страница приложения будет представлять страницу входа/регистрации. Не аутентифицированные пользователи не могут пользоваться функционалом приложения. Далее главная страница будет состоять из двух страниц (главная и профиль). На главной пользователь сможет просмотреть популярные и недавно вышедшие треки. На странице профиля пользователь сможет открыть список сохранённых плейлистов, альбомов и треков, а также загрузить свои треки.

Также будет обеспечен экран воспроизведения с кнопками управления, информацией о текущем треке и возможностью перемотки. Включите экран профиля пользователя и настройки.

3.2. Требования к программным интерфейсам

Для хранения данных (включая сами треки) и бэкенда будет использован Firebase. Необходимо использовать Firebase для аутентификации пользователей, стриминга треков, хранения треков, а также хранения плейлистов. Треки будут храниться с использованием Firebase Storage.

3.3. Интерфейсы оборудования

Будет использован API Firebase для взаимодействия с сервером. Интерфейс пользователя взаимодействует с экраном мобильного устройства, а Flutter обеспечивает адаптацию интерфейса под различные размеры экранов.

3.4. Интерфейсы связи и коммуникации

Коммуникация между приложением и сервером Firebase происходит посредством Firebase SDK по сети с использованием протоколов HTTPS и WebSocket.

3.5. Диаграмма прецедентов

Диаграмма прецедентов представлена на рисунке 3.1.

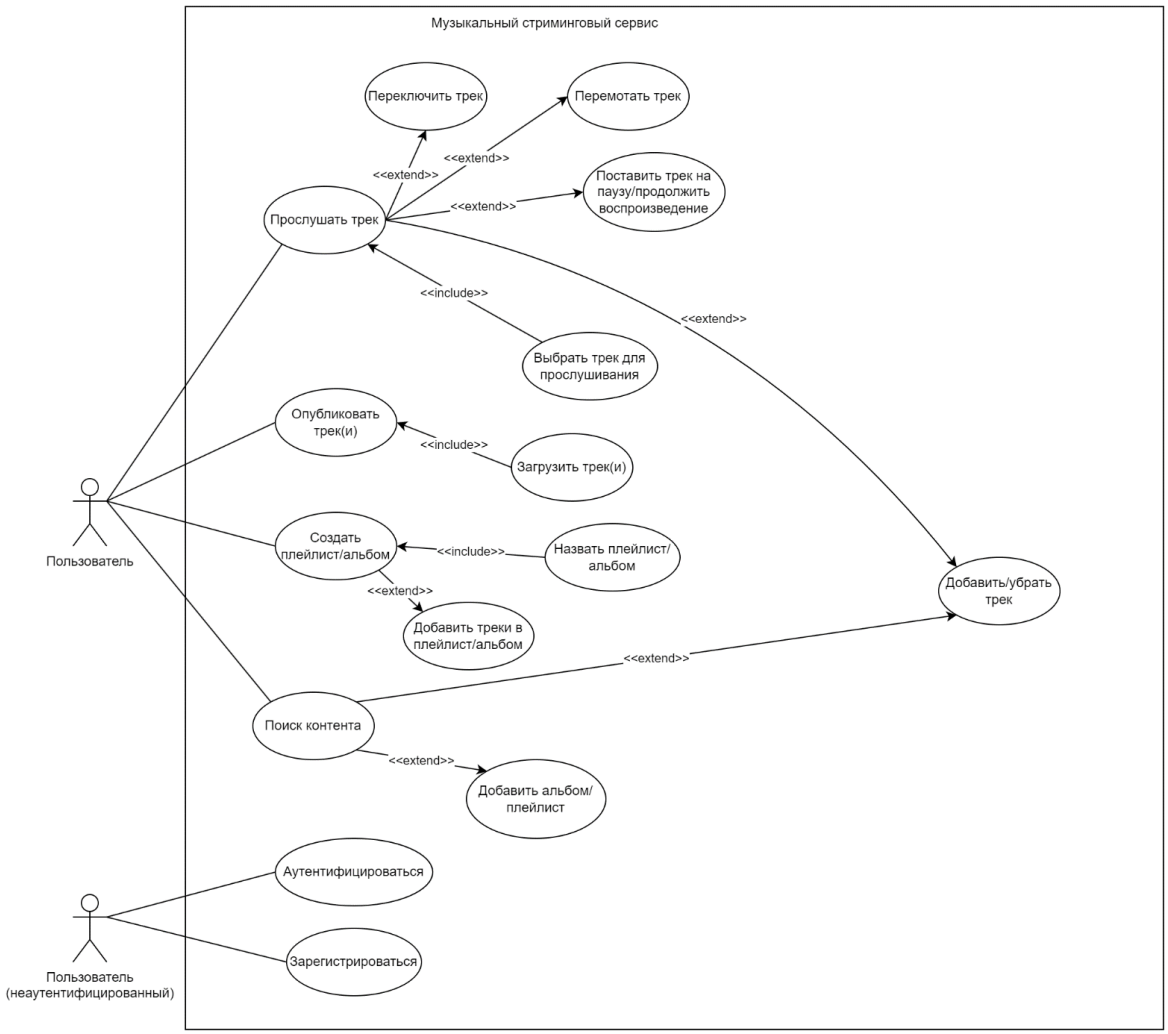


Рисунок 3.1. Диаграмма прецедентов

4. ФУНКЦИОНАЛЬНОСТЬ СИСТЕМЫ

4.1. Функциональные требования

Для прототипа приложения были выделены следующие функциональные требования:

4.1.1. Аутентификация пользователей

Цель: Приложение не предоставляет своего функционала неаутентифицированным пользователям поэтому цель аутентификации – предоставление пользователям доступа к функционалу.

Описание: Приложение должно предоставлять возможность регистрации пользователей и их входа по почте и паролю.

4.1.2. Аудиоплеер

Цель: Дать возможность пользователям управлять прослушиванием.

Описание: Аудиоплеер представляет собой интерфейс для прослушивания треков пользователями. Плеер должен предоставлять возможности перемотки играющего трека, приостановки и возобновления воспроизведения, добавления песни в аудиозаписи и переключения треков в очереди.

4.1.3. Главная страница

Цель: Дать возможность просмотра списка популярных треков среди пользователей

Описание: На главной странице пользователь может просматривать список популярных треков среди пользователей платформы. Отбор треков происходит по следующим критериям:

- Треку не больше двух недель с момента загрузки;
- Топ составляется на основе количества добавлений трека за 2 недели;

4.1.4. Загрузка треков

Цель: Дать пользователям возможность загружать и делиться собственными треками.

Описание: Каждый пользователь имеет право добавить любой собственный трек. Для этого необходимо создать меню на странице профиля, с помощью которого можно дать название загружаемому файлу, поставить статус доступности (доступен/недоступен для других пользователей) и загрузить сам файл из памяти мобильного устройства.

4.2. Бизнес-процесс «Прослушивание треков»

Процесс представляет собой главную функциональность приложения по прослушиванию треков. Пользователь выбирает трек для прослушивания и нажимает на трек, чтобы включить его. Происходит запрос к сервису, клиент получает данные о треке и начинается стриминг трека на устройство пользователя.

Подробное описание последовательности действий представлено на BPMN-диаграмме на рисунке 3.2.

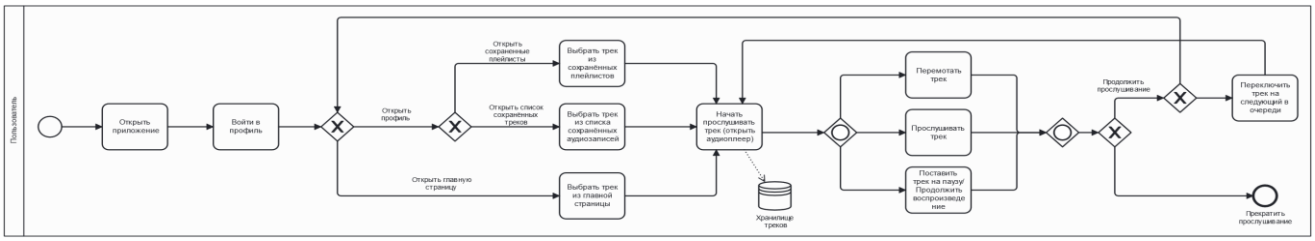


Рисунок 3.1. Прослушивание треков

4.3. Бизнес-процесс «Создание плейлиста»

Процесс представляет собой функциональность приложения по созданию плейлистов. Пользователь на странице личного профиля создаёт плейлист. Для плейлиста нужно указать название, его тип (альбом/плейлист) и доступ остальных пользователей к нему. Также в рамках бизнес-процесса пользователю нужно добавить треки в плейлист.

Подробное описание последовательности действий представлено на BPMN-диаграмме на рисунке 3.3.

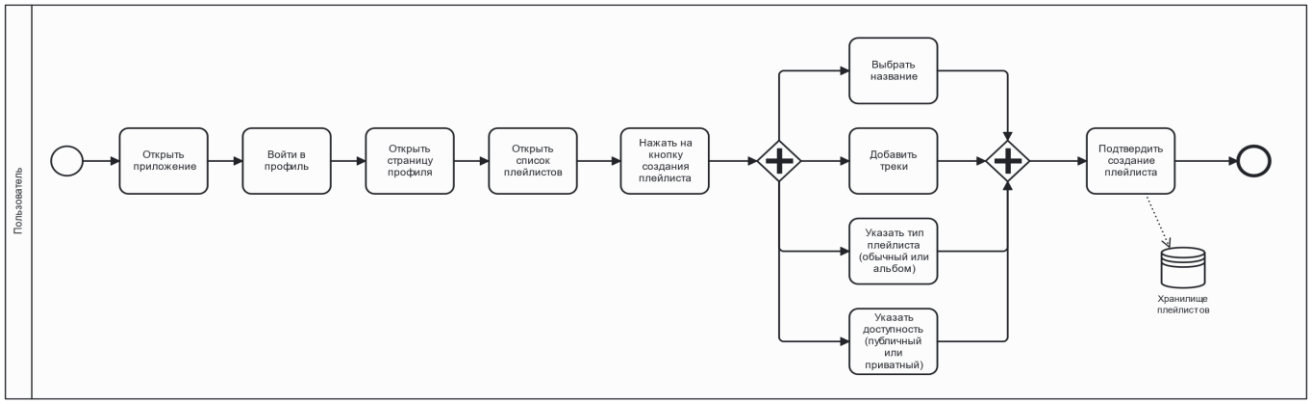


Рисунок 3.2. Создание плейлиста

4.4. Бизнес-процесс «Поиск и сохранение контента»

Процесс представляет собой функциональность приложения по поиску и сохранению треков и плейлистов в профиле пользователя. Пользователь может искать контент на главной странице приложения либо он может воспользоваться строкой поиска. В строке поиска на главной странице пользователь вводит название и по этому названию получает список треков, плейлистов и альбомов. На главной странице пользователь может просмотреть список популярных треков и недавних/популярных плейлистов. В результате пользователь может сохранить в свои аудиозаписи найденные альбомы или плейлисты.

Подробное описание последовательности действий представлено на BPMN-диаграмме на рисунке 3.4.

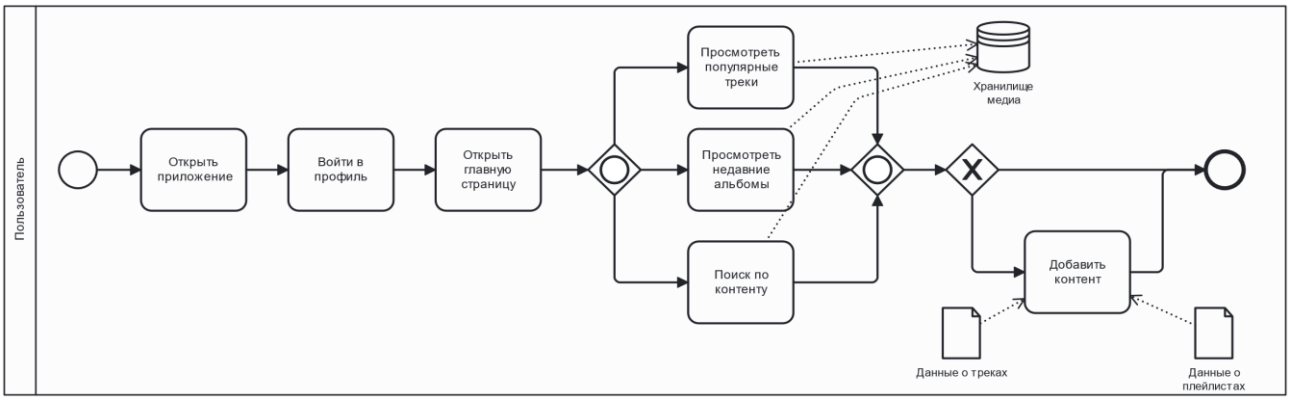


Рисунок 3.3. Поиск и сохранение контента

5. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

5.1. Требования к производительности

Время отклика приложения на пользовательские запросы не должно превышать 1 секунду в любых условиях сети и загруженности серверов. Приложение должно стабильно работать на устройствах, поддерживающих Android 7 и выше.

5.2. Требования к сохранности данных

Должна быть обеспечена конфиденциальность личных данных. Должна присутствовать возможность скрывать и публиковать в открытый доступ песни и плейлисты.

5.3. Критерии качества программного обеспечения

Программное приложение будет признано качественным, если в результате разработки будут соблюдены все требования технического задания.

Мобильное приложение должно быть оптимизировано для эффективного использования ресурсов устройства, потребляя минимальное количество оперативной памяти и процессорной мощности.

Лабораторная работа №4

Построение карты сайта

Цель работы: изучить процесс формирования структуры веб-приложения и построить карту сайта, иллюстрирующую структуру, для разрабатываемого проекта.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Карта сайта - это отдельная страница вашего сайта, на которой собраны ссылки на все внутренние страницы вашего сайта.

Обычно карта сайта создается для удобства посетителей. По ней легко найти нужный материал, просмотрев всю структуру сайта. Это касается HTML - карты.

А вот для поисковиков создается несколько иной формат карты сайта, так называемый sitemap XML. Он не читабелен для посетителей, так как представляет собой файл XML, в котором содержится список всех URL сайта.

Создается sitemap XML с одной целью - ускорить индексацию вашего сайта роботами поисковых систем. Поисковый робот часто не доходит до ссылок, расположенных на "глубине" (я имею ввиду страницы с уровнем вложенности больше трех). То есть глубокие ссылки остаются незамеченными.

Вот поэтому и создается специальная карта сайта в формате sitemap XML.

В рамках данной работы будем планировать структуру веб-приложения и построим примерную карту сайта (см. Рисунки 4.1 и 4.2).

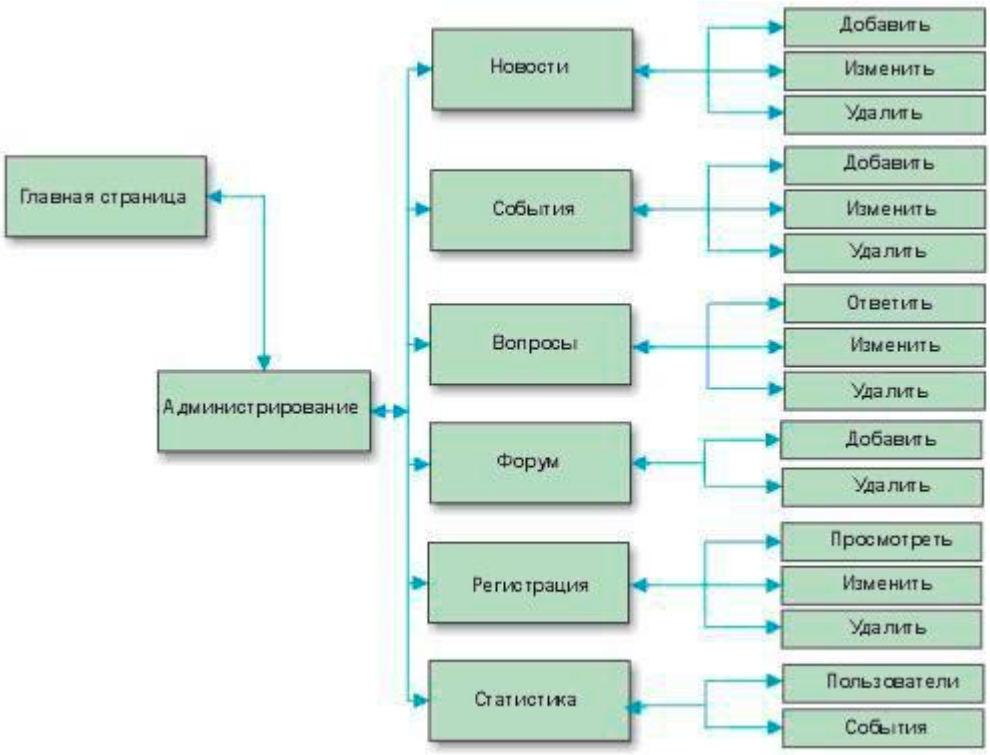


Рисунок 4.1. Пример карты сайта

Рисунок 4.2. Пример карты сайта

Задание для выполнения:

1. По выбранной ранее предметной области спланировать и нарисовать карту веб-приложения

2. Оформить отчет.

Содержание отчета:

1. Титульный лист;

2. Цель работы;

3. Ход работы (карта сайта и ее описание);

4. Выводы по работе.

Контрольные вопросы:

1) Что такое карта сайта?

2) Для чего необходима карта сайта?

Лабораторная работа №5

Построение диаграммы вариантов использования

Цель работы: изучить процесс управления требованиями к разрабатываемой системе, представление диаграмм в виде прецедентов, освоить построение диаграмм прецедентов и написание описаний прецедентов.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Технологический процесс управления требованиями

В ходе технологического процесса управления требованиями необходимо выполнить следующее:

1) Разработчикам и заказчикам необходимо выработать единое мнение о том, что должна делать система;

2) Разработчики должны полнее понять системные требования;

3) Должны быть определены границы системы;

4) Должна быть создана основа для планирования технического содержания итераций, а также оценки стоимости и времени разработки системы;

5) Исходя из нужд и целей пользователей, необходимо определить пользовательский интерфейс системы.

Для достижения этих целей требуется осуществить ряд последовательных (итеративных) этапов:

1) Выявление требований к системе

2) Выявление прецедентов и субъектов

- Распределение требований по субъектам и прецедентам
- Построение диаграммы прецедентов
- Составление документа описания прецедентов

Проектирование пользовательского интерфейса

Выявление требований к системе

Требование — это условие или характеристика, которой должна соответствовать система. Требования представляются как подробное текстовое описание функций системы. Для эффективного управления всеми требованиями необходимо иметь полное понимание нужд пользователей и других заинтересованных лиц, которым должна удовлетворять разрабатываемая система.

Виды требований:

- функциональные требования – выражают поведение системы;
- нефункциональные требования – набор требований параметры, качества которых нельзя сформулировать с помощью функциональных требований.

Категории нефункциональных требований: практичность, надежность, производительность, возможность поддержки.

Традиционно, требования представляются как подробное текстовое описание, относящееся к одной из категорий, и выражаются в форме «Система должна…».

Для эффективного управления всеми требованиями необходимо иметь полное понимание нужд пользователя и других заинтересованных сторон, которым должна удовлетворять разрабатываемая система. Это позволяет команде разработчиков ответить не только на вопрос что?, но и на вопрос почему?. Зная ответы на эти вопросы, команда сможет лучше их интерпретировать.

Выявление прецедентов и субъектов

Исходя из полученных требований производиться выявление субъектов

системы и прецедентов данной системы. Прецедент (use case) выполняет бизнес-функцию, которую может наблюдать внешний субъект.

Субъект (actor) — это некто или нечто (человек, машина и т.д.), взаимодействующие с прецедентом. Субъект взаимодействует с прецедентом, ожидая получить полезный результат.

Субъекты и прецеденты определяются в результате анализа требований. После того, как определены требования, выявляют субъекты и прецеденты. По полученным данным можно построить таблицу, которая распределяет требования по субъектам и прецедентам.

Построение диаграммы прецедентов

Диаграмма прецедентов приписывает прецеденты к субъектам; она также позволяет установить отношения между прецедентами, если таковые существуют.

Диаграмма использования (use case diagram) предназначена для отображения внешнего функционирования проектируемой системы и ее взаимодействия с внешним миром пользователями. Основой подхода являются так называемые блоки использования (use case), которые представляют собой некоторый набор функций системы, объединяемых в единое целое с точки зрения пользователя. Один блок использования не обязательно

представляет собой одну часть системы или даже единую группу функций. Он представляет собой именно понимание пользователем поведения системы.

Диаграммы использования являются широко применяемыми в различных технологиях проектирования. Их главная задача – спецификация требований к системе на начальных этапах проектирования, когда решаются наиболее общие задачи предназначения разрабатываемой системы.

Диаграмма состоит из следующих элементов:

- внешние пользователи (actors), это такие воздействия, которые передают или получают информацию для системы, это могут быть физические объекты разной природы от людей и механизмов до программных систем, один физический объект может описываться несколькими пользователями, если он взаимодействует с разными функциями,
- блоки использования (use case), это такие группы функций системы, которые объединяются в единое целое для внешнего пользователя,
- связи между блоками использования и связи между блоками использования и внешними пользователям.

Выделяют несколько стандартных видов отношений между актерами и вариантами использования:

- Отношение ассоциации (association relationship) служит для обозначения специфической роли актера в отдельном варианте использования. Другими словами, ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы.
- Отношение расширения (extend relationship) отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.
- Отношение обобщения (generalization relationship) между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов

использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

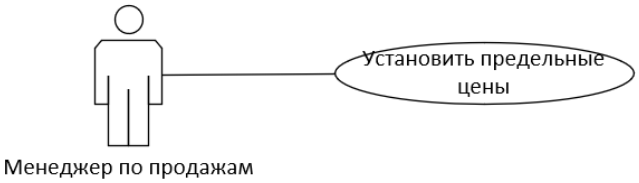
- Отношение включения (include relationship) между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в отношении включения.

Графически блоки использования обозначаются эллипсами с указанием имени внутри эллипса или рядом с ним. Внешние пользователи графически обозначаются как прямоугольники с табулятором "Пользователь" или в виде схематичной фигурки человека, с именем под ней. Элемент подсистемы - для оформления условных границ, объединяющих некоторую подгруппу элементов.

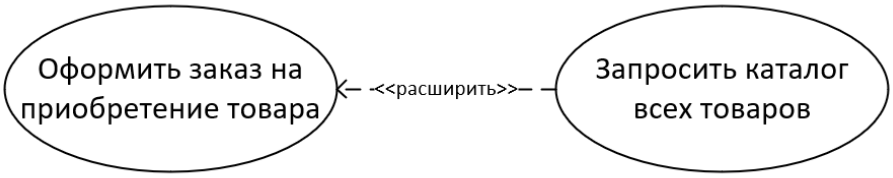
Рис. 5.1. Элементы диаграммы прецедентов

Графическое обозначение для связей следующее:

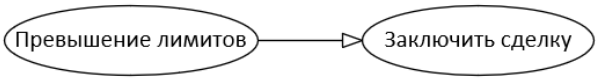
– отношение ассоциации обозначается сплошной линией. Эта линия может иметь дополнительные условные обозначения, такие, например, как имя и кратность. Кратность ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации. Кратность характеризует общее количество конкретных экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и, возможно, специального символа "*";



– отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом "extend" ("расширить");



– отношение обобщения обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования. Эта линия со стрелкой имеет специальное название — стрелка "обобщение".



– отношение включения обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом "include" ("включить").

Все блоки использования объединяются на рисунке в единый прямоугольник, очерчивающий рамки проектируемой системы.

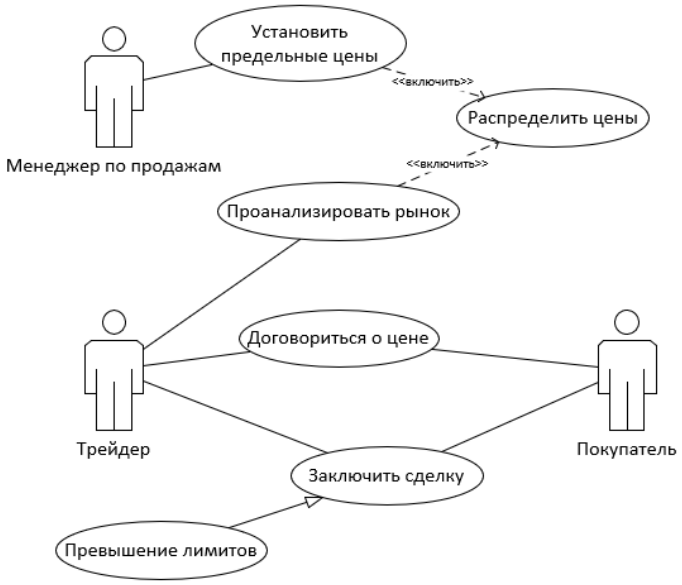


Рис. 5.2. Пример диаграммы прецедентов

Другими словами, диаграмма вариантов использования состоит из актеров, для которых система производит действие и собственно действия Use Case, которое описывает то, что актер хочет получить от системы.

Ответы на следующие вопросы позволят определить актеров, взаимодействующих с системой:

– кто взаимодействует с системой или использует систему;
– кто передает или принимает информацию в/из системы;
– кто является внешним по отношению к системе.

Каждый вариант использования показывает, как конкретный актер использует систему и в дальнейшем расширяется диаграммами состояний и последовательности действий.

Составление документа описания прецедентов

Структура документа, описывающего прецеденты, может варьироваться, однако типичное описание должно содержать следующие разделы:

1. Краткое описание;
2. Участвующие субъекты;
3. Предусловия, необходимые для инициирования прецедента;
4. Детализированное описание потока событий, которое включает:
   - основной поток событий;
   - альтернативные потоки для определения исключительных ситуаций;
5. Постусловия, определяющее состояние системы, по достижении которого прецедент завершается.

Документ описания прецедента развивается по ходу разработки. На ранней стадии определения требования составляется лишь краткое описание, остальные части документа создаются постепенно и итеративно.

Пример работы

Требования к системе (исходные данные)

В качестве примера, который возьмем за основу для реализации технологического процесса управления требованиями, рассмотрим систему Internet-магазин:

Производитель компьютеров предлагает возможность приобретения своей продукции через Internet. Клиент может выбрать компьютер на Web-странице производителя. Компьютеры подразделяются на серверы, настольные и портативные. Заказчик может выбрать любую конфигурацию. Компоненты конфигурации представляются как список. Для каждой конфигурации предоставляется цена.

Чтобы оформить заказ клиент должен заполнить информацию по доставке и оплате. В качестве платежных средств допускается использование кредитных карточек. После ввода заказа система отправляет клиенту по электронной почте сообщение с подтверждением получения заказа. Пока клиент ожидает прибытия компьютера, он может проверить состояние заказа в любое время. Серверная часть обработки заказа состоит из заданий, необходимых для проверки кредитоспособности и способа расчета клиента за покупку, из требования заказанной конфигурации, печати счета и подачи заявки на склад о доставке компьютера клиенту.

Выявление прецедентов и субъектов

Распределение требований по субъектам и прецедентам приведено в Таблице 5.1.

Таблица 5.1. Распределение требований по субъектам и прецедентам

	Требование	Субъект	Прецедент
1	Для знакомства со стандартной конфигурацией выбираемого компьютера клиент использует Web-страницу Internet-магазина. При этом также приводится цена конфигурации	Клиент	Отображение стандартной конфигурации компьютера
2	Клиент выбирает детали конфигурации, с которыми он хочет познакомиться, возможно, с намерением купить готовую или составить более подходящую конфигурацию. Цена для каждой конфигурации может быть подсчитана по требованию пользователя	Клиент	Составление конфигурации компьютера
3	Клиент может выбрать вариант заказа компьютера по Internet либо попросить, чтобы продавец связался с ним для объяснения деталей заказа, договорился о цене и т.п. прежде, чем заказ будет фактически размещен	Клиент, Продавец	Заказ сконфигурированного компьютера, Обращение с просьбой к продавцу
4	Для размещения заказа клиент должен заполнить электронную форму с адресами для доставки товара и отправки счет-фактуры, а также деталями, касающимися оплаты (кредитная карточка или чек)	Клиент	Заказ сконфигурированного компьютера, Проверка и прием платежа от клиента
5	После ввода заказ клиента в систему продавец отправляет на склад электронное требование, содержащее детали, касающиеся заказанной конфигурации	Продавец, Склад	Информирование склада о заказе
6	Детали сделки, включая номер заказа, номер счета клиента, отправляются по электронной почте клиенту, так что заказчик может проверить состояние заказа через Internet	Продавец, Клиент	Заказ сконфигурированного компьютера, Обновление состояния

7	Склад получает счет-фактуру от продавца и отправляет компьютер клиенту	Продавец, Склад	Печать счет- фактуры

Построение диаграммы прецедентов

Диаграмма прецедентов представлена на рисунке. 3.

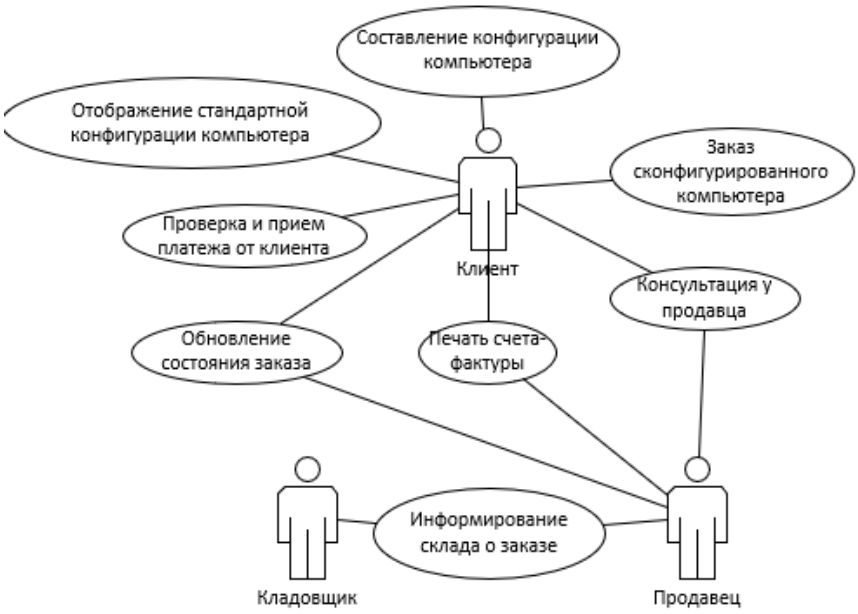


Рисунок. 5.3. Диаграмма прецедентов

Составление документа описания прецедентов

Составление конфигурации компьютера

Краткое Описание

Этот документ описывает процедуру построения конфигурации компьютера. Актер - Клиент.

Поток Событий

Основной Поток: Пользователь просматривает конфигурацию компьютера

1. Пользователь переходит по ссылке конфигурации компьютера со страницы «Каталог товаров»
2. Открывается страница «Конфигурация компьютера»
3. На странице подробно отображается информация о конфигурации выбранного компьютера, а также отображаться текущая цена.

4. Имеются ссылки на страницу «Каталог товаров», и «Заказ сконфигурированного компьютера»

Альтернативный Поток 1: Ошибка просмотра конфигурации компьютера.

Если при попытке просмотра требуемая конфигурация не была загружена, сообщение об ошибке должно быть отображено.

Кроме того, должна быть ссылка на страницу «Каталог товаров»

Предусловие

Открыть страницу «Конфигурация компьютера»

Постусловие

нет

Задание для выполнения:

1. Изучить теоретические данные по диаграмме прецедентов;
2. Провести анализ предметной области с целью выявления актеров и прецедентов;
3. Построить диаграмму прецедентов в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Сценарий выполнения UML);
4. Описать 2 прецедента на свой выбор (кроме регистрации и авторизации) по шаблону.
5. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, таблица распределения требований по субъектам и актерам, диаграмма прецедентов, описание 2 прецедентов);
4. Выводы по работе.

Контрольные вопросы:

1. Назовите и приведите условные обозначения основных элементов диаграмм прецедентов.
2. Перечислите виды отношений между элементами диаграммы прецедентов и приведите примеры их использования.

3. Объясните назначение разделов документа описания прецедентов.
4. Поясните взаимосвязь описания прецедентов и проектирования интерфейса пользователя.

Лабораторная работа №6

Построения диаграмм поведения

Цель работы: изучить методы динамического моделирования в языке UML, освоить построение диаграммы видов деятельности, диаграммы последовательностей и диаграммы коммуникации.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Диаграмма видов деятельности (activity diagram) отражает динамику системы и особенно полезна при описании поведения, включающего в себя большое количество параллельных процессов, а также для моделирования поведения системы в самом общем виде на этапе анализа (рисунок 6.1).

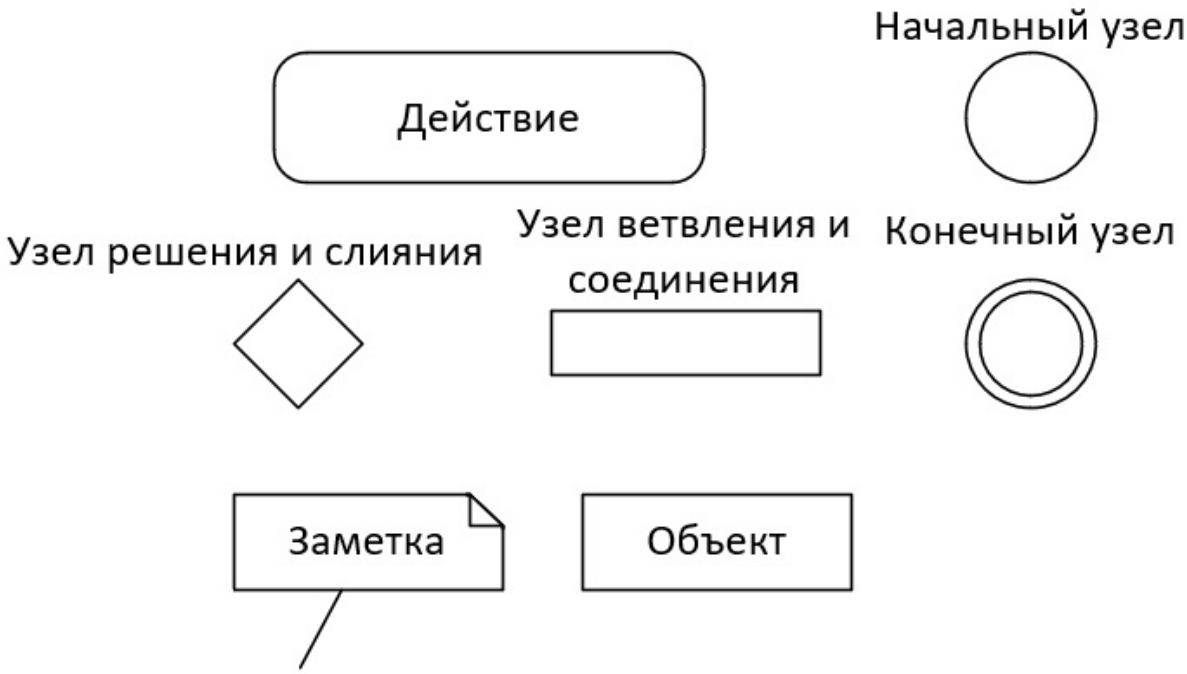


Рисунок 6.1. Узлы диаграммы видов деятельности

В данном представлении поведения системы упор делается на ее функциональные составляющие, или действия (Action). Действие представляет собой базовую единицу функциональности системы, не декомпозируемую в рамках содержащего их вида деятельности. Все действия являются предопределенными. Например, в языке UML определены действия для создания объектов, установки значений атрибутов, связывания объектов и т.д. Кроме того, имеются действия для выполнения работ, определяемых пользователем (в том числе и видов деятельности). Действия могут иметь входы и выходы,

называемые контактами (Pin), которые связываются ребрами потоков объектов. Контакты — это разновидность узлов объектов, поэтому они временно сохраняют значения данных в потоке. В простейшем случае для начала выполнения действия требуется наличие всех его входных данных.

Вид деятельности (Activity) — это спецификация поведения системы в виде координируемой последовательности действий.

Подобно всем разновидностям работ в UML, вид деятельности можно инициировать с помощью вызывающего действия (CallAction). Вид деятельности содержит параметры (ActivityParameterNode) для получения входных данных от вызывающего действия и предоставления ему выходных данных. Параметры не являются контактами, поскольку последние используются для соединения действий в потоке, а вид деятельности — это работа, вызываемая действием. Для доступа к значениям параметров из действий внутри вида деятельности параметры моделируются как особый вид узлов потоков объектов. Параметры размещаются на границе диаграммы, и ребра потока объектов соединяют их с контактами.

Раздел (ActivityPartition) представляет собой средство группировки действий в соответствии с некоторым признаком. Действие может входить одновременно в несколько разделов. Раздел может содержать несколько подразделов. Особым видом разделов являются так называемые внешние разделы, используемые для представления сущностей, внешних по отношению к системе. Внешние разделы помечаются стереотипом <<external>>. Наиболее часто разделы применяются на этапе определения требований для моделирования подразделений организационной структуры или отдельных исполнителей в бизнес-моделях. Так, в примере на рисунке 6.2 разделы «Отдел заказов», «Бухгалтерия» и «Покупатель» отражают исполнителей бизнес- процесса обработки заказа.

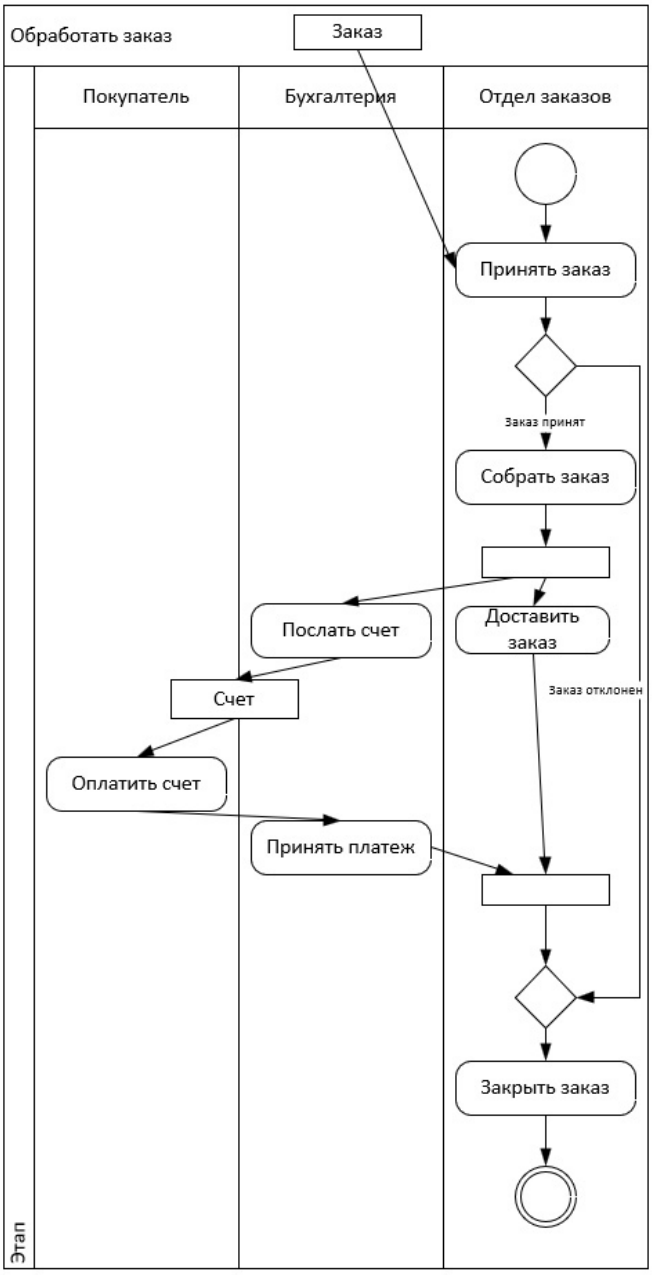


Рисунок 6.2. Пример диаграммы видов деятельности

Диаграмма видов деятельности представляет собой граф, узлы которого соединены ребрами, представляющими потоки управления и данных. Маркеры управления и данных двигаются вдоль ребер и обрабатываются узлами, направляются на другие узлы или временно сохраняются. Каждые узел и ребро определяют, когда через них могут проходить управляющие воздействия и значения данных. Эти правила движения маркеров можно комбинировать с целью прогнозирования поведения всего графа.

В диаграммах видов деятельности имеется три вида узлов.

Узлы действия оперируют получаемыми управляющими воздействиями и значениями данных и предоставляют управление и данные другим действиям.

Узлы управления маршрутизируют перемещение маркеров управления и данных по графу. Эти узлы содержат конструкции для выбора между альтернативными потоками, для параллельного движения по нескольким потокам и т.д.

Объектные узлы временно удерживают маркеры данных, которые ожидают продолжения движения по графу.

Узлы связываются направленными ребрами двух типов:

- ребра потоков управления связывают действия. Они обозначают, что действие на конце ребра со стрелкой не может начаться до того, как закончится исходное действие. По ребрам потоков управления могут проходить только маркеры управления;
- ребра потоков объектов соединяют узлы объектов (в том числе, контакты действий и параметры вида деятельности). По ребрам потоков объектов могут проходить только маркеры данных.

Действие активизируется только при наличии маркеров управления на всех входных ребрах потоков управления и маркеров данных на всех входных контактах. Перед выполнением действия соответствующие маркеры изымаются, а по его окончании на выходных ребрах потоков управления и выходных контактах размещаются маркеры управления и данных, после чего возможна активизация следующего действия.

Ветвление (DecisionNode) дает возможность показать разделение по условиям управляющих потоков. Данный узел имеет единственный вход и несколько выходов со сторожевыми условиями. Так как может выполниться только один из выходных переходов, то сторожевые условия должны взаимно исключать друг друга. Сторожевые условия изображаются в квадратных скобках около линии перехода. Если в качестве сторожевого условия используется [иначе], то это означает, что переход с данной меткой срабатывает в том случае, когда все другие переходы для данного ветвления являются ложными. Слияние (MergeNode) имеет несколько входов и единственный выход. Данный блок означает окончание условного поведения, которое было начато соответствующим ветвлением.

Параллельное поведение изображается при помощи разделений (ForkNode) и объединений (JoinNode). Последовательность выполнения параллельных действий является произвольной.

Конечный узел (ActivityFinalNode) завершает выполнение всех действий в виде деятельности (в том числе других видов деятельности, вызванных синхронно из текущего), удаляет все маркеры управления и данных, за исключением маркеров данных в выходных параметрах вида деятельности, и возвращает управление в вызвавшее его действие.

Конечный узел потока управления (FlowFinalNode) позволяет завершить отдельный поток управления, не затрагивая выполнение вида деятельности целиком. Это особенно полезно для моделирования видов деятельности, обрабатывающих потоки данных.

Диаграммы взаимодействия

Диаграммы взаимодействия (interaction diagrams) представляют собой модели, которые необходимы для описания поведения взаимодействующих групп объектов.

В языке UML существует четыре вида диаграмм взаимодействия: диаграмма последовательностей, диаграмма коммуникации, обзорная диаграмма взаимодействия (в пособии не рассматривается) и временная диаграмма (в пособии не рассматривается). Эти типы диаграмм дополняют друг друга, представляя взаимодействие элементов системы в динамике, но под различными углами зрения.

Диаграмма последовательностей

Диаграмма последовательностей (sequence diagram) отражает поток событий, происходящих при реализации некоторого прецедента, на этой диаграмме изображаются актеры, объекты, а также принимаемые и посылаемые ими сообщения (рисунок 6.3).

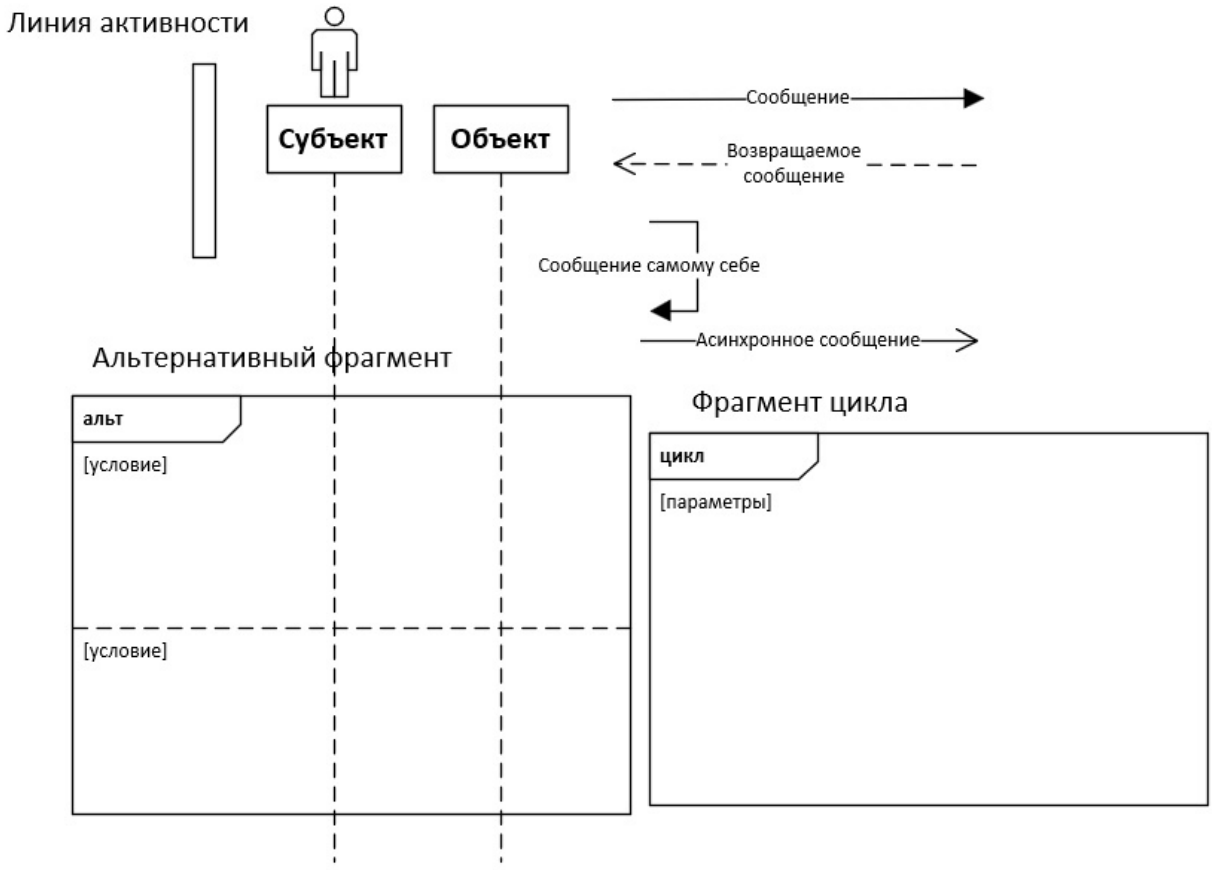


Рисунок 6.3. Элементы диаграммы последовательностей

При построении диаграммы последовательностей в первую очередь отражается

временная последовательность происходящих событий. На рис. 4 представлен пример диаграммы последовательностей, которая имеет два

измерения: вертикальное направление представляет время, горизонтальное – различные объекты.

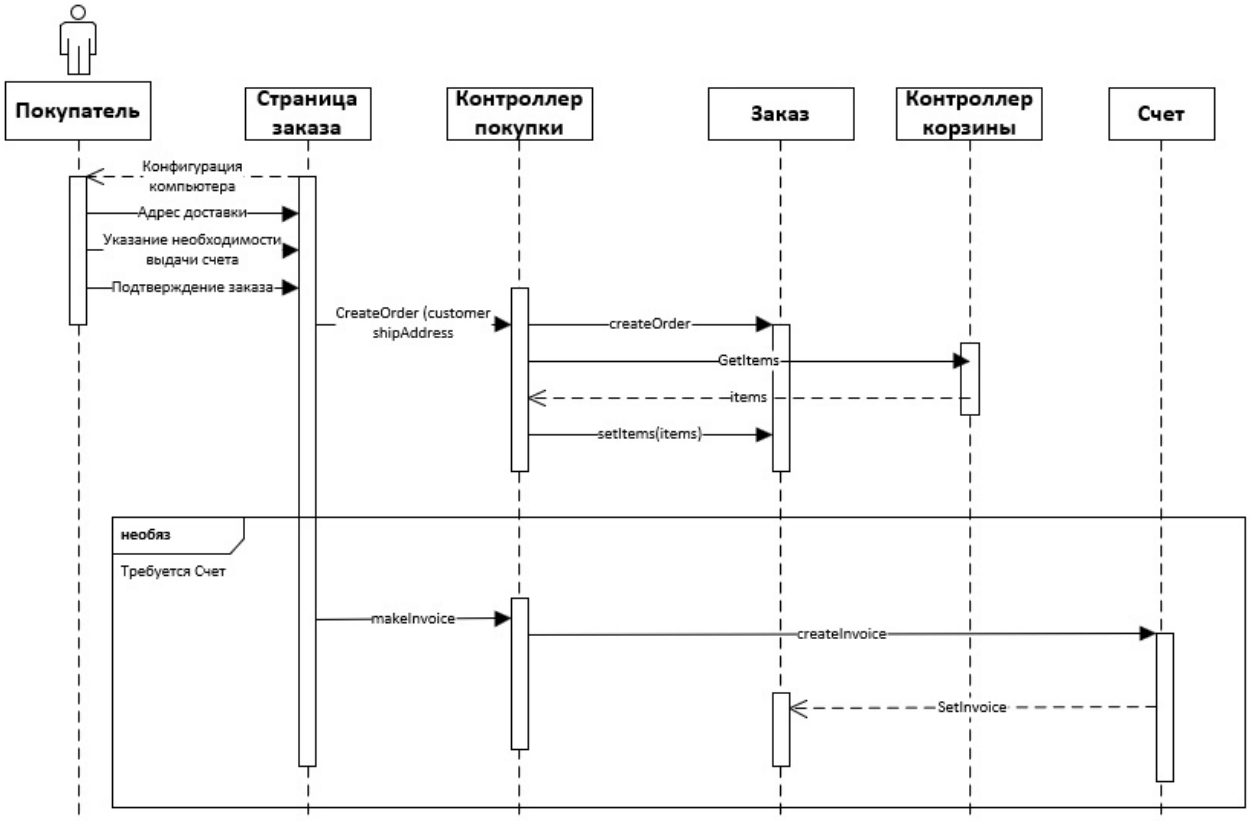


Рисунок 6.4. Пример диаграммы последовательностей

Объект (Object) — это экземпляр класса, конкретная сущность или образец, он может инициировать некоторые события, которые могут влиять на систему. На диаграмме последовательностей все объекты расположены последовательно в верхней ее части, за исключением объектов, создаваемых в результате тех или иных сообщений (примерами последних являются объекты Order и Invoice на рисунке 6.4).

От каждого объекта вниз отходит штриховая линия, называемая линией жизни (Lifeline) объекта. На ней показывают все, что происходит с объектом с момента его создания и до разрушения.

Частным случаем объекта может быть актер, он используется в том случае, когда необходимо показать связь участников и прецедентов.

Сообщения, передающиеся от одного объекта к другому, представляются стрелками между линиями жизни этих объектов. Порядок следования сообщений устанавливается сверху вниз. Каждое сообщение имеет как минимум имя сообщения, а также может иметь аргументы и некоторую управляющую информацию.

Для того чтобы показать момент времени, в который объект является активным, используются прямоугольники активности, их изображают поверх линии жизни.

Управляющая информация может быть представлена в виде условия, которое указывает, когда сообщение может быть передано, либо при помощи маркера итерации, показывающего, что сообщение посылается множество раз. Такая итерация дается в следующем виде: *[условие итерации].

В зависимости от типа сообщение изображается при помощи различных линий:

- вызов процедуры или другого потока управления;
- поток управления, показывает направление потока управления и последовательности шагов;
- ""- асинхронное стимулирующее воздействие;
- в возврат из процедуры, используется для того, чтобы показать, что каждая процедура возвращает управление после своего завершения. Обычно стрелка возврата указывается только в том случае, если это вносит в диаграмму дополнительную ясность.

В качестве элементов диаграммы последовательностей могут применяться так называемые фрагменты взаимодействия (InteractionFragment), которые по своим свойствам не отличаются от полного взаимодействия объектов и позволяют наглядным образом группировать последовательности сообщений.

Включение (InteractionUse) представляет собой ссылку на имеющееся в модели взаимодействие вместо копирования последнего на диаграмму, что поддерживает декомпозицию и повторное использование отдельных взаимодействий. Включение обозначается как фрагмент взаимодействия с оператором «ref», содержимое которого состоит из названия соответствующего взаимодействия.

Комбинированный фрагмент (CombinedFragment) представляет собой выражение над фрагментами взаимодействия, состоящее из оператора и операндов - фрагментов взаимодействия. С каждым операндом может быть связано сторожевое условие. В языке UML над фрагментами взаимодействий определены следующие основные операторы:

- alt – выполняется не более одного операнда, для которого сторожевое условие истинно;
- opt – необходимость выполнения единственного операнда определяется его сторожевым условием;
- break – единственный операнд выполняется вместо всех остальных сообщений в объемлющем фрагменте взаимодействия в случае, если его

сторожевое условие истинно;

- loop – циклическое выполнение единственного операнда, минимальное и максимальное количество итераций указывается в операнде;
- par – параллельное выполнение операндов при сохранении последовательности сообщений в каждом из них;
- strict – строго последовательное выполнение операндов;
- seq – условно последовательное выполнение операндов, при котором упорядочиваются только сообщения, относящиеся к одной и той же линии жизни; в последнем случае сообщение из первого операнда выполняется прежде сообщения из второго операнда и т.д.;
- critical – критический участок, операнды выполняются без перекрытия во времени с любыми другими сообщениями, относящимися к тем же линиям жизни, которые задействованы в операндах.

На рисунке 6.2 приведен пример условного комбинированного фрагмента opt, включающего единственный операнд, для которого определено необходимое сторожевое условие.

Инвариант состояния (StateInvariant) представляет собой ограничение на состояние объекта в процессе выполнения взаимодействия. Если данное ограничение не выполняется, то вся предыдущая последовательность сообщений считается неверной. Пример инварианта состояния приведен на рис. 5, где с его помощью показано, что возможна отмена только тех заказов, которые еще не доставлены.

Диаграммы последовательностей являются наглядным и легко читаемым средством описания функционирования системы. Они помогают быстрее разобраться в процессах поведения системы.

Диаграмма коммуникации

Диаграмма коммуникации (communication diagram) отображает ту же информацию, что и диаграмма последовательностей, но на диаграмме коммуникации зависимость от времени указывается посредством нумерации сообщений. На диаграмме коммуникации отражается распределение процессов между объектами и их зависимости друг от друга, что очень полезно при разработке различных проектов. Основной целью построения данной диаграммы является понимание структурной организации занятых в системе объектов, принимающих и передающих сообщения. На рис. 6.5 показан пример диаграммы коммуникации.

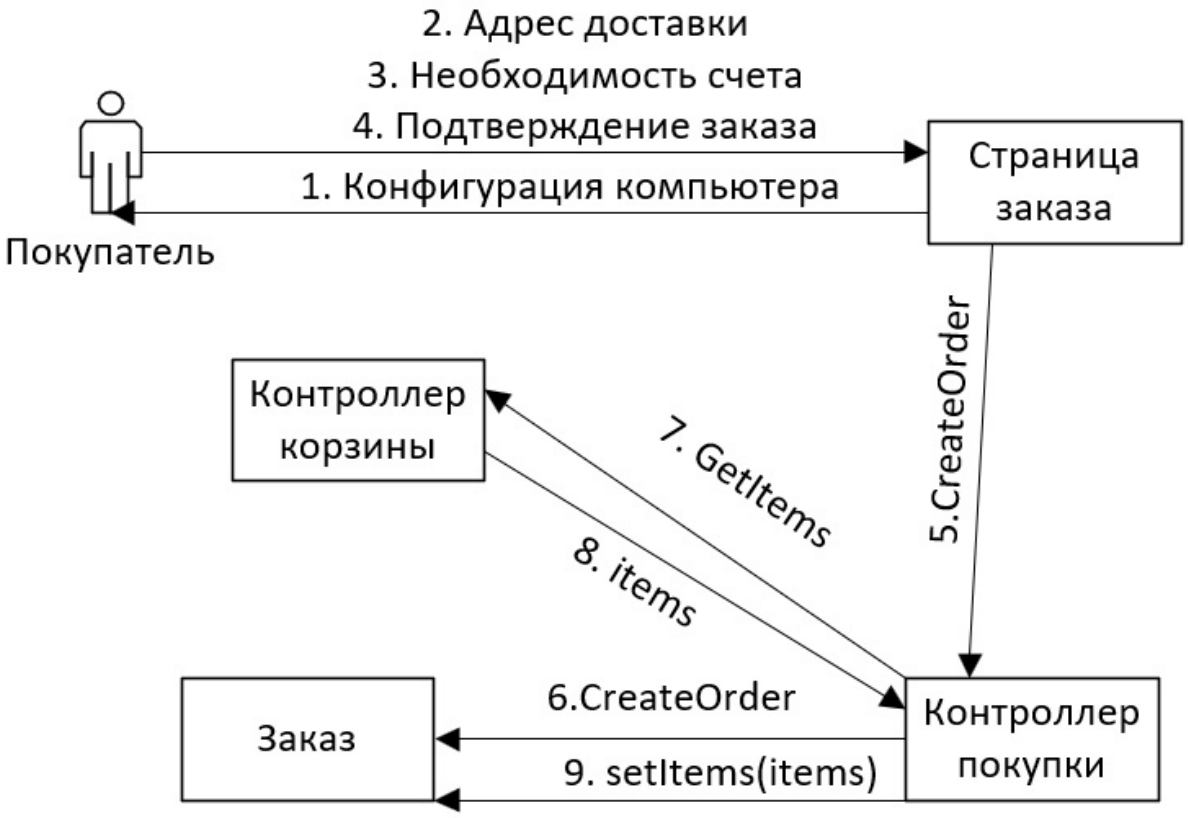


Рис. 6.5. Пример диаграммы коммуникации

Задание для выполнения:

1. Изучить теоретические данные по диаграммам видов деятельности, последовательностей и коммуникации;
2. Провести анализ предметной области с целью изучения процессов, проходящих в системе;
3. Построить диаграмму видов деятельности в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Деятельность UML);
4. Построить диаграмму последовательностей в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Последовательности UML);
5. Построить диаграмму коммуникации в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Связь UML);
6. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, диаграмма видов деятельности, диаграмма последовательностей, диаграмма коммуникации);
4. Выводы по работе.

Контрольные вопросы:

1) Перечислите виды диаграмм поведения в языке UML, поясните их основные отличия.
2) Дайте определение действия и деятельности, назовите их основные отличия.
3) Назовите и приведите условные обозначения основных элементов диаграмм последовательностей.
4) Поясните связь диаграмм поведения с моделью прецедентов.

Лабораторная работа №7

Построения диаграммы компонентов и диаграммы развертывания

Цель работы: изучить основы моделирования архитектуры системы на примере диаграммы компонентов и диаграммы развертывания и получить практические навыки их проектирования.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Компоненты — это физические части системы. Следовательно, проектирование компонент нельзя отделить от платформы реализации. Проектируемая система онлайновой торговли — это Web-приложение с сервером баз данных. Web-приложение - это Web-система, позволяющая ее пользователям работать в соответствии с заложенной в ней бизнес - логикой с помощью Web-браузера. Программа, поддерживающая бизнес-логику, может находить на сервере и/или на клиенте. Следовательно, Web-приложение - не что иное, как разновидность системы клиент-сервер с Web-узлом. Относительно нашего примера, рассмотрим типичную последовательность доступа к Web- страницам. Первая Web-страница, которую может посетить пользователь, — это Web-страница поставщика, на которой перечислены группы изделий (серверные, настольные системы, портативные компьютеры) и приводятся ссылки на Web-страницы, на которых представлены перечни изделий и дано краткое описание каждого изделия. Это единый функциональный блок, который может образовать компонент ProductList (Перечень изделий) и т.д. В соответствии с этим построим диаграмму компонентов. Важно отметить сходство этой диаграммы с диаграммой пакетов прецедентов, что неудивительно, поскольку пакеты прецедентов и компоненты представляют собой функциональные модули с четким границами.

Итак, на диаграмме компонентов отражаются программные компоненты системы.

Характер Internet-систем без установления прямого соединения делает развертывание Web-приложений значительно более сложной задачей, чем развертывание приложений баз данных в архитектуре клиент-сервер. Чтобы приступить к развертыванию, требуется установить Web-сервер в качестве пункта маршрутизации между всеми браузерами клиентов и базой данных. На рис. 7.3 показана диаграмма развертывания для нашего примера. Система онлайновой торговли развернута без отдельного сервера приложений.

Таким образом, диаграмма развертывания отражает систему в ее работе с программными и аппаратными ресурсами.

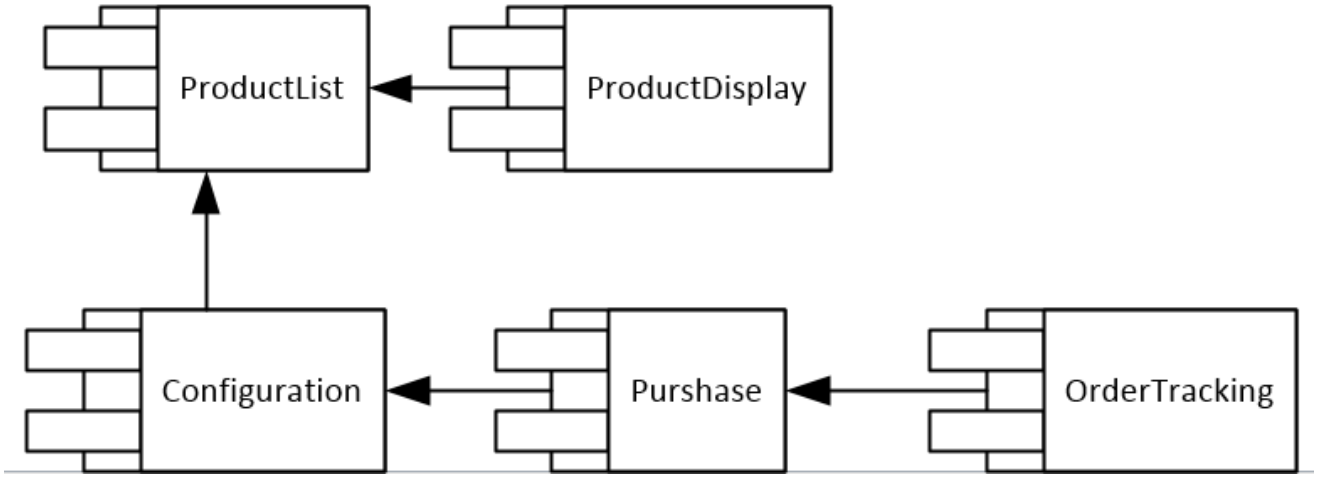


Рисунок 7.1. Пример диаграммы компонентов

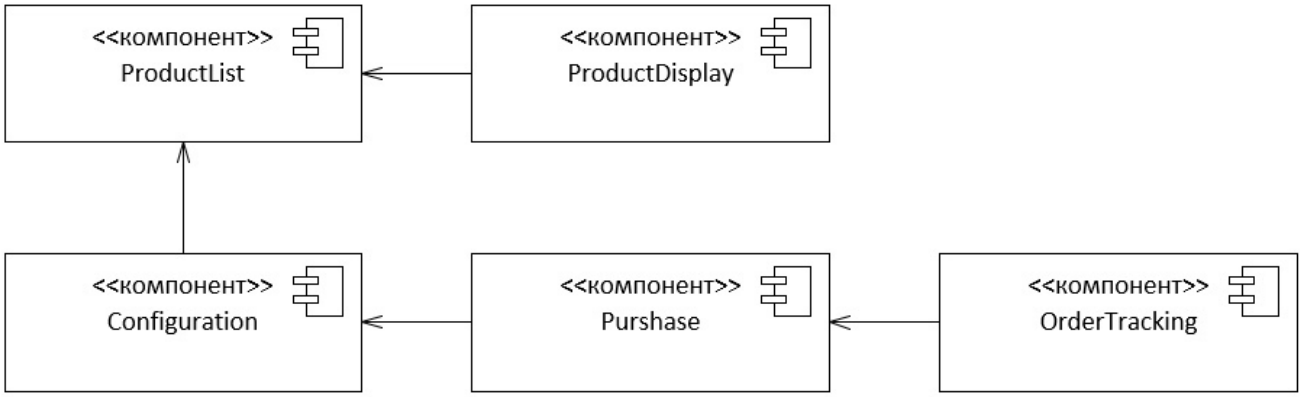


Рисунок 7.2. Пример диаграммы компонентов

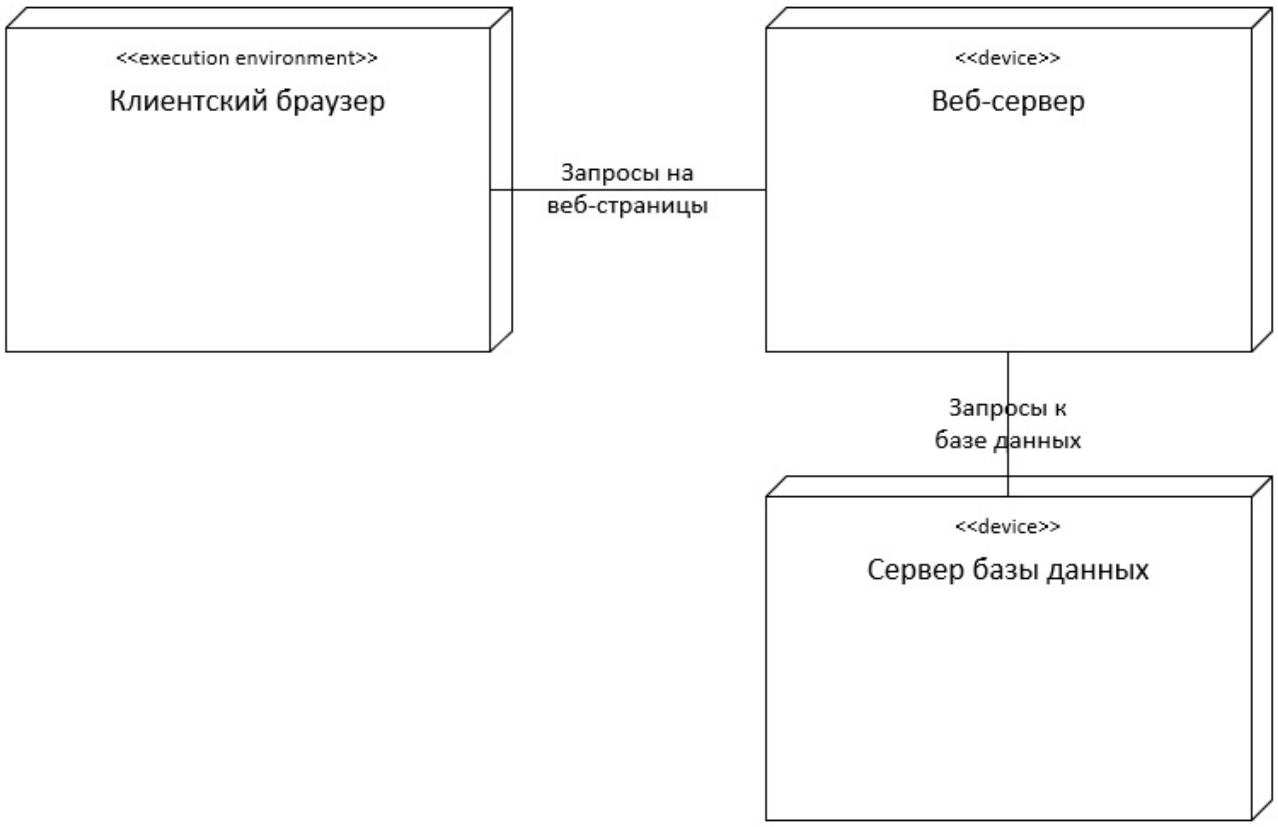


Рисунок 7.3. Пример диаграммы развертывания

Задание для выполнения:

1. Изучить теоретические данные по диаграммам компонентов и развертывания;
2. Провести анализ предметной области с целью выявления физических компонентов и узлов системы;
3. Построить диаграмму компонентов в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Компоненты UML);
4. Построить диаграмму развертывания в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Развертывание UML);
5. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;

3. Ход работы (описание предметной области, диаграмма компонентов, диаграмма развертывания);
4. Выводы по работе.

Контрольные вопросы:

1. Поясните сущность диаграмм компонентов.
2. Поясните сущность диаграмм развертывания.

Лабораторная работа №8

Построения диаграммы классов и диаграммы состояний

Цель работы: изучить процесс анализа и проектирования в части определения потенциальной архитектуры системы и освоить построение диаграммы классов и диаграммы состояний объектов классов.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Диаграмма классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные отношения между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Имеется два вида основных статических отношений:

- ассоциации (человек может сделать покупку в магазине);
- подтипы (корпоративный клиент является разновидностью клиента).

На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между объектами (рисунок 8.1).

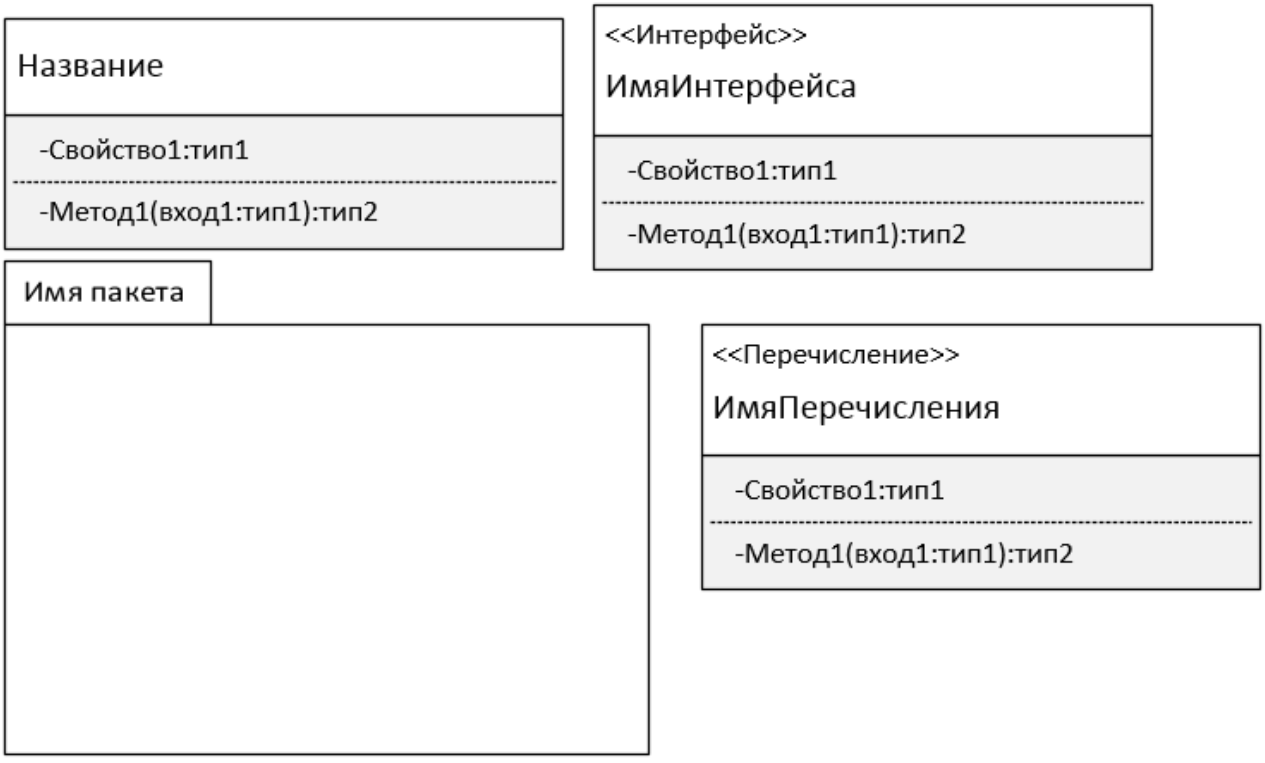


Рисунок 8.1. Элементы диаграммы классов

На рисунке 8.2 представлен пример диаграммы классов.

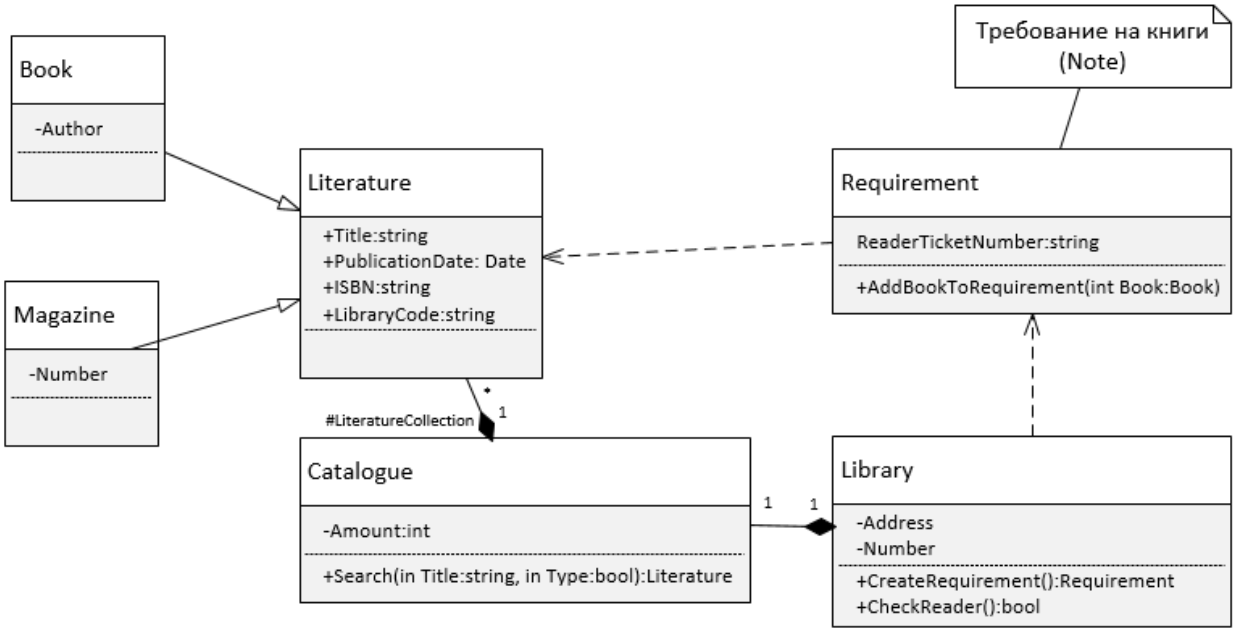


Рисунок 8.2. Пример диаграммы классов

Отношения ассоциации представляют собой отношения между экземплярами классов. Каждая ассоциация имеет два конца ассоциации**,** которыми она присоединяется к классам на диаграмме, а конец ассоциации, в свою очередь, обладает кратностью, которая показывает, сколько объектов может участвовать в данном отношении.

Разновидностью ассоциации является агрегирование. Агрегация – это когда класс в своей структуре использует часть другого класса. Например, можно сказать, что процессор или жесткий диск являются частью компьютера.

Существует более сильная разновидность агрегации – композиция. При композиции класс является целиком частью другого класса. Например, книга в каталоге целиком является частью каталога.

В общем случае кратность указывает верхнюю и нижнюю границы количества объектов, которые могут участвовать в отношении. Часто используемые варианты кратности:

1 - означает, что в ассоциации участвует один и только один экземпляр класса, с которым связана ассоциация;

* - в ассоциации может участвовать неограниченное число экземпляров класса;

0..1 - в ассоциации участвует либо один, либо ни одного экземпляра класса;

0..N - в ассоциации участвует от 0 до N экземпляров класса.

Стрелками в ассоциации обозначается направление навигации**,** таким образом, если в ассоциации присутствует стрелка, то она из симметричной преобразуется в одностороннюю.

Если навигация указана только в одном направлении, то такая ассоциация называется однонаправленной, а если навигация указана с обеих сторон, то ассоциация считается двунаправленной. Если ассоциация на диаграмме не имеет стрелок навигации, то она является двунаправленной.

Связь, заданная при помощи ассоциации, существует в течение всего жизненного цикла объектов, даже если соединяемые ею экземпляры классов могут изменяться во времени.

Существует еще один способ связи классов – обобщение. Обобщение – это отношение между общей сущностью (суперклассом, или родителем) и её конкретным воплощением (субклассом, или потомком). Обобщения иногда называют отношениями типа «является», имея в виду, что одна сущность (например, класс Книга (Book) или Журнал (Magazine)) является частным выражением другой, более общей (скажем, класса Литературное издание (Literature)).

Класс может иметь одного или нескольких родителей или не иметь их вовсе. Класс, у которого нет родителей, но есть потомки, называется базовым (base) или корневым (root), а тот, у которого нет потомков, - листовым (leaf). О классе, у которого есть только один родитель, говорят, что он использует одиночное наследование (Single inheritance); если родителей несколько, речь идет о множественном наследовании (Multiple inheritance).

Чтобы показать, что один из классов реализует поведение, специфицированное в другом классе, используют реализацию.

Для объединения часто повторяющихся групп блоков используют пакеты (Packages). В пакет можно поместить структурные, поведенческие, и даже другие группирующие сущности. В отличии от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки.

С целью повышения информативности диаграммы можно использовать заметки (Note).

Атрибуты являются элементами класса, определяющими его сущность. В синтаксисе UML описание атрибута выглядит следующим образом: <видимость><имя>:<тип>=<значение по умолчанию>. В примере на рис. 8 атрибутами являются: «Имя», «Адрес», «ЛимитКредита» и др.

Процессы, реализуемые классами, представляют собой *операции*. Синтаксис операции в UML выглядит следующим образом:

<видимость><имя>(<список параметров>):<выражение, возвращающее значение типа>(<строка свойств>), где:

видимость - принимает одно из трех значений: «+» -

общедоступная (public), «#» - защищенная (protected) либо «-» - закрытая (private);

имя - строка символов;

список параметров - содержит перечисленные через запятую параметры, которые описываются так же, как и атрибуты;

выражение, возвращающее значение типа - содержит перечисленные через запятую значения типов;

строка свойств - указывает свойства, которые имеются у данной операции.

При построении диаграммы прецедентов, являющейся наиболее общей концептуальной моделью проектируемой системы, применение русскоязычных терминов является не только оправданным с точки зрения описания структуры предметной области, но и эффективным с точки зрения взаимодействия с заказчиком и пользователями. При построении остальных типов диаграмм следует придерживаться разумного компромисса.

Диаграмма состояний

Диаграмма состояний (state machine diagram) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения, позволяя описать поведение объекта в различных прецедентах. Обычно диаграммы состояний строятся для единственного класса, чтобы показать динамику поведения единственного объекта. На рисунке 8.3 представлен пример диаграммы состояний.

Рисунок 8.3. Пример диаграммы состояний

Диаграмма состояний — это конечный автомат, реализованный средствами UML. Существует несколько разновидностей диаграмм состояний, в UML принята нотация Дэвида Харела. Рассмотрим основные элементы диаграммы состояний (рисунок 8.4).

Состояние (State) отображает одно из возможных состояний, в котором может находиться объект. Кроме имени в элементе State может содержаться также краткое описание состояния и деятельности, осуществляемой в этом

состоянии. В общем случае состояние определяют следующие характеристики:

входное воздействие - поведение, которое наступает при переходе объекта в данное состояние. Входное действие не прерывается и всегда выполняется до конца;

деятельность - поведение, которое реализует объект, находящийся в данном состоянии;

выходное действие - действие, которое выполняется при выходе объекта из текущего состояния.

Входные и выходные действия отображаются внутри графического элемента State под горизонтальной чертой, а друг от друга отделяются наклонной чертой «/» или двоеточием.

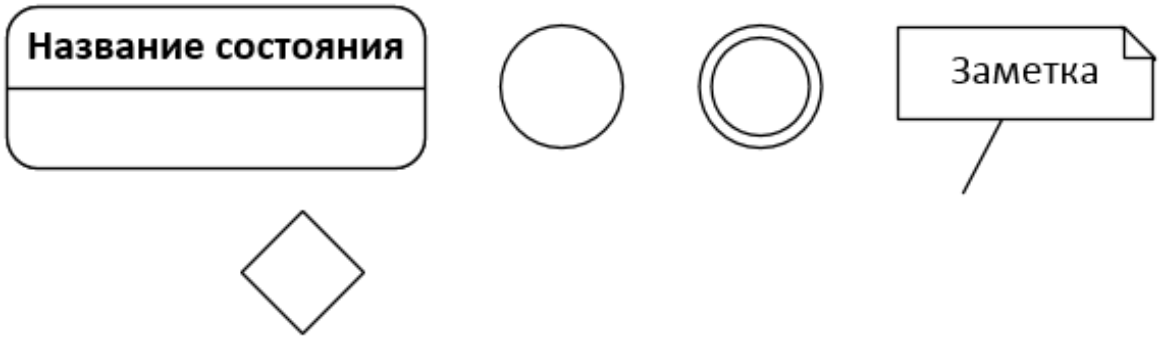


Рисунок 8.4. Элементы диаграммы состояний

Переход (Transition) объекта из одного состояния в другое отображается направленной стрелкой. Синтаксис метки перехода состоит из трех частей, каждая из которых является необязательной: Событие: [Сторожевое

условие]/Действие.

Событие (Event) — это некоторый факт, который инициирует переход из одного состояния в другое. События отображаются в виде поясняющей надписи около стрелки перехода.

Начальное состояние (Initial) — это состояние, в котором объект находится непосредственно после его создания. Начальное состояние — это обязательный элемент диаграммы, причем на диаграмме может быть только один такой элемент (изображается черным кружком), стрелка перехода соединяет его с первоначальным состоянием объекта.

Конечное состояние (Final) - состояние, в котором объект пребывает непосредственно перед его уничтожением. Это необязательный элемент, причем количество таких элементов на диаграмме не ограничено.

Заметка – это комментарий, поясняющие элементы диаграммы состояний.

Выбор – это элемент, позволяющий показать выбор следующего состояния при необходимости выполнения какого-либо условия.

Задание для выполнения:

1. Изучить теоретические данные по диаграмме классов и диаграмме состояний;
2. Провести анализ предметной области с целью выявления сущностей и связей между ними;
3. Построить диаграмму классов в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Класс UML);
4. Построить диаграмму состояний для выбранного в диаграмме классов объекта в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Конечный автомат UML);
5. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, список выделенных сущностей, диаграмма классов, диаграмма состояний (при необходимости пояснения к ним));
4. Выводы по работе.

Контрольные вопросы:

1. Дайте определение класса и его свойств (атрибутов и операций);
2. Дайте определение отношения обобщения между классами;
3. Дайте определение отношения ассоциации между классами и перечислите его основные характеристики;
4. Назовите виды структурных диаграмм в языке UML.

Лабораторная работа №9

Построения диаграммы IDEF0

Цель работы: изучить процесс формализации и описания бизнес-процессов с использованием нотации IDEF0 и получить практические навыки построения диаграмм по этой нотации.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

IDEF0 (Integrated Definition Function Modeling) – методология функционального моделирования. В основе IDEF0 методологии лежит понятие блока, который отображает некоторую бизнес-функцию. Четыре стороны блока имеют разную роль: левая сторона имеет значение "входа", правая - "выхода", верхняя - "управления", нижняя - "механизма" (рисунок 9.1).

Взаимодействие между функциями в IDEF0 представляется в виде дуги, которая отображает поток данных или материалов, поступающий с выхода одной функции на вход другой. В зависимости от того, с какой стороной блока связан поток, его называют соответственно "входным", "выходным", "управляющим".

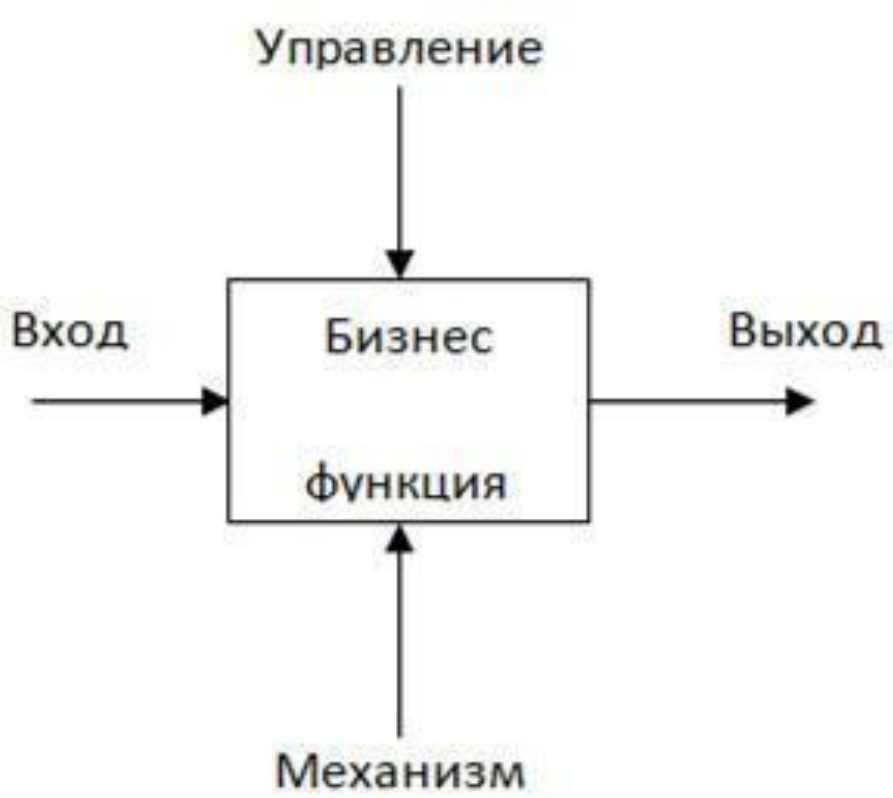


Рисунок 9.1. Функциональный блок

Принципы моделирования в IDEF0

В IDEF0 реализованы три базовых принципа моделирования процессов:

- принцип функциональной декомпозиции;
- принцип ограничения сложности;
- принцип контекста.

Принцип функциональной декомпозиции представляет собой способ моделирования типовой ситуации, когда любое действие, операция, функция могут быть разбиты (декомпозированы) на более простые действия, операции, функции. Другими словами, сложная бизнес-функция может быть представлена в виде совокупности элементарных функций. Представляя функции графически, в виде блоков, можно как бы заглянуть внутрь блока и детально рассмотреть ее структуру и состав (рисунок 9.2).

Принцип ограничения сложности. При работе с IDEF0 диаграммами существенным является условие их разборчивости и удобочитаемости. Суть принципа ограничения сложности состоит в том, что количество блоков на диаграмме должно быть не менее двух и не более шести. Практика показывает, что соблюдение этого принципа приводит к тому, что функциональные процессы, представленные в виде IDEF0 модели, хорошо структурированы, понятны и легко поддаются анализу.

Принцип контекстной диаграммы. Моделирование делового процесса начинается с построения контекстной диаграммы. На этой диаграмме отображается только один блок - главная бизнес-функция моделируемой системы. Если речь идет о моделировании целого предприятия или даже крупного подразделения, главная бизнес-функция не может быть сформулирована как, например, "продавать продукцию". Главная бизнес-функция системы — это "миссия" системы, ее значение в окружающем мире. Нельзя правильно сформулировать главную функцию предприятия, не имея представления о его стратегии.

При определении главной бизнес-функции необходимо всегда иметь ввиду цель моделирования и точку зрения на модель. Одно и то же предприятие может быть описано по-разному, в зависимости от того, с какой точки зрения его рассматривают: директор предприятия и налоговой инспектор видят организацию совершенно по-разному.

Контекстная диаграмма играет еще одну роль в функциональной модели. Она "фиксирует" границы моделируемой бизнес-системы, определяя то, как моделируемая система взаимодействует со своим окружением. Это достигается за счет описания дуг, соединенных с блоком, представляющим главную бизнес-функцию.

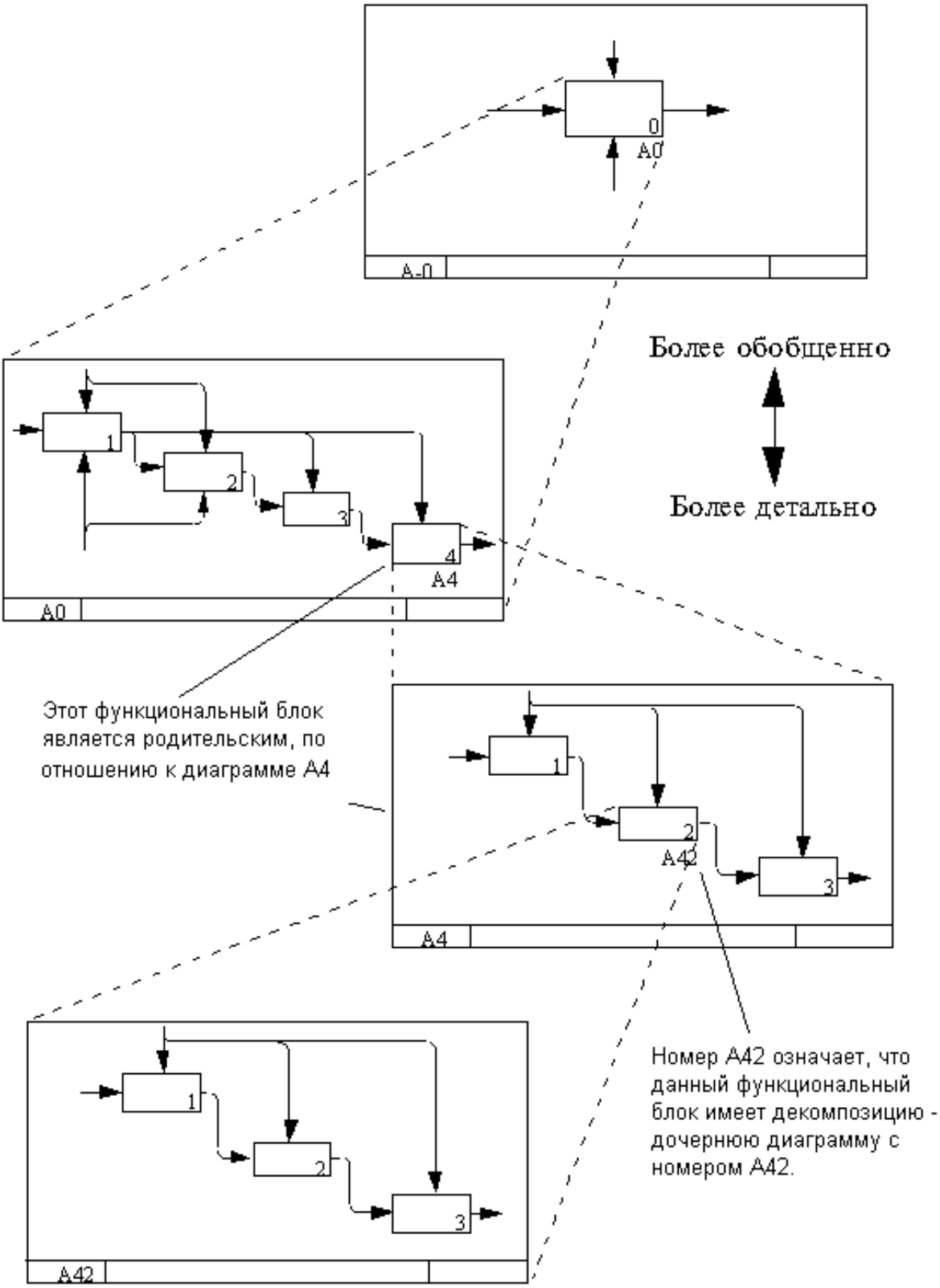


Рисунок 9.2. Декомпозиция функционального блока

Пример.

На рисунках 9.3 и 9.4 представлен пример построения функциональной диаграммы, описывающей изготовление изделия. Рисунок 9.3 - контекстная диаграмма. Рисунок 9.4 – первый уровень декомпозиции.

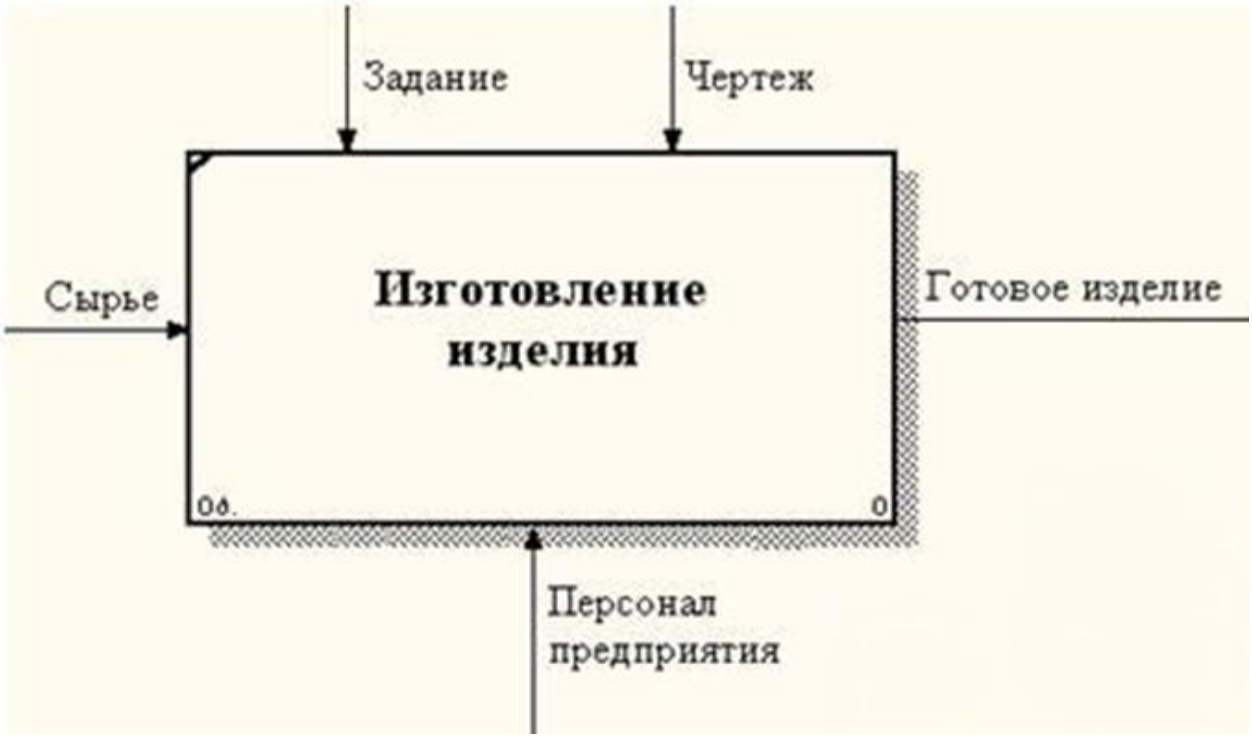


Рисунок 9.3. Контекстная диаграмма

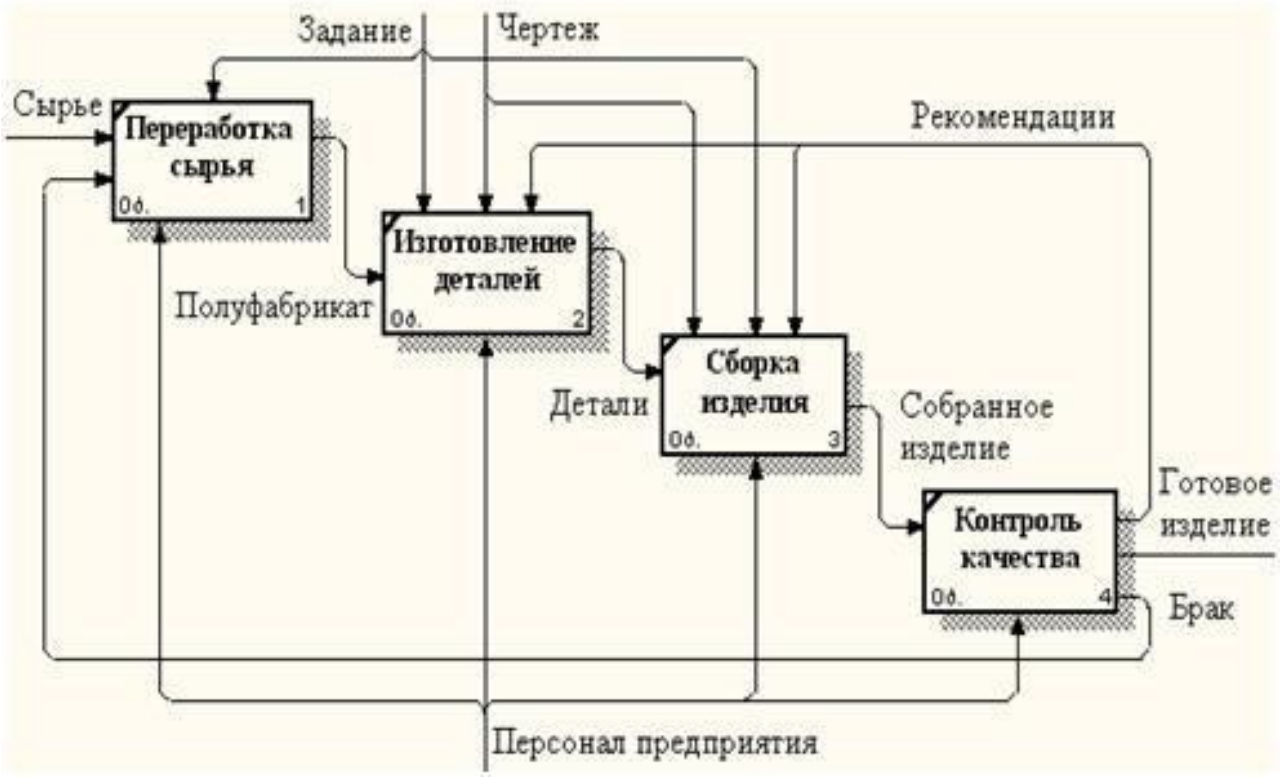


Рисунок 9.4. Диаграмма первого уровня декомпозиции

Применение IDEF0

Существует два ключевых подхода к построению функциональной модели: построение “как есть” и построение “как будет”.

Построение модели “как есть”. Обследование предприятия является обязательной частью любого проекта создания или развития корпоративной информационной системы.

Построение функциональной модели “как есть” позволяет четко зафиксировать, какие деловые процессы осуществляются на предприятии, какие информационные объекты используются при выполнении деловых процессов и отдельных операций. Функциональная модель “как есть” является отправной точкой для анализа потребностей предприятия, выявления проблем и "узких" мест и разработки проекта совершенствования деловых процессов.

Построение модели “как будет”. Создание и внедрение корпоративной информационной системы приводит к изменению условий выполнения отдельных операций, структуры деловых процессов и предприятия в целом. Это приводит к необходимости изменения системы бизнес-правил, используемых на предприятии, модификации должностных инструкций сотрудников. Функциональная модель “как будет” позволяет уже на стадии проектирования будущей информационной системы определить эти изменения. Применение функциональной модели “как будет” позволяет не только сократить сроки внедрения информационной системы, но также снизить риски, связанные с невосприимчивостью персонала к информационным технологиям.

Совершенно не обязательно каждый раз самим придумывать решения для стандартных задач. Всегда, когда Вы сталкиваетесь с необходимостью анализа той или иной функциональной системы (от системы проектирования космического корабля, до процесса приготовления комплексного ужина) – используйте годами проверенные и обкатанные методы. Одним из таких методов и является IDEF0, позволяющий с помощью своего простого и понятного инструментария решать сложные жизненные задачи.

Задание для выполнения:

1. Изучить теоретические данные по нотации IDEF0;
2. Провести анализ предметной области с целью выявления бизнес-процессов;
3. Построить диаграмму IDEF0 контекстного уровня и ее декомпозицию 1 уровня в MS Visio (выбрать набор элементов: Дополнительные фигуры – Блок-схема – Фигуры схемы IDEF0);
4. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;

3. Ход работы (описание предметной области, глоссарий, диаграмма IDEF0 контекстного уровня, декомпозиция первого уровня контекстной диаграммы IDEF0);
4. Выводы по работе.

Контрольные вопросы:

1. Для чего используются диаграммы IDEF0?
2. Перечислите и поясните основные элементы нотации IDEF0.

Лабораторная работа №10

Построения диаграммы IDEF3

Цель работы: изучить процесс формализации и описания бизнес-процессов с использованием нотации IDEF3 и получить практические навыки построения диаграмм по этой нотации.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Для описания логики взаимодействия информационных потоков используется IDEF3, называемая также work flow diagram, — методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Work flow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 — это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов IDEF3 не ограничивает аналитика чрезмерно жёсткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим процесс действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть документирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо документировать цель модели — те вопросы, на которые призвана ответить модель.

Диаграмма является основной единицей описания в IDEF3. Важно правильно построить диаграммы, поскольку они предназначены для чтения другими людьми (а не только автором).

Единицы работы — Unit of Work (UOW) — также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы (например, "Изготовление изделия"). Часто имя существительное в имени работы меняется в процессе моделирования, поскольку модель может уточняться и редактироваться. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее идентификатор не будет вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправленные и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи:

- Связь предшествования (Precedence) – показывает, что, прежде чем начнется работа-приемник, должна завершиться работа-источник. Обозначается сплошной линией.
- Связь отношения (Relational) - показывает связь между двумя работами или между работой и объектом ссылки. Обозначается пунктирной линией.
- Связь поток объектов (Object Flow) – показывает участие некоторого объекта в двух или более работах, как, например, если объект производится в ходе выполнения одной работы и потребляется другой работой. Обозначается стрелкой с двумя наконечниками.

Окончание одной работы может служить сигналом к началу нескольких *работ*, или же одна работа для своего запуска может ожидать окончания нескольких *работ*. Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы, используются *перекрестки* (Junction). Различают

перекрестки для слияния (*Fan*-in *Junction*) и разветвления стрелок (*Fan*-out *Junction*). Перекресток не может использоваться одновременно для слияния и для разветвления.

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства перекрестка при помощи диалога Junction Properties, который вызывается в контекстном меню перекрестка командой Definition/Note. В отличие от IDEF0 и DFD в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки. Перекрестки представлены в рисунке 10.1

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
&	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
&	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
O	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
O	Synchronous OR	Один или несколько предшествующих процессов завершаются одновременно	Один или несколько следующих процессов запускаются одновременно
X	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Рисунок 10.1. Перекрестки IDEF3

Пример диаграммы IDEF3 приведен на рисунке 10.2.

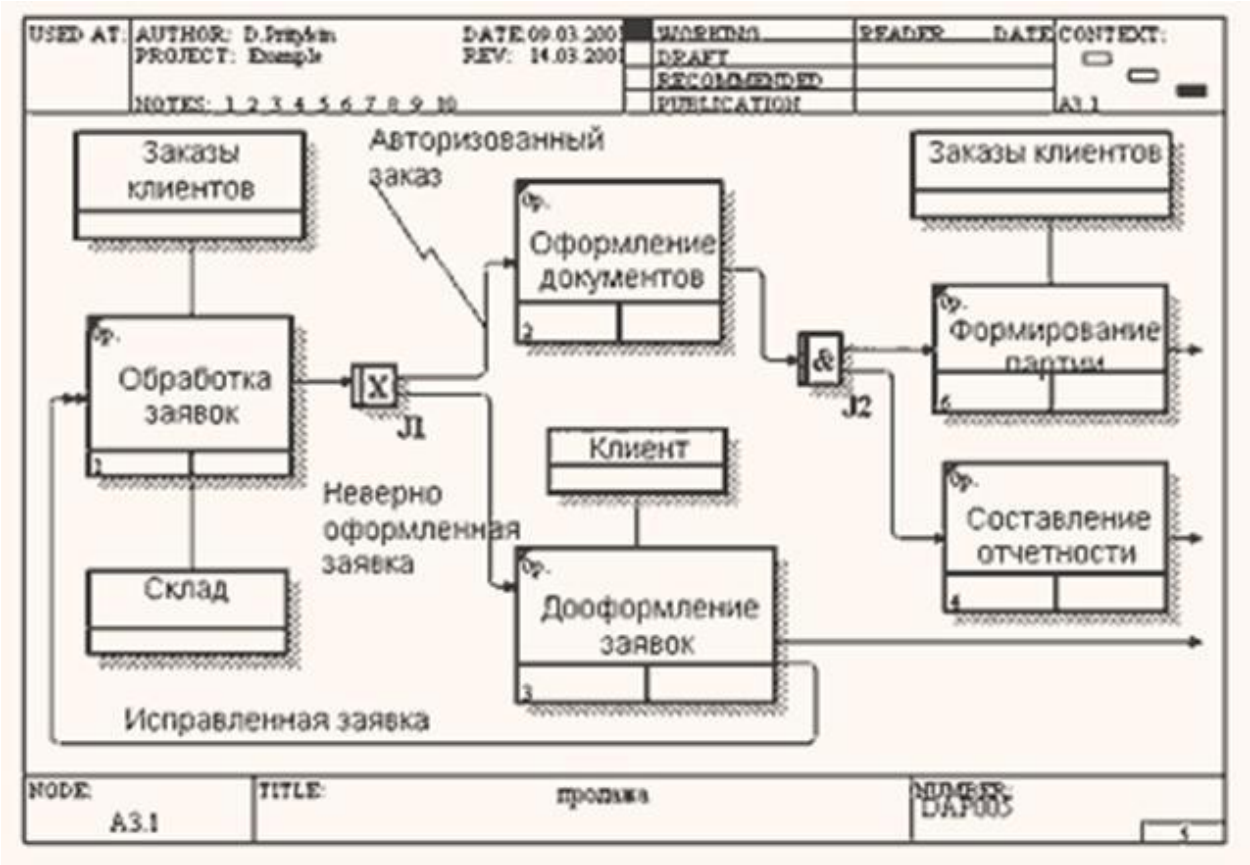


Рисунок 10.2. Диаграмма IDEF3

Задание для выполнения:

1. Изучить теоретические данные по нотации IDEF0;
2. Провести анализ предметной области с целью выявления бизнес-процессов;
3. Построить диаграмму IDEF0 контекстного уровня и ее декомпозицию 1 уровня в MS Visio (выбрать набор элементов: Дополнительные фигуры – Блок-схема – Фигуры схемы IDEF0);
4. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, глоссарий, диаграмма IDEF0 контекстного уровня, декомпозиция первого уровня контекстной диаграммы IDEF0);
4. Выводы по работе.

Контрольные вопросы:

1. Для чего используются диаграммы IDEF3?
2. Перечислите и поясните основные элементы нотации IDEF3;
3. Перечислите и поясните перекрестки;

Лабораторная работа №11

Построения диаграммы DFD

Цель работы: изучить процесс функционального проектирования программного обеспечения на примере нотации диаграммы потоков данных и получить практические навыки построения диаграмм DFD.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Диаграммы потоков данных (DFD)

Диаграммы потоков данных (Data Flow Diagrams - DFD) используются для описания движения документов и обработки информации как дополнение к IDEF0. В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой. DFD отражает функциональные зависимости значений, вычисляемых в системе, включая входные значения, выходные значения и внутренние хранилища данных. DFD — это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.

DFD содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

Диаграмма потоков данных содержит:

процессы, которые преобразуют данные;

потоки данных, переносящие данные;

активные объекты, которые производят и потребляют данные;

хранилища данных, которые пассивно хранят данные.

Процесс DFD преобразует значения данных и изображается в виде эллипса, внутри которого помещается имя процесса (рисунок 11.1).

Рисунок 11.1. Процессы

Поток данных соединяет выход объекта (или процесса) с входом другого объекта (или процесса) и представляет собой промежуточные данные вычислений. Поток данных изображается в виде стрелки между производителем и потребителем данных, помеченной именами соответствующих данных. Дуги могут разветвляться или сливаться, что означает соответственно разделение потока данных на части либо слияние объектов (рисунок 11.2).

Активным объектом является объект, который обеспечивает движение данных, поставляя или потребляя их.

Рисунок 11.2. Потоки данных

Хранилище данных — это пассивный объект в составе DFD, в котором данные сохраняются для последующего доступа. Хранилище данных допускает доступ к хранимым в нем данным в порядке, отличном от того, в котором они были туда помещены. Агрегатные хранилища данных, как, например, списки и таблицы, обеспечивают доступ к данным в порядке их поступления, либо по ключам.

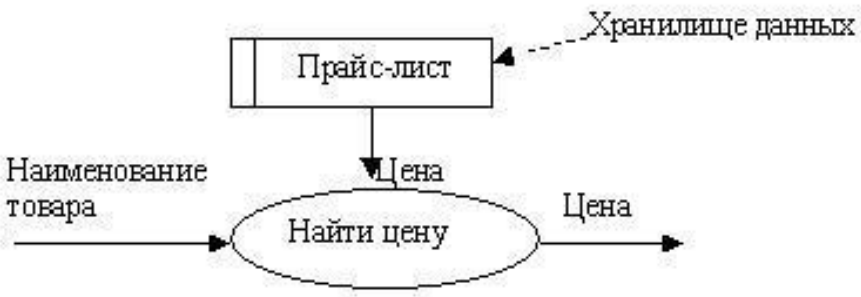


Рисунок 11.3. Хранилище данных

DFD показывает все пути вычисления значений, но не показывает в каком порядке значения вычисляются. Решения о порядке вычислений связаны с управлением программой, которое отражается в динамической модели. Эти решения, вырабатываемые специальными функциями, или предикатами, определяют, будет ли выполнен тот или иной процесс, но при этом не передают процессу никаких данных, так что их включение в функциональную модель необязательно. Тем не менее, иногда бывает полезно включать указанные предикаты в функциональную модель, чтобы в ней были отражены условия выполнения соответствующего процесса. Функция, принимающая решение о запуске процесса, будучи включенной в DFD, порождает в диаграмме поток управления и изображается пунктирной стрелкой.

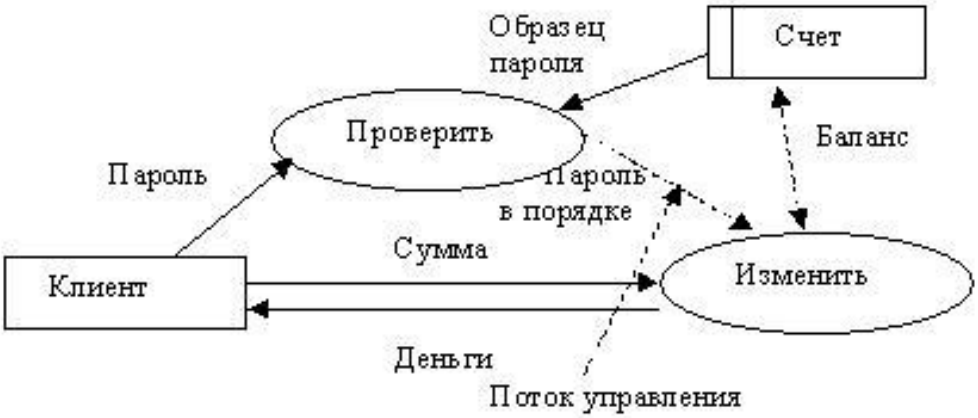


Рисунок 11.4. Потоки данных

Первым шагом при построении иерархии DFD является построение контекстных диаграмм. Обычно при проектировании относительно простых информационных систем строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и

источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и, кроме того, главный единственный процесс не раскрывает структуры распределенной системы.

Для сложных информационных систем строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не главный единственный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

При построении иерархии DFD переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т.п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

Ниже на рисунке 11.5 приведена диаграмма потоков данных верхнего уровня с ее последующим уточнением:

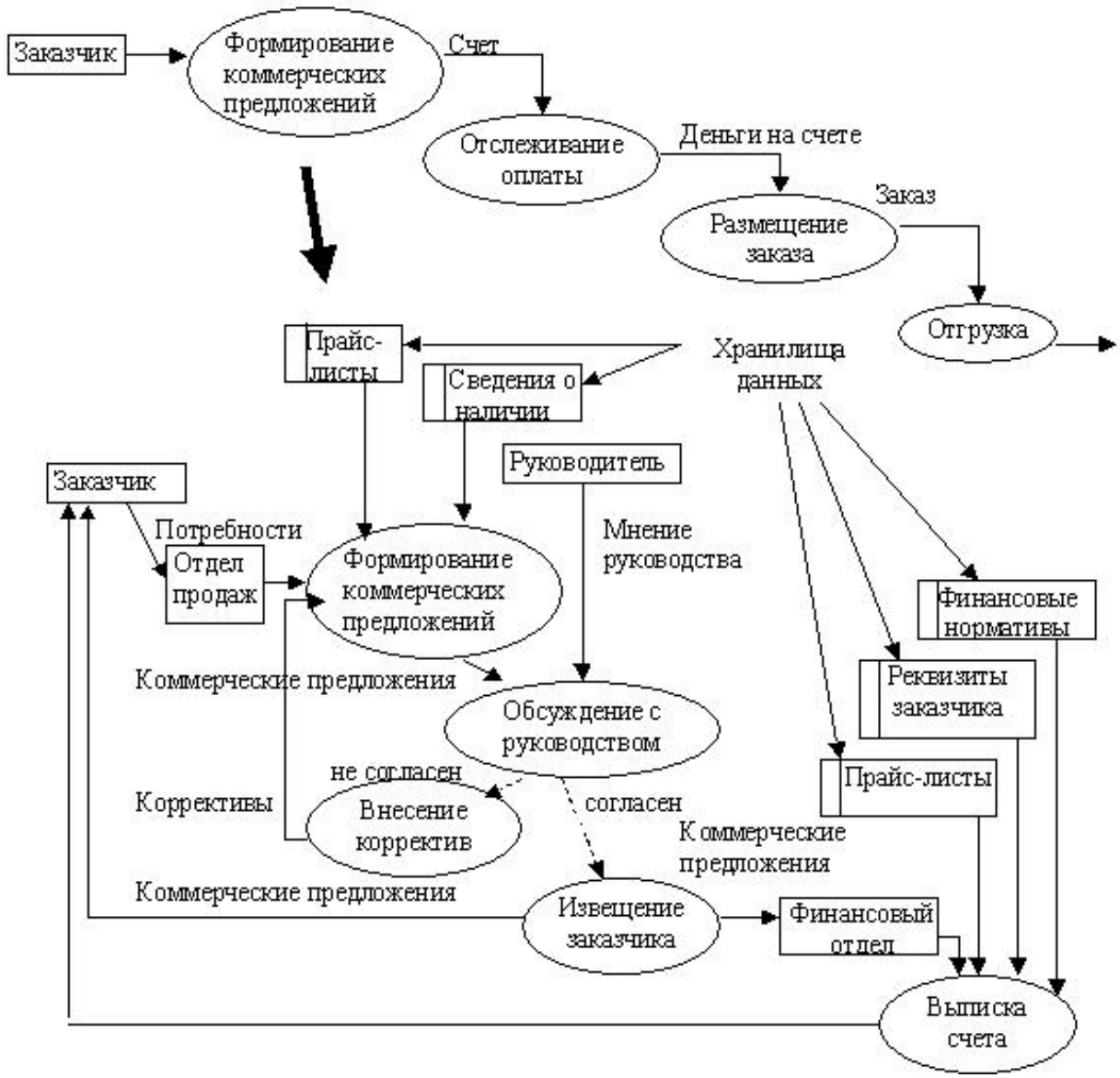


Рисунок 11.5. Диаграмма DFD контекстного уровня и декомпозиция первого уровня процесса «Формирование коммерческих предложений»

Задание для выполнения:

1. Изучить теоретические данные по диаграмме потоков данных;
2. Провести анализ предметной области с целью выявления процессов и информационных потоков между ними;
3. Построить диаграмму потоков данных контекстного уровня и ее декомпозицию 1 уровня в MS Visio (выбрать набор элементов: Дополнительные фигуры – Программы и базу данных – Программное обеспечение – Фигуры схемы потоков данных);
4. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, диаграмма DFD контекстного уровня, декомпозиция первого уровня контекстной диаграммы DFD);
4. Выводы по работе.

Контрольные вопросы:

1. Для чего используются диаграммы DFD?
2. Перечислите и поясните основные элементы диаграммы DFD.

Лабораторная работа №12

Построения ER диаграмм

Цель работы: изучить процесс моделирования баз данных путем построения диаграмм «сущность-связь».

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Понятие предметной области

Основным назначением информационных систем (ИС), в том числе и систем складирования данных, является оперативное обеспечение пользователя информацией о внешнем мире. Поэтому для ИС выделяют его фрагмент— предметную область, который и будет воплощен в системе. Информация о внешнем мире представляется в ИС в форме данных. Это ограничивает возможности смысловой интерпретации информации и конкретизирует семантику ее представления в ИС. Совокупность этих выделенных данных для ИС образует логическую модель предметной области, описывающие ее состояние с определенной точностью.

Важно понимать, что логическая модель предметной области создается на этапе анализа требований к ИС и не содержит предположений о технологии реализации хранилища или базы данных.

Понятие предметной области является одним из базовых понятий информатики и не имеет точного определения. Его использование в контексте ИС предполагает существование устойчивой во времени связи между именами, понятиями и определенными реалиями внешнего мира, не зависящей от самой ИС и ее круга пользователей. Таким образом, введение понятия предметной области ограничивает и делает обозримым пространство информационного поиска в ИС, и позволяет выполнять запросы за конечное время.

Совокупность реалий (объектов) внешнего мира — объектов, о которых можно задавать вопросы, — образует объектное ядро предметной области. Невозможно получить в ИС ответ на вопрос о том, что ей неизвестно.

Термин «объект» является первичным, неопределяемым понятием. Синонимами термина «объект» являются «реалия», «сущность», «вещь». Отметим, что термин «сущность» понимается далее несколько уже, как компонент определенной логической модели предметной области. Выделяемые в предметной области объекты превращаются

аналитиками (а не проектировщиками) в сущности. При этом сущность предметной области понимается как набор свойств (параметров, характеристик) объекта.

Сущности имеют связи между собой.

Моделирование методом «сущность-связь»

Результатом моделирования методом «сущность-связь», или ER-моделирования, является ER-модель. ER-модель представляется с помощью ER-диаграмм, которые являются графической нотацией для абстрагирования данных в виде сущностей, взаимосвязей и атрибутов. Таким образом, семантика предметной области представляется в ER-модели в терминах субъективных средств описания – сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов и т.д.

Сущность описывается с помощью данных, именуемых свойствами или атрибутами (attributes) сущности. Как правило, атрибуты являются определениями в высказывании о сущности и обозначаются именами существительными естественного языка.

Сущности вступают в связи друг с другом через свои атрибуты. Каждая группа атрибутов, описывающих одно реальное проявление сущности, представляет собой экземпляр сущности (instance). Иными словами, экземпляр сущности – это реализации сущности, отличающиеся друг от друга и допускающие однозначную идентификацию. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически, имя сущности дается по имени ее экземпляра.

Одним из основных компьютерных способов распознавания сущностей в ИС является присвоение сущностям идентификаторов (Entity identifier). Часто идентификатор сущности называют ключом. Задача выбора идентификатора сущности является семантически субъективной задачей. Поскольку сущность определяется набором своих атрибутов, для каждой сущности целесообразно выделить такое подмножество атрибутов, которое однозначно идентифицирует данную сущность.

Некоторые сущности имеют естественные идентификаторы. Например, естественным идентификатором счета-фактуры является его номер. Идентификаторы сущности могут быть составными — состоящими из нескольких атрибутов, и атомарными — состоящими из одного атрибута сущности.

Уникальный идентификатор сущности — это атрибут сущности, позволяющий отличать одну сущность от другой. Если сущность имеет несколько уникальных идентификаторов, так называемых возможных ключей, то проектировщик должен выбрать первичный ключ сущности.

Различают однозначные и многозначные атрибуты. Однозначными являются атрибуты, которые в пределах конкретного экземпляра сущности имеют только одно значение. В противном случае они считаются многозначными.

Важным моментом изучения модели предметной области проектировщиком является выделение многозначных атрибутов сущности. Это связано с тем, что реляционная модель не поддерживает многозначных атрибутов и они должны быть разрешены на последующих стадиях проектирования.

Каждый атрибут имеет домен (domain). Домен — это выражение, которое определяет значения, разрешенные для данного атрибута. Иными словами, домен — это область значений атрибута. Для каждого атрибута сущности должен быть определен домен. На уровне логического моделирования данных назначение домена атрибуту носит общий характер. Например, атрибут текстовый, числовой, бинарный, дата или "не определен". В последнем случае аналитик должен дать описание домена. На последующих стадиях тип домена конкретизируется, смысл понятия домена в физической модели ХД уже, чем его может понимать аналитик. Это связано с тем, что в рамках физической модели домен реализуется посредством механизма ограничения домена, а СУБД не понимает неопределенных доменов.

Сущности не существуют отдельно друг от друга. Между ними имеются реальные отношения (Relationship), которые должны быть отражены в модели предметной области. При выделении отношений акцент делается на фиксацию связей и их характеристик. Отношение (связь) представляет собой соединение (взаимоотношение) между двумя или более сущностями. Каждая связь реализуется через значения атрибутов сущностей. Обычно связь обозначается глаголом. Каждая связь также должна иметь свой уникальный идентификатор связи.

Связи характеризуются степенью связи и классом принадлежности сущности к связи. Степень (мощность) связи – это отношение числа сущностей, участвующих в образовании связи. Например, "один к одному", "один ко многим", "многие ко многим". На уровне логической модели допускается неопределенная или неразрешенная связь. Класс принадлежности сущности — это характер участия сущности в связи. Различают обязательные и необязательные классы принадлежности сущности к связи. Обязательным является такой класс принадлежности, когда экземпляры сущности участвуют в установлении связи в обязательном порядке. В противном случае сущность принадлежит к необязательному классу принадлежности. Для необязательного класса принадлежности сущности степень связи может быть равна нулю, т.е. экземпляр сущности можно связать с 0, 1 или несколькими экземплярами другой

сущности. Для обязательного класса принадлежности степень связи не может равняться нулю.

Отношения, связывающие сущность саму с собой, называются рефлексивными. Типичным примером рефлексивных отношений является определение структуры подчиненности в отношении «Сотрудники». Рефлексивные отношения чаще всего отражают иерархические отношения внутри структуры данных.

С точки зрения отношений различают слабые сущности (weak). Слабые сущности – это сущности, которые не могут присутствовать в базе данных, пока не существует связанного с ней экземпляра другой сущности. Примером такой сущности является заказ, который не может существовать без клиента. Слабые сущности имеют обязательный класс принадлежности, и степень связи такой сущности не может равняться нулю. Связь "заказ-клиент" является обязательной.

Выявление слабых сущностей и связанных с ними обязательных отношений необходимо для обеспечения целостности и согласованности данных. Так, например, неизвестному клиенту невозможно приписать заказ.

Иногда выделенная сущность несет в себе отношение включения или отношение "часть-целое". При этом существует некоторый атрибут, значения которого порождают разбиение множества экземпляров сущности на непересекающиеся подмножества — категории сущности. Категории сущности называют подтипами и выделяют в подчиненную в рамках отношения сущность, которая является категорией исходной сущности. Из исходной сущности выделяются общие для полученных категорий атрибуты, и таким образом выделяется сущность, которая становится супертипом. За выделенной сущностью-супертипом обычно оставляют наименование исходной сущности, хотя ее семантический смысл меняется.

Супертип с порожденными им подтипами является примером так называемой составной сущности. Составная сущность является логической конструкцией модели для представления набора сущностей и связей между ними как единого целого.

Пример. Сущность «автомобиль» можно разбить на следующие подтипы: автомобили с приводом на два колеса, автомобили с приводом на четыре колеса, автомобили с переключаемым приводом.

Примеры ER диаграмм физического и логического уровня представлены на рисунках 12.1 и 12.2.

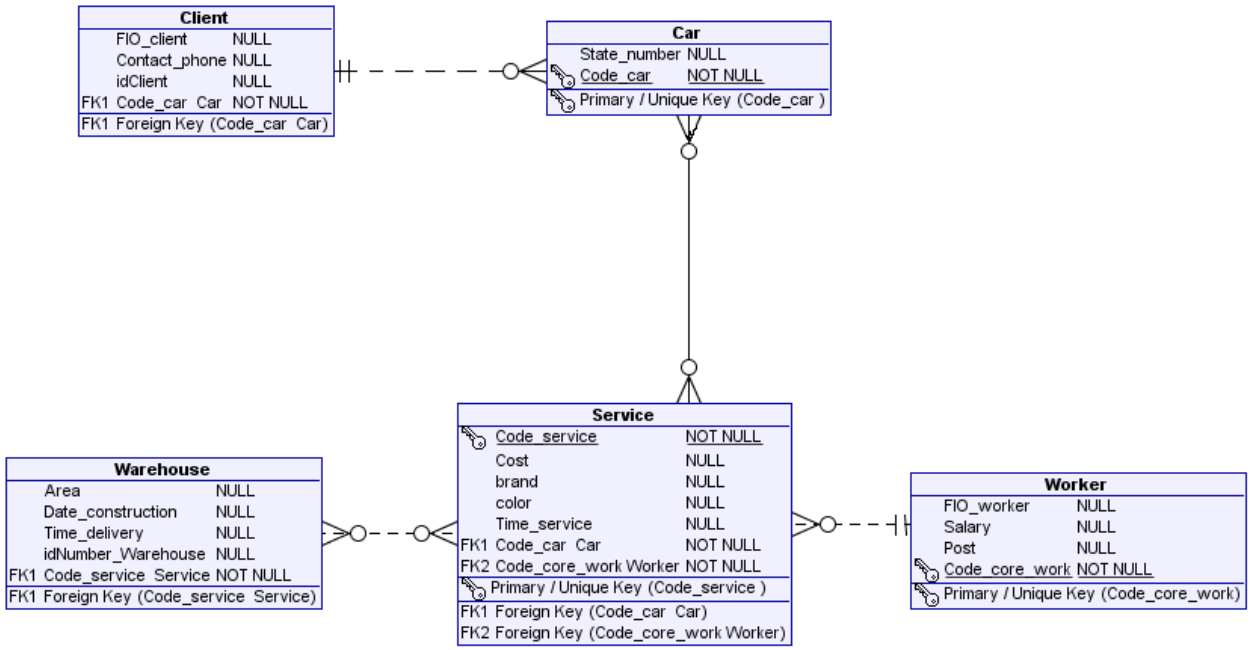


Рисунок 12.1 Логическая схема базы данных

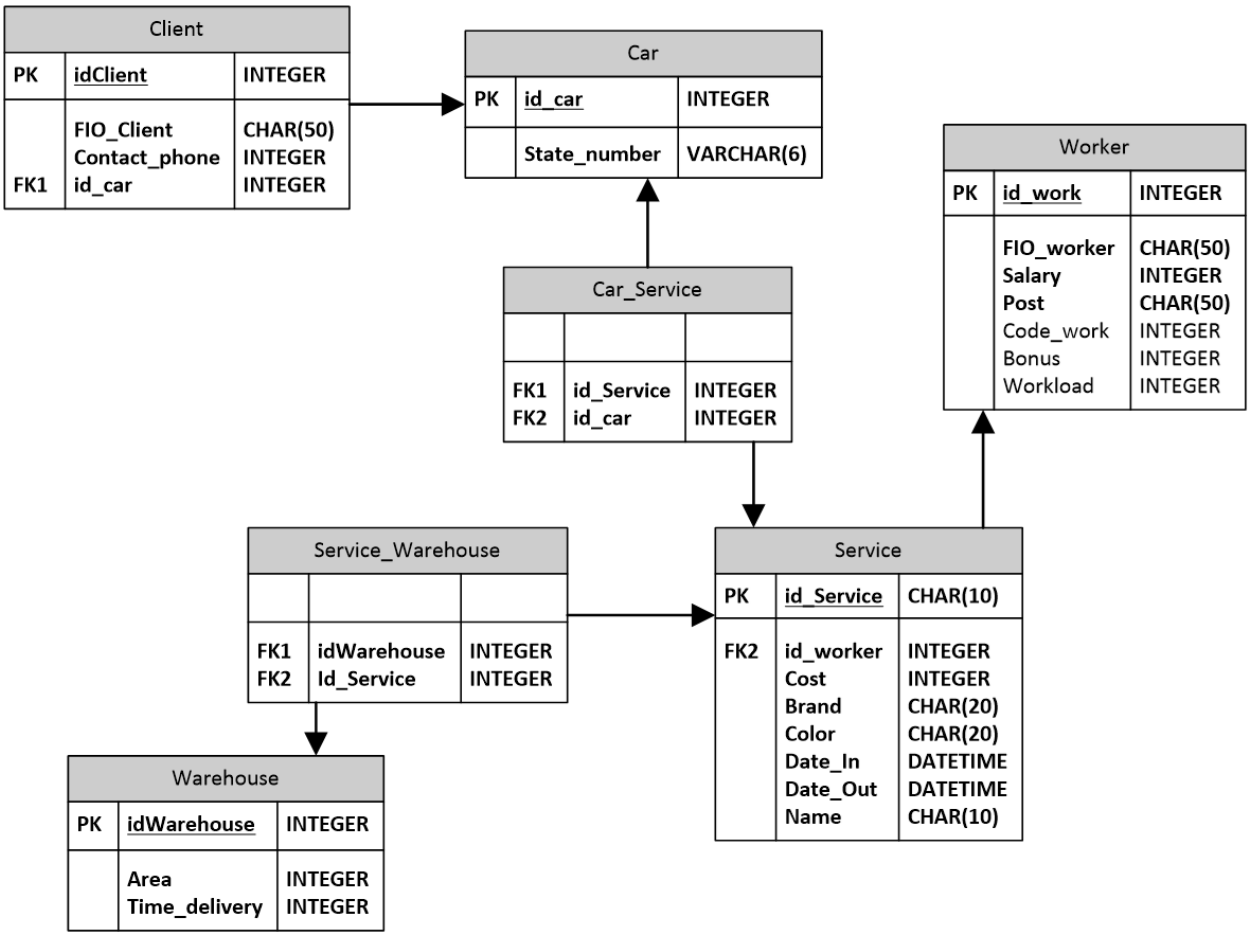


Рисунок 12.2 Физическая схема базы данных

Задание для выполнения:

1. Изучить теоретические данные по ER диаграммам
2. Построить схему базы данных логического и физического уровня;
3. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, ER диаграммы логического уровня и физичесого уровня);
4. Выводы по работе.

Контрольные вопросы:

1. Для чего используются ER диаграммы?
2. Перечислите и поясните основные элементы ER диаграммы.

Лабораторная работа №13

Построения организационной схемы

Цель работы: освоить методологию моделирования бизнес-процессов, основанную на архитектуре ARIS и анализа структуры предприятия, построив организационную модель подразделений и сотрудников.

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Для реализации комплексных задач, стоящих перед предприятиями, необходимо использовать бизнес - модели их деятельности. Бизнес - модели являются основой для анализа бизнес-процессов, для выявления требований к информационной системе, которая поддерживает организационную структуру и соответствует нормам международных стандартов серии *ISO* 9000:2000, а также для разработки и внедрения самой ИС.

Архитектура интегрированных систем (*ARIS*) – это целостный подход к разработке и анализу моделей бизнес-процессов. Модель в рамках этого подхода имеет несколько представлений:

• функциональное представление (*function view*);

• организационное представление (organization view);

• представление данных (data view);

• представление управления (control view).

Функциональное представление содержит описание выполняемых функций, перечень отдельных подфункций, а также существующие общие взаимосвязи и связи подчиненности, которые существуют между функциями.

Организационное представление показывает взаимодействие пользователей и организационных единиц, а также их связи и релевантные структуры, имеющие к ним отношение структуры. Представление данных включает описание модели данных предметной области. Представление управления введено для описания связей между представлениями. Интеграция этих связей в пределах отдельного представления позволяет учесть все связи без избыточности. Каждое из представлений является совокупностью моделей, описывающих соответствующий аспект предметной области. Среди основных видов моделей, которые используются в большинстве проектов, использующих архитектуру *ARIS*, можно выделить следующие:

• *Organizational Chart* – модель организационной структуры;

• *ValueAdded Chain Diagram* – модель цепочки добавленной стоимости;

• *Function Tree* – модель дерева функций;

• *extended Event Driven Process Chain* (*eEPC*) – расширенная событийноориентированная модель;

• *Function Allocation Diagram* (*FAD*) – модель описания функций;

• *ERM – Entity-Relationship Model* – модель данных;

• *Office Process* – офисная модель;

• *Indusrial Process* – производственная модель.

Модель организационной структуры

Модель организационной структуры относится к организационному представлению и обеспечивает описание статических отношений между различными структурными элементами, ответственными за выполнение функций на предприятии. В инструментальной среде ARIS имеются структурные элементы и типы соединений, которые служат для описания иерархической организационной структуры организации. Элементы модели организационной структуры приведены на рисунке 13.1.

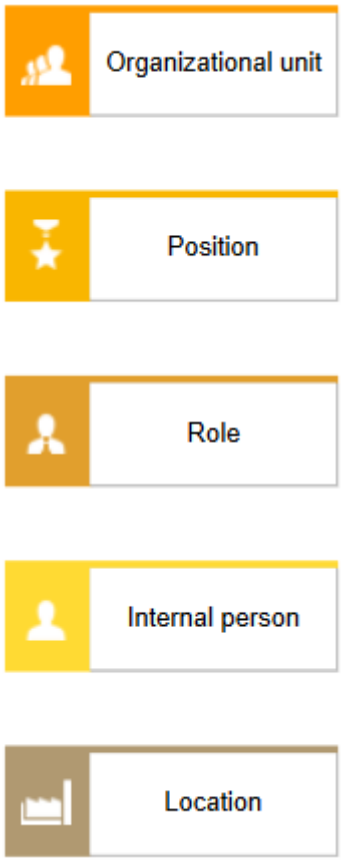


Рисунок 13.1 Основные элементы модели организационной структуры

Организационная единица (Organizational unit) – подразделение в организационной иерархии, например, отдел или местоположение. С его помощью можно показать, какие организационные единицы превосходят другие, и распределить их по ролям и персоналиям.

Должность (Position) – самая маленькая организационная единица в компании. Она назначается сотрудникам (персоналиям).

Роль (Role) – определяет задачи, свойства и привилегии пользователя. Она не зависит от конкретного человека. Роль может быть назначена нескольким лицам.

Лицо (Person) – конкретное лицо, которое выполняет определенную роль и может быть назначено в подразделение организации. Лицо обычно представляет существующего сотрудника в организации.

Местоположение (Location) – местоположением может быть фабрика, здание, а также офис или отдельное рабочее место в комнате. Местоположение относится к физическому месту.

При построении модели организационной структуры все структурные элементы связываются различными типами отношений, например:

• технический суперкласс для (*is technical superior to*);

• административно суперкласс для (*is disciplinary superior to*);

• ответственен за (*is responsible for*);

• является организационным менеджером для (*is Organization Manager for*) и др.

• технически вышестоящий для (*is technical superior to*);

• административно вышестоящий для (*is disciplinary superior to*).

Таким образом, организационная модель описывает иерархическую структуру организации, т.е. организационные единицы, связанные между собой коммуникационными отношениями и отчетностью. Цель представления иерархической структуры организации состоит в том, чтобы упростить

описание предприятия, объединив группы, выполняющие аналогичные задачи, в организационные единицы.

Для иллюстрации диаграмм в качестве объекта моделирования здесь и далее будет рассматриваться бизнес-процесс организации закупок у внешних поставщиков товарно-материальных ценностей (ТМЦ) для обеспечения основного производства типового машиностроительного предприятия.

Процесс закупок состоит в организации и контроле обеспечения товарноматериальными ценностями, хранении и передаче их в производство, выборе и оценке поставщиков.

Целью процесса закупок является обеспечение потребности производства материалами и комплектующими.

Владелец бизнес-процесса «Осуществлять закупки» – заместитель коммерческого директора.

Назначение бизнес-процесса закупок состоит в том, что потребитель должен получить товары заданного качества, в требуемом количестве, в нужное время, в нужном месте, от надежного поставщика с хорошим уровнем обслуживания (как до осуществления продажи продукции, так и после нее) и при заданном уровне общих затрат.

Согласно процессному подходу и методологии *ARIS* рассматриваемый бизнес-процесс можно представить в виде совокупности субпроцессов:

• выполнять текущую деятельность;

• анализировать результаты по закупкам;

• формировать управляющие воздействия.

На рисунке 13.2 представлена модель организационной структуры участников бизнес-процесса закупок (на уровне определения требований). Ее можно рассматривать как «проекцию» этого бизнес-процесса на общую организационную структуру предприятия.

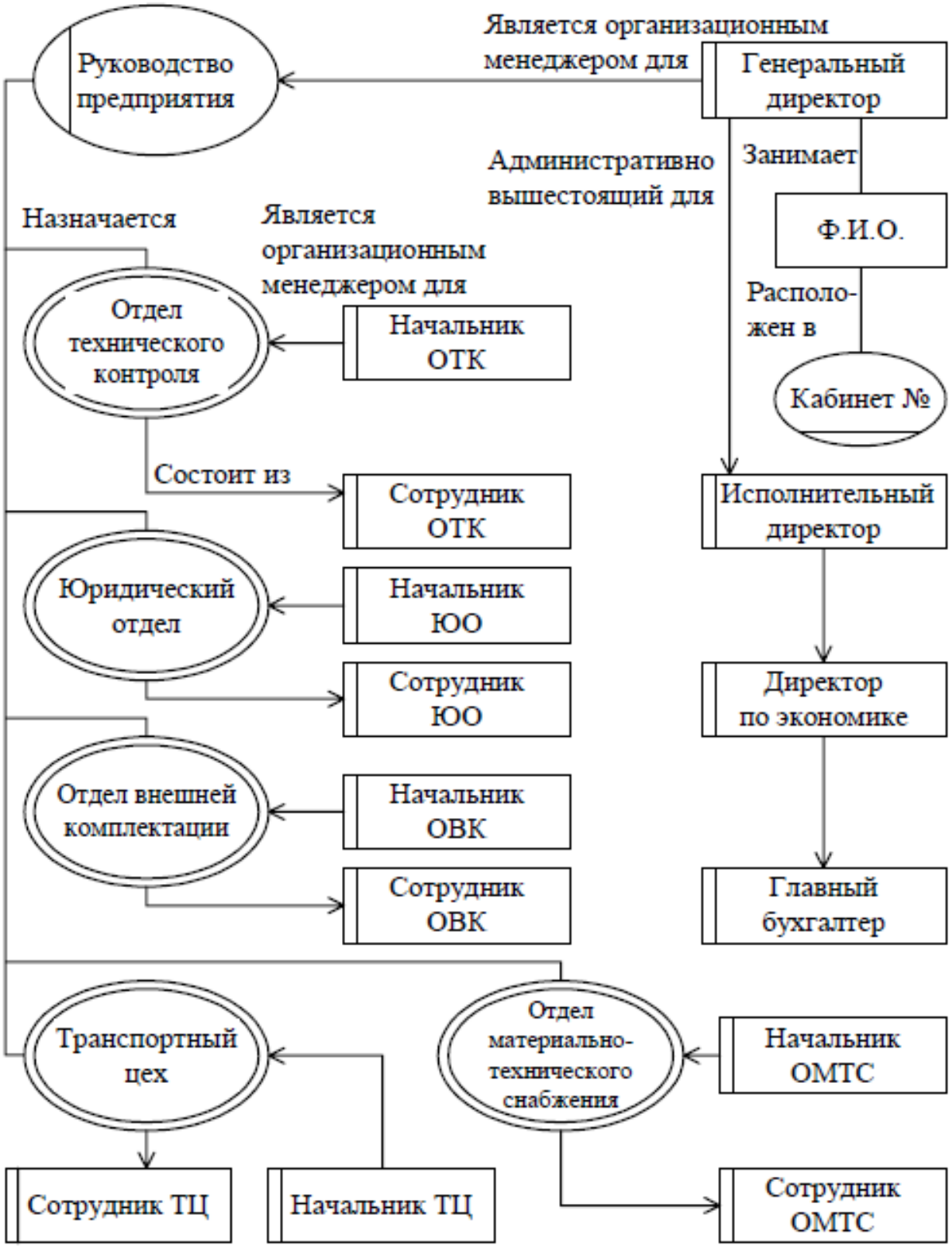


Рисунок 13.2 Модель организационной структуры участников бизнес-процесса закупок (в рамках старой версии нотации ARIS)

Модель организационной структуры участников бизнес-процесса закупок описывает организационную иерархию исполнителей (должностных лиц) функций бизнес-процесса. Структурные элементы модели связаны отношениями типов:

• «Является организационным менеджером для» – от позиции к организационной единице;

• «Административно вышестоящий» – между позициями на разных уровнях подчиненности;

• «Назначается» (*is assigned*) – от группы к организационной единице;

• «Занимает» (*Occupies*) – от личности к позиции;

• «Расположен в» (*is located at*) – от личности к местоположению («Кабинет №»);

• «Состоит из» (*is composed of*) – от группы к позиции.

Таким образом, на рассмотренной модели показаны организационные элементы (участники бизнес-процесса) и связи между ними. Описание процесса должно продолжаться до достижения необходимой степени «прозрачности», достаточной для корректного анализа и выработки эффективных управленческих решений.

На рисунке 13.3 представлен пример организационной структуры

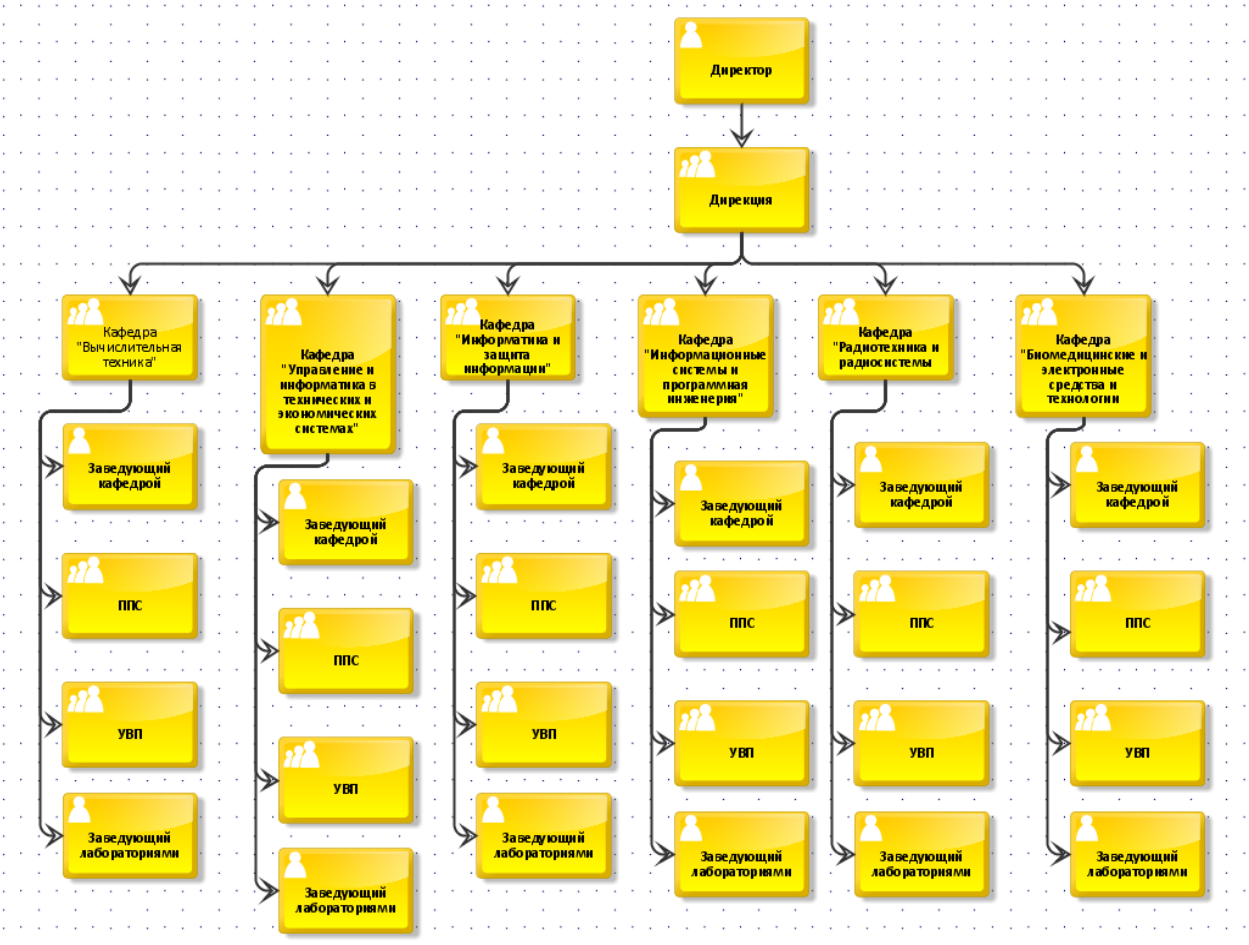


Рисунок 13.3 Организационная структура ИИТР

Задание для выполнения:

1. Изучить теоретические данные по ARIS и организационной модели
2. Построить организационную модель предприятия по вашей предметной области
3. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, организационная модель);
4. Выводы по работе.

Контрольные вопросы:

1. Что вызывает потребность в стандартизации процесса разработки моделей бизнес-процессов?
2. Перечислите представления модели в архитектуре *ARIS*?
3. Какие основные виды диаграмм используются для моделирования бизнес-процессов в среде *ARIS*?
4. Каковы особенности и назначение модели организационной структуры?

Лабораторная работа №14

Построения диаграммы BPMN

Цель работы: освоить методологию моделирования бизнес-процессов BPMN и построить диаграмму бизнес-процесса

Формируемые компетенции: ПК 1.1

Теоретические сведения:

BPMN — это нотация (или метод, хотя почти везде пишут «система») моделирования или описания бизнес-процессов. Бизнес-процесс представляет из себя логику (алгоритм) работы системы для достижения поставленной задачи. Соответственно, BPMN-диаграмма — это диаграмма, которая описывает бизнес-процесс. Такие диаграммы довольно просто и интуитивно читаются, особенно если разработчик бизнес-процесса (тот, кто моделировал диаграмму) проектировал его по всем правилам и стандартам, а также старался не нагружать его лишней информацией.

В отдельных случаях с помощью BPMN прорабатывают сложные процессы: разработчик проектирует процесс, описывает все условия, пробрасывает потоки и т. д. Для этого есть специальные среды разработки, например, Tibco BPM и Camunda BPM.

Перечислим основные виды графических элементы нотации:

1. **Элементы управления** (Flow Objects)
    - Шаги (Activities)
    - Шлюзы (Gateways)
    - События (Events)
2. **Соединяющие линии** (Connecting Objects)
    - Потоки управления (соответствуют ранее введенному понятию "переход")
    - Ассоциации
    - Потоки сообщений
3. **Роли-дорожки** (Pools - Swimlanes)
    - Пулы
    - Дорожки
4. **Данные** (Data)
    - Объекты данных
    - Хранилища данных
    - Сообщения
5. **Артефакты** (Artifacts)

- Группы
- Ассоциации
- Аннотации

Особенность нотации BPMN - использование различных сочетаний элементов. Например, (если окружность означает событие, конвертик - сообщение, предмет белого цвета - пассивное поведение, а черного цвета - активное поведение) то белый конвертик, изображенный в окружности, будет означать событие получения сообщения, а белый конвертик, изображенный в окружности - событие отправки сообщения.

Шаги бизнес-процесса в нотации BPMN могут быть задачами или подпроцессами. Подпроцессы могут быть внутренними или внешними.

Типы используемых в нотации BPMN задач перечислены на рисунке 14.1

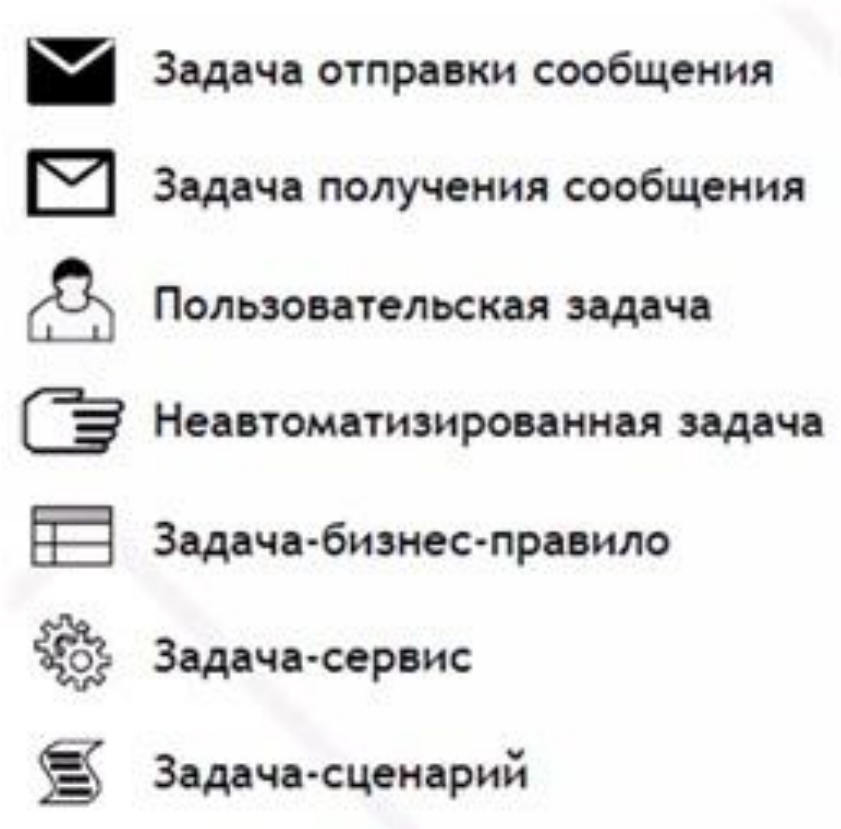


Рисунок 14.1 Типы шагов

Задачи "Пользовательская задача" и "Неавтоматизированная задача" соответствуют узлам-действиям.

Шлюзы - это элементы, в которых происходит деление-слияние точек управления или выбор перехода, по которому точка управления будет перемещена дальше. Используемые в нотации шлюзы приведены на рисунке 14.2

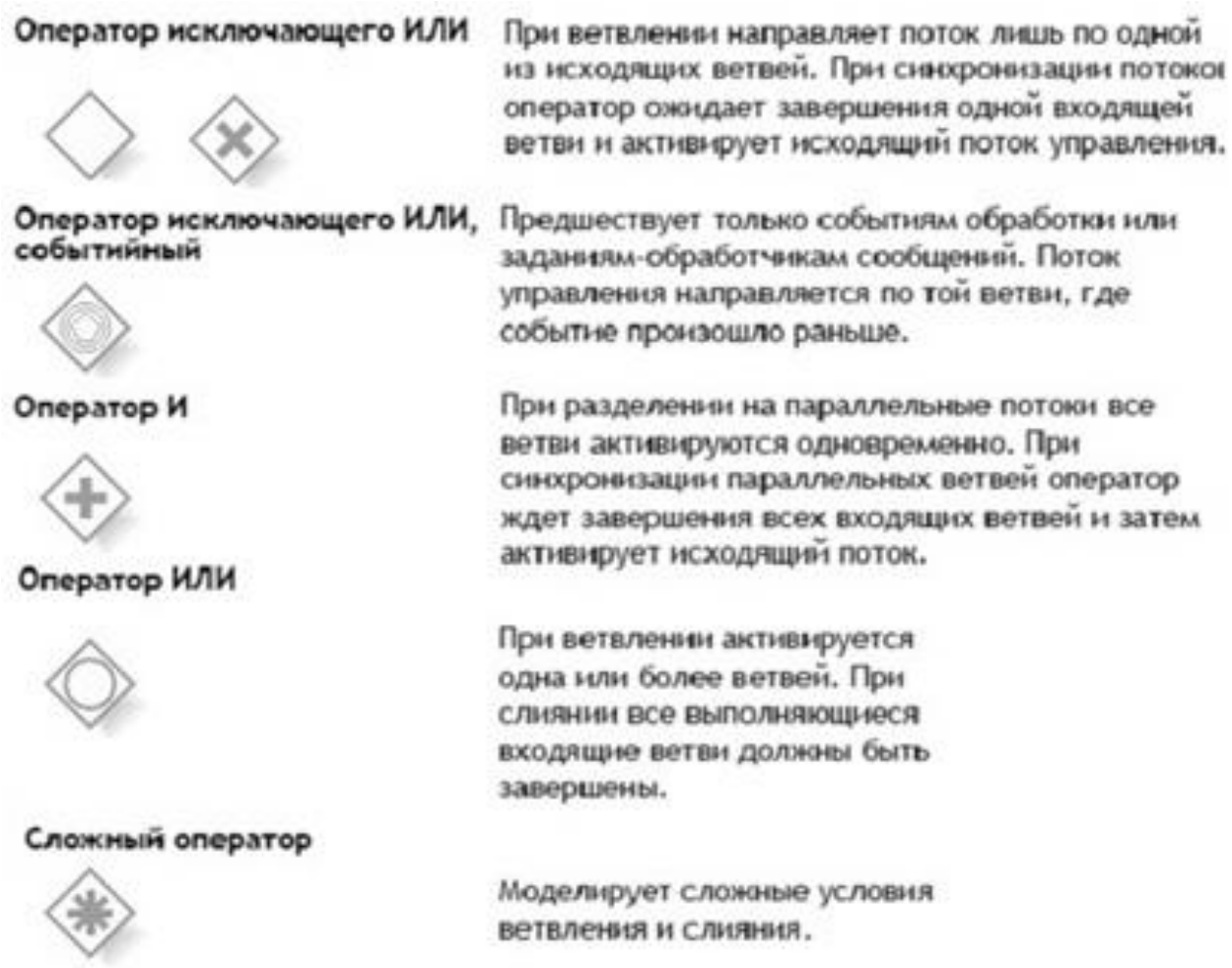


Рисунок 14.2 Шлюзы

Поведение шлюзов "Оператор исключающего или" и "Оператор и" соответствует поведению элементов "Исключающий шлюз" и "Параллельный шлюз".

Применяемые в нотации BPMN обозначения для потоков управления (переходов) приведены на рисунке 14.3.

Рисунок 14.3. Обозначения для потоков управления, используемые на схемах приватных выполняемых бизнес-процессов (исполнимых бизнес-процессов)

В нотации BPMN применяется три типа событий. Эти типы приведены на рисунке 14.4.

Рисунок 14.4. Типы событий нотации BPMN

На рисунке 14.5 приведены использующиеся в BPMN события, полученные добавлением к типу события его "внутренней" части.

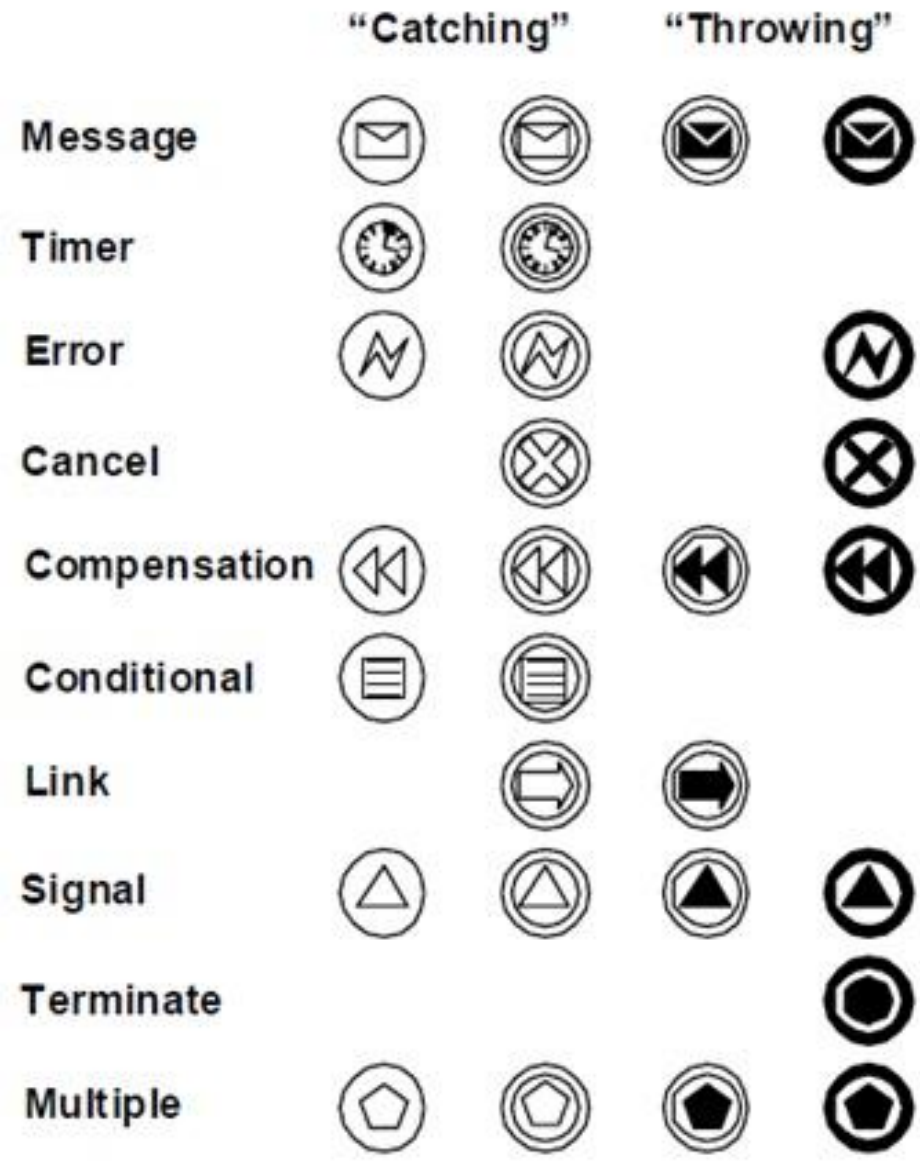


Рисунок 14.5. События

Из всех событий наиболее часто используются события "окончание бизнес-процесса" и "завершение потока управления".

Конечное событие "окончание бизнес-процесса" обозначается "толстой" окружностью с кружком внутри. В случае прихода точки управления в элемент "окончание бизнес-процесса" все точки управления экземпляра бизнес-процесса удаляются и экземпляр сразу завершается.

Конечное событие "завершение потока управления" обозначается "толстой" окружностью без каких-либо элементов внутри. В случае прихода точки управления в элемент "завершение потока управления" пришедшая в элемент точка управления удаляется, а состояния других точек управления экземпляра бизнес-процесса не изменяется. Если в экземпляре бизнес-процесса не осталось точек управления, то он считается завершенным.>

На рисунке 14.6 представлены использующиеся в BPMN элементы, имеющие отношение к перспективе ресурсов - пулы и дорожки.

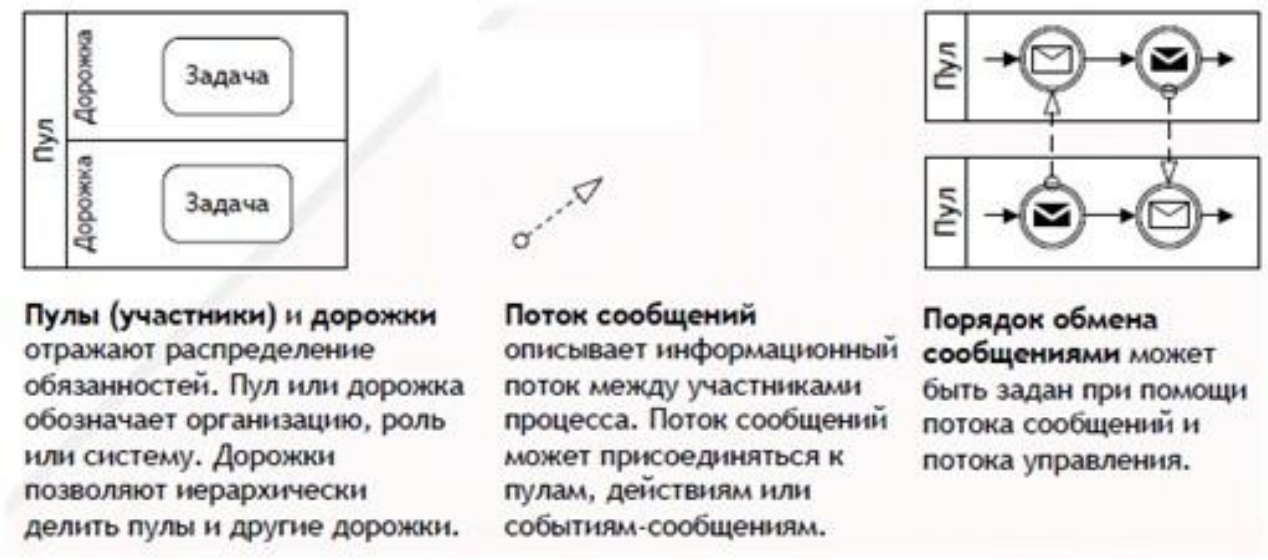


Рисунок 14.6. Пулы и дорожки

Дорожки соответствуют ролям перспективы ресурсов исполнимых бизнес-процессов. Понятие "пул" соответствует определению бизнес-процесса. Применяется в случае, если на диаграмме изображены схемы нескольких бизнес-процессов.

Сообщения и сигналы

Взаимодействие между экземплярами бизнес-процессов в BPMN 2.0 осуществляется при помощи сообщений и сигналов.

Сообщения реализуют передачу информации от экземпляра бизнес-процесса - отправителя к другому экземпляру бизнес-процесса - получателю. Сигналы реализуют широковещательную рассылку, в случае сигнала отправленная информация будет предназначена для приёма всеми экземплярами бизнес-процессов, "подписанными" на данный сигнал.

Для сообщений и сигналов в нотации предусмотрены графические элементы, являющиеся источниками и получателями информации. Эти элементы встраиваются в перспективу потока управления (являются элементами схемы бизнес-процесса, соединенными переходами с другими элементами).

Сообщения между двумя бизнес-процессами могут (не обязательно) изображаться пунктирными стрелками с небольшой окружностью у отправителя сообщения и стрелочкой у получателя. (см. рисунок 14.7)

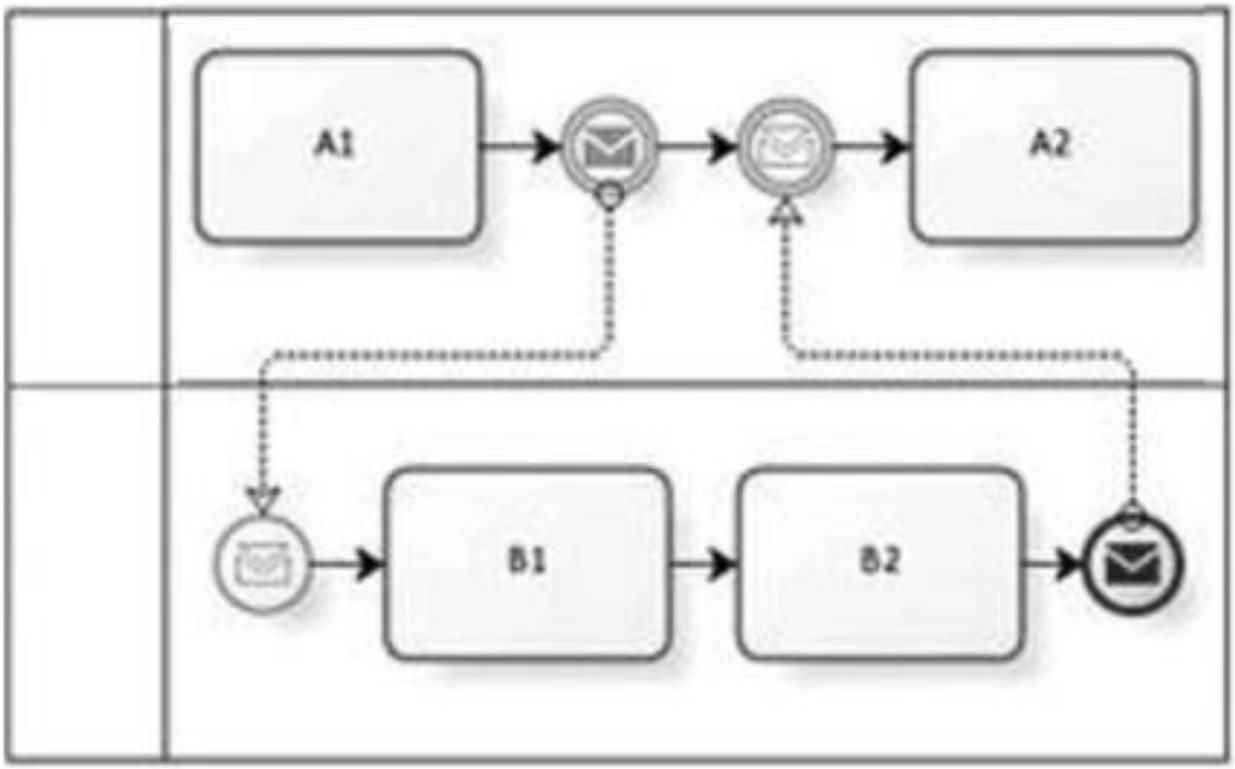


Рисунок 14.7. Обмен сообщениями между двумя бизнес-процессами

Шлюзы

Рассмотрим более подробно узлы, имеющие форму ромба (т. е. шлюзы) и связанные с ними конструкции.

Исключающий шлюз (или "Оператор исключающего ИЛИ") представлен на рисунке 14.8.

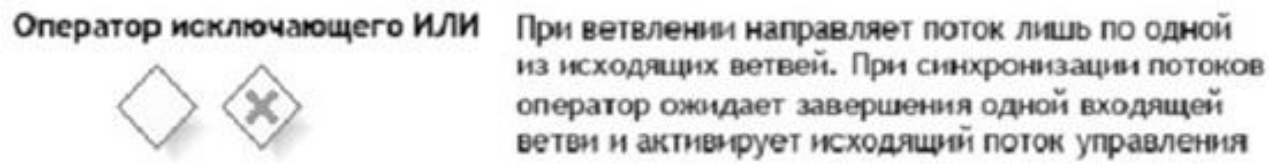


Рисунок 14.8. Исключающий шлюз

Нотация BPMN позволяет для экономии места на схеме бизнес-процесса как бы "накладывать" исключающий шлюз на предыдущий узел-действие. При этом "ромбик" не рисуется, а выходящие из него стрелочки присоединяются с предыдущему узлу-действию, при этом в основании стрелочек помещается маленький ромбик, либо стрелочка помечается косой чертой как "путь по умолчанию". См. рисунок 14.9.

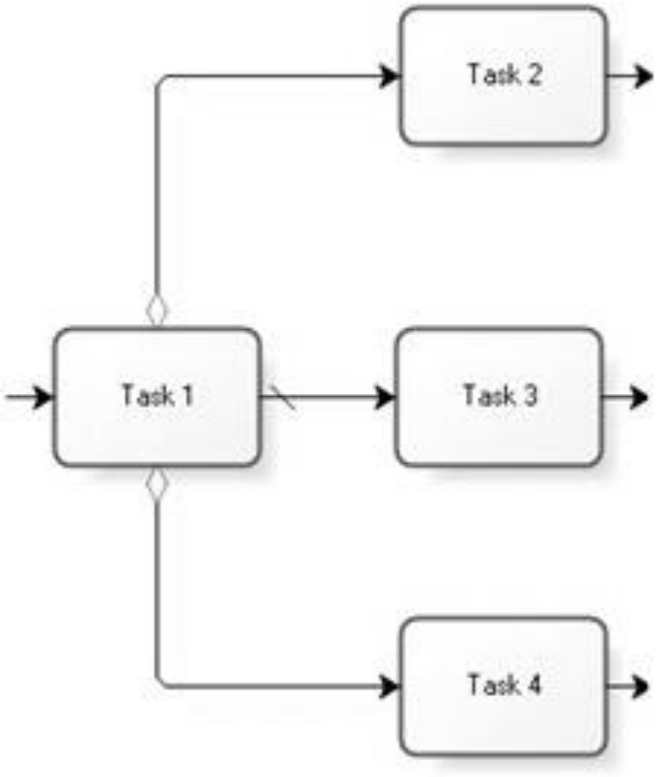


Рисунок 14.9. Неявный исключающий шлюз, совмещенный с узлом-действием

Исключающий шлюз также используется для соединения потоков управления. См. рисунок 14.10.

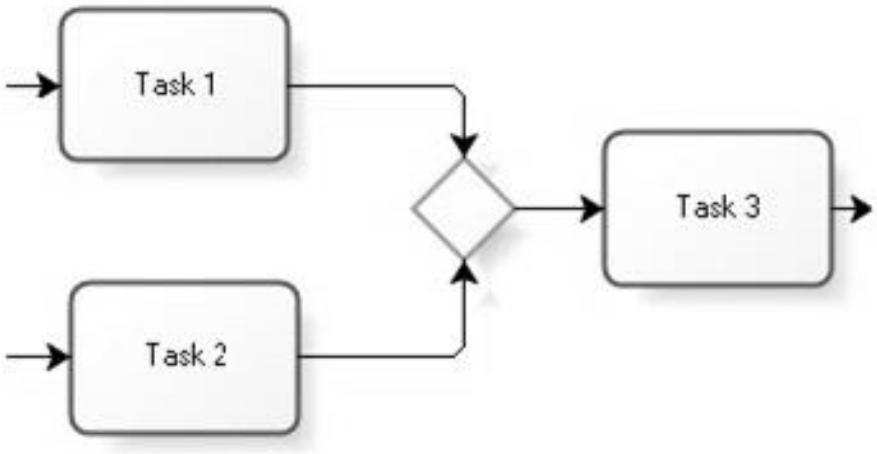


Рисунок 14.10. Использование исключающего шлюза для соединения различных маршрутов точек управления

В случае соединения потоков управления нотация также позволяет"накладывать" исключающий шлюз на предыдущий узел-действие. Ромбики у оснований переходов при этом не используются. См. рисунок 14.11.

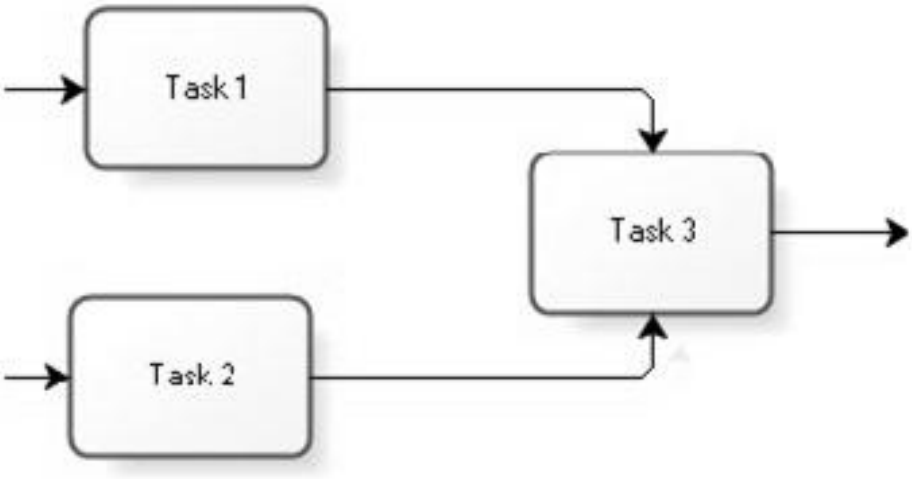


Рисунок 14.11. Совмещение исключающего шлюза с узлом-действием для соединения различных маршрутов точек управления

Хорошим стилем является использование парных исключающих шлюзов для ветвления и соединения потоков управления, когда это позволяет логика бизнес-процесса. См. рисунок 14.12.

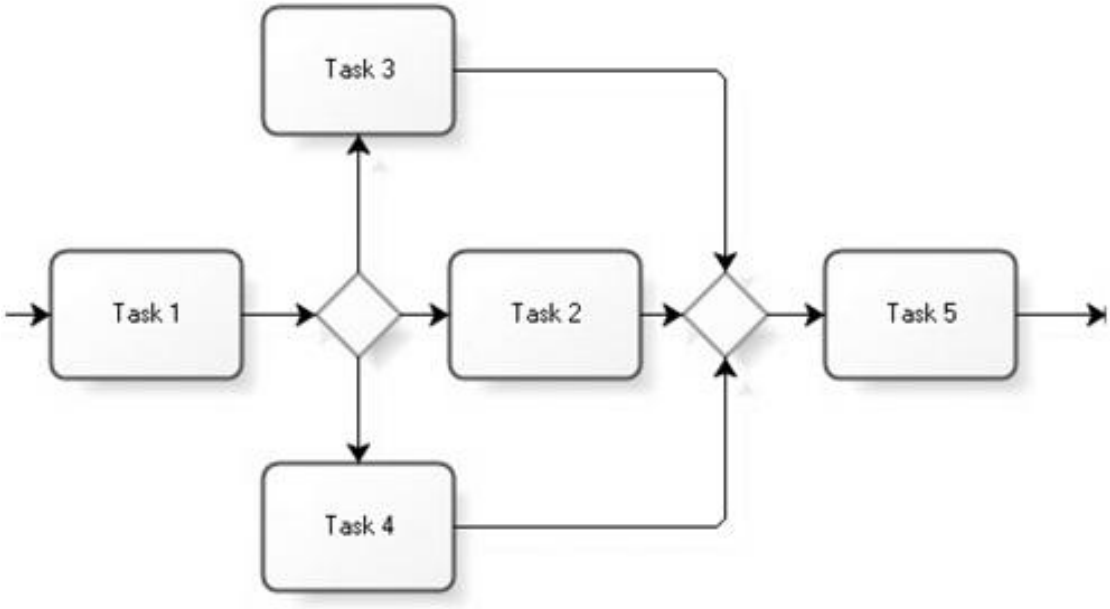


Рисунок 14.12. Пример использования парных исключающих шлюзов для ветвления и соединения потоков управления

Параллельный шлюз (или "Оператор И") представляет собой ромбик, внутри которого изображен плюсик. См. рисунок 14.13

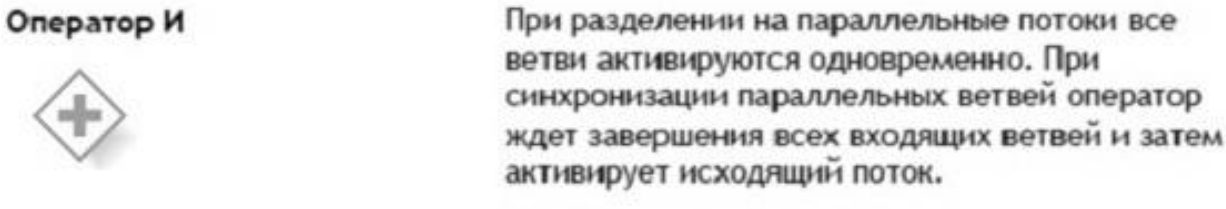


Рисунок 14.13. Параллельный шлюз

В случае, если из узла-действия выходит одновременно несколько стрелок (без маленьких ромбиков в их основании), то в соответствии с нотацией BPMN 2.0 происходит распараллеливание точек управления. То есть, на узел-действие как бы налагается последующий параллельный шлюз. См. рисунок 14.14

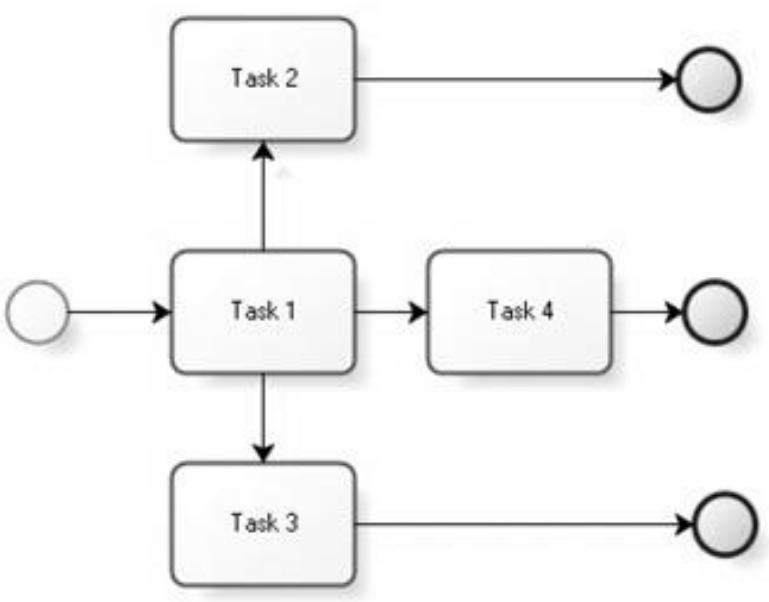


Рисунок 14.14. Пример использования неявного распараллеливания

Нотация допускает еще два типа шлюзов - "Оператор ИЛИ" и "Сложный оператор". В случае "Оператора ИЛИ" внутрь ромбика вписывается окружность. См. рисунок 14.15. Оператор может породить точки управления не на всех исходящих из узла переходах (но обязательно должен хотя бы на одном). При этом на схеме бизнес-процесса обязательно должен присутствовать парный "Оператор ИЛИ", в котором все вышедшие точки должны собраться, после этого они будут удалены и на исходящий переход будет помещена одна точка управления.

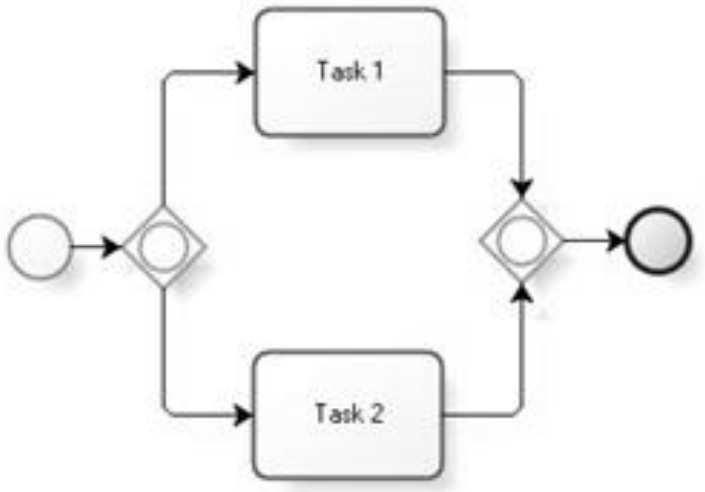


Рисунок 14.15. "Оператор ИЛИ"

В случае использования ромбика со звездочкой внутри, в элементе может использоваться нестандартное условие синхронизации. См. рисунок 14.16.

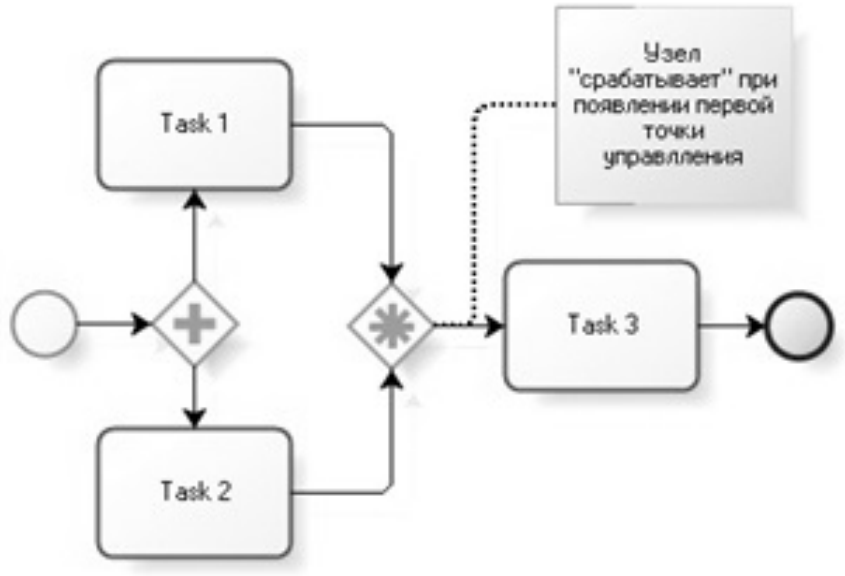


Рисунок 14.16. Пример сложной синхронизации

Если в ромбик вписана двойная окружность, содержащая пятиугольник, то такой элемент соответствует "Ветвлению по событиям". То есть точка управления "ждет" в элементе наступления одного из событий, находящихся на исходящих из узла переходах. В случае наступления события точка управления перемещается в соответствующий событию элемент. См. рисунок 14.17.

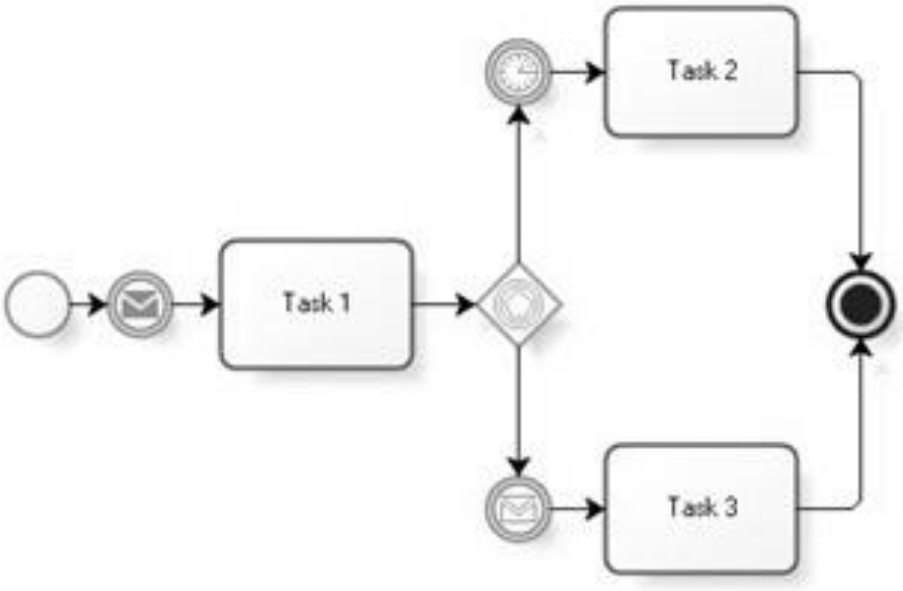


Рисунок 14.17. Пример ветвления по событиям

Подпроцессы, циклы и мультидействия

В нотации BPMN прямоугольники со скругленными углами могут обозначать узлы-действия, циклы и подпроцессы. Что именно обозначает элемент, зависит от маркера, находящегося внутри прямоугольника. Если маркер отсутствует, то элемент обозначает узел-действие. На рисунке 14.18 представлены наиболее часто использующиеся маркеры для циклов и подпроцессов.

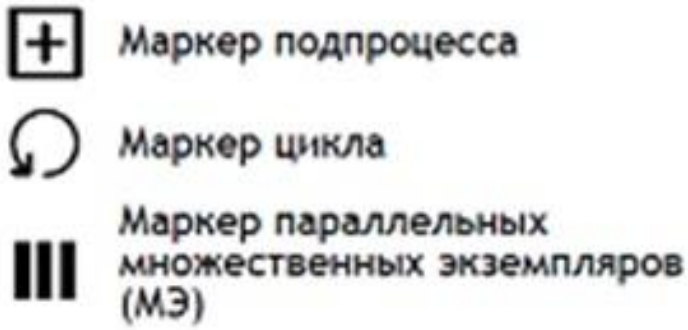


Рисунок 14.18. Маркеры для обозначения циклов и подпроцессов, используемые на элементах - шагах бизнес-процесса

Подпроцессы используются как для уменьшения сложности базовых (родительских) бизнес-процессов путем разделения их на части, так и для повторного использования одинаковых участков в разных бизнес-процессах. Подпроцессы могут быть внутренними и внешними. Внутренние подпроцессы в основном используются для уменьшения схемы базового бизнес-процесса путем декомпозиции ее на несколько подпроцессов. Внутренний подпроцесс использует те же переменные и роли, что и базовый бизнес-процесс.

Внешний подпроцесс - фактически это самостоятельный бизнес-процесс, запущенный базовым бизнес-процессом в одном из своих элементов. У внешнего подпроцесса кроме собственной схемы (перспективы потока управления) присутствуют все элементы остальных перспектив исполнимого бизнес-процесса: собственные роли, переменные и т.п. Для внешнего подпроцесса генерируется свой собственный экземпляр. Более того, базовый бизнес-процесс может порождать несколько экземпляров подпроцессов. В этом случае точка управления базового процесса ждет в породившем подпроцессы элементе, пока все порожденные экземпляры подпроцесса завершатся и только после этого переходит дальше по исходящему переходу. Такие подпроцессы называются мультидействиями и мультиподпроцессами, для них предусмотрен специальный маркер. См. рисунок 14.18.

Если внутри элемента-прямоугольника находится маркер цикла (См. рисунок 14.18.), то это значит что элемент соответствует подпроцессу, действия которого будут последовательно повторяться, пока не выполнится условия выхода из цикла.

На рисунке 14.19. приведены примеры цикла и мультидействия.

Рисунок 14.19. Примеры цикла и мультидействия

На рисунке 14.20 приведен пример участка бизнес-процесса, иллюстрирующий использование мультидействия. В данном примере один заказ клиента может содержать товары, находящиеся на разных складах, при этом на всех складах процедура получения товара одна и та же.

Рисунок 14.20. Пример использования мультидействия

В нотации BPMN 2.0 внутренние подпроцессы могут изображаться как в свернутом, так и развернутом виде. В случае развернутого вида маркер подпроцесса на

элементе не устанавливается, внутри элемента рисуется схема подпроцесса. См. рисунок 14.21

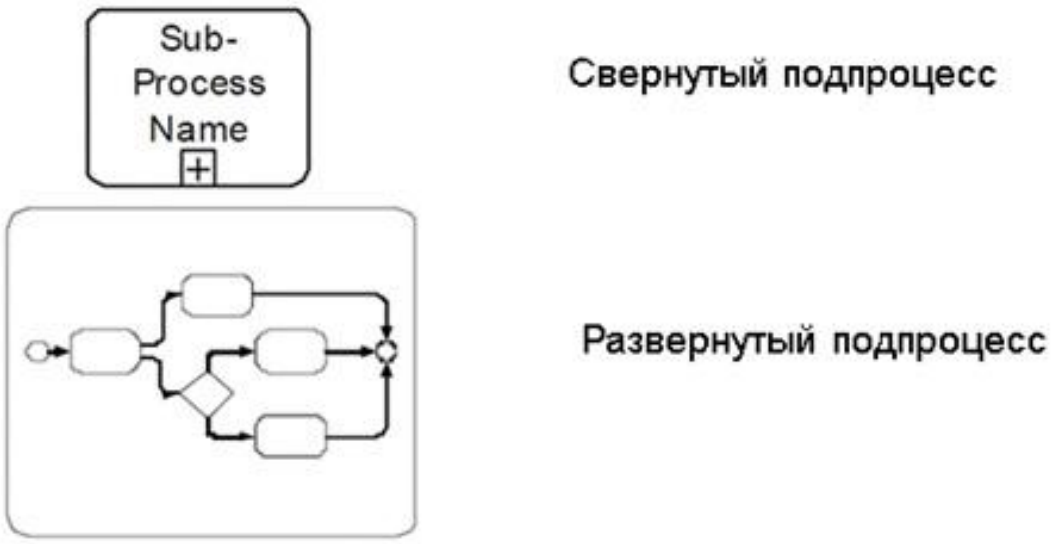


Рисунок 14.21. Свернутый и развернутый подпроцесс

Исключительные ситуации, Компенсации, Артефакты

Исключительные ситуации соответствуют событиям, которые делают дальнейшее исполнение экземпляра бизнес-процесса невозможным. Для исключительной ситуации в BPMN 2.0 есть специальный элемент-событие. При возникновении исключительной ситуации нормальный ход выполнения бизнес-процесса прекращается и управление передается элементу, который выполняет обработку исключительной ситуации. См. рисунок 14.22.

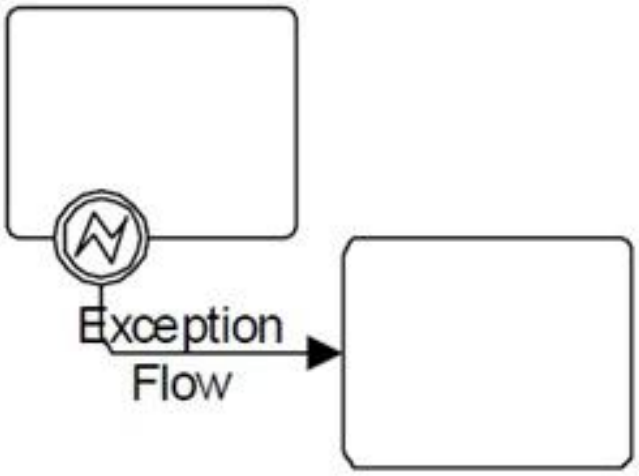


Рисунок 14.22. Исключительная ситуация

Компенсация используются в случаях, когда после возникновения исключительной ситуации возникает необходимость в откате данных в состояние, предшествующее началу выполнения этапа бизнес-процесса. Например, в случае отказа клиента туристической компании от поездки, туристическая компания должна произвести отмену сделанного для данного клиента бронирования гостиницы. Элемент, выполняющий откат, связывается с соответствующим узлом пунктирной стрелкой (См. рисунок 14.23).

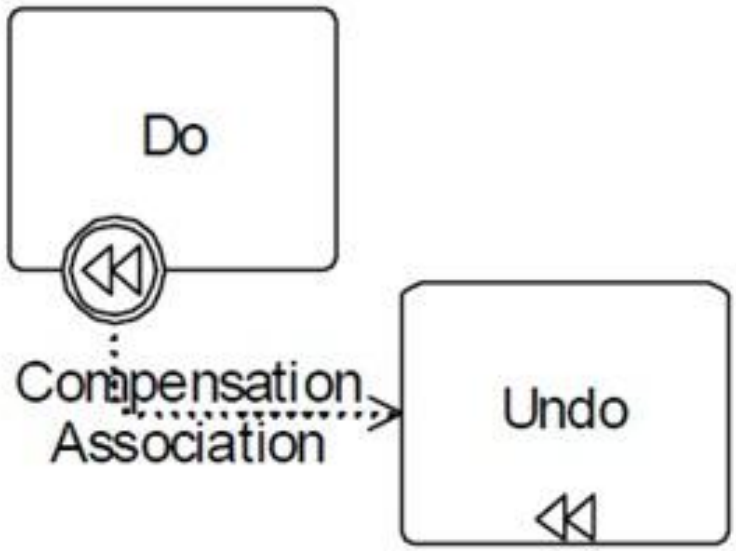


Рисунок 14.23. Использование компенсации

Артефакты не выполняют в бизнес-процессе никаких действий. Они служат для иллюстрации. В нотации BPMN 2.0 используются следующие артефакты: Аннотация, объект данных, группа (См. рисунок 14.24).

Рисунок 14.24. Артефакты

Аннотация соответствует комментарию к элементу бизнес-процесса. Объект данных не оказывает влияния на поток управления, однако, содержит информацию о данных, используемых при выполнении бизнес-процесса. Группа служит для свободной группировки графических элементов бизнес-процесса. Представляет собой прямоугольник с закругленными углами, который рисуется пунктирной линией (См. рисунок 14.24).

На рисунке 14.25 приведен пример схемы процесса, построенной по нотации BPMN

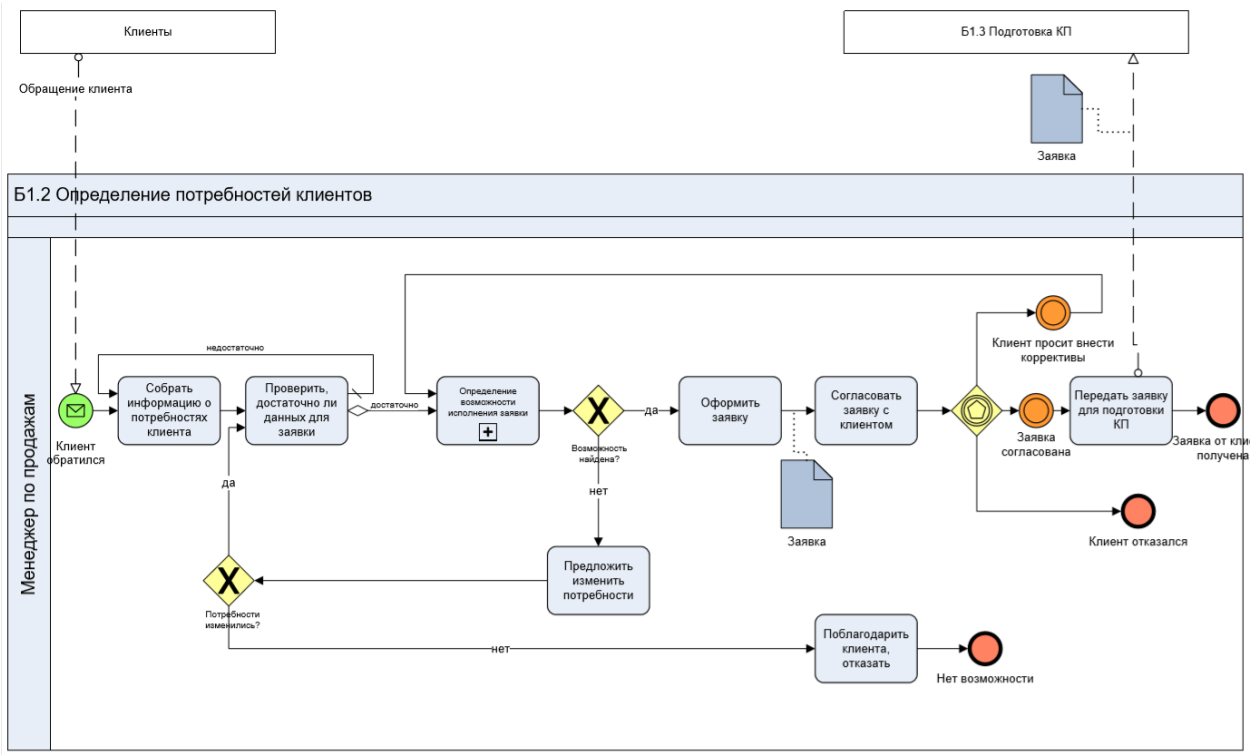


Рисунок 14.25 Пример схемы процесса

Задание для выполнения:

1. Изучить теоретические данные по BPMN;
2. Построить схему процесса по нотации BPMN
3. Оформить отчет.

Содержание отчета:

1. Титульный лист;
2. Цель работы;
3. Ход работы (описание предметной области, схема процесса);
4. Выводы по работе.

Контрольные вопросы:

1. Какие объекты доступны при построении диаграммы бизнес-процесса в нотации BPMN?
2. Опишите возможные типы стартовых событий и приведите примеры их использования.
3. Опишите возможные типы промежуточных событий и приведите примеры их использования.
4. Опишите возможные типы конечных событий и приведите примеры их использования.
5. Опишите возможные типы задач бизнес-процесса и приведите примеры их использования.
6. Опишите возможные типы шлюзов и приведите примеры их использования.

Лабораторная работа №15

Построение диаграммы цепочки добавленной стоимости

Цель работы: освоить методологию моделирования ARIS и построить диаграмму цепочки добавленной стоимости

Формируемые компетенции: ПК 1.1

Теоретические сведения:

Модель цепочки добавленной стоимости относится к представлению управления. В общем виде цепочка добавленной стоимости – это согласованный набор видов деятельности, создающих ценность, начиная от исходных источников сырья вплоть до готовой продукции (услуги), доставленной конечному пользователю. Модель цепочки добавленной стоимости описывает бизнес-процессы, непосредственно участвующие в формировании величины добавленной стоимости предприятия. Цепочка добавленной стоимости является объектом типа функция. Модель имеет иерархическую структуру и состоит из совокупности взаимосвязанных диаграмм. Основные элементы модели цепочки добавленной стоимости приведены в таблице 15.1.

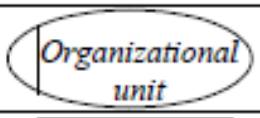
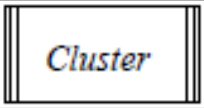
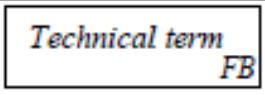
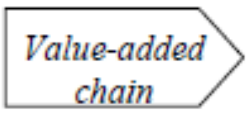
Представление процесса в виде модели цепочки добавленной стоимости в среде *ARIS* подчиняется определенным правилам:

• функции могут размещаться в функциональной последовательности в соответствии с добавлением стоимости;

• между функциями могут устанавливаться связи/отношения;

•функции могут быть разделены на подфункции.

Таблица 15.1 Основные элементы модели цепочки добавленной стоимости

Обозначение	Описание
	Организационная единица
	Кластер
	Специальный технический термин (относящийся к бизнес-процессу)
	Звено цепочки добавленной стоимости

Модель цепочки добавленной стоимости целесообразно использовать также для описания основных бизнес-процессов предприятия и размещения их в определенной последовательности. На более низких уровнях детализации можно представить элементы каждого бизнес-процесса в виде отдельной диаграммы цепочки добавленной стоимости. Отдельно от структуры процессов можно описать организационные элементы и потоки данных.

Детальное рассмотрение функций, определенных в цепочке добавленной стоимости бизнес-процесса, можно осуществлять путем применения процессных цепочек. Наиболее часто применяемый тип моделей представления процессных цепочек – расширенная событийно-ориентированная модель (*eEPC – extended Event Driven Process Chain*).

В соответствии с процессным подходом для уменьшения сложности представления основной деятельности предприятия как комплекса бизнеспроцессов необходимо разработать иерархию моделей, а именно модель верхнего уровня всего предприятия и модели отдельных бизнес-процессов. Модель цепочки добавленной стоимости целесообразно использовать для описания модели верхнего уровня, поэтому моделирование бизнес-процесса нужно начинать именно с построения такой модели. Пример диаграммы трехуровневой структуры бизнес-процесса закупок приведен на рисунке 15.1.

Первый уровень состоит из одной функции, представляющей собой, собственно, сам бизнес-процесс «Осуществлять закупки».

Второй уровень цепочки добавленной стоимости состоит их трех связанных действий: «Выполнять текущую деятельность», «Анализ результатов по закупкам» и «Формировать управляющее воздействие».

Третий уровень образуют функции, относящиеся к текущей деятельности – осуществлению закупок товарно-материальных ценностей (ТМЦ).

Кроме функций на диаграмме модели имеются элементы: технический термин, организационная единица и кластеры. Между элементами модели установлены связи типов:

• «Является иерархически вышестоящим для» – от функции верхнего уровня иерархии к функциям нижнего уровня;

• «Является предшествующим для» (*is predecessor of*) – для задания функциональной последовательности в соответствии с добавлением стоимости;

• «Является входом для» (*is input for*) – для задания входной/управляющей информации для функций;

• «Имеет выход из» (*has output of*) – от функции к кластеру;

• «Выполняет» (*executes*) – от организационной единицы к функции. Таким образом, на модели, представленной на рисунке 15.1, в самом общем виде показан бизнес-процесс (его исполнители, информационные объекты и управляющая информация).

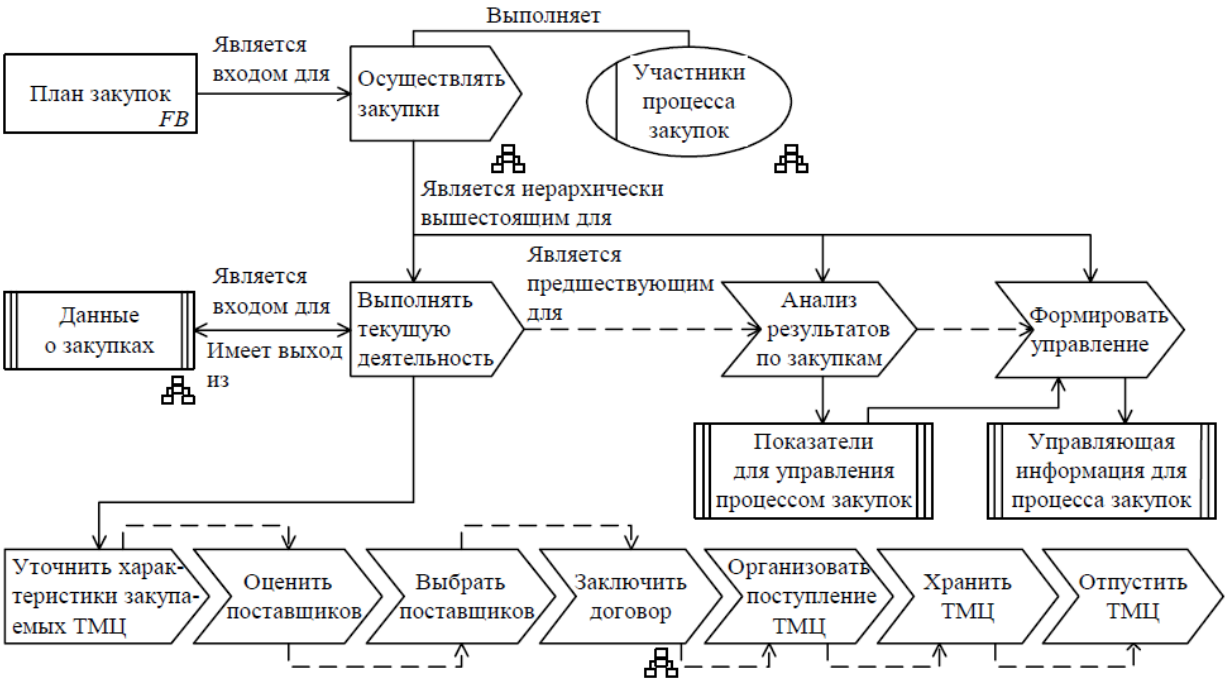


Рисунок 15.1 Модель цепочки добавленной стоимости бизнес-процесса «Осуществлять закупки»

Задание для выполнения:

1. Изучить теоретические данные по ARIS и модели цепочки добавленной стоимости;
2. Построить схему модели цепочки добавленной стоимости для бизнес-процесса выбранной ранее предметной области
3. Оформить отчет.

Содержание отчета:

5. Титульный лист;
6. Цель работы;
7. Ход работы (описание предметной области, модель цепочки добавленной стоимости);
8. Выводы по работе.

Контрольные вопросы:

1. Что вызывает потребность в стандартизации процесса разработки моделей бизнес-процессов?

2. Перечислите представления модели в архитектуре *ARIS*?

3. Какие основные виды диаграмм используются для моделирования бизнес-процессов в среде *ARIS*?

4. Каковы особенности и назначение модели цепочки добавленной стоимости?

СПИСОК ЛИТЕРАТУРЫ

1. Трусов, А. В. Технология проектирования информационных систем : учебное пособие / А. В. Трусов, В. А. Трусов. - М ; Вологда : Инфра-Инженерия. ЭБС «ZNANIUM.COM» : [сайт]. – URL: https://znanium.com/catalog/product/2100456 (дата обращения: 10.03.2025).

2. Брежнев, Р. В. Методы и средства проектирования информационных систем и технологий : учебное пособие / Р. В. Брежнев. - Красноярск : Сиб. федер. ун-т. ЭБС «ZNANIUM.COM» : [сайт]. – URL: https://znanium.com/catalog/product/1819341 (дата обращения: 10.03.2025).

3. Коваленко, В. В. Проектирование информационных систем : учебное пособие / В.В. Коваленко. — 2-е изд., перераб. и доп. — М : ИНФРА-М. ЭБС «ZNANIUM.COM» : [сайт].– URL: https://znanium.com/catalog/product/1894610 (дата обращения: 10.03.2025).