

УДК 004.415.2
ББК 32.971.32-02
П 79

Авторы-составители: А. Н. Семенюта, д-р техн. наук, профессор;
Л. В. Ятченко, ассистент

Рецензенты: С. Н. Курочка, канд. техн. наук, доцент, зав. кафедрой
информационных технологий Гомельского
государственного технического университета
им. П. О. Сухого;
С. М. Мовшович, канд. техн. наук, доцент Белорусского
торгово-экономического университета потребительской
кооперации

Рекомендован к изданию научно-методическим советом учрежде-
ния образования «Белорусский торгово-экономический университет
потребительской кооперации». Протокол № 4 от 11 февраля 2014 г.

Проектирование информационных систем : практикум для ре-
П 79 лизации содержания образовательных программ высшего образова-
ния I ступени / авт.-сост. : А. Н. Семенюта, Л. В. Ятченко. – Гомель :
учреждения образования «Белорусский торгово-экономический уни-
верситет потребительской кооперации», 2015. – 80 с.
ISBN 978-985-540-281-8

Издание содержит краткие теоретические сведения и задания на построение мо-
делей информационных систем с использованием функционально-модульного и объ-
ектно-ориентированного подходов.

Практикум предназначен для студентов специальности 1-26 03 01 «Управление
информационными ресурсами»

УДК 004.415.2
ББК 32.971.32-02

ISBN 978-985-540-281-8

© Учреждение образования «Белорусский
торгово-экономический университет
потребительской кооперации», 2015

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

При проектировании современных информационных систем используются два основных подхода. Первый подход называют функционально-модульным или структурным. В его основу положен принцип функциональной декомпозиции, при которой структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами. Второй подход, объектно-ориентированный, использует объектную декомпозицию. При этом структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. При этом общепризнано, что специалист в области информационных технологий должен владеть каждым из них.

Цель данного практикума – сформировать у студентов навыки практического применения функционально-модульного и объектно-ориентированного подходов при проектировании информационных систем с использованием среды Microsoft Visio.

Лабораторная работа 1

ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ В СТАНДАРТЕ IDEF0

Цели работы:

- изучение основных принципов моделирования систем на основе использования стандарта IDEF0;
- изучение методов создания IDEF0-диаграмм в среде Microsoft Visio.

1. Краткие сведения из теории

При создании любой информационной системы не обойтись без обследования объекта, на котором будет использоваться создаваемая система.

Специалисты по информационным технологиям при исследовании организаций часто используют соответствующие методологии, позволяющие понять работу объекта в целом путем построения его функциональной модели.

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Функциональная направленность означает, что функции системы исследуются независимо от механизмов, которые обеспечивают их выполнение. В целом такой подход направлен на изучение того, что делает исследуемая система, а не каким конкретно способом. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

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

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, т. е. нужно определить, что будет в дальнейшем рассматриваться как компоненты системы, а что – как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ.

Другими словами, первоначально необходимо определить область моделирования. Описание области как системы в целом, так и ее

компонентов является основой построения модели. Хотя предполагается, что в процессе моделирования область может корректироваться. В основном она должна быть сформулирована изначально, поскольку именно область определяет направление моделирования и время, когда должна быть закончена модель.

Таким образом, можно сказать, что методология функционального моделирования IDEF0 – это технология описания исследуемой системы в целом как множества взаимозависимых функций. «Функциональная» точка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации.

Первый шаг при построении модели IDEF0 заключается в определении назначения модели – набора вопросов, на которые должна отвечать модель, например:

- Почему необходимо моделировать данный процесс?
- Что можно будет выявить после разработки данной модели?

На втором шаге обычно определяются границы моделирования, которые предназначены для обозначения ширины охвата предметной области и глубины детализации и являются логическим продолжением уже определенного назначения модели.

На следующем шаге определяется предполагаемая целевая аудитория, для которой создается модель. От этого зависит уровень детализации, с которым должна создаваться модель. Перед построением модели необходимо иметь представление о том, какие сведения о предмете моделирования уже известны, какие дополнительные материалы и (или) техническая документация для понимания модели могут быть необходимы для целевой аудитории, какие язык и стиль изложения являются наиболее подходящими.

Далее определяется точка зрения, с позиций которой создается модель. Точка зрения выбирается таким образом, чтобы учесть уже обозначенные границы моделирования и назначение модели. Однажды выбранная точка зрения остается неизменной на протяжении всего процесса построения модели.

Непосредственная работа с элементами модели происходит с использованием правил графического языка IDEF0, в основе методологии которого лежат четыре основных понятия.

Первое из них – *функциональный блок (Activity Box)*. Графически функциональный блок изображается в виде прямоугольника и моделирует некоторую конкретную функцию в рамках рассматриваемой системы, например подготовку данных (рисунок 1).

Подготовка
данных

Рисунок 1 – Пример функционального блока

Название каждого функционального блока должно быть образовано с использованием глаголов или отглагольных существительных. Важно подбирать имена функциональных блоков так, чтобы они отражали точку зрения, используемую для моделирования.

Вторым понятием методологии IDEF0 является *интерфейсная дуга (Arrow)*. Графически интерфейсная дуга выглядит как однонаправленная стрелка. Интерфейсная дуга отображает элемент исследуемой системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком. Каждая интерфейсная дуга должна иметь свое уникальное наименование. Для названий интерфейсных дуг, как правило, употребляются имена существительные. С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т. д.) или потоки данных и информации (документы, данные, инструкции и т. д.).

На IDEF0-диаграммах возможны четыре типа стрелок (рисунок 2):

- стрелка входа (*Input*) – то, что потребляется в ходе выполнения процесса;
- стрелка управления (*Control*) – ограничения и инструкции, влияющие на ход выполнения процесса;
- стрелка выхода (*Output*) – то, что является результатом выполнения процесса;
- стрелка механизма исполнения (*Mechanism*) – то, что используется для выполнения процесса, но остается неизменным.

Каждый тип стрелки на диаграммах IDEF0 соединяется с определенной стороной функционального блока. Как и в случае с функциональными блоками, присвоение имен всем стрелкам на диаграмме является необходимым условием для понимания сути изображенного.

Стрелки входа. Вход представляет собой сырье или информацию, потребляемую или преобразуемую функциональным блоком для производства выхода. Стрелки входа всегда направлены в левую сторону прямоугольника, обозначающего в IDEF0 функциональный блок.

Наличие входных стрелок на диаграмме не является обязательным, так как возможно, что некоторые блоки ничего не преобразуют и не изменяют. Примером блока, не имеющего входа, может служить блок *Принятие решения руководством*, где анализируется несколько факторов, но ни один из них непосредственно не преобразуется и не потребляется в результате принятия какого-либо решения.

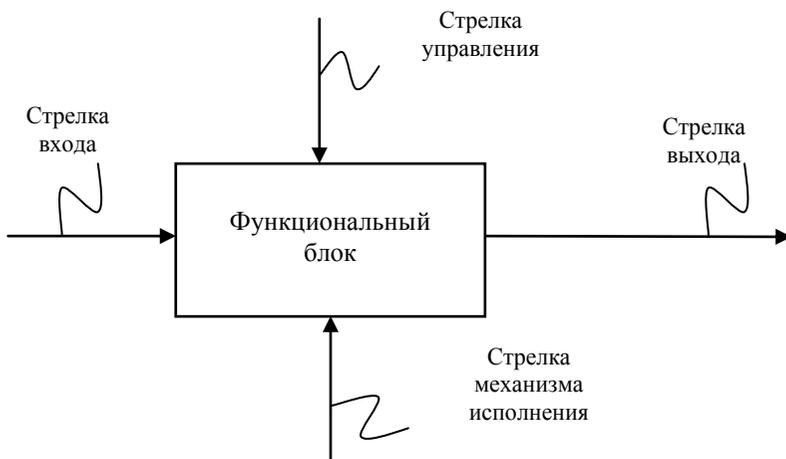


Рисунок 2 – Типы стрелок на диаграммах IDEF0

Стрелки управления. Стрелки управления отвечают за регулирование того, как и когда выполняется функциональный блок. Так как управление контролирует поведение функционального блока для обеспечения создания желаемого выхода, каждый функциональный блок должен иметь как минимум одну стрелку управления. Стрелки управления всегда входят в функциональный блок сверху.

Управление часто существует в виде правил, инструкций, законов, политики, набора необходимых процедур или стандартов. Влияя на работу блока, оно само остается неизменным. Может оказаться, что целью функционального блока является как раз изменение того или иного правила, инструкции, стандарта и т. п. В этом случае стрелка, содержащая соответствующую информацию, должна рассматриваться не как управление, а как вход функционального блока.

Управление можно рассматривать как специфический вид входа. В случаях, когда неясно, относится стрелка к входу или управлению, предпочтительно относить ее к управлению до момента, пока неясность не будет разрешена.

Стрелки выхода. Выход – это продукция или информация, получаемая в результате работы функционального блока. Каждый блок должен иметь как минимум один выход. Действие, которое не имеет никакого четко определяемого выхода, желательно не моделировать вообще.

Стрелки механизма исполнения. Механизмом выступает ресурс, который непосредственно исполняет моделируемое действие. В качестве механизмов исполнения могут моделироваться ключевой персонал, техника и (или) оборудование. Стрелки механизма исполнения могут отсутствовать, в случае если оказывается, что они не являются необходимыми для достижения поставленной цели моделирования.

Третьим основным понятием стандарта IDEF0 является *декомпозиция (Decomposition)*. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели. Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой и обозначается идентификатором «А-0». В пояснительном тексте к диаграмме должна быть указана цель построения диаграммы (в виде краткого описания) и зафиксирована точка зрения.

Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Например, функциональные модели одной и той же организации, с точек зрения главного технолога и финансового директора, будут существенно различаться по направленности их детализации.

В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подверга-

ется детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы, и называется «дочерней» по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется «дочерним блоком»). В свою очередь, функциональный блок-предок называется «родительским блоком» по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит, – «родительской диаграммой».

Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. Важно отметить, что в каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0-модели. Наглядно принцип декомпозиции представлен на рисунке 3.

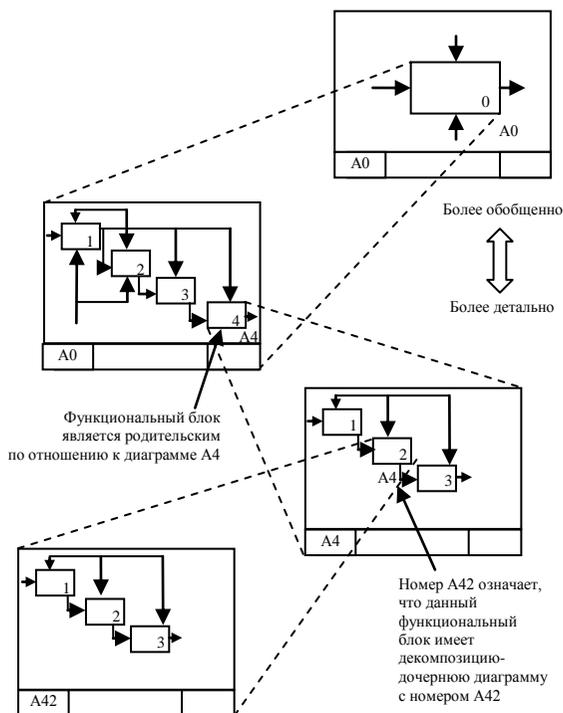


Рисунок 3 – Декомпозиция функциональных блоков IDEF0-диаграммы

Часто бывают случаи, когда отдельные интерфейсные дуги не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот – отдельные дуги не имеют практического смысла выше какого-то уровня. Например, интерфейсную дугу, изображающую деталь на входе в функциональный блок *Обработать на токарном станке*, не имеет смысла отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. С другой стороны, случается необходимость избавиться от отдельных «концептуальных» интерфейсных дуг и не детализировать их глубже некоторого уровня. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Изображение «туннеля» в виде двух квадратных скобок вокруг начала интерфейсной дуги означает, что эта дуга не была унаследована от функционального родительского блока и появилась (из «туннеля») только на этой диаграмме. В свою очередь, такое же изображение вокруг конца (стрелки) интерфейсной дуги в непосредственной близости от блока-приемника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет.

Последним из понятий IDEF0 является *гlossарий (Glossary)*. Для каждого из элементов IDEF0 – диаграмм, функциональных блоков, интерфейсных дуг – существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т. д., которые характеризуют объект, отображенный данным элементом (см. рисунок 3).

Этот набор называется гlossарием и является описанием сущности данного элемента. Например, для управляющей интерфейсной дуги *распоряжение об оплате* гlossарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т. д. Гlossарий гармонично дополняет диаграммы необходимой информацией.

2. Пример

В практикуме в качестве учебного примера рассматривается сквозная задача по проектированию информационной системы для решения проблемы сокращения времени выполнения заказов клиентов вымышленной организации ООО «Компьютер». Эта организация

занимается сборкой, а затем продажей собранных компьютеров. Для этого она закупает комплектующие для сборки компьютеров от нескольких поставщиков, собирает из них настольные компьютеры и ноутбуки, а затем продает их.

Предположим, что на основании проведенного анкетирования клиенты данной организации недовольны сроками исполнения их заказов. Руководство ООО «Компьютер» не может оставить эту ситуацию без внимания и хотело бы использовать возможности современных информационных технологий для решения этой проблемы.

Предложения по использованию современных информационных технологий должны основываться на результатах анализа деятельности организации. Поэтому первым шагом по обоснованию предлагаемых решений по созданию необходимых информационных систем всегда является исследование объекта, для которого планируется создание информационной системы.

Анализ деятельности любого объекта предполагаемого внедрения информационной системы с использованием IDEF0-методологии всегда начинается с построения контекстной диаграммы (диаграммы нулевого уровня), которая изображает исследуемый объект как единое целое со всеми связями с внешней средой.

Исходя из имеющейся информации можно сделать вывод, что контекстная IDEF0-диаграмма для ООО «Компьютер» может иметь вид, представленный на рисунке 4.

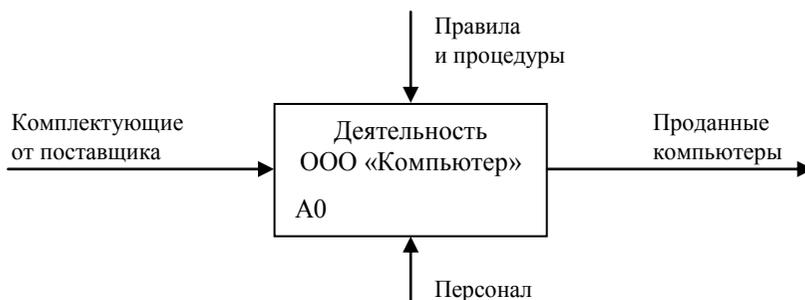


Рисунок 4 – Контекстная диаграмма IDEF0-модели деятельности ООО «Компьютер»

Для построения данной диаграммы в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio.

2. В появившемся окне выберите слева категорию шаблонов *Блок-схема* и в ней – элемент *Схема IDEF0* (рисунок 5).

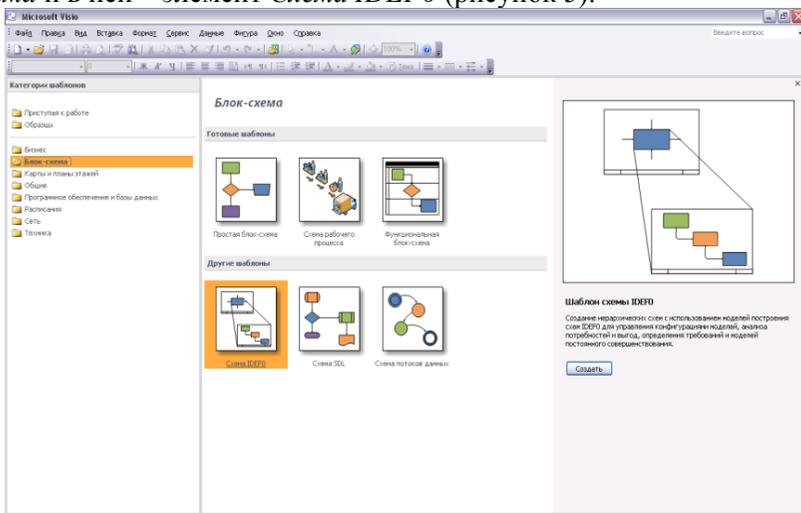


Рисунок 5 – Диалоговое окно выбора требуемой блок-схемы

3. Нажмите кнопку *Создать* в правой части экрана, в результате чего на экране появится окно, в левой части которого будет отображен набор графических символов для построения IDEF0-диаграмм (рисунок 6), а в правой части – лист (рабочее поле) для построения соответствующей диаграммы.

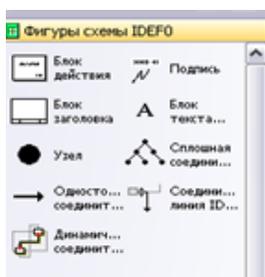
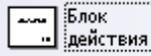


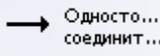
Рисунок 6 – Графические символы для построения IDEF0-диаграмм

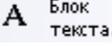
4. Для создания контекстной диаграммы IDEF0-модели деятельности ООО «Компьютер» с использованием мыши перетащите на рабочее поле графический символ



чье поле графический символ

5. В появившемся диалоговом окне *Специальные настройки* укажите имя процесса – *Деятельность ООО «Компьютер»*.

6. С помощью элемента  создайте необходимые стрелки контекстной диаграммы.

7. С помощью элемента  или двойного нажатия левой кнопкой мыши по стрелке введите необходимые названия стрелок: *Комплекующие от поставщиков*, *Правила и процедуры*, *Проданные компьютеры*, *Персонал*.

8. Для удобства использования диаграмм строящейся модели переименуйте рабочую страницу. Для этого внизу окна двойным нажатием левой кнопкой мыши по названию страницы *Страница-1* (или с помощью контекстного меню *Переименовать страницу*) выделите текущее имя и введите новое – *Контекстная диаграмма* (рисунок 7).

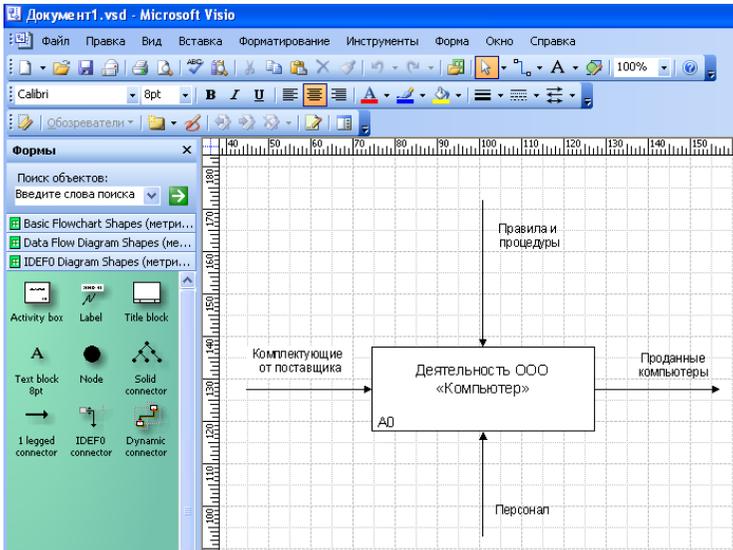


Рисунок 7 – Диалоговое окно *Контекстная диаграмма*

Контекстная диаграмма не показывает деталей деятельности ООО «Компьютер». Поэтому далее предполагаем, что в результате более глубокого изучения исследуемого объекта (дообследования ООО «Компьютер») получена следующая информация:

- менеджер по продажам принимает заказы от клиентов, а затем передает их в сборочный цех для выполнения;
- техники-сборщики из комплектующих, получаемых со склада, собирают заказанные компьютеры;
- собранные компьютеры передаются на склад для отгрузки клиентам;
- работник склада отгружает клиентам собранные компьютеры, а также принимает присылаемые от поставщиков комплектующие.

На основании этой информации можно сделать вывод о том, что основными функциями, реализуемыми ООО «Компьютер», являются:

- *Прием заказов;*
- *Сборка компьютеров;*
- *Отгрузка готовых компьютеров;*
- *Прием комплектующих.*

Полученная информация позволяет декомпозировать контекстную диаграмму ООО «Компьютер» в соответствии с рисунком 8.

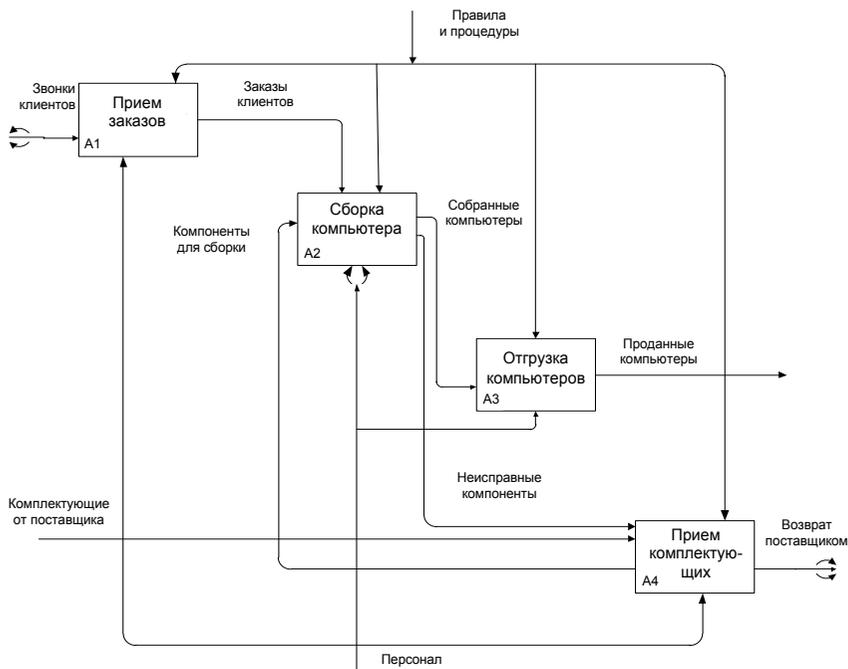


Рисунок 8 – Диаграмма декомпозиции первого уровня деятельности ООО «Компьютер»

Для построения данной диаграммы в среде Microsoft Visio выполните следующие действия:

1. Выполните команду *Вставка/Создать страницу*. В появившемся диалоговом окне *Параметры страницы* на вкладке *Свойства страницы* введите название создаваемой диаграммы – *Диаграмма декомпозиции первого уровня деятельности* (рисунок 9). Затем поставьте флажок *Открыть страницу в новом окне* и нажмите *ОК*.

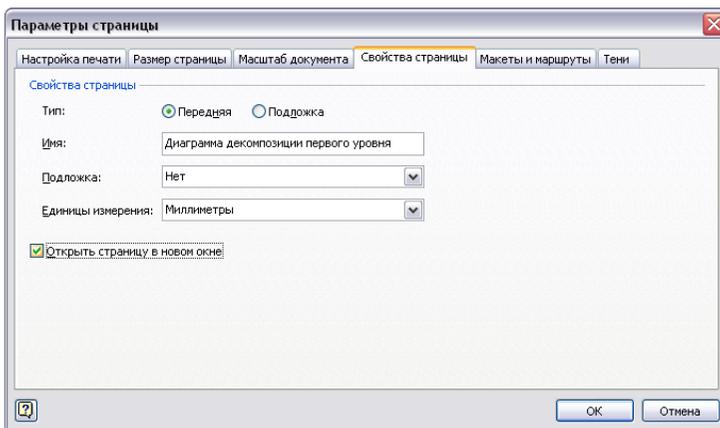
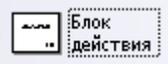
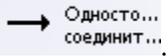
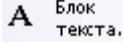
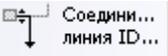


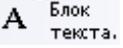
Рисунок 9 – Диалоговое окно *Параметры страницы*

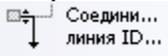
2. С помощью элемента  на рабочем поле создайте четыре блока: *Прием заказов*, *Сборка компьютеров*, *Отгрузка готовых компьютеров*, *Прием комплектующих*, а в качестве их идентификаторов – *A1, A2, A3, A4* соответственно.

3. Создайте необходимые стрелки на диаграмме с помощью элемента  мента

4. Подпишите стрелки с помощью элемента  **А** *Блок текста*, или с помощью двойного нажатия левой кнопкой мыши по соответствующей стрелке.

5. С помощью элемента  создайте все остальные стрелки.

6. Введите необходимые подписи к стрелкам (*Звонки клиентов*, *Правила и процедуры*, *Заказы клиентов*, *Собранные компьютеры*, *Компоненты для сборки*, *Неисправные компоненты*, *Персонал*) с помощью элемента  **А** *Блок текста*, или с помощью двойного нажатия левой кнопкой мыши по соответствующей стрелке.

Примечание – Для создания разветвленных стрелок *Правила и процедуры* и *Персонал* используйте элемент , объединяя их через точки соединения на соответствующих стрелках .

7. Для создания туннелированной стрелки *Звонки клиентов* выделите ее и в контекстном меню выберите команду *Туннелировать вход*. Выполните аналогичные действия для стрелки *Возврат поставщикам* с той лишь разницей, что в контекстном меню надо выбрать команду *Туннелировать выход*.

3. Задание

Постройте диаграмму декомпозиции функционального блока *Сборка компьютеров*, которая должна иметь вид, представленный на рисунке 10.

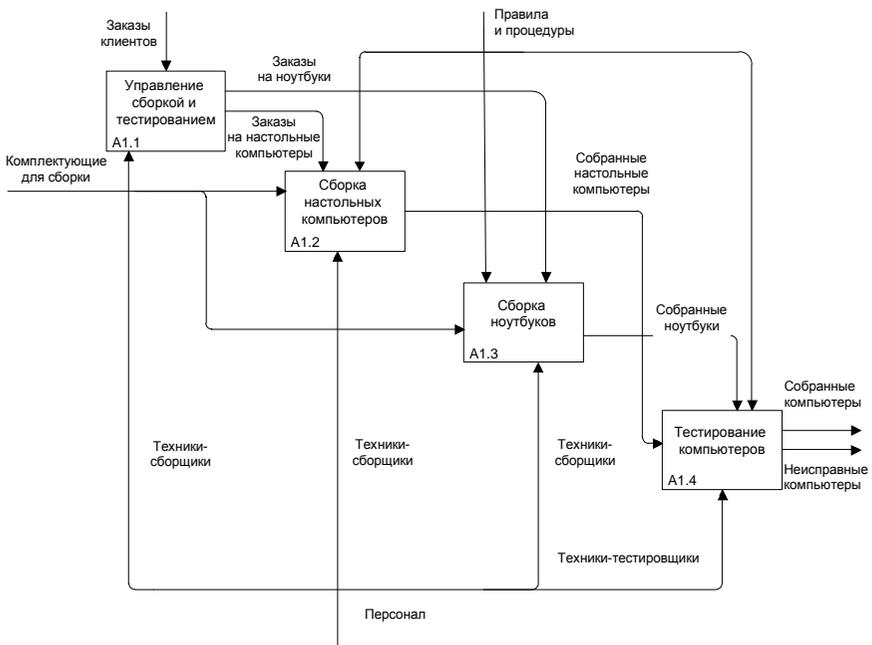


Рисунок 10 – Диаграмма декомпозиции функционального блока «Сборка компьютеров»
Лабораторная работа 2
ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Цели работы:

- изучение основных принципов построения диаграмм вариантов использования;
- изучение методов создания диаграмм вариантов использования в среде Microsoft Visio.

1. Краткие сведения из теории

Процесс проектирования любой информационной системы начинается с формулирования требований к создаваемой информационной системе (что должно быть реализовано в создаваемой системе). Эти требования обычно вырабатываются на основе анализа функционирования объекта предполагаемого внедрения информационной системы.

Для наглядного представления требований к создаваемой информационной системе в последнее время используются так называемые диаграммы вариантов использования (use case).

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой и отражает представление о поведении системы с точки зрения пользователя. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать, или целей, которые он преследует по отношению к разрабатываемой системе.

Цель построения диаграмм вариантов использования – документирование функциональных требований к системе в самом общем виде.

На рисунке 11 показан пример такой диаграммы для банковской информационной системы. На данной диаграмме человеческие фигурки обозначают действующих лиц, овалы – собственно варианты использования, а линии и стрелки – различные связи между действующими лицами и вариантами использования.

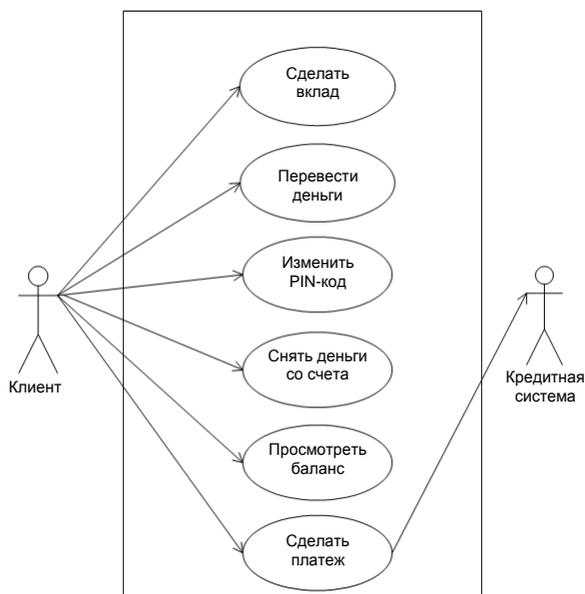


Рисунок 11 – Пример диаграммы вариантов использования

Из рисунка 11 можно сделать вывод о том, что с данной системой (существующей или создаваемой) взаимодействуют *Клиент* и *Кредитная система*. Кроме этого из диаграммы следует, что изучаемая система предназначена для выполнения следующих шести основных действий (функций):

- *Сделать вклад.*
- *Перевести деньги.*
- *Изменить PIN-код.*
- *Снять деньги со счета.*
- *Просмотреть баланс.*
- *Сделать платеж.*

В целом данная диаграмма отражает функциональные требования (что должна делать система) с точки зрения пользователя.

Таким образом, варианты использования – это функции, выполняемые системой, а действующие лица – это заинтересованные лица по отношению к создаваемой системе.

Диаграммы вариантов использования показывают, какие действующие лица инициируют варианты использования. Из них также видно, кто кому передает необходимую информацию. Например, из ри-

сунка 11 следует, что вариант использования *Сделать платеж* предоставляет *Кредитной системе* информацию об оплате по кредитной карточке.

Действующие лица могут играть различные роли по отношению к варианту использования. Они могут пользоваться его результатами, а также непосредственно в нем участвовать. Значимость различных ролей действующего лица зависит от того, каким образом используются его связи.

Обычно источником идентификации вариантов использования служат внешние события. Следует начать с перечисления всех событий, происходящих во внешнем мире, на которые система должна каким-то образом реагировать. Какое-либо конкретное событие может повлечь за собой реакцию системы, не требующую вмешательства пользователей, или, наоборот, вызвать чисто пользовательскую реакцию. Идентификация событий, на которые необходимо реагировать, помогает идентифицировать варианты использования.

Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Для последующего проектирования системы требуются более конкретные детали. Эти детали описываются в документе, называемом «сценарий варианта использования» или «поток событий». Целью потока событий является подробное документирование процесса взаимодействия действующего лица с системой, реализуемого в рамках варианта использования.

Описание потока событий в общем случае включает следующие разделы:

- *Краткое описание.*
- *Предусловия (pre-conditions).*
- *Основной поток событий.*
- *Альтернативные потоки событий.*
- *Постусловия (post-conditions).*
- *Расширения (extensions).*

1. *Краткое описание.* Каждый вариант использования должен иметь краткое описание того, что в нем происходит. Например, вариант использования *Перевести деньги* может содержать следующее описание: «Вариант использования *Перевести деньги* позволяет клиенту или служащему банка переводить деньги с одного счета до востребования или сберегательного счета на другой счет».

2. *Предусловия.* Предусловия варианта использования – это такие условия, которые должны быть выполнены, прежде чем вариант ис-

пользования начнет выполняться. Например, таким условием может быть выполнение другого варианта использования или наличие у пользователя прав доступа, требуемых для начала работы.

3. *Основной и альтернативный потоки событий.* Конкретные детали вариантов использования описываются в основном в альтернативных потоках событий. Поток событий поэтапно описывает, что должно происходить во время выполнения заложенной в варианты использования функциональности. Поток событий уделяет внимание тому, что будет делать система, а не как она будет это делать, причем описывает все это с точки зрения пользователя. Основной поток событий описывает нормальный ход событий (при отсутствии ошибок) и при наличии нескольких возможных вариантов хода событий может разветвляться на подчиненные потоки. Альтернативные потоки описывают отклонения от нормального хода событий (ошибочные ситуации) и их обработку.

Например, в банковской информационной системе потоки событий варианта использования *Снять деньги со счета* могут выглядеть следующим образом:

Основной поток событий.

1. Вариант использования начинается, когда клиент вставляет свою карточку в банкомат.

2. Банкомат выводит приветствие и предлагает клиенту ввести свой персональный PIN-код.

3. Клиент вводит PIN-код.

4. Банкомат подтверждает введенный код.

5. Банкомат выводит список доступных действий:

- сделать вклад;
- снять деньги со счета;
- перевести деньги.

6. Клиент выбирает пункт *Снять деньги со счета*.

7. Банкомат запрашивает, сколько денег надо снять.

8. Клиент вводит требуемую сумму.

9. Банкомат определяет, имеется ли на счету достаточно денег.

10. Банкомат вычитает требуемую сумму из счета клиента.

11. Банкомат выдает клиенту требуемую сумму наличными.

12. Банкомат возвращает клиенту его карточку.

13. Банкомат печатает чек для клиента.

14. Вариант использования завершается.

Альтернативный поток событий 1. Ввод неправильного PIN-кода.

4a1. Банкомат информирует клиента, что код введен неправильно.

4а2. Банкомат возвращает клиенту его карточку.

4а3. Вариант использования завершается.

Альтернативный поток событий 2. Недостаточно денег на счете.

9а1. Банкомат информирует клиента, что денег на его счете недостаточно.

9а2. Банкомат возвращает клиенту его карточку.

9а3. Вариант использования завершается.

Альтернативный поток событий 3. Ошибка в подтверждении запрашиваемой суммы.

9б1. Банкомат сообщает пользователю, что при подтверждении запрашиваемой суммы произошла ошибка, и дает ему номер телефона службы поддержки клиентов банка.

9б2. Банкомат заносит сведения об ошибке в журнал ошибок. Каждая запись содержит дату и время ошибки, имя клиента, номер его счета и код ошибки.

9б3. Банкомат возвращает клиенту его карточку.

9б4. Вариант использования завершается.

4. *Постусловия.* Постусловиями называются такие условия, которые всегда должны быть выполнены после завершения варианта использования.

5. *Расширения.* Этот пункт присутствует, если в основном потоке событий имеют место относительно редко встречающиеся ситуации (частные случаи).

2. Пример

Будем считать, что на основании результатов обследования деятельности ООО «Компьютер» выявлено, что в данной организации время выполнения заказов клиентов не контролируется. Отсюда вытекает предложение о целесообразности создания информационной системы, которая отслеживала бы реальное время прохождения заказов внутри организации. Тогда на основании этой информации руководство могло бы при необходимости выработать соответствующие меры по сокращению времени выполнения заказов.

Для конкретизации задачи создания автоматизированной системы контроля времени выполнения заказов выявлены события, на которые должна реагировать создаваемая система, а именно:

- Менеджер по продажам вводит в систему требуемую информацию о заказе и времени его поступления.

- Менеджер по продажам фиксирует время передачи заказа оператору-сборщику.

- Менеджер по продажам фиксирует время передачи собранного компьютера работнику склада (для упрощения задачи предполагается, что менеджеру по продажам время передачи собранного компьютера работнику склада сообщает оператор-сборщик).

- Менеджер по продажам фиксирует время выдачи заказа клиенту (также предполагается, что менеджеру по продажам время выдачи заказа клиенту сообщает работник склада).

- Менеджер по продажам хочет узнать статус заказа (на каком этапе выполнения находится заказ).

- Менеджер по продажам или директор организации хочет получить статистический отчет о сроках исполнения заказов.

Таким образом, на основании предъявленных требований можно сделать вывод о том, что в создаваемой системе необходимо иметь шесть вариантов использования:

- *Зафиксировать поступление заказа.*
- *Зафиксировать дату и время передачи заказа оператору сборки.*
- *Зафиксировать дату и время передачи заказа работнику склада.*
- *Зафиксировать дату и время выдачи заказа клиенту.*
- *Определить статус заказа.*
- *Получить статистику о сроках исполнения заказов.*

Для построения диаграммы вариантов использования создаваемой системы в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней – элемент *Схема модели UML* (рисунок 12).

2. Нажмите кнопку *Создать* в правой части экрана. Слева появится диалоговое окно Microsoft Office Visio *Фигуры схемы модели UML* (рисунок 13). Для создания диаграммы вариантов использования щелкните левой кнопкой мыши на пакете *Сценарий выполнения UML* в левой части рабочего окна.

3. С помощью графического элемента *Граница системы*  пакета *Сценарий выполнения UML* создайте границы проектируемой системы.

4. С помощью элемента *Сценарий выполнения*  пакета *Сценарий выполнения UML* поместите в границах системы новый вариант использования.

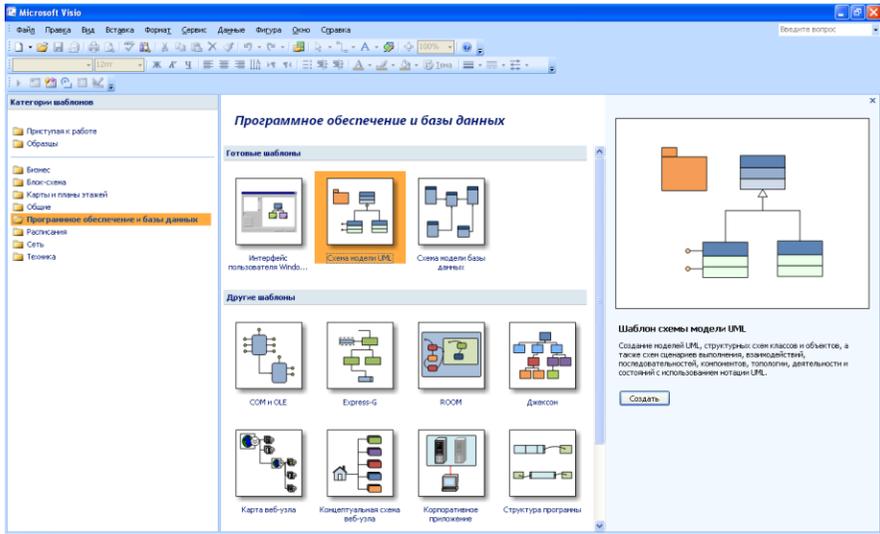


Рисунок 12 – Диалоговое окно выбора требуемой блок-схемы

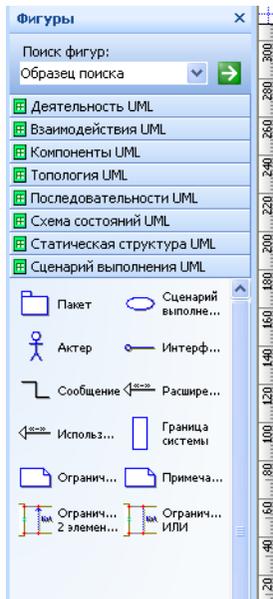


Рисунок 13 – Диалоговое окно Microsoft Office Visio.
Фигуры-схемы UML

5. Двойным нажатием левой кнопкой мыши (либо через контекстное меню – *Свойства*) выберите вариант использования, в появившейся вкладке *Свойства сценария выполнения UML* в поле *Имя* введите *Зафиксировать поступление заказа* (рисунок 14).

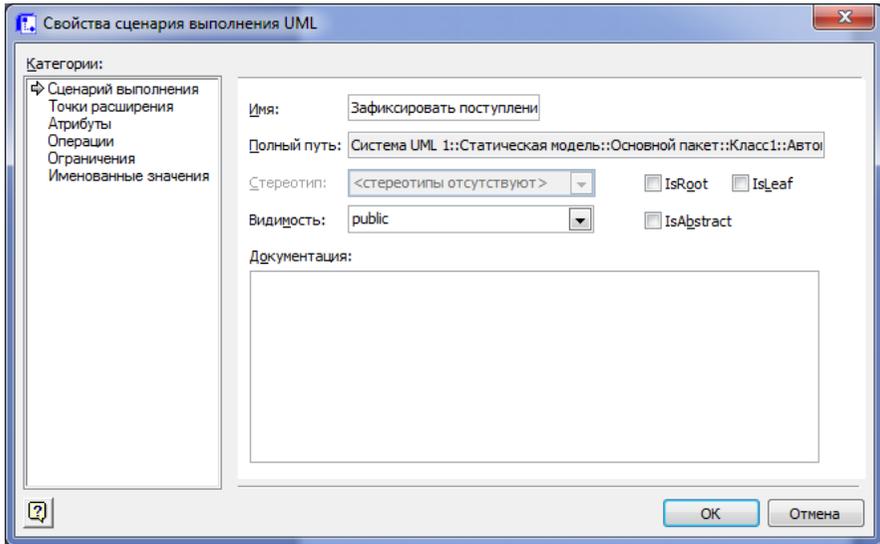


Рисунок 14 – Окно свойства сценария выполнения UML

6. С помощью элемента *Актер*  поместите на диаграмму действующее лицо. Двойным нажатием левой кнопкой мыши (либо через контекстное меню – *Свойства*) по данному элементу в появившейся вкладке *Свойства актера UML* в поле *Имя* введите *Менеджер по продажам*.

7. Переместите элемент *Сообщение*  на диаграмму. Используйте его для соединения элементов *Актер* и *Сценарий выполнения*. Для правильного отображения данного элемента сделайте следующее:

7.1. Двойным нажатием левой кнопкой мыши откройте окно *Свойства ассоциации UML* и укажите имя *Участствует в*. Затем в таблице *Окончания ассоциаций* в строчке с окончанием *Конец2* установите флажок в поле *IsNavigable* или *Перемещаемый* (рисунок 15).

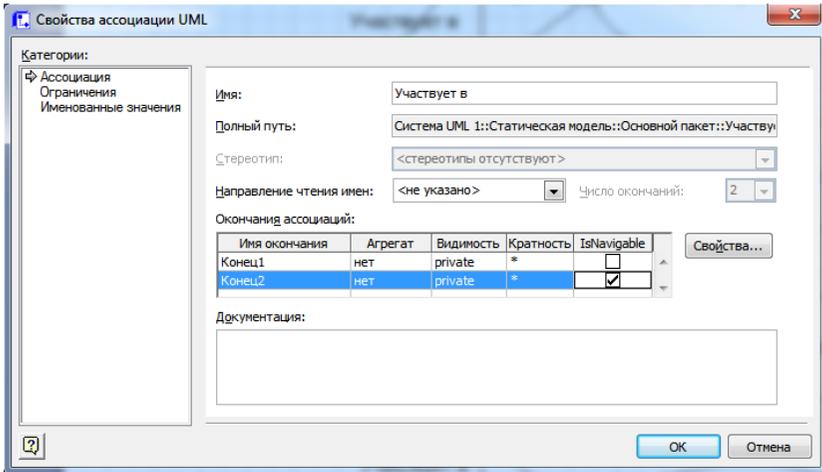


Рисунок 15 – Свойства ассоциации UML

7.2. Нажатием правой кнопкой мыши по элементу *Сообщение* откройте контекстное меню и выберите *Параметры отображения фигуры*. Установите флажки, как показано на рисунке 16.

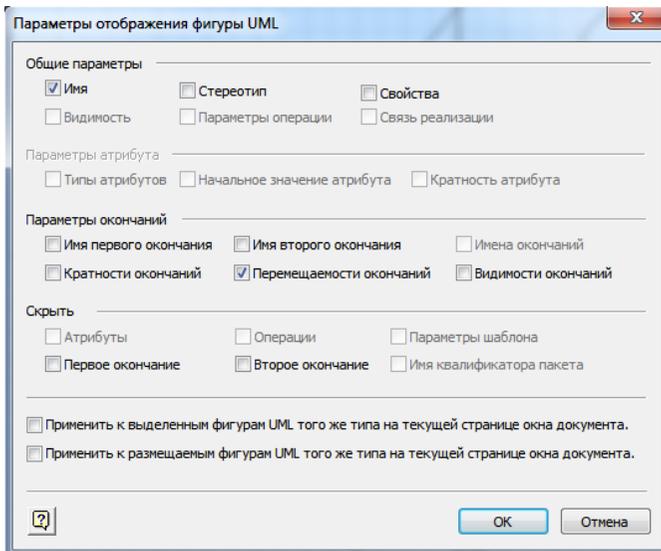


Рисунок 16 – Параметры отображения фигуры UML

8. В результате проделанных операций получим диаграмму с первым вариантом использования (рисунок 17).



Рисунок 17 – Диаграмма с одним из вариантов использования

Для конкретизации данного варианта использования дополним его основным сценарием:

- Вариант использования начинается, когда клиент сообщает менеджеру по продажам, что он готов сделать новый заказ.
- Менеджер по продажам запрашивает у клиента необходимую персональную информацию, требующуюся для оформления заказа.
- Если клиент обращается в первый раз, то персональная информация о данном клиенте заносится в базу данных системы.
- Клиент сообщает менеджеру по продажам информацию о заказе (требуемую конфигурацию аппаратных средств и необходимое программное обеспечение).
- Менеджер по продажам вводит в систему информацию о поступившем заказе (требуемую конфигурацию аппаратных средств, необходимое программное обеспечение, дату и время его приема).
- Вариант использования завершается.

Для внесения этого сценария на построенную ранее диаграмму (см. рисунок 17) запустите окно *Документация (UML/Вид/Документация)* и выделите вариант использования. Затем перейдите в окно *Документация* и введите текст сценария (рисунок 18).

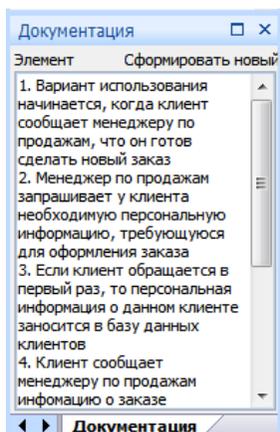


Рисунок 18 – Сценарий варианта использования *Зафиксировать дату и время поступления нового заказа*

3. Задание

Постройте окончательную диаграмму вариантов использования создаваемой системы, которая должна иметь вид, представленный на рисунке 19.

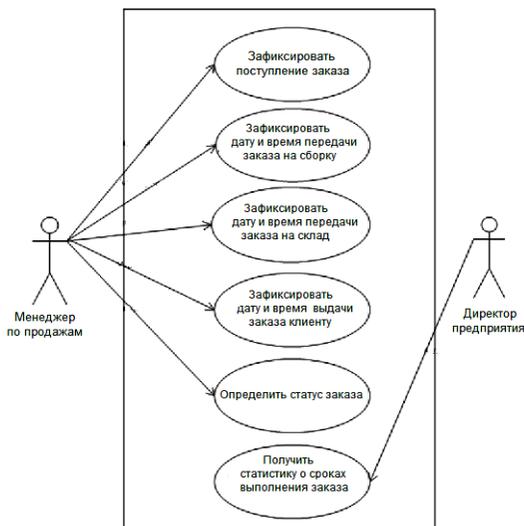


Рисунок 19 – Диаграмма вариантов использования автоматизированной системы контроля времени выполнения заказов

Лабораторная работа 3

МОДЕЛИ ДАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Цели работы:

- изучение принципов моделирования данных, хранящихся в информационных системах;
- изучение методов создания диаграмм моделей данных в среде Microsoft Visio.

1. Краткие сведения из теории

В процессе проектирования информационной системы необходимо определить информацию или данные, которые должны храниться и использоваться в ней. Для графического отображения таких данных традиционно используются диаграммы типа «сущность-связь».

Диаграмма «сущность-связь» (Entity Relationship Diagram – ERD) графически представляет структуру данных проектируемой информационной системы, позволяет рассмотреть систему целиком и выявить требования, необходимые для ее разработки, касающиеся хранения необходимых данных.

ER-диаграммы состоят из трех элементов: сущностей, атрибутов и взаимосвязей.

Сущность – это субъект, место, вещь, событие или понятие, содержащее информацию. Точнее, сущность – это набор (объединение) объектов, называемых экземплярами.

Сущности отображаются при помощи прямоугольников, содержащих имя. Имена принято выражать существительными в единственном числе, взаимосвязи – при помощи линий, соединяющих отдельные сущности. Взаимосвязь показывает, что данные одной сущности ссылаются или связаны с данными другой сущности.

В приведенном на рисунке 20 примере сущность *Читатель* представляет всех возможных читателей, а сущность *Книга* – все книги, которые могут быть взяты читателем.

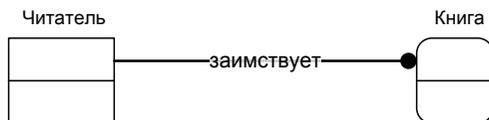


Рисунок 20 – Пример диаграммы «сущность-связь»
без атрибутов сущностей

Логические взаимосвязи представляют собой связи между сущностями. Они определяются глаголами показывающими, как одна сущность относится к другой.

Некоторые примеры взаимосвязей:

- *Команда включает много игроков.*
- *Самолет перевозит много пассажиров.*
- *Продавец продает много продуктов.*

Во всех этих случаях взаимосвязи отражают взаимодействие между двумя сущностями, называемое «один-ко-многим». Это означает, что один экземпляр первой сущности взаимодействует с несколькими экземплярами другой сущности. При использовании стандарта IDEF1X для моделирования данных взаимосвязи «один-ко-многим» отображаются линиями, соединяющими две сущности с точкой на одном конце и глаголом, располагаемым над линией.

Кроме взаимосвязи «один-ко-многим» существует еще один тип – «многие-ко-многим». Этот тип связи описывает ситуацию, при которой экземпляры сущностей могут взаимодействовать с несколькими экземплярами других сущностей. Связь «многие-ко-многим» используют на первоначальных стадиях проектирования. Этот тип взаимосвязи отображается сплошной линией с точками на обоих концах. Связь «многие-ко-многим» может не учитывать определенные ограничения системы, поэтому может быть заменена на «один-ко-многим» при последующем пересмотре проекта.

Если взаимосвязи между сущностями были правильно установлены, то можно составить предложения, их описывающие. Например, по модели, показанной на рисунке 20, можно составить следующие предложения:

- *Читатель заимствует книги.*
- *Много книг заимствовано одним читателем.*

Составление таких предложений позволяет проверить соответствие полученной модели требованиям и ограничениям создаваемой системы.

Каждый экземпляр сущности обладает набором характеристик. Так, каждый читатель может иметь имя, адрес, телефон и т. д. Все эти характеристики называются атрибутами сущности. На рисунке 21 в качестве примера показана ER-диаграмма, включающая в себя атрибуты сущностей.

Дальнейшее уточнение модели данных производится путем определения ключевых полей каждой сущности, в результате чего строится диаграмма модели данных с ключевыми атрибутами. Основное внимание здесь уделяется определению так называемых первичных

ключей каждой сущности. Первичный ключ предназначен для уникальной идентификации каждого экземпляра сущности.



Рисунок 21 – Пример диаграммы «сущность-связь» с атрибутами сущностей

Для определения первичного ключа сущности сначала необходимо выделить группу атрибутов, которые потенциально могут стать первичным ключом (потенциальные ключи), затем произвести отбор атрибутов для включения в состав первичного ключа согласно следующим рекомендациям:

- Первичный ключ должен быть подобран таким образом, чтобы по значениям атрибутов, в него включенных, можно было точно идентифицировать экземпляр сущности.
- Никакой из атрибутов первичного ключа не должен иметь нулевое значение.
- Значения атрибутов первичного ключа не должны меняться. Если значение изменилось, значит, это уже другой экземпляр сущности.

При выборе первичного ключа можно внести в сущность дополнительный атрибут и сделать его ключом. Так, для определения первичного ключа часто используют уникальные номера, которые могут автоматически генерироваться системой при добавлении экземпляра сущности в базу данных. Применение уникальных номеров облегчает процесс индексации и поиска в базе данных.

На диаграммах «сущность-связь» каждая сущность содержит горизонтальную линию, разделяющую атрибуты на две группы. Атрибуты, расположенные над линией, называются первичным ключом. При проведении связи между двумя сущностями в дочерней сущности автоматически образуются внешние ключи (foreign key). Связь образует ссылку на атрибуты первичного ключа в дочерней сущности, и эти атрибуты образуют внешний ключ в дочерней сущности. Атрибуты внешнего ключа обозначаются символами (FK) после своего имени.

2. Пример

Построим диаграммы «сущность-связь» для рассматриваемой в практикуме задачи создания автоматизированной системы контроля времени выполнения заказов в ООО «Компьютер».

Исходя из анализа требований, предъявленных к проектируемой системе, можно сделать вывод о том, что она должна содержать данные о клиентах и их заказах. Это определяет сущности, информация о которых должна храниться в создаваемой системе (сущности *Клиент* и *Заказ*). Предполагая, что один клиент может иметь несколько заказов, сделаем вывод о том, что связь между данными сущностями будет «один-ко-многим».

Для построения ER-диаграммы в стандарте IDEF1X в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора категории шаблона выберите категорию *Программное обеспечение и базы данных* и в ней – элемент *Схема модели базы данных*. Нажмите кнопку *Создать* в правой части экрана (рисунок 22).

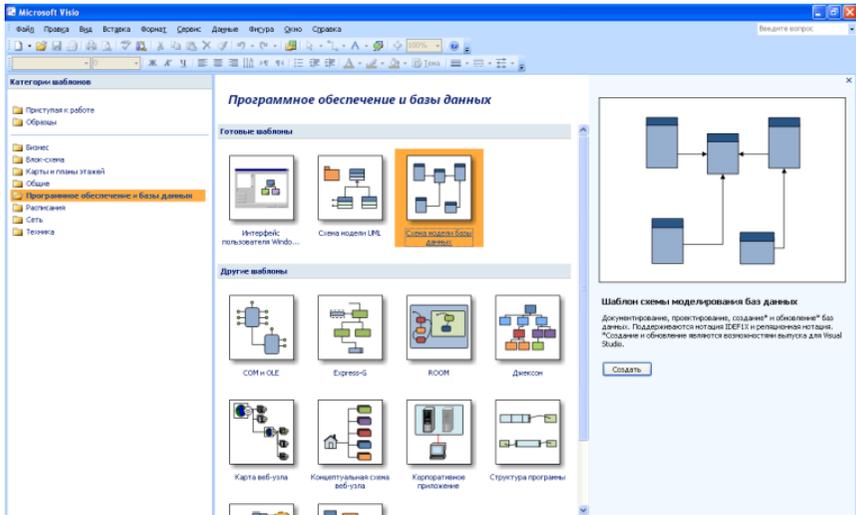


Рисунок 22 – Диалоговое окно выбора требуемой блок-схемы

2. Для задания используемой при построении диаграммы «сущность-связь» нотации выполните команду главного меню *База данных/Параметры/Документ*. В открывшемся окне на вкладке *Общие* в меню *Набор*

символов установите переключатель на *IDEFIX*. Диаграммы «сущность-связь» используются для построения логических моделей информационных систем, поэтому на вкладке *Общие* в качестве имен, видимых на схеме, укажите *концептуальные имена* (рисунок 23).

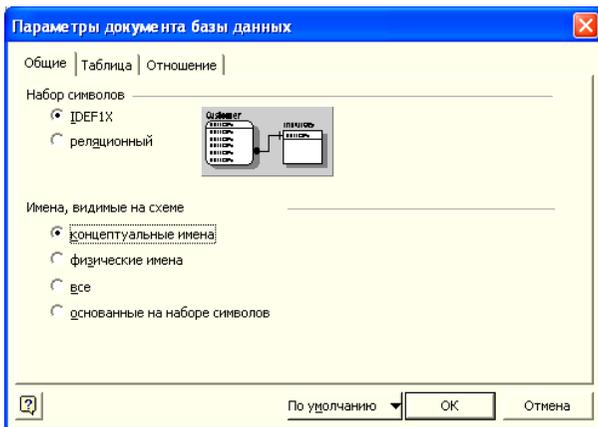


Рисунок 23 – Настройка параметров модели

3. Для настройки отображения в модели мощности связи и ее имени в закладке *Отношение* окна *Параметры документа базы данных* отметьте соответствующие пункты (рисунок 24), а затем нажмите кнопку *OK*.

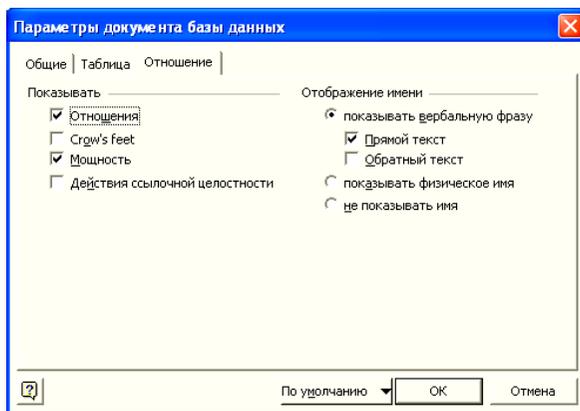


Рисунок 24 – Настройка вида отношений информационной модели

4. Для создания сущности *Клиент* перетащите элемент *Сущность*



на рабочее поле, нажмите правую кнопку мыши на сущности и в открывшемся меню (рисунок 25) выберите пункт *Свойства базы данных*.

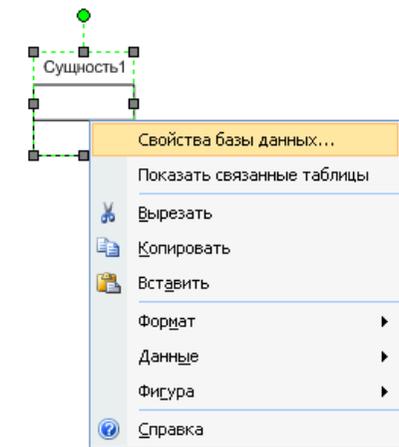


Рисунок 25 – Контекстное меню элемента *Сущность*

5. В категории *Определение* окна *Свойства базы данных* введите физическое имя сущности *Клиент* (рисунок 26).

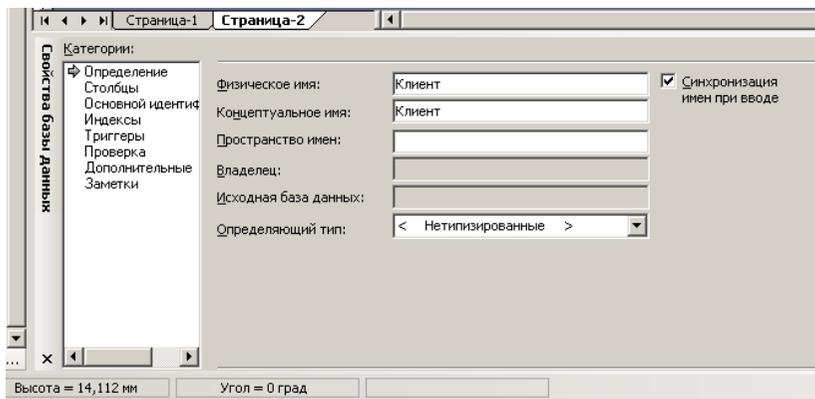


Рисунок 26 – Свойства сущности *Клиент*

6. Аналогичным образом создайте сущность *Заказ*, в результате чего на рабочем поле появятся две сущности (рисунок 27).

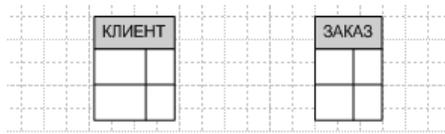


Рисунок 27 – Созданные сущности

7. Для отображения связи «один-ко-многим» между сущностями данной модели на создаваемой диаграмме необходимо перетащить на рабочую область элемент *Отношение* , поднести один конец стрелки к середине родительской сущности (*Клиент*), другой – к дочерней (*Заказ*). При этом каждая связанная сущность будет подсвечена красным цветом (рисунок 28).



Рисунок 28 – Создание связи между сущностями

8. Щелкните правой кнопкой мыши по созданному элементу *Отношение* и в открывшемся меню выберите пункт *Свойства базы данных*. В категории *Имя* окна *Свойства базы данных* введите вербальную фразу *Размещает* (рисунок 29).

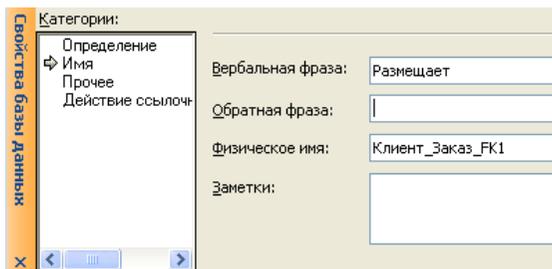


Рисунок 29 – Имя созданной связи

9. Перейдите на категорию *Прочее* окна *Свойства базы данных* и укажите тип отношения *идентифицирующее* и мощность связи *0 или более* (рисунок 30).

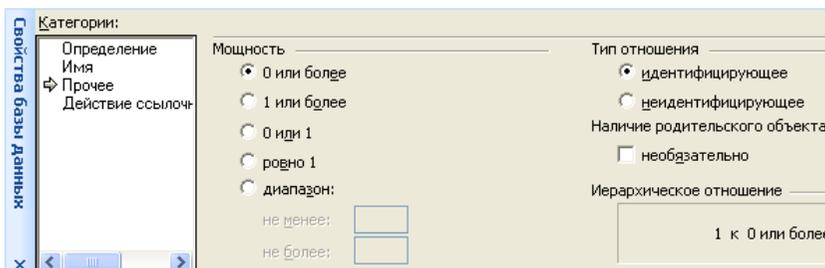


Рисунок 30 – Мощность и тип отношения логической связи *Клиент – Заказ*

В результате проделанных операций получим требуемую ER-диаграмму в нотации IDEF1X (рисунок 31).



Рисунок 31 – ER-диаграмма в нотации IDEF1X

Предположим далее, что в результате дальнейшей работы над проектируемой системой выявлены атрибуты, их типы и размеры, а также первичные ключи каждой сущности системы (таблица).

Сущности и их атрибуты

Сущности	Атрибуты	Тип и размер поля
Клиент	Код клиента (первичный ключ)	Char(5)
	ФИО клиента	Char(6)
	Телефон	Char(13)
Заказ	Код заказа (первичный ключ)	Char(5)
	Конфигурация компьютера	Char(50)
	Время принятия заказа	DateTime
	Время передачи заказа на сборку	DateTime

Окончание таблицы

Сущности	Атрибуты	Тип и размер поля
	Время передачи заказа на склад	DateTime
	Время выдачи заказа клиенту	DateTime

Для внесения этой информации в модель данных создаваемой системы выполните следующее:

1. Щелкните правой кнопкой мыши по сущности *Клиент* и в открывшемся меню выберите пункт *Свойства базы данных*. Перейдите в категорию *Столбцы* и добавьте названия атрибутов, выберите соответствующие типы и размеры согласно приведенной выше таблице, поставьте флажки в столбце *Обязательное* для всех атрибутов, а также укажите ключевое поле данной сущности, поставив флажок в столбце *PK*. Затем установите радиоточку в положение *Показывать: Физический тип данных* (рисунок 32).

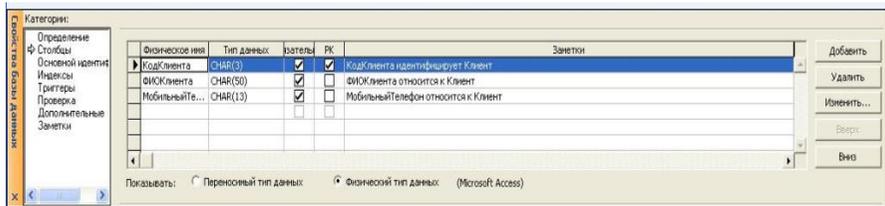


Рисунок 32 – Атрибуты сущности *Клиент*

2. Аналогичным образом введите атрибуты сущности *Заказ* и получите диаграмму модели данных, основанной на ключах для создаваемой информационной системы (рисунок 33).

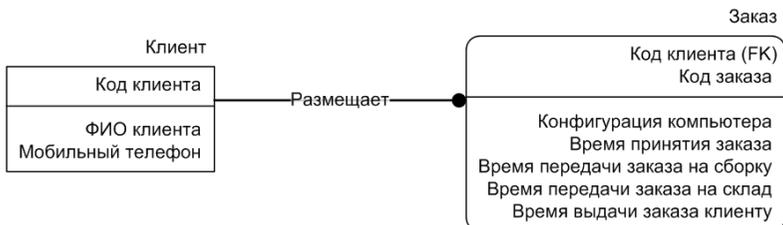


Рисунок 33 – **ER-диаграмма модели данных системы построения на основе стандарта IDEF1X**

3. Задание

Постройте в среде Microsoft Visio ER-диаграмму автоматизированной системы отслеживания прохождения заказов в ООО «Компьютер» в нотации «птичья лапка» (рисунок 34).



Рисунок 34 – ER-диаграмма в нотации «птичья лапка»

Лабораторная работа 4 ДИАГРАММЫ ПОТОКОВ ДАННЫХ

Цели работы:

- изучение основных принципов построения диаграмм потоков данных;
- изучение методов создания диаграмм потоков данных в среде Microsoft Visio.

1. Краткие сведения из теории

Суть диаграмм потоков данных (DFD) состоит в том, что они показывают, какие данные циркулируют внутри исследуемой системы, а также какие процессы их обрабатывают. По существу, диаграммы потоков данных – это модели функционирования информационных систем.

Для построения диаграмм потоков данных используются четыре базовых элемента:

- внешние сущности;
- процессы;
- потоки данных;
- хранилища данных.

Внешние сущности являются источниками или приемниками информации, поступающей на вход или снимаемой с выхода исследуемой системы. На диаграммах DFD внешние сущности изображаются квадратами (рисунок 35) и обычно размещаются у их краев.



Рисунок 35 – Пример обозначение внешней сущности на диаграммах потоков данных

Внешняя сущность может быть любым объектом за пределами исследуемой системы (организации целиком, отделы организаций, люди). Одна внешняя сущность может одновременно передавать информацию на вход исследуемой системы и принимать информацию с выходов исследуемой системы. Одна внешняя сущность может повторяться на одной и той же диаграмме несколько раз. Этот прием полезно применять для сокращения количества линий, соединяющих объекты на диаграмме.

На диаграммах DFD процесс преобразует информацию, поступающую на его вход в информацию, выдаваемую на его выход. Процессы изображаются в виде прямоугольников с закругленными углами, например, как показано на рисунке 36.

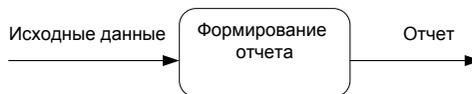


Рисунок 36 – Пример обозначения процесса и потоков данных на диаграммах потоков данных

Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, *Сформировать отчет*) или отглагольное существительное (например, *Формирование отчета*).

Потоки данных внутри исследуемой системы представляются в виде стрелок с соответствующими названиями (см. рисунок 36).

Стрелки могут входить и выходить из процесса в любой части. При построении DFD-диаграмм также используются двунаправлен-

ные стрелки, которые нужны для отображения взаимодействия между блоками, например диалога типа «приказ – результат выполнения».

В хранилище данных можно временно поместить информацию, которая перемещается между процессами. При этом способ ее помещения и извлечения не уточняется (рисунок 37).



Рисунок 37 – Пример обозначения хранилища данных на диаграммах потоков данных

Построение DFD-диаграмм исследуемой системы, также как и IDEF0-диаграмм, всегда начинается с построения контекстной диаграммы.

Контекстная DFD-диаграмма обычно состоит из одного функционального блока со стрелками, соединенными с внешними сущностями. Функциональный блок на этой диаграмме обычно имеет имя, совпадающее с именем исследуемой системы.

Пример контекстной диаграммы системы приведен на рисунке 38.



Рисунок 38 – Пример контекстной DFD-диаграммы

Контекстная диаграмма четко фиксирует исследуемую систему вместе с информационными потоками, связывающими ее с внешним миром. Она идентифицирует источники или приемники информации из исследуемой системы (внешние сущности), а также единственный процесс, отражающий главную цель или природу исследуемой системы насколько это возможно. И хотя контекстная диаграмма выглядит тривиальной, несомненная ее полезность заключается в том, что она устанавливает границы анализируемой системы.

Далее процесс декомпозиции процесса, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации. Получившаяся диаграмма отображает основные процессы создаваемой и называется «дочерней» по отношению к контекстной диаграмме. Аналогично каждый из процессов дочерней диаграммы может быть детализирован далее. Такая процедура последовательной декомпозиции производится до тех пор, пока не будет получен комплект диаграмм, удовлетворяющих исследователя.

Для примера на рисунке 39 представлена возможная декомпозиция контекстной DFD-диаграммы, приведенной на рисунке 38.

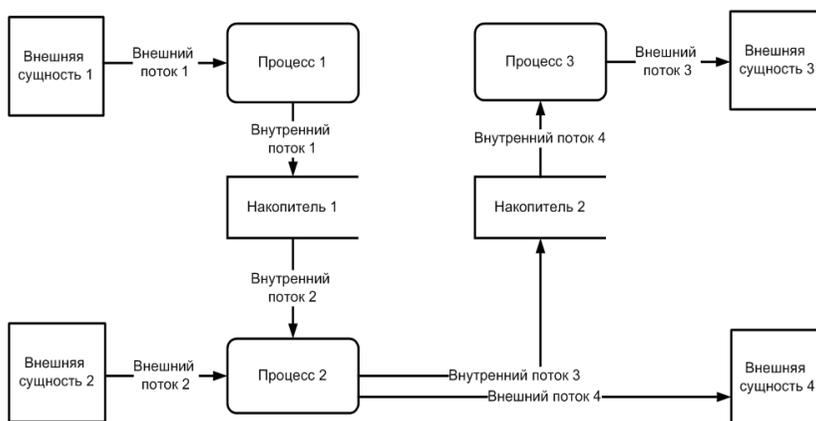


Рисунок 39 – Пример декомпозиции контекстной DFD-диаграммы

2. Пример

Построим диаграммы потоков данных для рассматриваемой в практикуме задачи создания автоматизированной системы контроля времени исполнения заказов в ООО «Компьютер».

Исходя из анализа требований, предъявленных к проектируемой системе, можно сделать вывод о том, что она должна взаимодействовать с двумя внешними сущностями, а именно с менеджером по продажам и директором организации. Проанализировав входящие и исходящие потоки данных создаваемой системы, изобразим соответствующую контекстную диаграмму (рисунок 40).

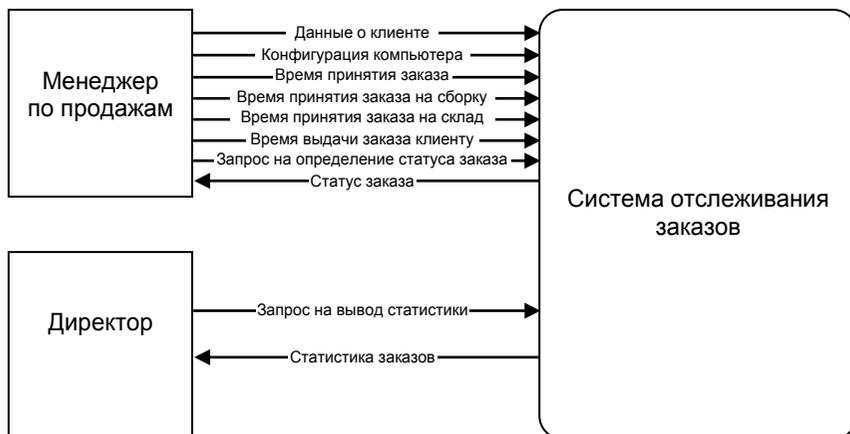


Рисунок 40 – Контекстная диаграмма автоматизированной системы контроля времени выполнения заказов в ООО «Компьютер»

Построим в среде Microsoft Visio эту диаграмму. Для этого выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора категории шаблона выберите *Программное обеспечение и базы данных* и в ней – элемент *Схема модели потоков данных*. Нажмите кнопку *Создать* в правой части экрана (рисунок 41).

2. Перетащите элемент *Интерфейс*  на рабочее поле. С помощью двойного щелчка левой кнопкой мыши по данному элементу перейдите в режим редактирования и задайте имя *Менеджер по продажам*.

3. Выполните аналогичные действия для создания внешней сущности *Директор*.

4. Перетащите элемент *Процесс*  на рабочее поле. С помощью двойного щелчка левой кнопки мыши перейдите в режим редактирования и задайте ему имя (см. рисунок 40).

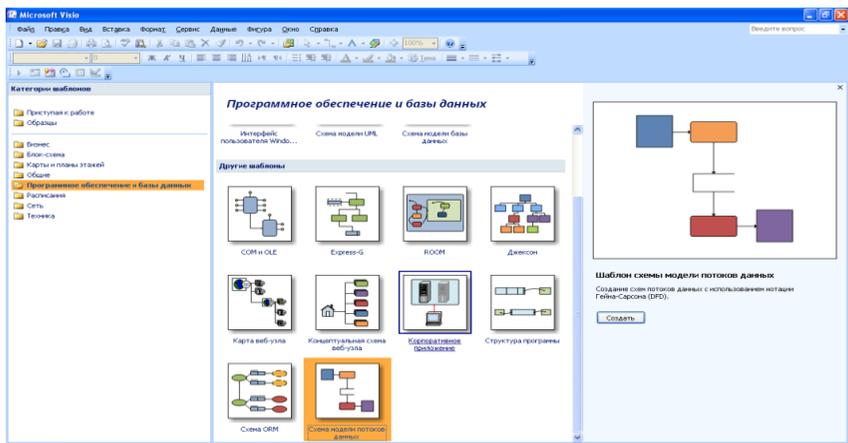


Рисунок 41 – Диалоговое окно выбора требуемой блок-схемы

5. Добавьте на диаграмму с помощью элемента *Поток данных*



необходимые потоки данных. Двойным нажатием левой кнопкой мыши по данным элементам задайте соответствующие имена: *Данные о клиенте, Конфигурация компьютера, Время принятия заказа, Время передачи заказа на сборку, Время передачи заказа на склад, Время выдачи заказа клиенту, Запрос на определение статуса заказа, Статус заказа, Запрос на вывод статистики, Статистика заказов*. В результате проделанных операций получите окончательный вид контекстной диаграммы проектируемой системы (см. рисунок 40).

Контекстная диаграмма не показывает деталей внутренней структуры проектируемой системы, поэтому она нуждается в дальнейшей декомпозиции. Проанализировав события, на которое должна реагировать создаваемая система, можно предложить следующую декомпозицию контекстной диаграммы автоматизированной информационной системы контроля времени исполнения заказов в ООО «Компьютер» (рисунок 42).

Для построения декомпозиции контекстной диаграммы в среде Microsoft Visio выполните следующие действия.

1. Выберите *Вставка/Страница*. В диалоговом окне *Параметры страницы* укажите свойства новой страницы в соответствии с рисунком 43.

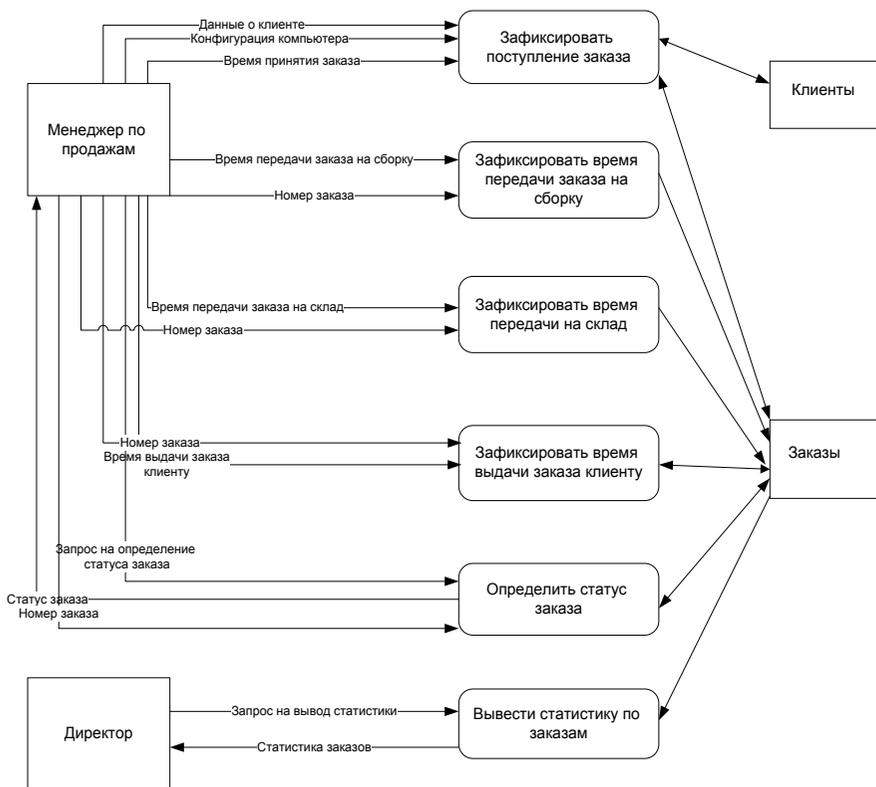


Рисунок 42 – Декомпозированная контекстная диаграмма автоматизированной системы контроля времени выполнения заказов в ООО «Компьютер»

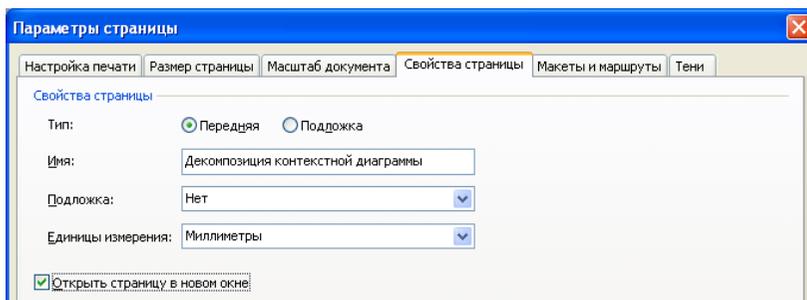


Рисунок 43 – Диалоговое окно *Параметры страницы*

2. С помощью элемента *Процесс*  добавьте на рабочее поле окна процессы *Зафиксировать поступление заказа*, *Зафиксировать время передачи заказа на сборку*, *Зафиксировать время передачи заказа на склад*, *Зафиксировать время выдачи заказа клиенту*, *Определить статус заказа*, *Вывести статистику по заказам*.

3. С помощью элемента *Хранилище данных*  добавьте на рабочее поле накопители и задайте им имена *Клиенты* и *Заказы*.

4. Соедините все добавленные объекты, используя элемент *Поток данных* , как показано на рисунке 42.

5. Для создания двунаправленных стрелок кликните правой кнопкой мыши на нужной стрелке и выберите *Формат/Линия*. В открывшемся окне установите концы линий, как показано на рисунке 44.

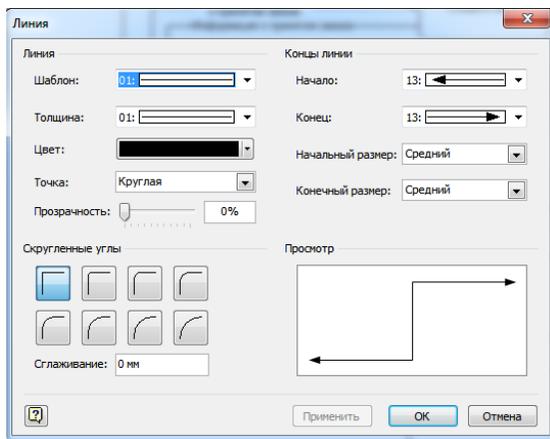


Рисунок 44 – Окно форматирования линии

В результате проделанных операций получим окончательный вид контекстной диаграммы проектируемой информационной системы.

3. Задание

Постройте в среде Microsoft Visio дальнейшую декомпозицию процесса *Зафиксировать поступление заказа*, которая должна иметь вид, представленный на рисунке 45.

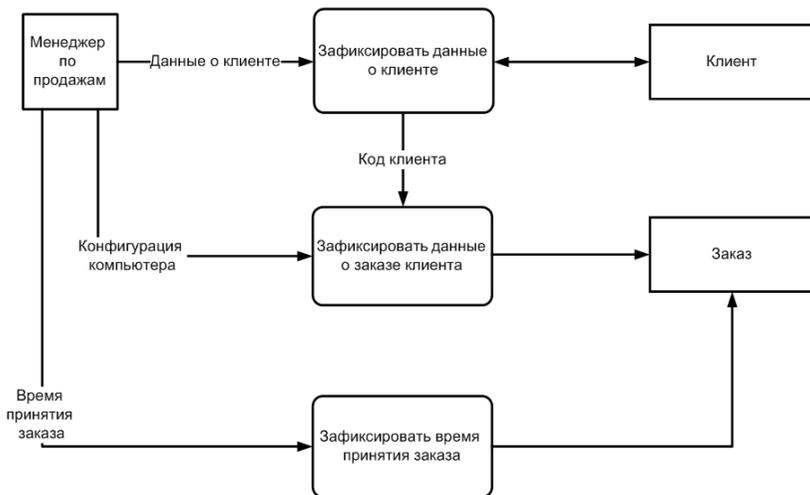


Рисунок 45 – Декомпозиция процесса *Зафиксировать поступление заказа*

Лабораторная работа 5 ДИАГРАММЫ КЛАССОВ

Цели работы:

- изучение основных принципов построения диаграмм классов;
- изучение методов создания диаграмм классов в среде Microsoft Office Visio.

1. Краткие сведения из теории

Диаграммы классов являются центральным звеном методологии объектно-ориентированного анализа и проектирования информационных систем. Диаграммы классов содержат информацию об объектах системы и статических связях между ними, отражают декларативные знания о предметной области, оперируют понятиями класса, объекта, отношения, тем самым представляя логический аспект проекта.

Диаграммы классов служат для следующих целей:

- моделирования данных (анализ предметной области позволяет выявить основные характерные для нее сущности и связи между ними);

- представления архитектуры проектируемой системы (можно выделить архитектурно значимые классы и показать их на диаграммах, описывающих архитектуру);

- моделирования навигации экранов (на таких диаграммах показываются пограничные классы и их логическая взаимосвязь).

В зависимости от степени детализации используется два вида диаграмм классов:

- Диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов.

- Диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Классом называется именованное описание совокупности объектов с общими атрибутами, операциями, связями. Графически класс изображается в виде прямоугольника. У каждого класса должно быть имя, уникально отличающее его от всех других классов. При формировании имен классов допускается использование произвольной комбинации букв, цифр и знаков препинания. На практике рекомендуется использовать в качестве имен классов короткие и осмысленные существительные, каждое из которых начинается с заглавной буквы. Условным обозначением класса на диаграммах служит прямоугольник, разбитый на три секции: имена классов, атрибуты классов, операции классов (рисунок 46).

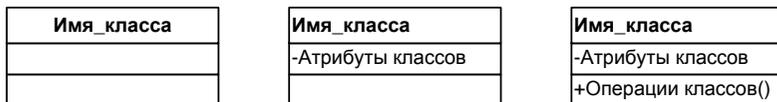


Рисунок 46 – Примеры изображений классов

Секция имени не может быть опущена. Прочие секции могут быть пустыми или отсутствовать вовсе.

Атрибутом класса называется именованное свойство класса, характеризующее множество значений, которые могут принимать экземпляры этого класса. Класс может иметь любое число атрибутов (или не иметь ни одного атрибута). Свойство, выражаемое атрибутом, является свойством моделируемой сущности, общим для всех объек-

тов данного класса. Любой атрибут любого объекта класса должен иметь некоторое значение.

Имена атрибутов представляются в разделе класса, расположенном под именем класса. На имена атрибутов (имя атрибута может быть произвольной текстовой строкой) накладываются ограничения, однако рекомендуется использовать короткие прилагательные и существительные, отражающие смысл соответствующего свойства класса. Первое слово в имени атрибута рекомендуется писать с прописной буквы, а все остальные слова – с заглавной. Пример описания класса с указанными атрибутами показан на рисунке 47.

Человек
-Пол
-Дата рождения
-Фамилия
-Имя
-Отчество
-Место рождения

Рисунок 47 – Атрибуты класса *Человек*

Каждый атрибут может иметь признак видимости:

- общий (public) – атрибут доступен для всех клиентов класса, на диаграммах помечается символом «+»;
- защищенный (protected) – атрибут доступен только для подклассов и друзей класса;
- секретный (private) – атрибут доступен только для друзей класса.

Операцией класса называется некоторая услуга, которую можно запросить у любого объекта этого класса. Операция – это абстракция того, что можно делать с объектом. Класс может содержать любое число операций, в частности не содержать ни одной операции. Набор операций класса является общим для всех объектов данного класса.

Операции класса описываются в разделе, расположенном ниже раздела с атрибутами. Для именованной операции рекомендуется использовать глагольные формы, соответствующие ожидаемому поведению объектов данного класса. Описание операции может также содержать имена и типы всех параметров, а если операция является функцией, то и тип ее значения, как класс *Человек* с определенными

операциями. На рисунке 48 приведен пример операций применительно ко всем классам сущности *Человек*.

Человек
-Пол
-Дата рождения
-Фамилия
-Имя
-Отчество
-Место рождения
+Выдать возраст()

Рисунок 48 – Операции класса *Человек*

Ассоциацией называется структурная связь, показывающая, что объекты одного класса некоторым образом связаны с объектами другого класса. В ассоциации могут связываться два класса, и тогда она называется бинарной. Допускается создание ассоциаций, связывающих сразу n классов (они называются n -арными ассоциациями). Графически ассоциация изображается в виде линии, соединяющей класс сам с собой или с другими классами.

Ассоциации может быть присвоено имя, характеризующее природу связи. Имя связи располагается над линией связи. Например, из рисунка 49 следует, что *Студент* учится в *Университете*.



Рисунок 49 – Пример ассоциации

Обобщением называется связь между общей сущностью, называемой суперклассом (или родителем), и более специализированной разновидностью этой сущности, называемой подклассом (или потомком). Класс-потомок наследует все атрибуты и операции класса-предка, но в нем могут быть определены дополнительные атрибуты и операции.

Объекты класса-потомка могут использоваться везде, где могут использоваться объекты класса-предка. Это свойство называют полиморфизмом по включению, имея в виду, что объекты потомка можно

считать включаемыми во множество объектов класса-предка. Графически обобщения изображаются в виде сплошной линии с большой незакрашенной стрелкой, направленной к суперклассу. На рисунке 50 показан пример иерархии обобщения: у каждого подкласса имеется только один суперкласс.

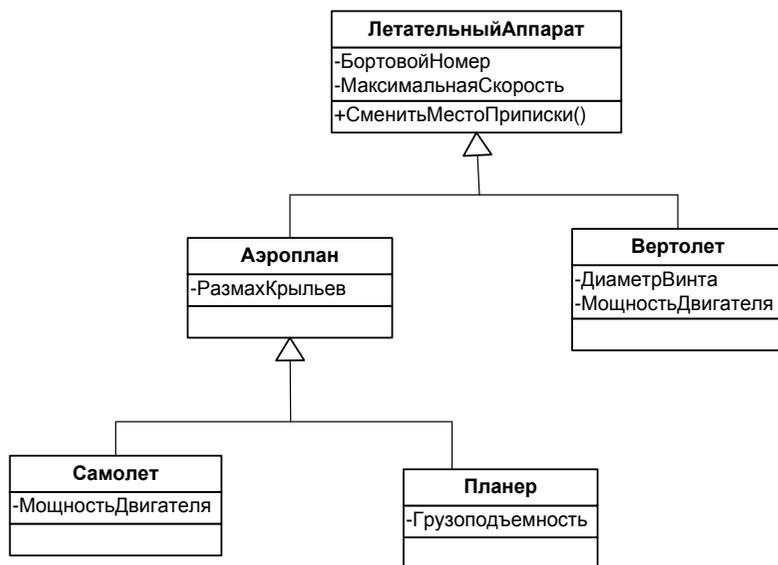


Рисунок 50 – Иерархия одиночного наследования классов

2. Пример

Исходя из анализа требований, предъявляемых к рассматриваемой в практикуме автоматизированной системе контроля времени выполнения заказов в ООО «Компьютер», можно сделать вывод о том, что она должна содержать информацию о клиентах и их заказах. Это определяет классы, которые должны иметься в создаваемой системе, – классы *Клиент* и *Заказ*. Предполагая, что один клиент может иметь несколько заказов, сделаем вывод о том, что связь между этими классами относится к типу «один-ко-многим». Принимая во внимание ранее выявленные указанные классы (см. лабораторную работу 3),

можно изобразить диаграмму классов проектируемой системы (рисунок 51).



Рисунок 51 – Диаграмма классов проектируемой системы

Для построения данной схемы в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней – элемент *Схема модели UML*. Нажмите кнопку *Создать* (рисунок 52).

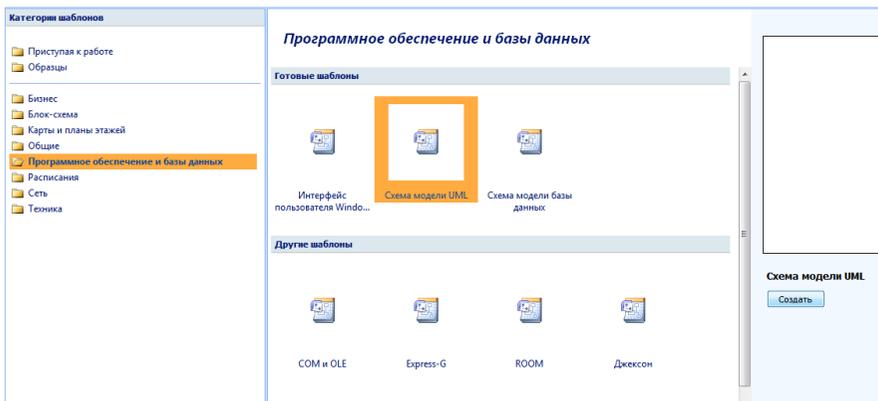


Рисунок 52 – Выбор шаблона

2. С помощью элемента **Класс**  пакета *Статическая структура UML* поместите на страницу два объекта. Двойным нажатием левой кнопкой мыши по элементу откройте *Свойства класса UML*, где в поле *Имя* введите *Клиент* и *Заказ* соответственно. На вкладке *Атрибуты* добавьте:

- для класса *Клиент*: *Код клиента*, *ФИО*, *Телефон*;

- для класса *Заказ*: *Код заказа*, *Конфигурация компьютера*, *Время принятия заказа*, *Время передачи заказа на сборку*, *Время передачи заказа на склад*, *Время передачи клиенту* (рисунок 53).

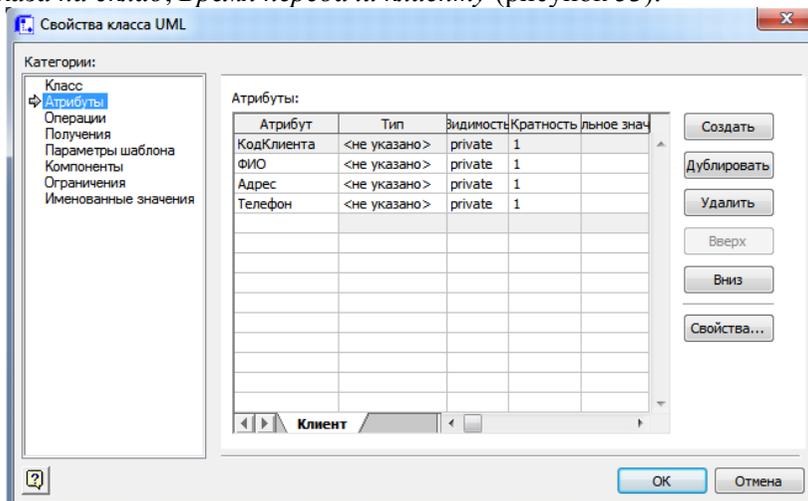


Рисунок 53 – Окно свойств класса UML

3. С помощью элемента *Двуместная ассоциация*  пакета *Статическая структура UML* соедините ранее созданные два класса. Двойным нажатием левой кнопкой мыши по элементу вызовите *Свойства ассоциации UML* и задайте окончания ассоциаций, как показано на рисунке 54.

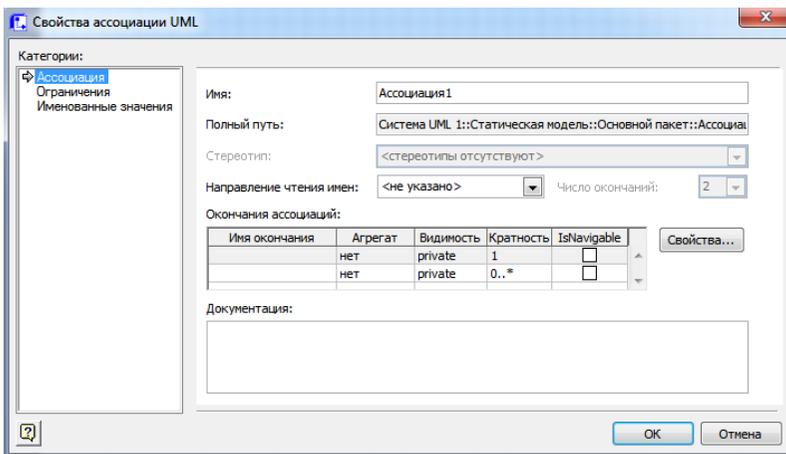


Рисунок 54 – Окно свойств ассоциации UML

3. Задание

Дополните диаграмму классов проектируемой системы (см. рисунок 51) необходимыми операциями, как показано на рисунке 55.



Рисунок 55 – Диаграмма классов

Лабораторная работа 6 ДИАГРАММЫ ВЗАИМОДЕЙСТВИЯ

Цели работы:

- изучение основных принципов построения диаграмм взаимодействия;
- изучение методов создания диаграмм взаимодействия в среде Microsoft Office Visio.

1. Краткие сведения из теории

Диаграммы взаимодействия (interaction diagrams) являются моделями, описывающими поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображаются ряд объектов и те сообщения, которыми они обмениваются между собой.

Проиллюстрируем данный подход на примере достаточно простого варианта использования, который описывает следующее поведение:

- *Окно Ввода Заказа* посылает *Заказу* сообщение «приготовиться».
- *Заказ* посылает данное сообщение каждой *Строке заказа* в данном *Заказе*.
- Каждая *Строка заказа* проверяет состояние определенного *Запаса товара*.

Если данная проверка удовлетворяется (результат – true), то *Строка заказа* удаляет соответствующее количество товара из *Запаса*. В противном случае количество *Запаса* снижается до уровня повторного заказа, и *Запас* запрашивает новую поставку товара.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams).

На диаграмме последовательности объект изображается в виде прямоугольника на вершине пунктирной вертикальной линии (рисунок 56). Эта вертикальная линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице – сверху вниз. Каждое сообщение помечается, как минимум, именем сообщения; при желании можно добавить аргументы, некоторую управляющую информацию и показать самоделегирование (self-delegation) – сообщение, которое объект посылает

самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

Из всей возможной управляющей информации два ее вида имеют существенное значение. Во-первых, это условие, показывающее, когда посылается сообщение, например: [*нуженПовторныйЗаказ = «true»*]. Сообщение посылается только при выполнении данного условия. Другой полезный управляющий маркер – это маркер итерации, показывающий, что сообщение посылается много раз для множества объектов-адресатов (например, * *приготовиться*).

Диаграммы последовательности очень просты и наглядны (в этом заключается самое большое их достоинство), они существенно помогают разобраться в процессе поведения системы.

Диаграмма содержит возврат, означающий не новое сообщение, а возврат из сообщения (рисунок 56). На диаграмме возврат отличается от обычных сообщений тем, что его стрелка не сплошная, а имеет вид пары линий.

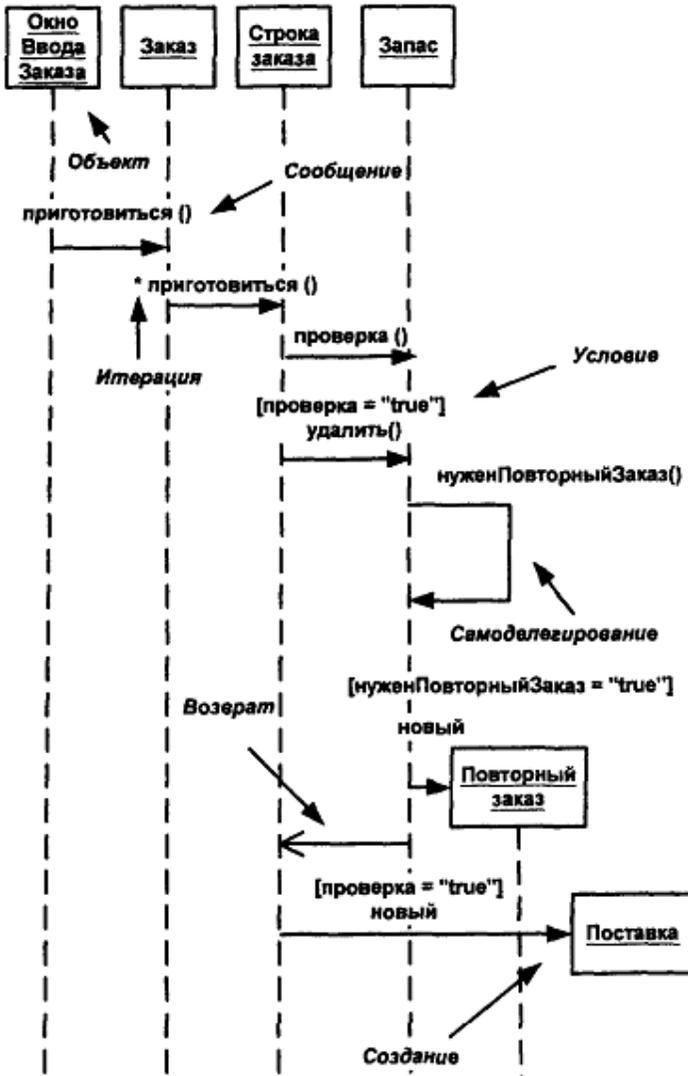


Рисунок 56 – Диаграмма последовательности

Вторым видом диаграммы взаимодействия является кооперативная диаграмма, на которой экземпляры объектов показаны в виде пиктограмм. Как и на диаграмме последовательности, здесь стрелки обозначают сообщения, обмен которыми осуществляется в рамках

данного варианта использования, однако их временная последовательность указывается путем нумерации сообщений.

Нумерация сообщений делает восприятие их последовательности более трудным, чем в случае расположения линий на странице сверху вниз. С другой стороны, такое пространственное расположение позволяет более легко отразить некоторые другие моменты. Например, можно показать взаимосвязь объектов, перекрывающиеся компоненты или другую информацию.

Для кооперативных диаграмм можно использовать один из нескольких вариантов нумерации. Часто применяется десятичная схема нумерации (рисунок 57), поскольку в этом случае понятно, какая операция вызывает какую операцию.

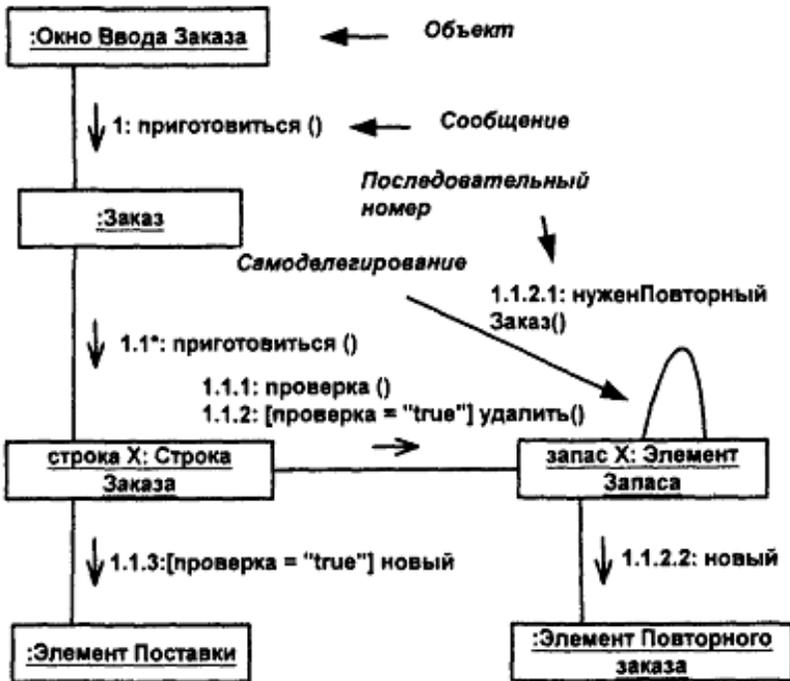


Рисунок 57 – Кооперативная диаграмма с десятичной нумерацией

Независимо от используемой схемы нумерации на диаграмме можно разместить такого же рода управляющую информацию, как и на диаграмме последовательности.

На рисунке 57 можно увидеть различные формы схемы именования объектов на контекстной диаграмме. Общая форма имеет вид *<ИмяОбъекта: ИмяКласса>*, где либо имя объекта, либо имя класса может отсутствовать. При отсутствии имени объекта остается двоеточие, чтобы было понятно, что это имя класса, а не объекта.

У разных разработчиков имеются различные предпочтения относительно диаграмм взаимодействия. В диаграмме последовательности делается акцент именно на последовательность сообщений: легче наблюдать порядок, в котором происходят различные события. На кооперативной диаграмме можно использовать пространственное расположение объектов для того, чтобы показать их статическое взаимодействие.

Одним из принципиальных свойств любой формы диаграммы взаимодействия является их простота. Посмотрев на диаграмму, можно легко увидеть все сообщения.

Диаграммы взаимодействия следует использовать, когда нужно описать поведение нескольких объектов в рамках одного варианта использования.

2. Пример

Диаграмма последовательности для варианта использования *Зафиксировать поступление заказа* имеет вид, представленный на рисунке 58.

Для построения данной схемы в среде Microsoft Visio выполните следующие действия:

1. Добавьте новую страницу с помощью контекстного меню, вызываемого щелчком правой кнопкой мыши по текущему названию страницы, и выбора пункта *Добавить страницу...* (рисунок 59).

2. С помощью элемента *Актер*  пакета *Сценарий выполнения UML* (Файл – Фигуры – Программное обеспечение и базы данных – Программное обеспечение – Сценарий выполнения UML) поместите на страницу объект, как показано на рисунке 58. Двойным нажатием левой кнопкой мыши по элементу открываем *Свойства актера UML*, где в поле *Имя* нужно вести *Менеджер по продажам* (рисунок 60).

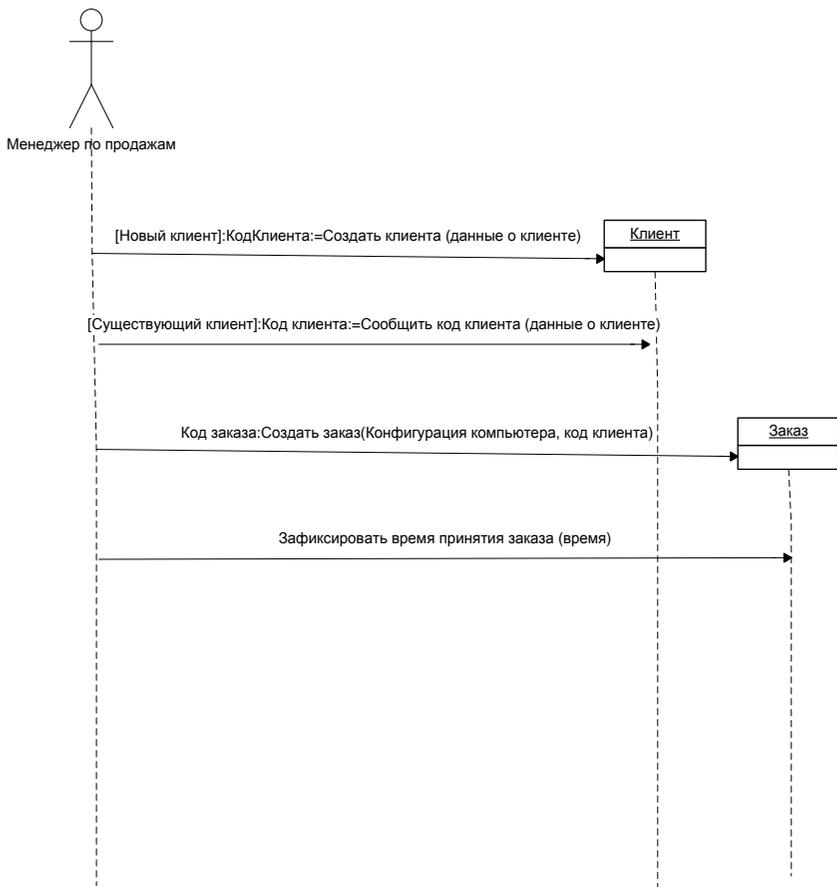


Рисунок 58 – Диаграмма последовательности для варианта использования *Зафиксировать поступление заказа*

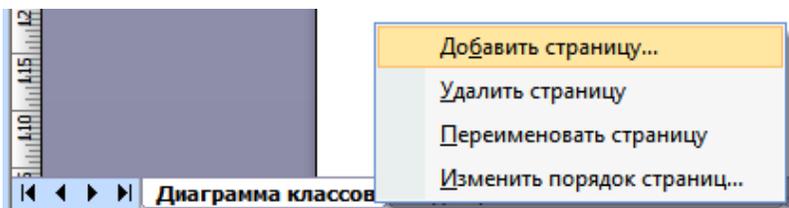


Рисунок 59 – Добавление страницы

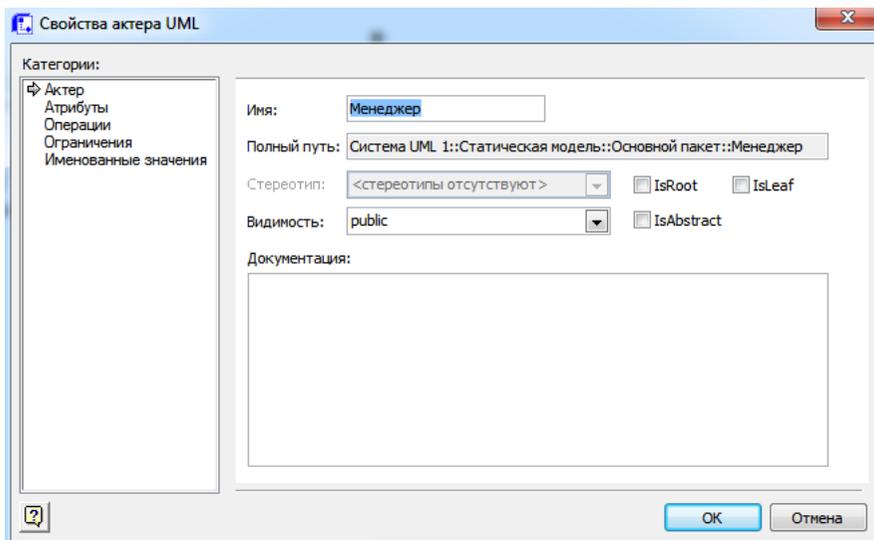


Рисунок 60 – Окно свойств актера UML

3. С помощью элемента *Линия жизни*  пакета *Последовательности UML* (Файл – Фигуры – Программное обеспечение и базы данных – Программное обеспечение – Последовательности UML) добавьте объекты на страницу, а затем присоедините к каждому из актеров: *Клиент* и *Менеджер* соответственно.

4. С помощью элемента *Линия жизни объекта*  пакета *Последовательности UML* поместите на страницу два объекта, как показано на рисунке 58. Двойным нажатием левой кнопкой мыши по одному элементу вызовите *Свойства роли классификатора UML* и задайте ему имя *Клиенты* (рисунок 61). Второй элемент назовите *Заказы*.

5. С помощью элемента *Объект*  пакета *Сценарий выполнения UML* поместите на страницу объект, как показано на рисунке 58. Двойным нажатием левой кнопкой мыши по элементу откройте *Свойства объекта UML*, где в поле *Имя* введите *Клиент* (рисунок 62).

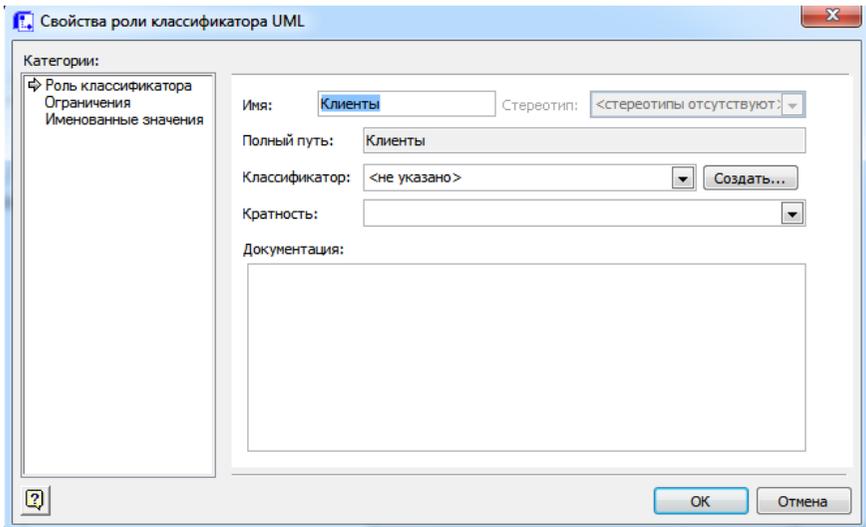


Рисунок 61 – Окно свойств роли классификатора UML

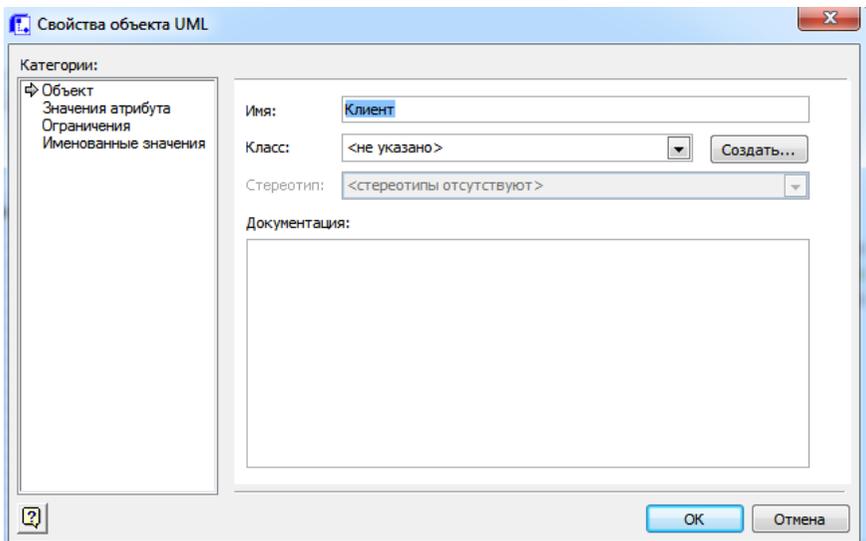


Рисунок 62 – Окно свойств объекта UML

6. С помощью элемента *Сообщение*  пакета *Последовательности UML* поместите на страницу четыре объекта, соедините линии жизни, как показано на рисунке 58. Двойным нажатием левой кнопкой мыши по элементу вызовите *Свойства сообщения UML* и задайте имена: *Оформить заказ*, [*Новый клиент*]: *Создать клиента*, [*Существующий клиент*]: *Сообщить код*, *Создать заказ* (рисунок 63).

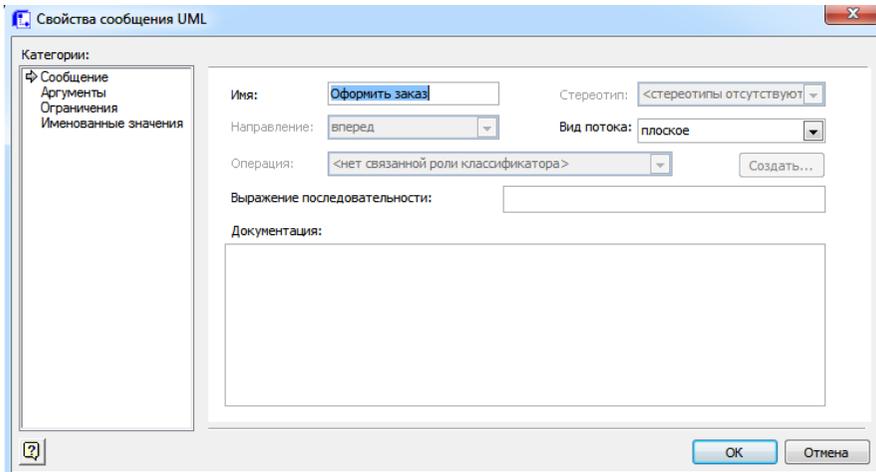


Рисунок 63 – Окно свойств сообщения UML

7. С помощью элемента *Текст*  на палитре инструментов *Стандартная* добавим комментарии для каждого сообщения из нашей схемы, как показано на рисунке 58.

Диаграмма взаимодействия для варианта использования *Зафиксировать поступление заказа* имеет вид, представленный на рисунке 64.

Для построения данной схемы в среде Microsoft Visio выполните следующие действия:

1. Добавьте новую страницу с помощью контекстного меню, вызываемого щелчком правой кнопкой мыши по текущему названию страницы, и выбора пункта *Добавить страницу* (рисунок 65).

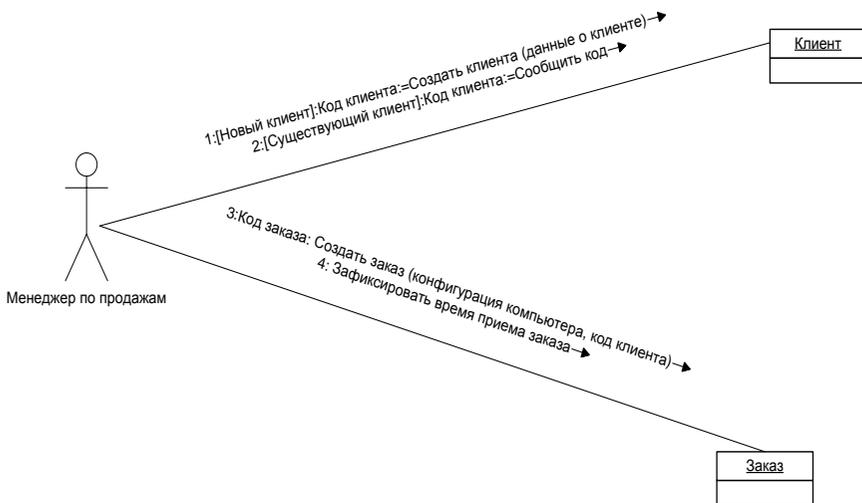


Рисунок 64 – Диаграмма взаимодействия для варианта использования **Зафиксировать поступление заказа**

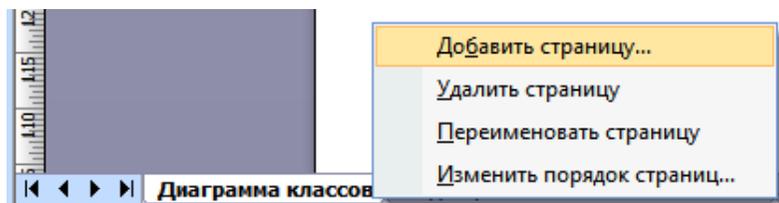


Рисунок 65 – Добавление страницы

2. С помощью элемента **Актер**  пакета **Сценарий выполнения UML** (Файл – Фигуры – Программное обеспечение и базы данных – Программное обеспечение – Сценарий выполнения UML) поместите на страницу два объекта, как показано на рисунке 64. Двойным нажатием левой кнопкой мыши по одному элементу откройте **Свойства актера UML**, где в поле **Имя** введите **Клиент**. Второй элемент назовите **Менеджер** (рисунок 66).

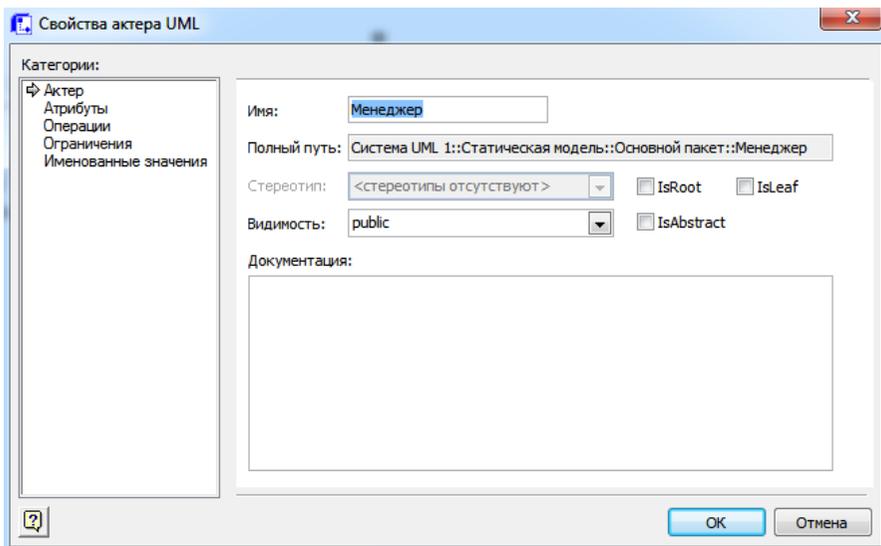


Рисунок 66 – Окно свойств актера UML

3. С помощью элемента *роль классификатора*  пакета *Взаимодействия UML* (Файл – Фигуры – Программное обеспечение и базы данных – Программное обеспечение – Взаимодействия UML) добавьте два объекта на страницу как показано на рисунке 64. Двойным нажатием левой кнопкой мыши по элементу вызовите *Свойства роли классификатора UML* и задайте ему имя *Клиент*. Второй элемент назовите *Заказ* (рисунок 67).

4. С помощью элемента *Роль ассоциации*  пакета *Взаимодействия* добавьте четыре объекта на страницу, как показано на рисунке 64. Двойным нажатием левой кнопкой мыши по элементу вызовите *Свойства роли ассоциации UML*, перейдите в категорию *Сообщения* и задайте для каждой роли ассоциации свои сообщения: *Оформить заказ*, [*Новый клиент*]: *Создать клиента*, [*Существующий клиент*]: *Сообщить код*, *Создать заказ* (рисунок 68).

5. С помощью элемента *Текст*  на палитре инструментов *Стандартная* добавьте комментарии для каждой роли ассоциации нашей схемы, как показано на рисунке 68.

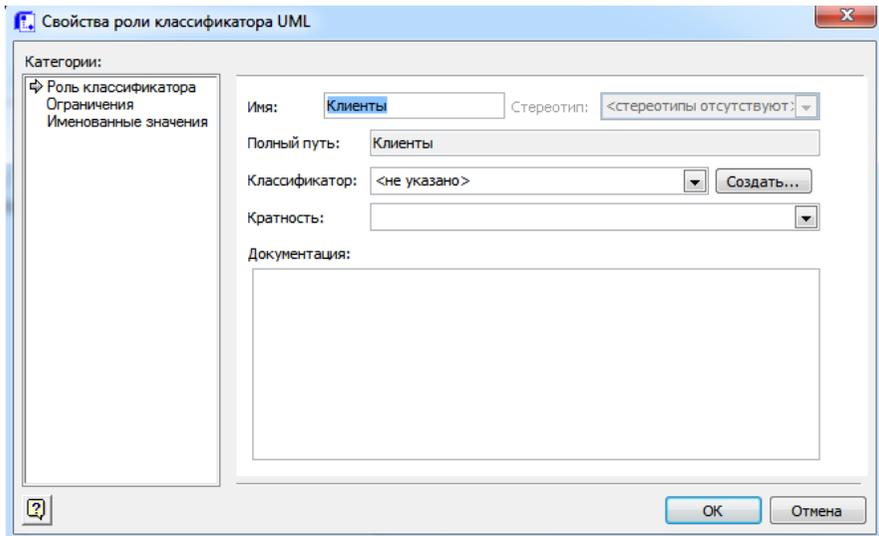


Рисунок 67 – Окно свойств роли классификатора UML

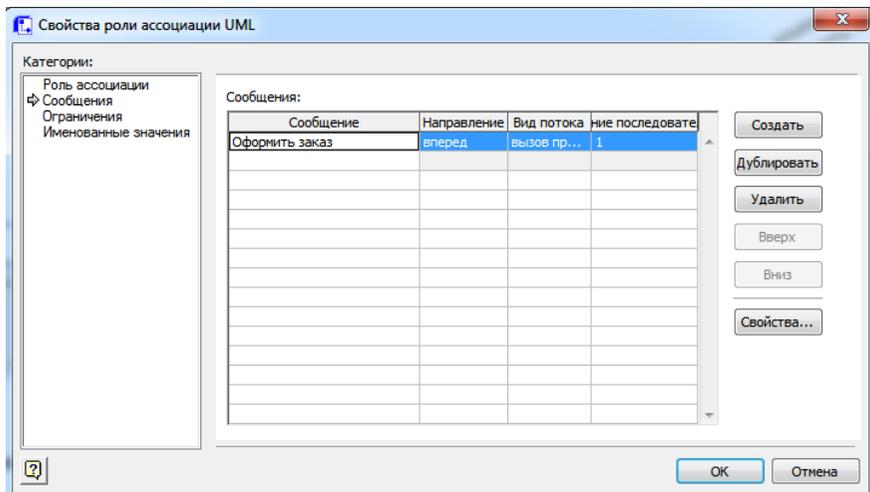


Рисунок 68 – Окно свойств роли ассоциации UML

3. Задание

Изобразите в среде Microsoft Visio диаграмму взаимодействия для варианта использования *Зафиксировать время передачи заказа на сборку*, как показано на рисунке 69.

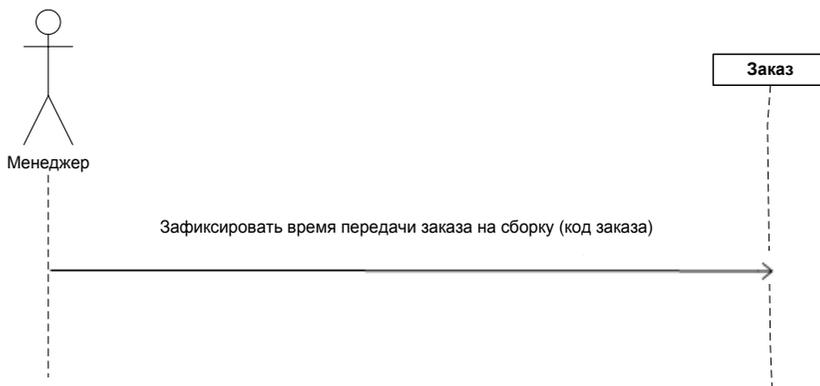


Рисунок 69 – Диаграмма последовательности для варианта использования *Зафиксировать время передачи заказа на сборку*

Лабораторная работа 7 ДИАГРАММЫ СОСТОЯНИЙ

Цели работы:

- изучение основных принципов построения диаграмм состояний;
- изучение методов создания диаграмм состояний в среде Microsoft Office Visio.

1. Краткие сведения из теории

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На рисунке 70 приведен пример диаграммы состояний для банковского счета. Из данной диаграммы видно, в каких состояниях может

существовать счет. Можно также видеть процесс перехода счета из одного состояния в другое. Например, если клиент требует закрыть открытый счет, он переходит в состояние *Закрыт*. Требование клиента называется событием (event), именно такие события и вызывают переход из одного состояния в другое. Если клиент снимает деньги с открытого счета, он может перейти в состояние *Превышение кредита*. Это происходит, только если баланс по этому счету меньше нуля, что отражено условием [*отрицательный баланс*] на диаграмме. Заключенное в квадратных скобках ограничивающее условие (guard condition) определяет, когда может или не может произойти переход из одного состояния в другое.

На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан.

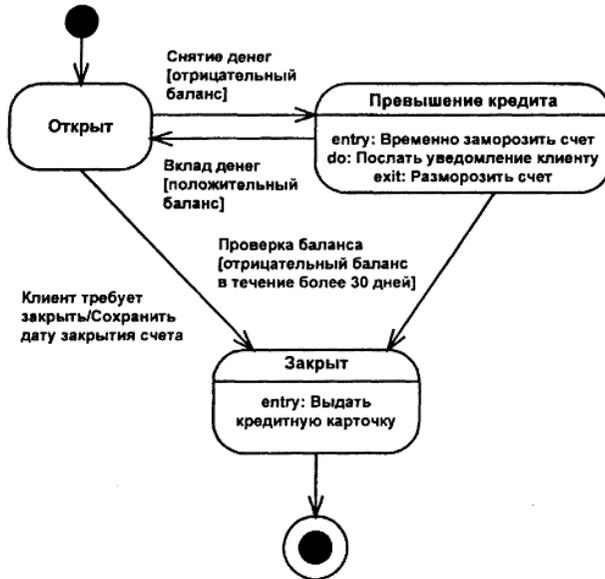


Рисунок 70 – Диаграмма состояний для банковского счета

Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть только одно

начальное состояние, а конечных состояний может быть столько, сколько нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. На рисунке 70 при превышении кредита клиенту посылается соответствующее сообщение. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions). С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Рассмотрим каждый из них в контексте диаграммы состояний для класса *Банковский счет*. Деятельность (activity) – это поведение, реализуемое объектом, пока он находится в данном состоянии. Например, когда счет находится в состоянии *Закрит*, происходит возврат кредитной карточки пользователю.

Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово *do* (выполнять) и двоеточие.

Входное действие (entry action) – это поведение, которое выполняется, когда объект переходит в данное состояние. Когда счет в банке переходит в состояние *Превышение кредита*, выполняется действие *Временно заморозить счет* независимо от того, откуда объект перешел в это состояние. Таким образом, данное действие осуществляется не после того, как объект перешел в это состояние, а как часть этого перехода. В отличие от деятельности входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово *entry* (вход) и двоеточие.

Выходное действие (exit action) подобно входному, однако оно осуществляется как составная часть процесса выхода из данного состояния. Так, при выходе объекта *Account* из состояния *Превышение кредита* независимо оттого, куда он переходит, выполняется действие *Разморозить счет*. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым. Выходное действие изображают внутри состояния, ему предшествует слово *exit* (выход) и двоеточие. Переходом (transition) называется перемещение объекта из одного состояния в другое. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся на последующем. Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные пе-

реходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.

У перехода существует несколько спецификаций, основными из которых являются события, ограждающие условия и действия. Событие (event) вызывает переход из одного состояния в другое. Событие *Клиент требует закрыть* вызывает переход счета из открытого в закрытое состояние. События размещают на диаграмме вдоль линии перехода. На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям. Ограничивающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В примере событие *Вклад денег* переведет счет из состояния *Превышение кредита* в состояние *Открыт*, но только если баланс будет больше нуля. В противном случае переход не осуществится. Ограничивающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки. Ограничивающие условия задавать необязательно. Однако, если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия, чтобы понять, какой путь перехода будет автоматически выбран. Действие может быть не только входным или выходным, но и частью перехода. Например, при переходе счета из открытого в закрытое состояние выполняется действие *Сохранить дату закрытия счета*.

Действие изображают вдоль линии перехода после имени события, ему предшествует косая черта. Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

2. Пример

Исходя из анализа имеющейся информации можно сделать вывод о том, что объект *Заказ* информационной системы контроля времени исполнения заказов в ООО «Компьютер» может находиться в следующих состояниях:

- На оформлении (ожидает передачу на сборку).
- На сборке.
- На складе.
- Выдан.

Эта информация позволяет изобразить диаграмму состояний объекта *Заказ* информационной системы контроля времени исполнения заказов в ООО «Компьютер» (рисунок 71).



Рисунок 71 – Диаграмма состояний объекта *Заказ* информационной системы контроля времени выполнения заказов в ООО «Компьютер»

Для построения этой диаграммы в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней – элемент *Схема модели UML* (рисунок 72).

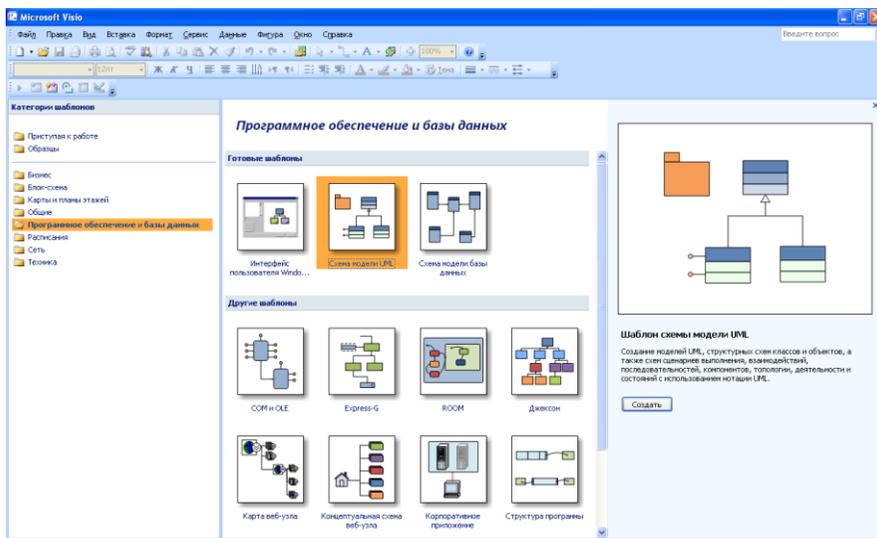


Рисунок 72 – Шаблон программного обеспечения и базы данных

2. Нажмите кнопку *Создать* в правой части экрана. Слева появится диалоговое окно Microsoft Office Visio *Фигуры схемы модели UML* (рисунок 73). Для создания диаграммы состояний нужно щелкнуть левой кнопкой мыши на пакете *Схема состояний UML* в левой части рабочего окна.

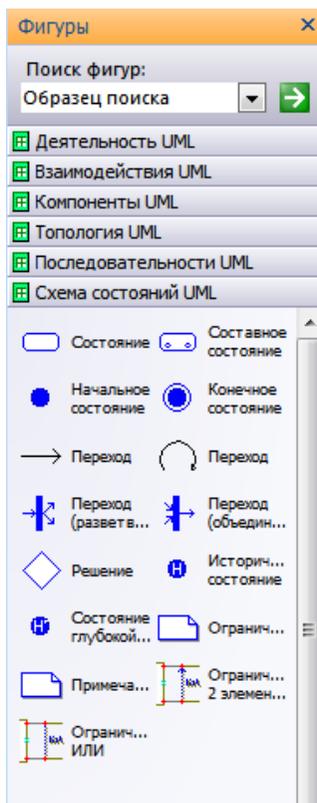


Рисунок 73 – Окно Microsoft Office Visio *Фигуры* схемы UML

3. На рабочую область поместите два элемента пакета *Схема состояний UML*: *Начальное действие*  и *Конечное действие* .

4. С помощью элемента *Состояние*  пакета *Схема состояний UML* поместите на рабочую область между элементами *Начальное состояние* и *Конечное состояние* следующие состояния: *Заказ на оформлении*, *Заказ на сборке*, *Заказ на складе*, *Заказ выдан*. Имена

задайте с помощью поля *Имя* окна *Свойства состояния UML*, которое вызывается двойным нажатием левой кнопкой мыши по состоянию либо через контекстное меню – *Свойства* (рисунок 74).

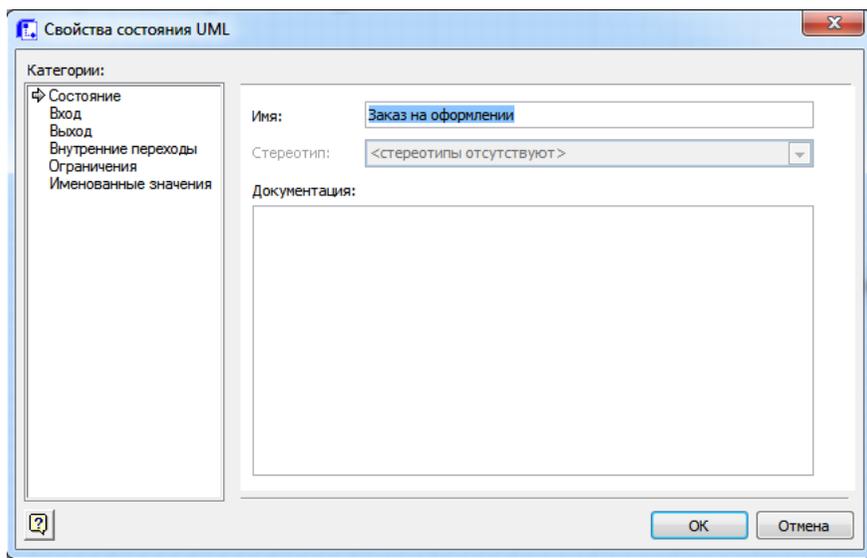


Рисунок 74 – Окно свойств состояния UML

5. С помощью элемента *Переход* \longrightarrow пакета *Схема состояний UML* поместите на диаграмму переход, затем соедините начало стрелки с начальным состоянием и конец стрелки с состоянием *Заказ на оформлении*, как показано на рисунке 75.

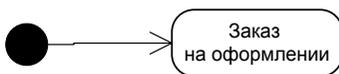


Рисунок 75 – Соединение состояний посредством элемента *Переход*

6. Откройте окно свойств перехода UML двойным нажатием левой кнопкой мыши по переходу либо через контекстное меню – *Свойства*. Нажмите кнопку *События...*, затем кнопку *Создать...* и выберите *Событие вызова* в окне выбора типов событий. Нажмите кнопку *OK* (рисунок 76).

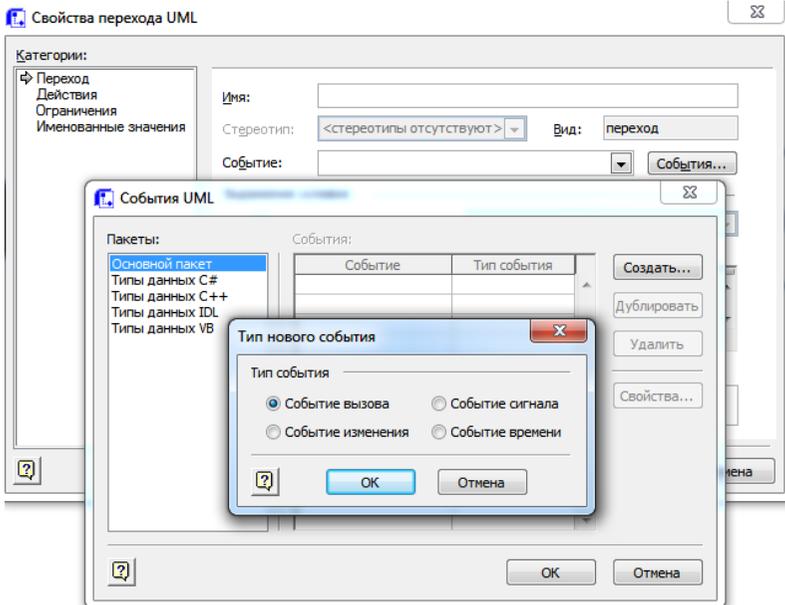


Рисунок 76 – Процесс выбора типа события для перехода

7. В появившемся окне *Свойства события* вызова UML задайте имя события *Создать заказ* в поле *Имя*, затем нажмите два раза кнопку *OK* и в последнем открытом окне *Свойства перехода UML* выберите соответствующее событие *Основной пакет: Создать заказ* из выпадающего списка по полю *Событие*. Нажмите кнопку *OK* (рисунок 77).

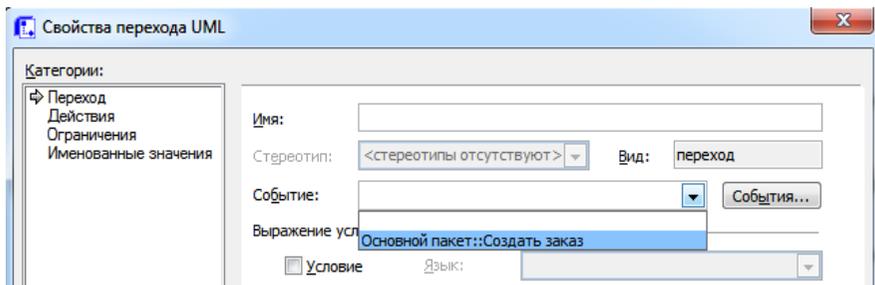


Рисунок 77 – Выбор соответствующего события в окне свойств перехода UML

8. Прodelайте пункты 5–7 для создания и соединения остальных переходов (*Зафиксировать время передачи на сборку, Зафиксировать время передачи на склад, Зафиксировать время передачи клиенту*) с состояниями, как показано на рисунке 71.

Лабораторная работа 8 ВЫБОР АРХИТЕКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Цели работы:

- изучение основных типов современных архитектур информационных систем;
- изучение методов создания диаграмм архитектур информационных систем в среде Microsoft Visio.

1. Краткие сведения из теории

Архитектура информационной системы – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

Исторически первыми появились информационные системы на основе файл-серверной архитектуры. Файл-серверная архитектура представляет собой наиболее простой случай распределенной обработки данных, согласно которой на сервере располагаются только файлы данных. На клиентской части находятся не только приложения пользователей, но и средства управления базами данных (рисунок 78).



Рисунок 78 – Структура информационной системы с файл-сервером

Применение архитектуры файл-сервер привлекает своей простотой, удобством использования и доступностью. Она представляет интерес для малых рабочих групп, нередко и сегодня используется в информационных системах небольших организаций.

Однако файл-серверная архитектура информационных систем имеет ряд недостатков. Она не позволяет в полной мере обеспечить конфиденциальность доступа и целостность данных. По сети файлы передаются целиком, независимо от того, какая часть содержащихся в них данных нужна пользователю. Это сильно перегружает сеть и уменьшает быстродействие системы. Невысока и надежность системы на основе файл-серверов. Сбой на одной из рабочих станций в момент записи файла приводит к потере или искажению данных. Для обеспечения непротиворечивости данных приходится блокировать файлы, что также приводит к замедлению работы.

Информационные системы с клиент-серверной архитектурой позволяют избежать проблем файл-серверных приложений. При такой архитектуре сервер базы данных, расположенный на компьютересервере, обеспечивает выполнение основного объема обработки данных. Клиентское приложение формирует запросы к серверу базы данных, как правило, в виде инструкций языка SQL. Сервер извлекает из базы запрошенные данные и передает на компьютер клиента. Главное достоинство такого подхода – значительно меньший объем передаваемых данных.

Большинство конфигураций информационных систем типа «клиент-сервер» использует двухуровневую модель, в которой клиент обращается к серверу (рисунок 79).

Обеспечение безопасности данных – очень важная функция для успешной работы информационной системы. Если у базы данных слабая система безопасности, любой достаточно подготовленный пользователь может нанести серьезный ущерб работе организации. Следует отметить, что защита данных в файл-серверной информационной системе изначально не может быть обеспечена на должном уровне.

Безопасность же современных серверов баз данных, организованная с помощью самой операционной системы, путем ограничения доступа пользователей через представления, с использованием схем, имен входов, ролей, шифрования базы данных, заслуживает похвалы.

В настоящее время архитектура «клиент-сервер» широко признана и находит применение для организации работы приложений как для

рабочих групп, так и для информационных систем масштаба организации.

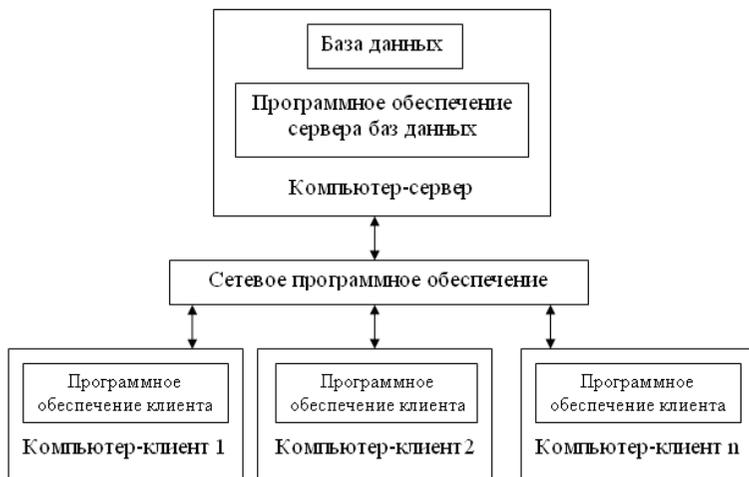


Рисунок 79 – Структура информационной системы с клиент-сервером

2. Пример

Исходя из анализа построенных логических моделей проектируемой системы можно сделать вывод о том, что данные модели можно реализовать на основе использования любой современной персональной системы управления базами данных (например, Microsoft Access), установленной на компьютере менеджера по продажам, доступ к которой должен иметь компьютер директора. Другими словами, создаваемую информационную систему можно реализовать на основе файл-серверной архитектуры с совмещенным (невыделенным) сервером (рисунок 80).

Для построения данной схемы в среде Microsoft Visio выполните следующие действия:

1. Запустите Microsoft Office Visio. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней – элемент *Корпоративное приложение* (рисунок 81).

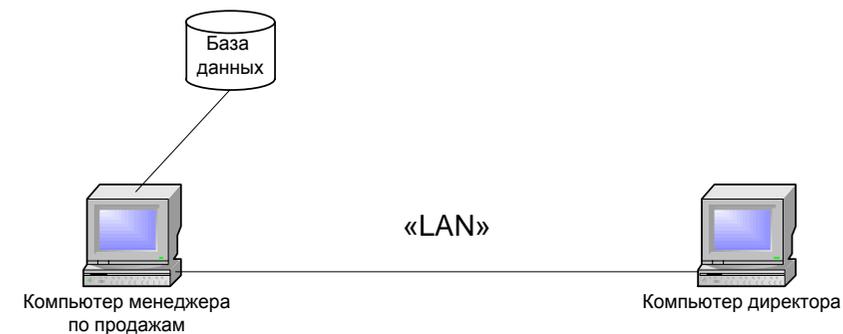


Рисунок 80 – Архитектура автоматизированной системы контроля времени исполнения заказов

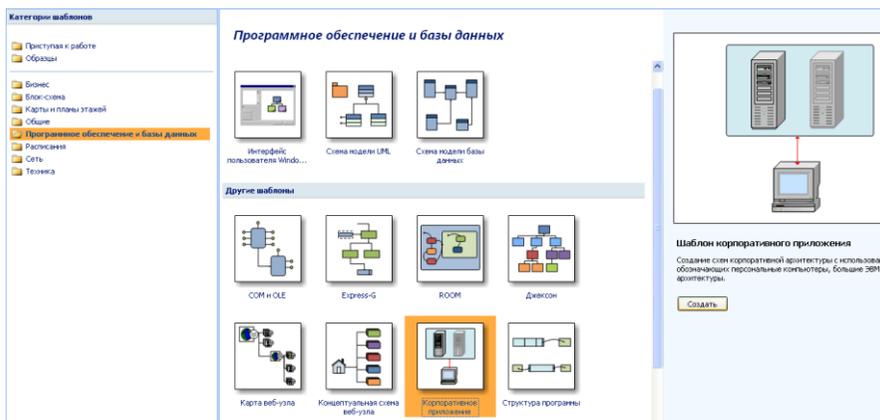


Рисунок 81 – Выбор шаблона

2. С помощью элемента *Рабочая станция*  пакета *Корпоративное приложение* поместите на страницу два объекта. Двойным нажатием левой кнопкой мыши по элементу задайте соответствующие имена: *Компьютер менеджера по продажам* и *Компьютер руководителя* (рисунок 82).

3. С помощью элемента *Хранилище данных*  пакета *Корпоративное приложение* поместите на страницу данный объект. Двойным

нажатием левой кнопки мыши по элементу задайте имя *База данных*.



Рисунок 82 – Компьютеры создаваемой системы

4. С помощью элемента *Линия связи* пакета *Корпоративное приложение* соедините элементы модели. Для того чтобы убрать направления стрелок, кликните правой кнопкой мыши на нужной стрелке и выберите *Формат/Линия*. В открывшемся окне удалите концы линий, выбрав вариант *00:Нет* из списка в полях *Начало* и *Конец* (рисунок 83).

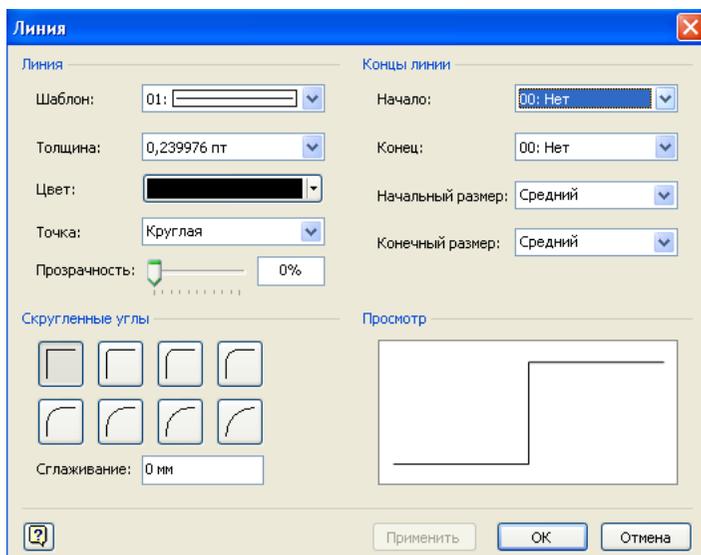


Рисунок 83 – Окно форматирования линии

3. Задание

В среде Microsoft Visio изобразите перспективную архитектуру информационной системы ООО «Компьютер» типа «клиент-сервер», которая должна иметь вид, представленный на рисунке 84.

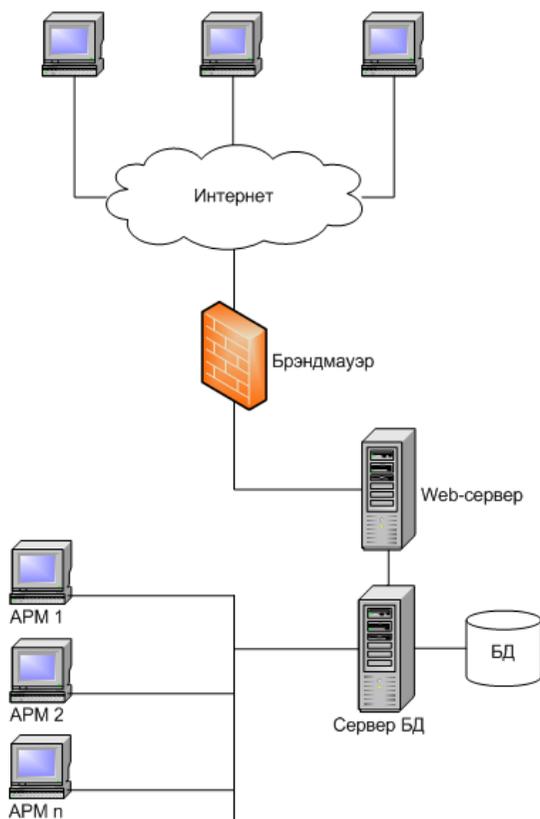


Рисунок 84 – Архитектура перспективной информационной системы ООО «Компьютер»

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ : [пер. с англ.] / Г. Буч. – 2-е изд. – М. : Бинوم ; СПб. : Невский диалект, 1999.

Буч, Г. Язык UML. Руководство пользователя : [пер. с англ.] / Г. Буч, Дж. Рамбо, А. Джекобсон. – М. : ДМК, 2000.

Вендров, А. М. Проектирование программного обеспечения экономических информационных систем : учеб. / А. М. Вендров. – М. : Финансы и статистика, 2000.

Калашян, А. Н. Структурные модели бизнеса: DFD-технологии / А. Н. Калашян, Г. Н. Калянов. – М. : Финансы и статистика, 2003.

Кармайкл, Э. Быстрая и качественная разработка программного обеспечения : [пер. с англ.] / Э. Кармайкл, Д. Хейвуд. – М. : Вильямс, 2003.

Коберн, А. Современные методы описания функциональных требований к системам : [пер. с англ.] / А. Коберн. – М. : ЛОРИ, 2002.

СОДЕРЖАНИЕ

Пояснительная записка.....	3
Лабораторная работа 1. Функциональное моделирование в стандарте IDEF0	4
Лабораторная работа 2. Диаграммы вариантов использования	17
Лабораторная работа 3. Модели данных информационных систем ...	29
Лабораторная работа 4. Диаграммы потоков данных	38
Лабораторная работа 5. Диаграммы классов.....	46
Лабораторная работа 6. Диаграммы взаимодействия.....	53
Лабораторная работа 7. Диаграммы состояний	66
Лабораторная работа 8. Выбор архитектуры информационной системы	74
Список рекомендуемой литературы	78

Учебное издание

**ПРОЕКТИРОВАНИЕ
ИНФОРМАЦИОННЫХ СИСТЕМ**

**Практикум
для реализации содержания образовательных
программ высшего образования I степени**

Авторы-составители:
Семенюта Андрей Николаевич
Ятченко Лариса Владимировна

Редактор М. П. Любошенко
Компьютерная верстка Л. Ф. Барановская

Подписано в печать 06.08.15. Формат 60 × 84 ¹/₁₆.
Бумага типографская № 1. Гарнитура Таймс. Ризография.
Усл. печ. л. 4,65. Уч.-изд. л. 5,00. Тираж 110 экз.
Заказ №

Издатель и полиграфическое исполнение:
учреждение образования «Белорусский торгово-экономический
университет потребительской кооперации».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/138 от 08.01.2014.
Просп. Октября, 50, 246029, Гомель.
<http://www.i-bteu.by>

**БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»**

Кафедра информационно-вычислительных систем

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

**Практикум
для реализации содержания образовательных
программ высшего образования I степени**

Гомель 2015