МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Колледж экономики, управления и права

Методические указания
по организации практических занятий и самостоятельной работы
по МДК.01.02
Методы и средства проектирования информационных систем

Специальность

09.02.04 Информационные системы (по отраслям)

Ростов-на-Дону
2021

Методические указания по **МДК.01.02 Методы и средства проектирования информационных систем** разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.04 Информационные системы (по отраслям), предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

Составитель (автор): Д.А. Морозюк преподаватель колледжа ЭУП

Рассмотрены на заседании предметной (цикловой) комиссии специальностей 09.02.04 Информационные системы (по отраслям) и 09.02.05 Прикладная информатика (по отраслям)

Протокол № ___ от _____ 2021 г.

Председатель П(Ц)К _____ С.В. Шинакова
личная подпись

и одобрены решением учебно-методического совета колледжа.

Протокол № 6 от 31 мая 2021 г.

Председатель учебно-методического совета колледжа
_____ С.В.Шинакова
личная подпись

Рекомендованы к практическому применению в образовательном процессе.

СОДЕРЖАНИЕ

Практическая работа №1
Разработка технического задания

Цель: знакомство со стандартами, регламентирующими жизненный цикл разработки ИС. Приобретение практических навыков при разработке программного документа Техническое задание.

1. Теоретические сведения

Для обеспечения потребностей по разработке программного обеспечения (ПО) ИС необходимо взаимосвязанное совершенствование технических решений, технологий проектирования и программирования, инструментальных средств, а также обучения специалистов.

Стандарты для процессов, инструментальных средств и данных, которые отражают лучшую практику и защищают от неблагоприятных последствий, играют определенную роль в обеспечении указанных потребностей, в частности, поддерживая доверие потребителя к продуктам, услугам и технологиям разработки ПО.

Использование стандартов при разработке ПО ИС позволит обеспечить:

- повышение надежности;
- повышение эффективности применения и снижение затрат в сфере сопровождения программных средств (ПС);
- увеличение коэффициента повторного использования ПС общего назначения;
- повышение объективности оценок качества ПС;
- создание условий для конкуренции разработчиков на внутреннем рынке ПС;
- обеспечение конкурентоспособности отечественных ПС на мировом рынке.

Стандарты комплекса ГОСТ 34 на создание и развитие автоматизированных систем (АС) – обобщенные, но воспринимаемые как весьма жесткие по структуре жизненного цикла (ЖЦ) и проектной документации. Наиболее популярными можно считать ГОСТ 34.601-90 **(**Автоматизированные системы. Стадии создания) и ГОСТ 34.602-89 (Техническое задание на создание АС). Введение единой, достаточно качественно определенной терминологии, наличие достаточно разумной классификации работ, документов, видов обеспечения и др. способствует более полной и качественной стыковке разных систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС.

В последние годы в стране идет интенсивная работа по гармонизации государственных стандартов Российской Федерации с международными стандартами ISO в области создания нормативной базы управления жизненным циклом программных средств и информационных систем. В основе стандартизации - ГОСТ Р ИСО/МЭК 12207-99 "Информационная технология. Процессы жизненного цикла программных средств", который является аутентичным переводом международного стандарта ISO/IEC 12207: 1995-08-01.

Техническое задание (ТЗ) на АС - утвержденный в установленном порядке документ, определяющий цели, требования и основные исходные данные необходимые для разработки АС и содержащий предварительную оценку экономической эффективности.

ТЗ содержит основные технические требования, предъявляемые к изделию и исходные данные для разработки; в ТЗ указываются назначение объекта, область его применения, стадии разработки документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов, полученных в ходе предпроектного обследования.

Как инструмент коммуникации в связке общения заказчик-исполнитель, техническое задание позволяет:

А) Обеим сторонам:

- представить готовый продукт;
- выполнить проверку готового продукта по пунктам (приёмочное тестирование – проведение испытаний);
- уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением испытаний).

Б) Заказчику:

- осознать, что именно ему нужно, опираясь на существующие на данный момент технические возможности и свои ресурсы;
- требовать от исполнителя соответствия продукта всем условиям, оговорённым в ТЗ.

В) Исполнителю:

– правильно понять суть задачи, показать заказчику «технический облик» будущего программного изделия или АС;
– спланировать выполнение проекта и работать по намеченному плану;
– отказаться от выполнения работ, не указанных в ТЗ.

2. Задание к выполнению

В соответствии с индивидуальным вариантом, определяющим предметную область на основании ГОСТ 34.602-89, разработать документ Техническое задание на создание АИС.

3. Варианты

1. ИС «Отдел кадров»;
2. ИС «Агентство аренды»;
3. ИС «Аптека»;
4. ИС «Ателье»;
5. ИС «Аэропорт»;
6. ИС «Библиотека»;
7. ИС «Кинотеатр»;
8. ИС «Поликлиника»;
9. ИС «Автосалон»;
10. ИС «Таксопарк».

4. Контрольные вопросы

1. Этапы развития проекта ИС?
2. Понятие модели жизненного цикла ПО ИС?
3. Обзор моделей жизненного цикла, их недостатки и преимущества?
4. Обзор и особенности методологии *RAD*?
5. Обзор стандартов, регламентирующих ЖЦ ПО ИС?
6. Назначение, содержание и степень адаптивности стандарта ГОСТ 34.601-90?
7. Стадии и этапы создания АС в соответствии с ГОСТ 34.601-90?
8. Назначение, содержание стандарта ГОСТ 34.602-89?
9. Назначение, содержание и степень адаптивности стандарта ГОСТ Р ИСО/МЭК 12207?
10. Содержание одного из основных процессов ЖЦ ПО ИС по ГОСТ Р ИСО/МЭК 12207 – Разработка?
11. Содержание стандарта ISO/IEC 15288?
12. Назначение программного документа Техническое задание на создание АС. Порядок разработки, согласования и утверждения документа?
13. Состав и содержание документа ТЗ?

Практическая работа №2
ER-моделирование в нотации Чена

Цель: Изучение методов построения логической модели базы данных выбранной предметной области

1. Теоретические сведения

В настоящее время существуют две группы методов проектирования схемы базы данных:

• методы с использованием диаграмм сущность-связь;

• методы функциональных и многозначных зависимостей.

Методы, основанные на использовании диаграмм сущность-связь, обеспечивают относительно простой способ формирования схемы базы данных. Однако после проектирования схемы базы данных требуется дополнительная проверка, находятся ли построенные таблицы в соответствующей нормальной форме (обычно схема базы данных приводится к третьей нормальной форме).

Методы с использованием функциональных и многозначных зависимостей являются более формализованными. Однако эти методы могут приводить к построению схемы базы данных с таблицами в труднопонимаемой со стороны пользователя форме. Кроме того, сами алгоритмы проектирования схемы имеют не полиномиальную временную сложность, что ограничивает их использование для баз данных с большим количеством атрибутов.

Модели сущность-связь основаны на выделении в предметной области, для которой осуществляется проектирование базы данных, различных типов объектов, информацию о которых требуется хранить в базе данных.

Набор однотипных объектов предметной области образует сущность. Между сущностями могут быть установлены информационные связи (зависимости), которые также могут быть учтены при проектировании схемы базы данных. Совокупность сущностей и связи между ними составляют информационную модель данных предметной области (Entity-Relationship диаграмму).

В настоящее время существует несколько приемов выделения сущностей и связей – нотации Чена, Мартина, Баркера, IDEF1X и т.д.

Основные определения

Сущность (Entity) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Атрибут (Attribute) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе.

Связь (Relationship) – поименованная ассоциация между сущностями, значимая для рассматриваемой предметной области.

Нотация Чена

В нотации Чена различают зависимые и независимые сущности.

Сущность называется ***независимой***, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями.

Сущность называется ***зависимой*** от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

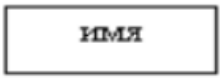
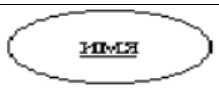
Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности.

Сущности и связи могут иметь атрибуты. Для каждой сущности находится атрибут (или набор атрибутов), значение которого однозначно определяет экземпляр сущности. Этот атрибут является ключом сущности.

Связь также может иметь ключевой атрибут. В ряде случаев для удобства организации связей в состав атрибутов сущности вводится искусственный ключ (обычно число). Ключевой атрибут (набор атрибутов)

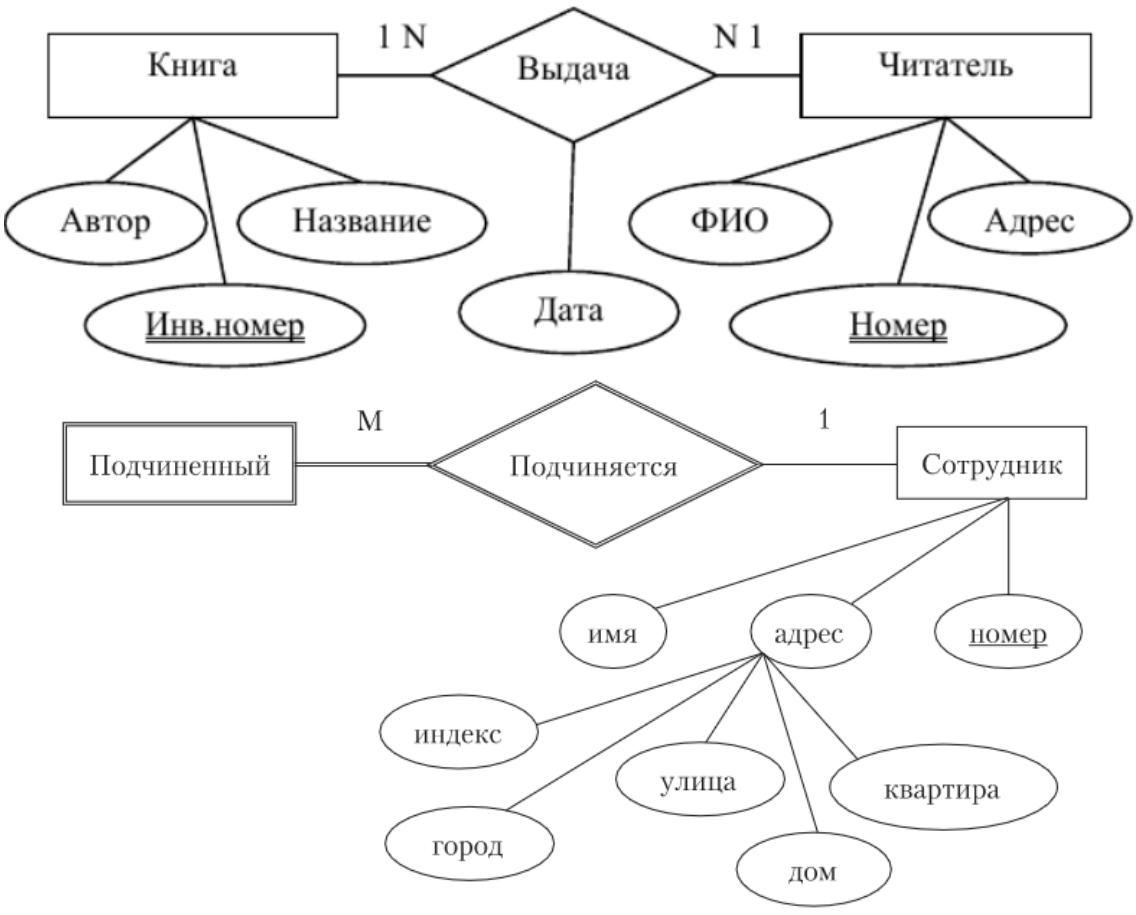
на диаграмме отмечается двумя линиями снизу, внешние ключи отмечаются одной линией. В таблице представлены основные элементы, используемые для формирования ER-диаграммы в нотации Чена.

Таблица 1 – Элементы ER-диаграммы в нотации Чена

Элемент диаграммы	Описание
	Независимая сущность
	Зависимая сущность
	Связь
	Идентифицирующая связь
	Атрибут
	Первичный ключ
	Внешний ключ

Связь имеет кардинальное число, определяющее, какое количество экземпляров одной сущности имеет информационную связь с экземплярами другой сущности.

На рисунках представлен пример ER-диаграммы в нотации Чена.



2. Методика выполнения

В качестве примера возьмем ***базу данных университета для описания учебного процесса.***

1. Проанализируйте предметную область.

База данных создаётся для информационного обслуживания сотрудников и студентов университета. БД должна содержать данные о сотрудниках, студентах, группах, предметах, оценках.

В соответствии с предметной областью система строится с учётом следующих особенностей:

- на одной должности работают несколько сотрудников;
- один сотрудник преподает несколько предметов;
- один предмет преподают несколько сотрудников;
- в одной группе несколько студентов;
- один студент получает оценки по нескольким предметам;
- по одному предмету оценку получают несколько студентов.

Для инфологического проектирования воспользуемся методом «сущность-связь». Для того, чтобы представить, как устроена предметная область нужно задать множество объектов реального мира (главная проблема что считать объектом). Объект – семантическое понятие, которое может быть полезно при обсуждении устройств реального мира. Сущность реального мира – объекты – не обязательно материальны – важно понятие существенно и различимо для других. Между объектами могут возникать связи трех видов:

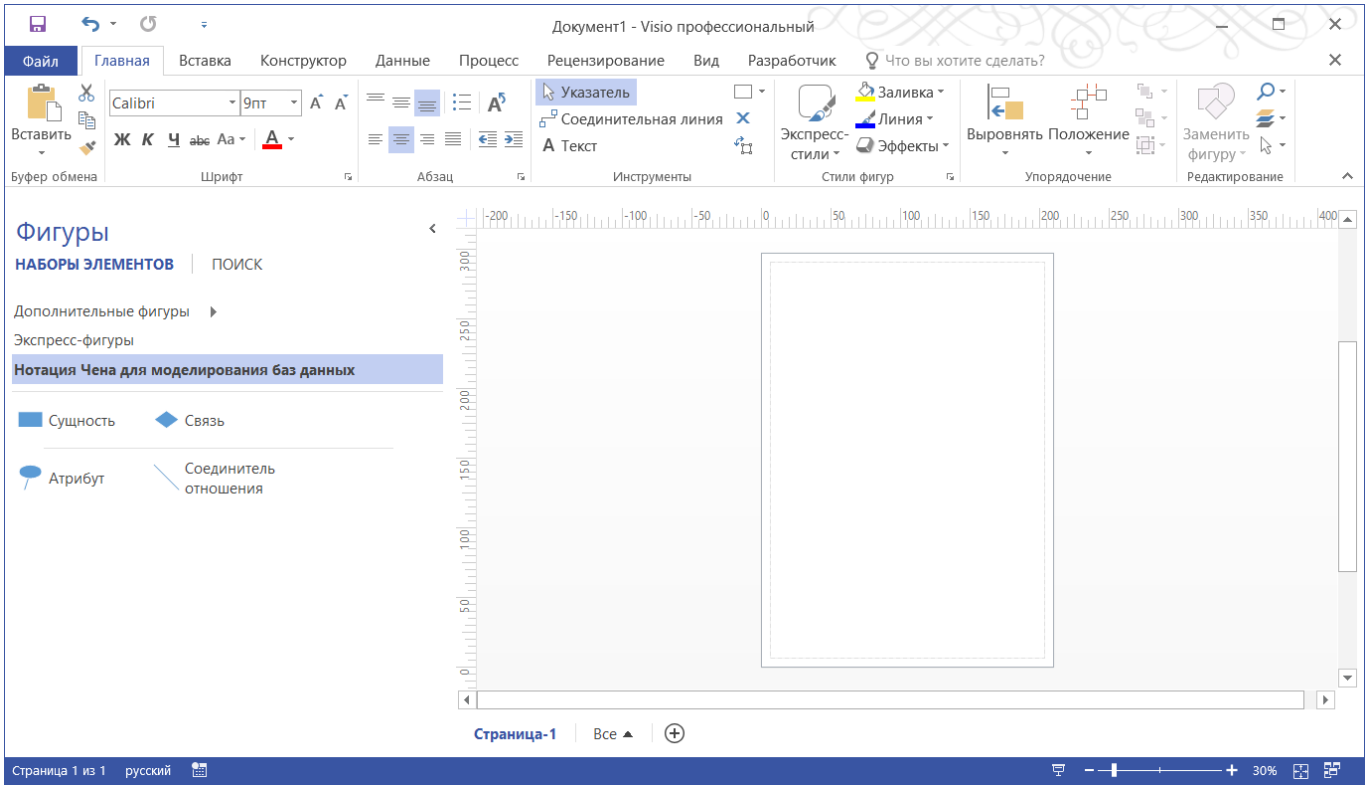
- один к одному 1:1 (пациент: место в палате);
- один к многим 1:n и многие к одному n:1 (абонент : номер телефона);
- многие ко многим n:n (пациент : хирург).

Выделим базовые сущности этой предметной области:

- Сотрудники. Атрибуты сотрудников – табельный номер (PK), ФИО, пол, дата рождения, адрес и телефон. Для сотрудника необходимо хранить код должности (FK).
- Должности. Атрибуты – ID (PK), название.
- Студенты. Атрибуты– номер зачетки (PK), ФИО, пол, дата рождения, адрес, телефон. Для студента необходимо хранить код группы (FK).
- Группы. Атрибуты– ID (PK), название.
- Предметы. Атрибуты – ID (PK), название. Для предмета необходимо хранить код сотрудника (FK).
- Оценки. Атрибуты – ID (PK), название.

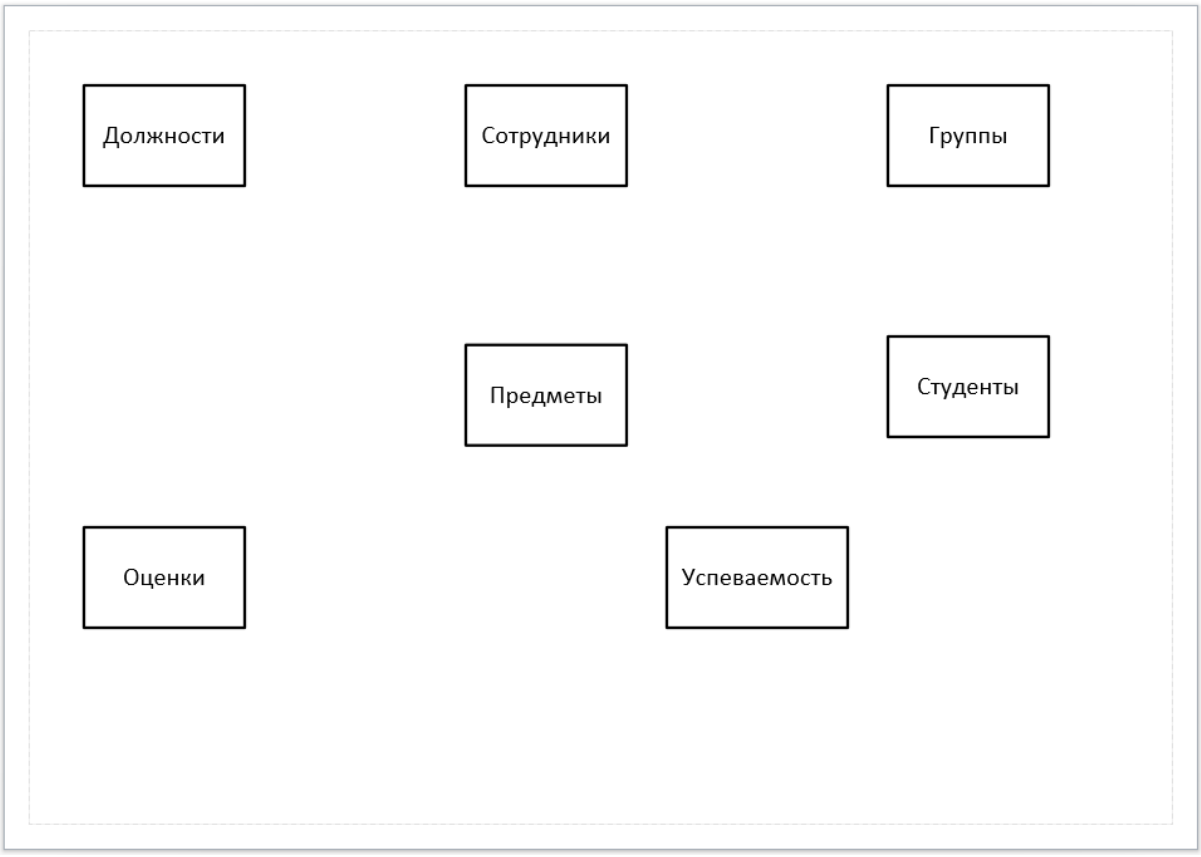
Для отражения успеваемости в системе нужно учитывать по какому предмету каким студентом какая оценка получена. Для успеваемости необходимо хранить код студента (FK), код предмета (FK), код оценки (FK).

2. Запустите MS Visio.
3. На закладке выбора шаблона выберите категорию *Базы данных* и в ней элемент *Нотация Чена*. Нажмите кнопку *Создать* в правой части экрана.
4. Окно программы примет вид

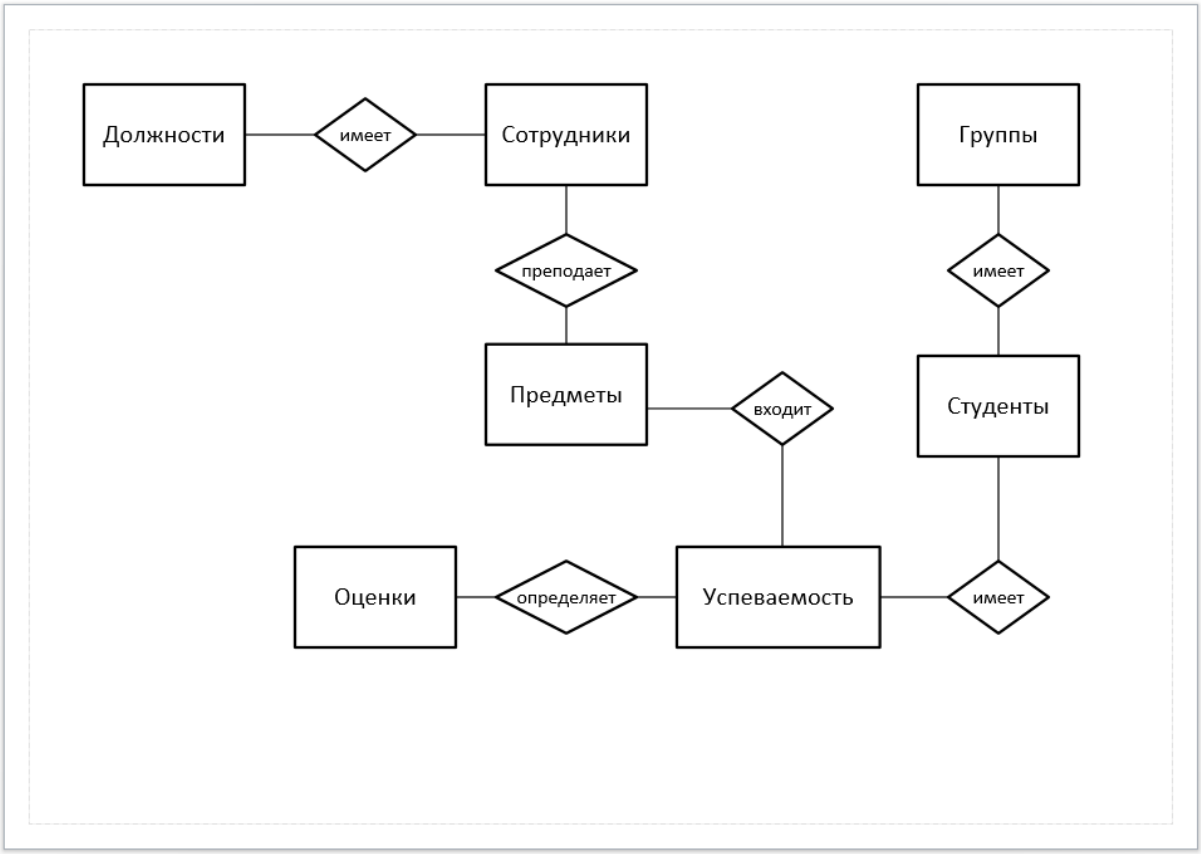


5. В соответствии с анализом предметной области необходимо добавить на рабочий лист четыре фигуры Сущность.

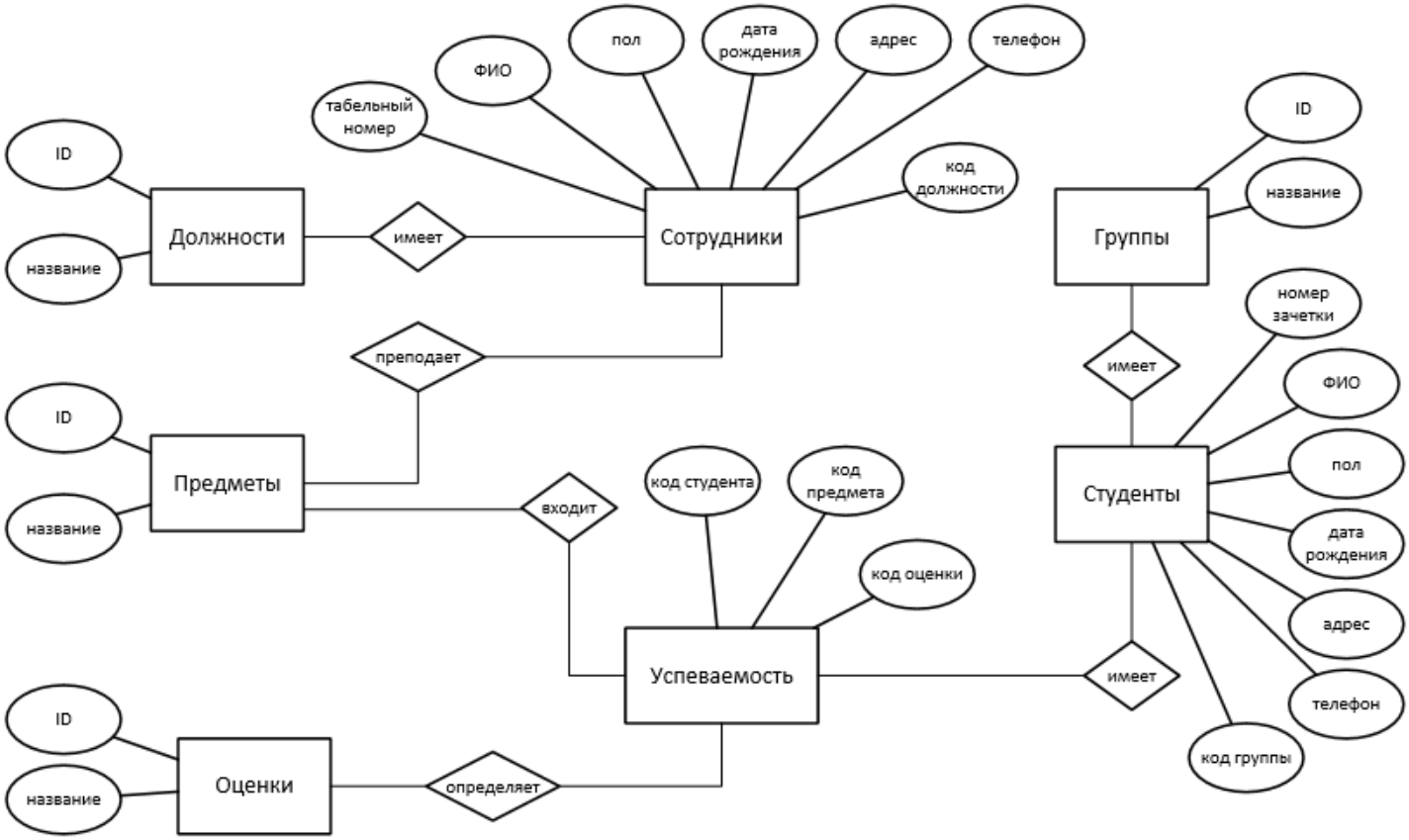
6. Для добавления текста фигуре необходимо дважды щелкнуть по ней левой кнопкой мыши.



7. Далее необходимо добавить связи между сущностями. Для этого необходимо использовать фигуру ромб.



8. На данном этапе получена общая диаграмма Чена. Для ее детализации необходимо добавить атрибуты ко всем сущностям. Атрибуты изображаются в виде овала

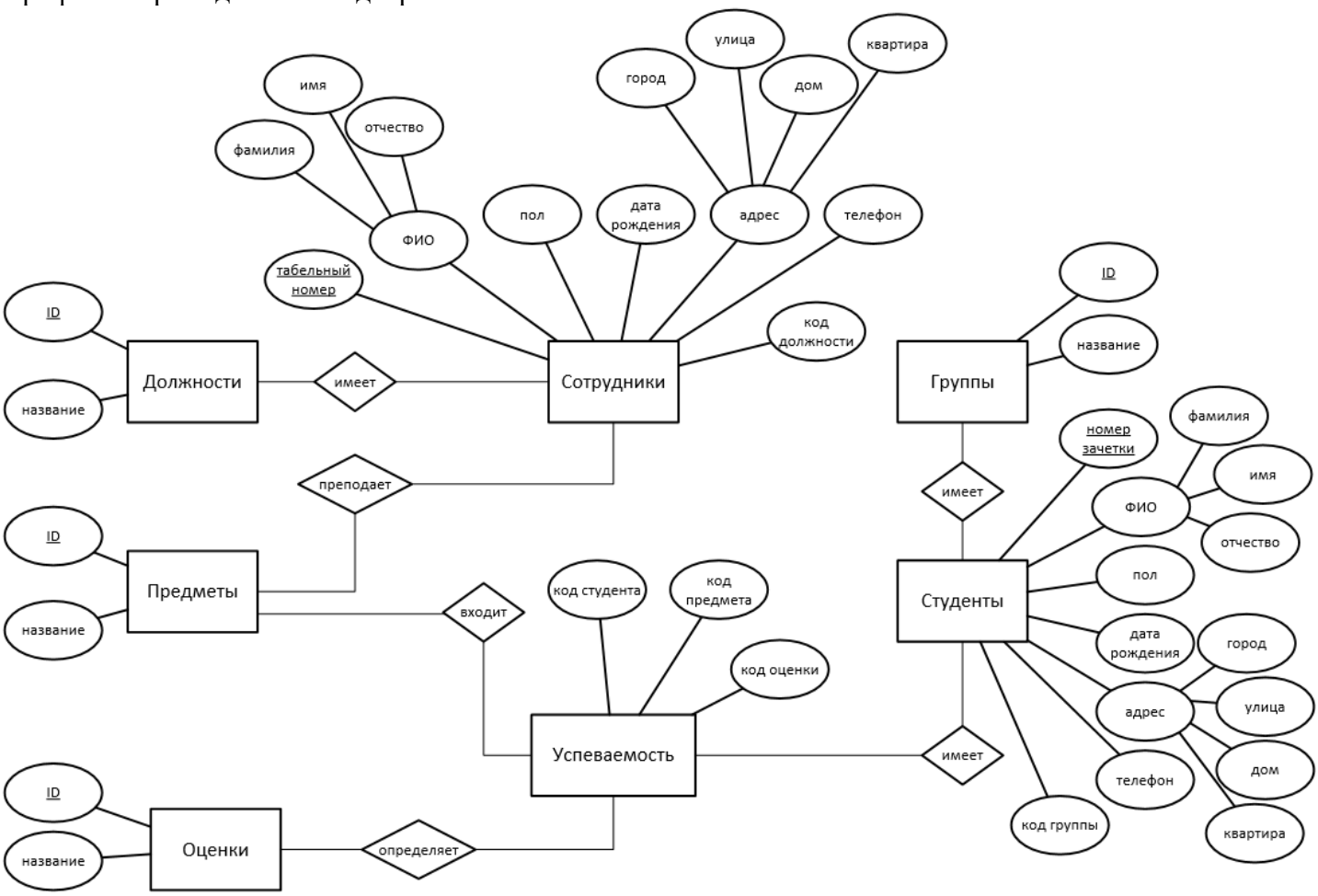


9. Далее необходимо разбить на дополнительные атрибуты ФИО и адрес для сотрудников и студентов, а также указать первичный и внешний ключ.

Для разбиения составного атрибута необходимо добавить фигуру овал и соединить с нужным атрибутом.

Для указания первичного ключа необходимо нажать правой кнопкой мыши на атрибуте и в контекстном меню выбрать пункт Задать первичный ключ.

Для указания внешнего ключа необходимо выбрать нужный атрибут и дополнительных параметрах шрифта выбрать двойное подчеркивание текста.



3. Варианты

1. Проектирование ИС «Отдел кадров»;
2. Проектирование ИС «Агентство аренды»;
3. Проектирование ИС «Аптека»;
4. Проектирование ИС «Ателье»;
5. Проектирование ИС «Аэропорт»;
6. Проектирование ИС «Библиотека»;
7. Проектирование ИС «Кинотеатр»;
8. Проектирование ИС «Поликлиника»;
9. Проектирование ИС «Автосалон»;
10. Проектирование ИС «Таксопарк»;
11. Проектирование ИС «Издательство»;
12. Проектирование ИС «Прокат велосипедов»;
13. Проектирование ИС «Спортивный клуб».

4. Контрольные вопросы

1 Группы методов проектирования схемы базы данных.
2 Приемы выделения сущностей и связей.
3 Основные определения ER-диаграмм.
4 Элементы диаграммы в нотации Чена.

Практическая работа №3
ER-моделирование в нотации IDEF1X

Цель: освоение технологии построения информационной модели логического и физического уровней в нотации IDEF1X с использованием пакета Microsoft Visio.

1 Задачи

Основными задачами практической работы являются:

- приобретение студентами навыков построения информационной модели логического уровня;
- нормализации полученной модели;
- построения информационной модели физического уровня.

2 Краткие теоретические сведения

Понятие информационной модели. Уровни информационной модели

Методология IDEF1X – язык для семантического моделирования данных, основанный на концепции «сущность-связь».

Различают два уровня информационной модели: **логический** и **физический**.

Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.

Различают 3 подуровня логического уровня модели данных, отличающиеся по глубине представления информации о данных:

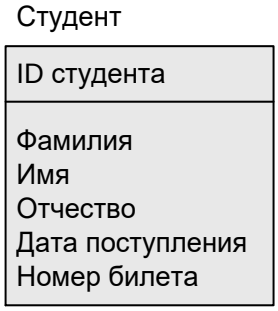
- диаграмма сущность-связь (Entity-Relationship Diagram (ERD);
- модель данных, основанная на ключах (Key Based Model (KB);
- полная атрибутивная модель (Fully Attributed Model (FA).

Физическая модель отражает физические свойства проектируемой базы данных (типы данных, размер полей, индексы). Параметры физической информационной модели зависят от выбранной системы управления базами данных (СУБД).

Основные элементы информационной модели логического уровня

Сущности и атрибуты

Сущность – это множество **реальных** или **абстрактных объектов** (людей, предметов, документов и т.п.), **обладающих общими атрибутами или характеристиками**. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. *Именование сущности* осуществляется с помощью *существительного в единственном числе*. При этом имя сущности должно отражать **тип** или **класс** объекта, а не его **конкретный экземпляр** (например, **Студент**, а не **Петров**).



Любая сущность характеризуется набором атрибутов (**свойств**).

Атрибут сущности – характеристика сущности, то есть свойство реального объекта. Например, на рисуке атрибутами сущности **«Студент»** являются **«ID студента»**, **«Фамилия»**, **«Имя»**, **«Отчество»**, **«Дата поступления»** и **«Номер билета»**.

В свою очередь, *атрибуты сущности* делятся на 2 вида: *собственные* и *наследуемые. Собственные* атрибуты являются уникальными в рамках модели. *Наследуемые* атрибуты передаются от сущности-родителя при установке связи с другими сущностями.

Первичный ключ (Primary Key, PK). Каждая сущность должна обладать *атрибутом* или *комбинацией атрибутов*, чьи значения *однозначно определяют* каждый *экземпляр сущности*. Эти атрибуты образуют *первичный ключ* сущности.

Внешний ключ (Foreign Key, FK). Если между двумя сущностями *имеется специфическое отношение* связи или *категоризации*, то *атрибуты*, входящие в *первичный ключ родительской* или *общей сущности*, *наследуются* в качестве *атрибутов сущностью-потомком* или *категориальной сущностью* соответственно. Эти атрибуты и называются внешними ключами. Наследуемый атрибут может использоваться в сущности в качестве части или целого первичного ключа, альтернативного ключа или не ключевого атрибута.

Отношения в IDEF1X-модели

При построении информационной модели различают следующие типы отношений между сущностями: *идентифицирующее, не идентифицирующее, не специфическое (многие-ко-многим) и отношения категоризации*.

Мощность отношения служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Нормализация данных

Нормализация – это процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Процесс нормализации сводится к последовательному приведению структур данных к нормальным формам – формализованным требованиям к организации данных.

Первая нормальная форма (1НФ). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра.

Вторая нормальная форма (2НФ). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый не ключевой атрибут полностью зависит от первичного ключа (не может быть зависимости от части ключа).

Третья нормальная форма (3 НФ). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой не ключевой атрибут не зависит от другого не ключевого атрибута (не должно быть зависимости между не ключевыми атрибутами).

3 Рекомендации по выполнению

Практическая работа выполняется в соответствии с индивидуальным вариантов в Microsoft Visio.

Данное занятие может выполняться на основе результатов функционального моделирования предметной области.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

4 Методика выполнения практического занятия

Упражнение 1. Построение логической информационной модели уровня «сущность-связь»

1. Составление пула – списка потенциальных сущностей

Информационная модель может быть построена на основе функциональной модели или без нее.

Список потенциальных сущностей (при использовании программного продукта MS Office Visio для функционального моделирования) должен быть составлен вручную. В случае использования CASE-средства *AllFusion Process Modeler* отчет по интерфейсным дугам генерируется автоматически. Список потенциальных сущностей для рассматриваемого примера будет представлен таблицей вида.

Теперь из этого списка необходимо выделить сущности, остальные интерфейсные дуги будут преобразованы в атрибуты сущностей.

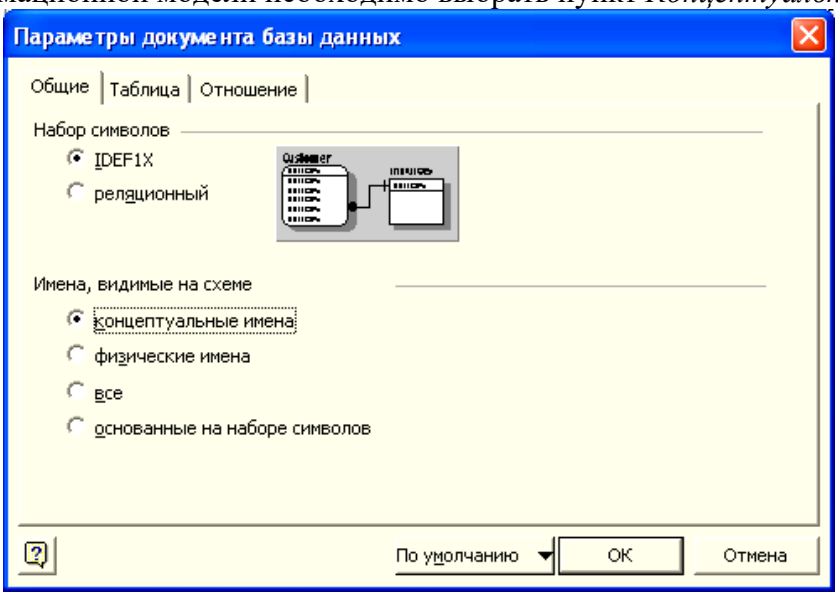
В качестве сущностей выделим следующие:

1) задание;
2) пояснительная записка;
3) курсовая работа;
4) положение о курсовом проектировании;
5) студент;
6) преподаватель;
7) график;
8) методические указания.

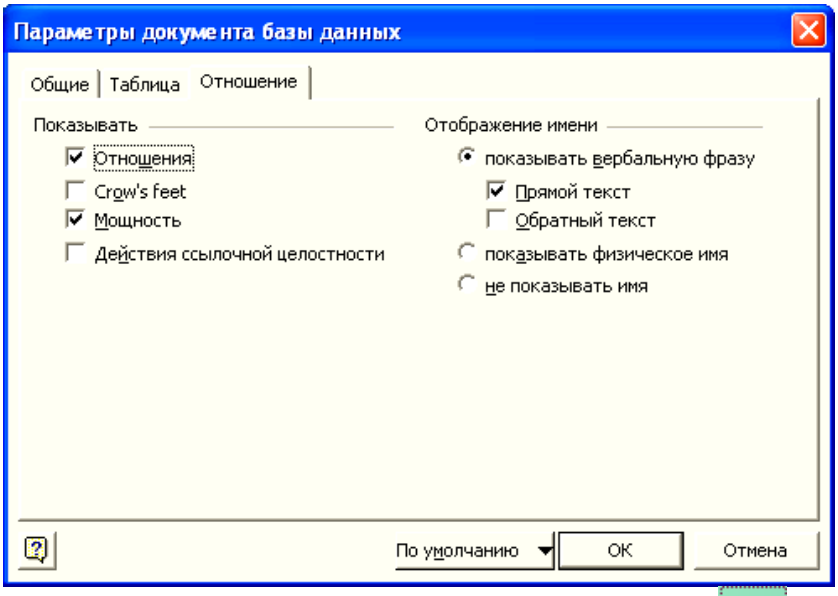
Arrow Name
Варианты заданий
График
Графическая часть
Задание
Замечания, дополнения
Курсовая работа
Литература
Методические указания
Оценка за курсовую работу
Положение о курсовом проектировании
Пояснительная записка
Преподаватель
Расчеты
Список литературы
Студент

2. Создание логической модели «сущность-связь»

1. Запустите MS Visio.
2. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней элемент *Схема модели базы данных*. Нажмите кнопку *Создать* в правой части экрана.
3. Установите необходимые параметры страницы (масштаб, ориентация страницы).
4. MS Visio поддерживает различные нотации моделей баз данных. Для того чтобы задать нотацию IDEF1X, необходимо выбрать пункты меню База данных → Параметры → Документ. В открывшемся окне на вкладке Общие установить переключатель в меню Набор символов на IDEF1X. Меню Имена, видимые на схеме, позволяет указать, какие имена атрибутов сущности будут отображены на диаграмме (концептуальные, физические или оба варианта одновременно). В данном случае для логического представления информационной модели необходимо выбрать пункт *Концептуальные имена*.

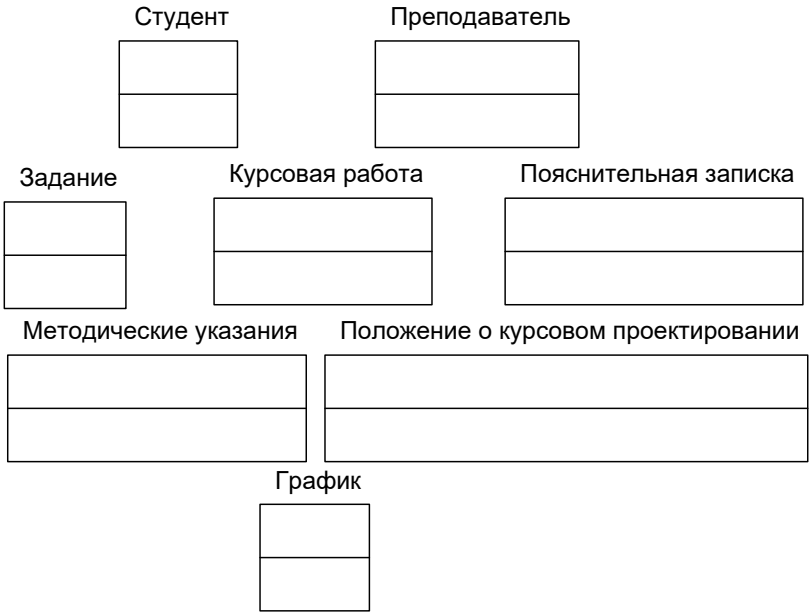


В закладке *Отношение* окна *Параметры документа базы данных* в меню *Показывать* отметить галочкой пункт *Мощность*, в меню *Отображение вида* выбрать пункт *Показывать вербальную фразу*, снять галочку в пункте *Обратный текст*. Данные настройки позволят отобразить имя и мощность связи в модели.



5. Для того чтобы создать сущность, необходимо перетащить элемент  на рабочее поле. Переход в режим редактирования сущности осуществляется двойным щелчком по сущности или по нажатию правой кнопки мыши и выбора пункта меню *Свойства базы данных*.

Чтобы задать имя сущности, в окне *Свойства базы данных* нужно выбрать категорию *Определение*, снять галочку в пункте *Синхронизация имен при вводе* (в противном случае, физическое и логическое имя сущности будут совпадать, что по практическим соображениям не всегда удобно) и задать концептуальное имя сущности. Руководствуясь данным алгоритмом, создадим 8 сущностей, определенных в пункте 1.



6. Далее необходимо установить связи между сущностями.

Сначала составим описание предметной области на естественном языке.

Любой студент должен выполнить одну или несколько курсовых работ.

Каждая курсовая работа должна выполняться одним студентом (в идеале).

Каждая курсовая работа выполняется в соответствии с методическими указаниями и положением о курсовом проектировании.

Курсовая работа сдается по графику.

Курсовая работа оформляется в виде пояснительной записки.

Преподаватель проводит консультации, проверяет и ставит оценку за курсовую работу.

Таким образом, сформулируем имена связей:

СТУДЕНТ выполняет ***КУРСОВУЮ РАБОТУ***.

ПРЕПОДАВАТЕЛЬ проверяет ***КУРСОВУЮ РАБОТУ***.

КУРСОВАЯ РАБОТА выполняется в соответствии с ***ЗАДАНИЕМ***.

КУРСОВАЯ РАБОТА оформляется в виде ***ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ***.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ определяют требования к ***КУРСОВОЙ РАБОТЕ***.

КУРСОВАЯ РАБОТА организуется согласно ***ПОЛОЖЕНИЮ ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ***.

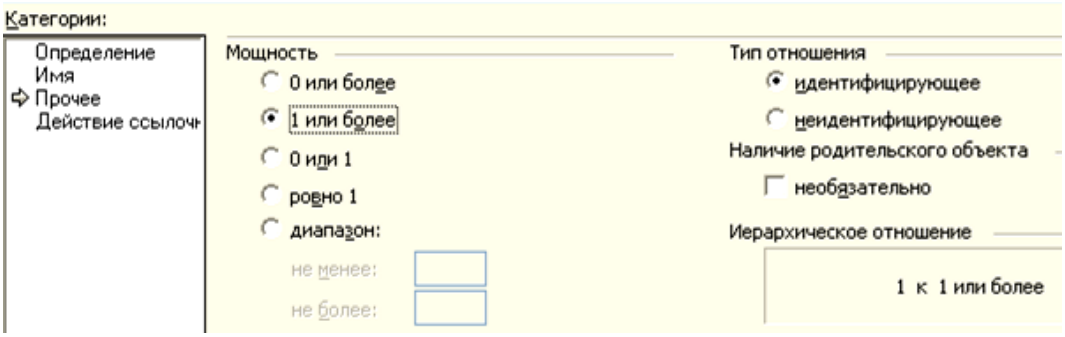
КУРСОВАЯ РАБОТА сдается по ***ГРАФИКУ***.

Во всех случаях сущность *Курсовая работа* является дочерней, за исключением связи с сущностью *Пояснительная записка*. Определим типы связей и построим модель (см. рис. 6). В дальнейшем можно будет подкорректировать связи между сущностями.

Чтобы установить **связи** между сущностями, необходимо перетащить на рабочую область элемент  , поднести один конец стрелки к родительской сущности, другой – к дочерней.

Примечание. При правильном связывании каждая сущность будет подсвечена красным цветом.

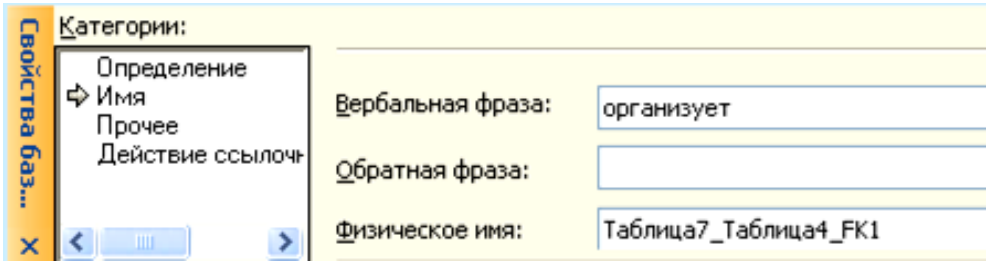
В MS Office Visio по умолчанию используется *не идентифицирующее* отношение. Чтобы изменить **тип связи**, необходимо двойным щелчком по связи открыть окно *Свойства базы данных* и в категории в категории *Прочее* указать тип отношения (идентифицирующее, не идентифицирующее). В этой же категории указывается мощность связи.



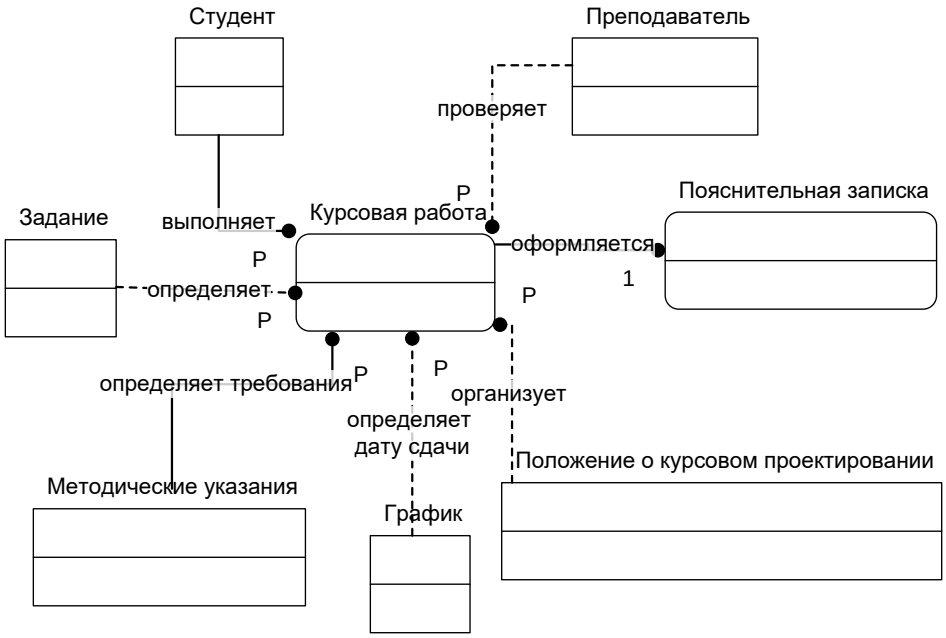
Примечание. Кроме того, при не идентифицирующем отношении нужно указать, является ли наличие родительской сущности обязательным (т.е. может ли существовать экземпляр дочерней сущности, если не существует экземпляра родительской). Если наличие родительского объекта является необязательным, графически это отобразится в виде не закрашенного ромба со стороны родительской сущности.

Следующий шаг – в категории *Имя* в поле *Вербальная фраза* нужно указать имя отношения (рис. 7). Также можно указать имя связи в поле *Обратная фраза* для спецификации отношения потомок-родитель (в нашем случае обратная фраза отображаться не будет).

Примечание. Все изменения при закрытии окна свойств сохраняются автоматически.



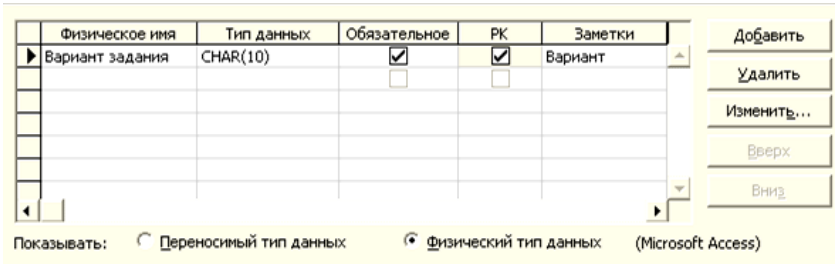
После определения имен, типов связей и задания мощностей получим информационную модель.



Упражнение 2. Разработка логической модели данных, основанной на ключах

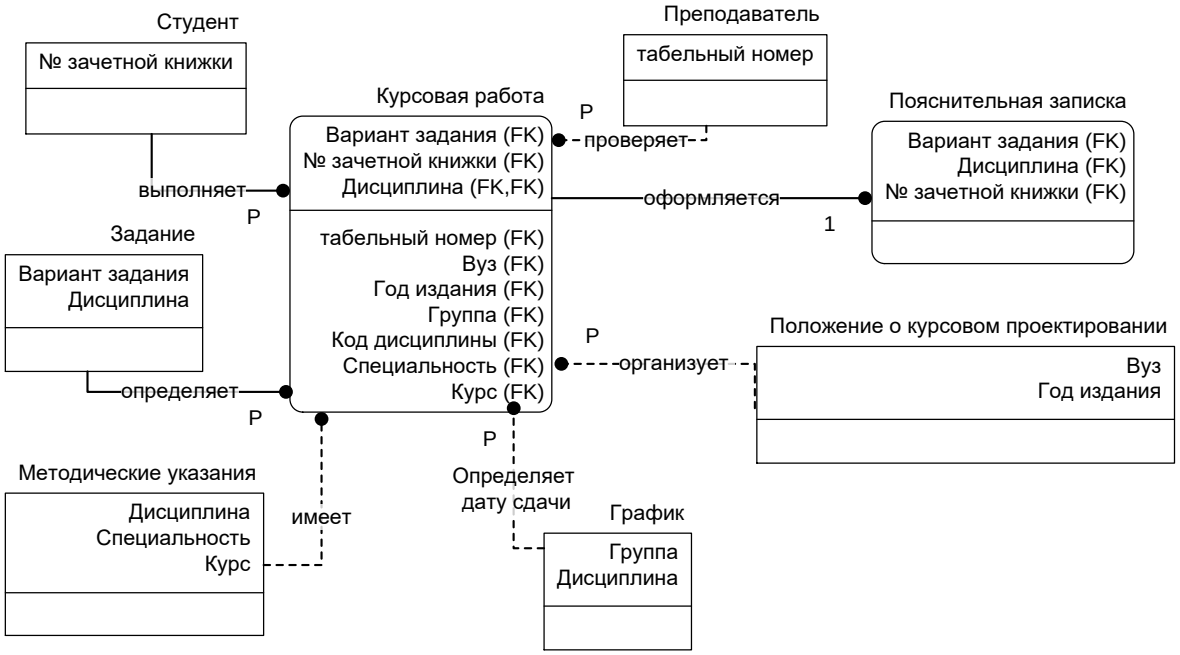
1. Необходимо определить ключевые атрибуты для каждой сущности, обращая внимание на то, что дочерние сущности наследуют ключевые атрибуты от родительских.

Для этого двойным щелчком мыши по сущности откроем окно редактирования ее свойств, перейдем в категорию *Столбцы*, по нажатию кнопки *Добавить* введем имя поля (например, для сущности *Задание* ключевым атрибутом будет являться *Вариант задания*). Чтобы сделать атрибут ключевым, необходимо отметить галочкой пункт *РК*. Данное поле становится обязательным автоматически.

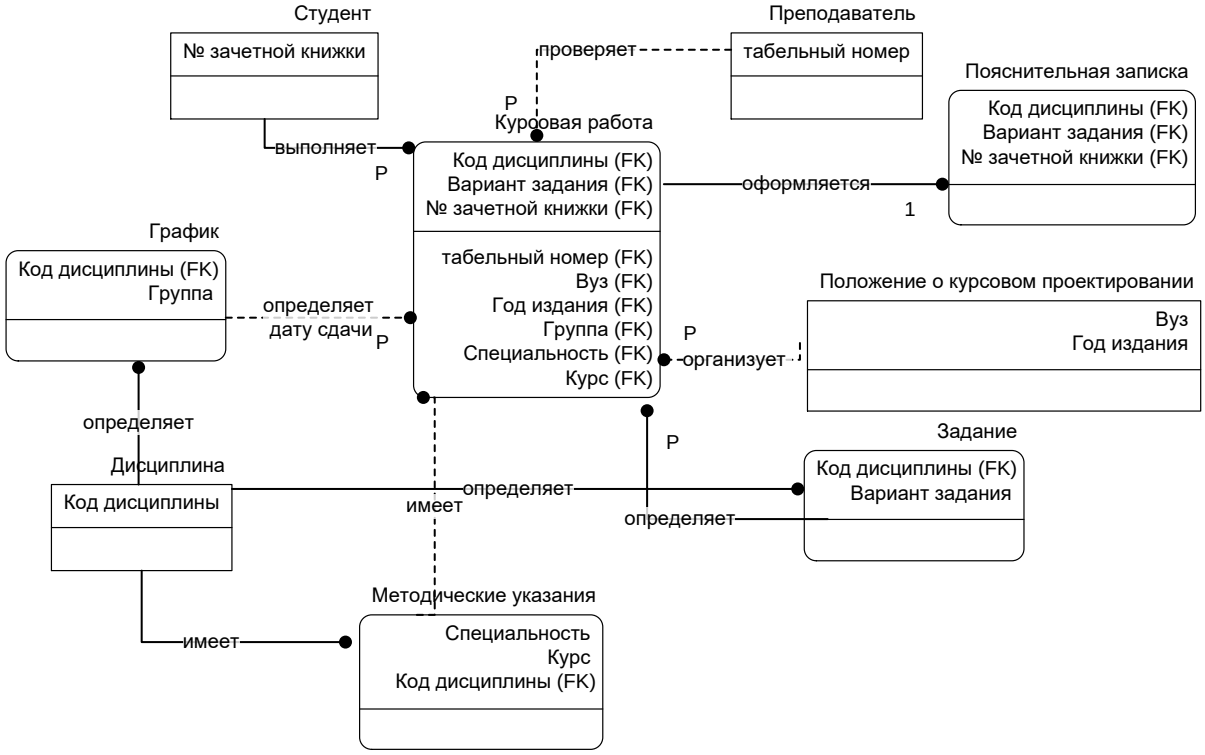


Аналогичным образом зададим ключевые атрибуты для всех сущностей информационной модели.

Как видно из рисунка по сравнению с информационной моделью уровня «сущность-связь», был изменен тип связи между сущностями *Методические указания* и *Курсовая работа*, поскольку ключевые атрибуты сущности *Методические указания* для сущности *Курсовая работа* будут являться избыточными (зная номер зачетной книжки, можно узнать специальность и курс, на котором учится студент).



2. Кроме того, отметим, что три сущности (Задание, График, Методические указания) содержат одинаковые атрибуты Дисциплина. Это является некорректным. Чтобы устранить данную ошибку, выделим одноименную сущность и свяжем ее идентифицирующими связями с вышеуказанными сущностями.



Упражнение 3. Создание полной атрибутивной модели

Для того чтобы получить полную атрибутивную модель, необходимо дополнить сущности не ключевыми атрибутами.

Примечание. Если атрибут не является обязательным, нужно убедиться, что в окне *Свойства базы данных* в категории *Столбцы* в пункте *Обязательное* не стоит галочка. Не обязательные к заполнению атрибуты справа от имени имеют пометку *(О)*.

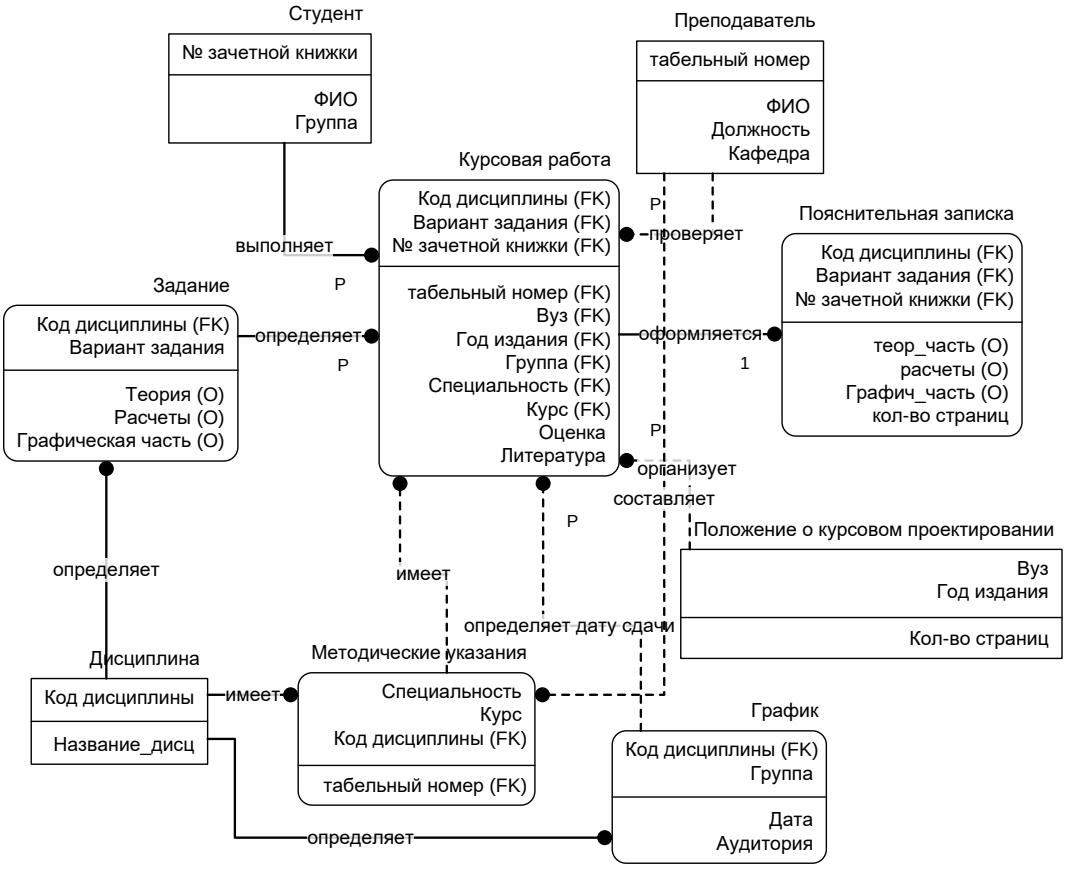
Упражнение 4. Нормализация полной атрибутивной модели

1. Проверим, все ли атрибуты имеют атомарные значения, т.е. среди атрибутов не должно встречаться повторяющихся групп, нескольких значений для каждого экземпляра (например, номер телефона_1, номер телефона_2). Видим, что атрибут *Авторы* в сущности *Методические указания* не удовлетворяет требованиям 1 НФ (у методических указаний может быть несколько авторов). Необходимо выделить сущность, которая будет содержать сведения об авторах методических указаний. Поскольку авторами всегда являются преподаватели вузов, новую сущность выделять не имеет смысла, свяжем сущности *Методические указания* и *Преподаватель*, предварительно удалив атрибут *Авторы*. Остальные атрибуты соответствуют 1 НФ. Атрибутивная модель, приведенная к 1 НФ.

2. Приведем модель ко 2 НФ. Проверим, все ли атрибуты зависят от составного ключа, а не от его части. Проверка показала, что все не ключевые атрибуты сущностей полностью зависят от составного ключа. Значит, модель удовлетворяет требованиям 2 НФ.

3. Проверим, есть ли транзитивная зависимость между не ключевыми атрибутами. Проверка показала, что взаимозависимости между не ключевыми атрибутами нет. Таким образом, модель, представленная на рисунке 13, приведена к 3 НФ.

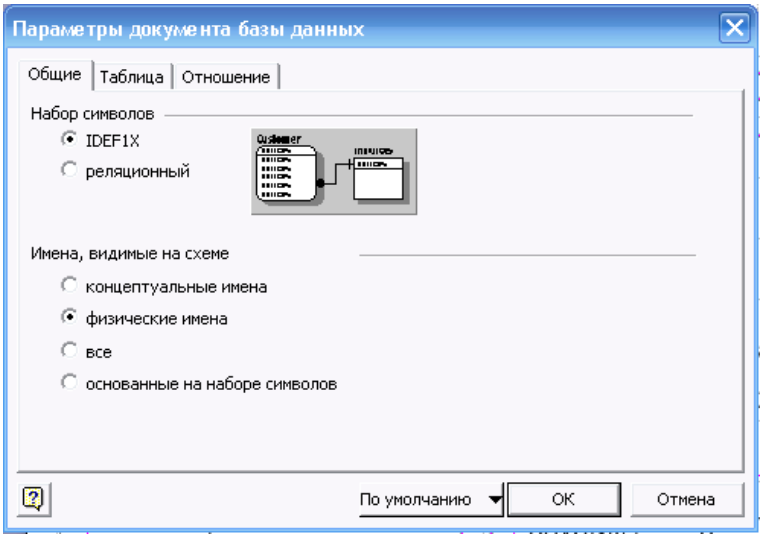
Примечание. К нормализации относились также действия, выполненные в п. 2 упражнения 2.

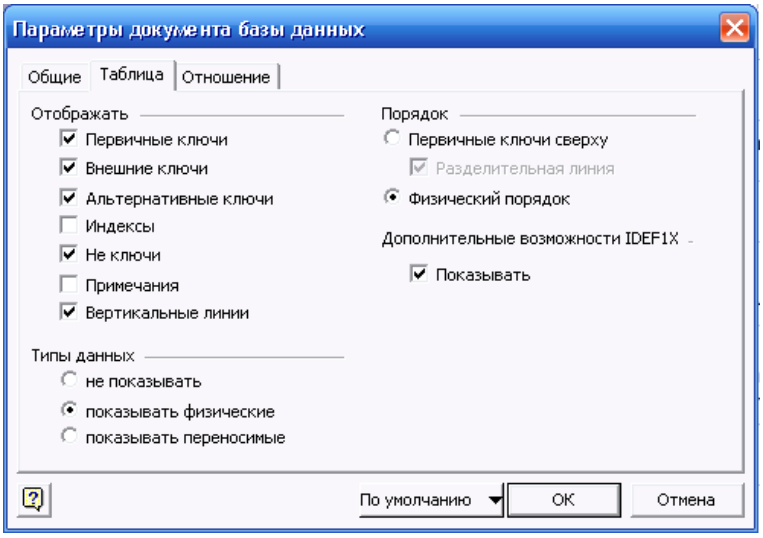


Упражнение 5. Создание физической модели

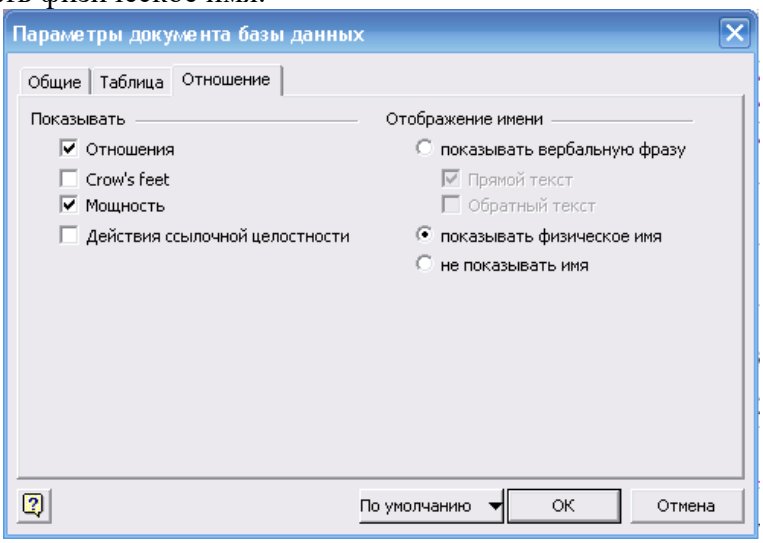
1. Необходимо переключиться на физический уровень представления информационной модели. Для этого нужно выбрать пункты меню *База данных* → *Параметры* → *Документ*. В открывшемся окне на вкладке *Общие* установить переключатель в меню *Имена, видимые на схеме*. В данном случае для физического представления информационной модели необходимо выбрать пункт *Физические имена*.

2. В закладке *Таблица* окна *Параметры документа базы данных* в меню *Отображать* выбрать пункт *Вертикальные линии,* в меню *Типы данных – Показывать физические* и в меню *Порядок – Физический порядок*.

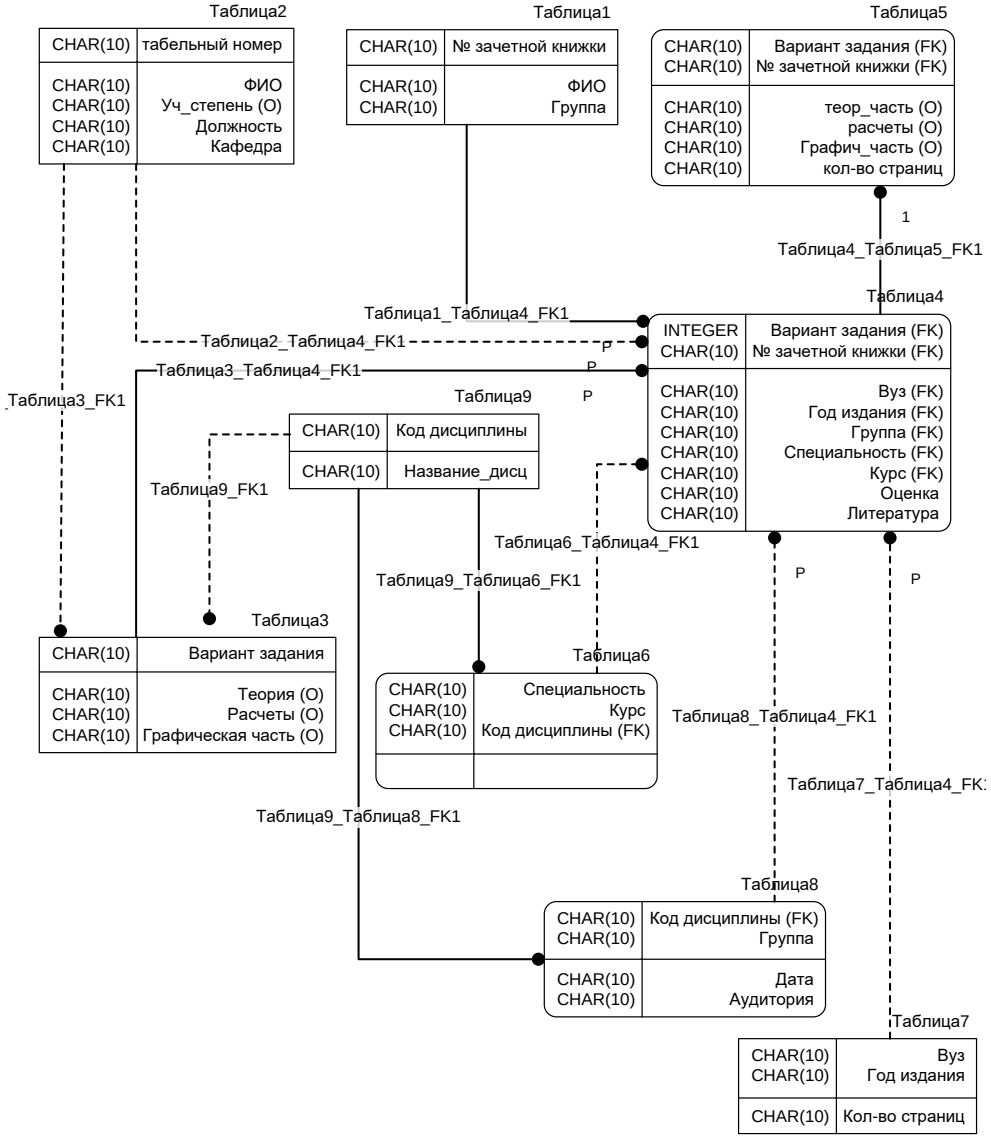




3. В закладке Отношение окна Параметры документа базы данных в меню Отображение вида выбрать пункт Показывать физическое имя.



4. Для каждого атрибута (поля) необходимо определить тип данных.



Типы данных можно представить в виде правил, ограничивающих вид сведений, которые могут быть введены в каждый столбец таблицы базы данных. Например, чтобы в поле, которое предназначено только для дат, нельзя было ввести имя, этому полю назначается тип данных «Дата».

Примечание *(Выбор между переносимыми и физическими типами данных).*

Переносимые типы данных — это обобщенные типы данных, соответствующие в разных системах баз данных простым, совместимым между собой физическим типам.

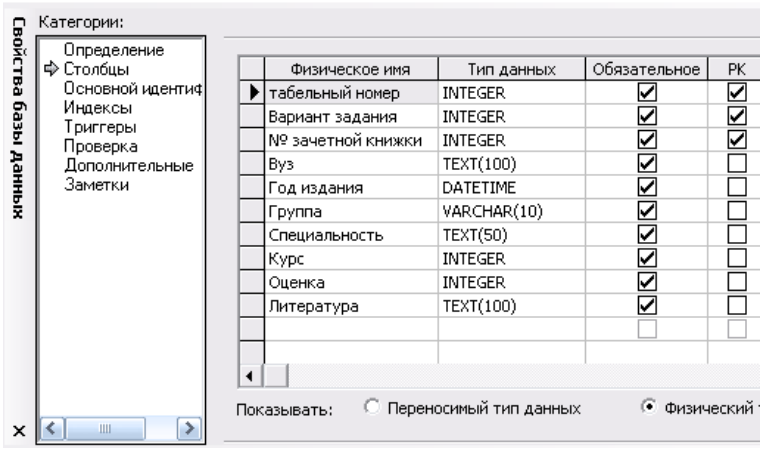
Физические типы данных — это типы данных, поддерживаемые целевой базой данных.

Щелкните сущность, содержащую атрибуты, для которых требуется установить типы данных.

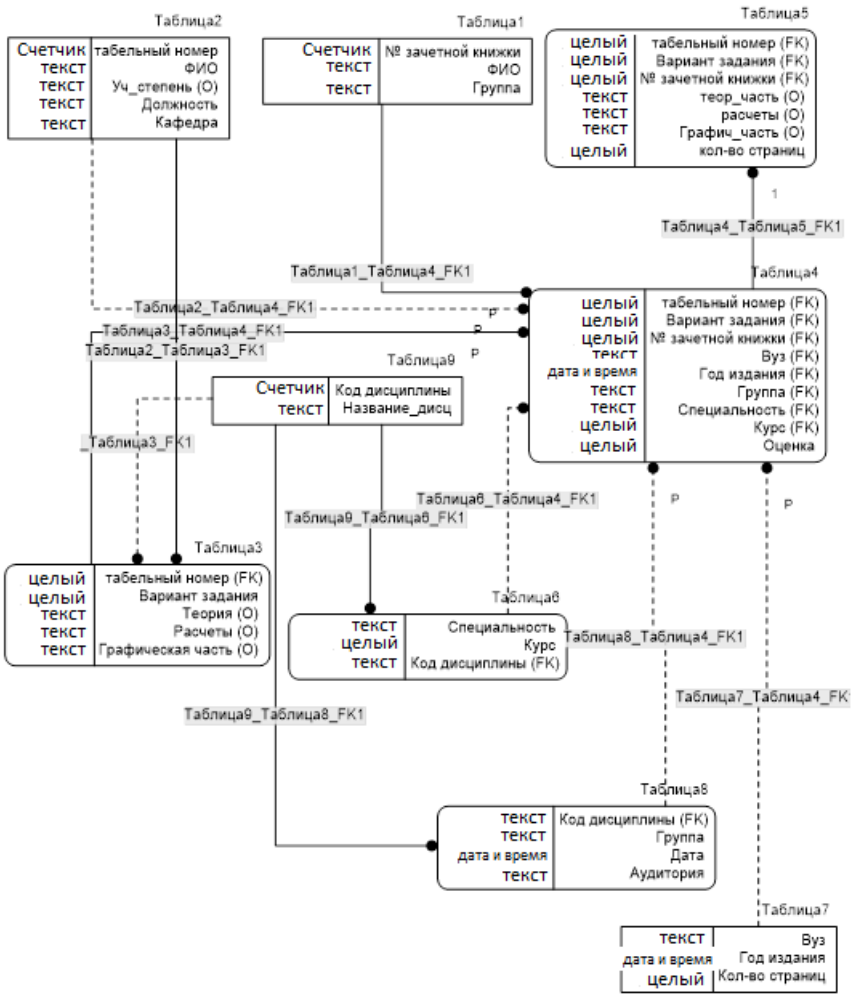
В окне *Свойства базы данных* в списке *Категории* выберите вариант *Столбцы*.

Под списком столбцов установите переключатель в положение *Физический тип данных*.

В группе *Тип данных* для каждого атрибута выберите необходимый вариант из множества альтернатив. Описание типов данных приведено в *Приложении А*.



Таким образом, проделав все вышеперечисленные действия, получим информационную модель физического уровня, на основе которой может быть сгенерирована схема БД (в нашем случае в MS Office Access).



5 Задание

В соответствии с вариантом задания, определенным преподавателем, последовательно выполнить следующие действия:

1) создать информационную модель логического уровня (выполнить упражнения 1-3). Минимальное количество сущностей – 4;

2) провести нормализацию полученной модели (упражнение 4);

3) на основе нормализованной логической модели построить информационную модель физического уровня (упражнение 5).

6 Варианты

1. Проектирование ИС «Отдел кадров»;

2. Проектирование ИС «Агентство аренды»;
3. Проектирование ИС «Аптека»;
4. Проектирование ИС «Ателье»;
5. Проектирование ИС «Аэропорт»;
6. Проектирование ИС «Библиотека»;
7. Проектирование ИС «Кинотеатр»;
8. Проектирование ИС «Поликлиника»;
9. Проектирование ИС «Автосалон»;
10. Проектирование ИС «Таксопарк»;
11. Проектирование ИС «Издательство»;
12. Проектирование ИС «Прокат велосипедов»;
13. Проектирование ИС «Спортивный клуб».

7 Порядок выполнения работы

Для выполнения работы необходимо:

а) повторить правила техники безопасности при работе с вычислительной техникой;

б) изучить соответствующий раздел лекционного курса, а также теоретическую часть настоящего методического указания;

в) выполнить лабораторную работу согласно описанной в пункте 5 методике в соответствии с вариантом задания;

г) в соответствии с требованиями, приведенными в разделе 8 практикума, оформить отчет по лабораторной работе;

е) защитить лабораторную работу в соответствии с требованиями преподавателя.

8 Требования к содержанию и оформлению отчета

Отчет должен содержать:

1) титульный лист;
2) название практического занятия, цель;
3) пул – список потенциальных сущностей;
4) нормализованную информационную модель логического уровня;
5) информационную модель физического уровня;
6) выводы по проделанной работе.

Отчет должен быть представлен в бумажном виде.

9 Контрольные вопросы

1. Какие диаграммы позволяет строить нотация IDEF1X?
2. Для чего предназначена диаграмма «сущность-связь»?
3. Чем отличается полная атрибутивная модель от диаграммы «сущность-связь»?
4. Какие виды отношений существуют и чем они отличаются?
5. Что представляет собой нормализация?
6. В чем разница между логическим уровнем модели данных и физическим?

ПРИЛОЖЕНИЕ А

Типы данных Microsoft Office Access

1. Символьные типы

Символьные типы используются для представления как строк символов, так и отдельных символов.

Таблица Б.1 – Перечень символьных типов

Тип данных	**Назначение**	**Размер**
CHAR	Строковый тип	до 32767 байт. по умолчанию 1 байт
VARCHAR	Тоже, что и CHAR	
LONG VARCHAR	Символьный тип произвольной длины. Аналог MEMO-полям в dBase, FoxPro, Access	Длина произвольная. Ограничена максимальным размером файлов базы данных (2 гигобайта)
TEXT	Тоже, что и LONG VARCHAR	

2. Числовые типы

Числовые типы предназначены для обозначения целых, вещественных и денежных типов.

Таблица Б.2 – Перечень числовых типов

Тип данных	**Диапазон значений**	**Точность - число знаков после запятой**	**Размер**
INTEGER	от -2 147 483 648 до +2 147 483 647	0	4 байта
SMALLINT	от -32 768 до +32 767	0	2 байта
REAL	от -3.4 e-38 до 3.4 e+38	до 6	4 байта
DOUBLE	от -1.797 e-308 до +1.797 e+308	до 15	8 байт
DECIMAL	числа, состоящие из N цифр с M цифрами в дробной части. По умолчанию N=30, M=6	M	сколько требуется
NUMERIC	Тоже, что и DECIMAL		

3. Типы дата/время

Типы дата/время предназначены для хранения времени, дат и дат совместно с временем.

Таблица Б.3 – Форматы представления данных типа дата/время, определяемые по умолчанию

Тип данных	**Формат, используемый по умолчанию**
DATETIME	'YYYY-MM-DD HH:NN:ss.SSS'

- **YYYY** – четыре цифры, обозначающие год:
- **MM** – две цифры, обозначающие месяц:
- **DD** – две цифры, обозначающие день:
- **HH** – две цифры, обозначающие часы:
- **NN** – две цифры, обозначающие минуты:
- **ss** – две цифры, обозначающие секунды:
- **SSS** – три цифры, обозначающие доли секунд.

По умолчанию составляющие времени HH, NN, ss, SSS принимаются равными нулю, а DD - единице.

4. Двоичные типы

Двоичные типы предназначены для представления двоичных данных, включая изображения и другую информацию, не обрабатываемую собственными средствами СУБД.

Таблица Б4 -

Таблица Б.4 – Двоичные типы

Тип данных	**Назначение**	**Размер**
BIT	Тип для представления значений 0 и 1. Аналог полей типа Logical в dBase, FoxPro	1 байт
BINARY	Тоже, что и CHAR, за исключением операций сравнения. В отличии от CHAR, данные этогопо умолчанию 1 байт типа сравниваются на полное совпадение двоичных кодов байтов	до 32767 байт
LONG	Тип для представления двоичных данных произвольной длины	Длина произвольная. Ограничена максимальным размером файлов базы данных (2 гиго- байта)

Практическая работа №4
Функциональное моделирование предметной области в нотации IDEF0 с помощью Ramus Educational

Цель: получение навыков создания и редактирования функциональных моделей в нотации IDEF0.

1 Задачи

Основными задачами практической работы являются:
– приобретение студентами навыков построения функциональной модели;
– приобретение студентами навыков редактирования функциональной модели.

2 Краткие теоретические сведения

1. Основные сведения по методологии IDEF0

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Цель моделирования. Модель не может быть построена без четко сформулированной цели. Пример цели: «Описать функциональность предприятия с целью написания спецификаций ИС».

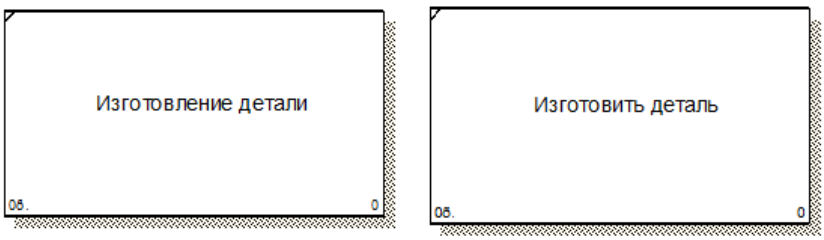
Точка зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Цель и точка зрения документируются.

Основные элементы IDEF0-модели

В основе методологии IDEF0 лежат 4 основных понятия:
– функциональный блок;
– интерфейсная дуга (стрелка);
– декомпозиция;
– глоссарий.

1. Функциональный блок

Функциональные блоки обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Графически функциональные блоки изображаются в виде прямоугольников. Все блоки должны быть названы и определены. Имя функционального блока должно быть выражено сочетанием отглагольного существительного, обозначающего процесс, или глаголом:



а) б)

Определение функционального блока заносится в глоссарий или словарь работ (Activity Dictionary).

Все функциональные блоки модели нумеруются. Номер состоит из префикса и числа. Может использоваться префикс любой длины, но обычно используется префикс А. Контекстная (корневая) работа (функциональный блок) имеет номер А0.

2. Интерфейсная дуга (стрелка – Arrow)

Взаимодействие функциональных блоков с внешним миром и между собой описывается в виде интерфейсных дуг (стрелок). Стрелки представляют собой некую информацию и обозначаются существительными (например, «Заготовка», «Изделие») или именуемыми сочетаниями (например,

«Готовое изделие»). Все стрелки должны быть определены. Определения заносятся в словарь стрелок – глоссарий (Arrow Dictionary).

В IDEF0 различают 4 типа стрелок.

Каждая стрелка имеет свое расположение относительно функционального блока.

Вход (Input) – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Стрелка Input рисуется входящей в левую грань работы.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Рисуется как входящая в верхнюю грань работы.

Выход (Output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Изображается исходящей из правой грани работы.

Механизм (Mechanism) – ресурсы, которые выполняют работу, например, персонал предприятия, станки, устройства и т.д. Рисуется как входящая в нижнюю грань работы.

3. Глоссарий

Набор определений, ключевых слов и т.д., которые характеризуют каждый объект модели.

4. Декомпозиция

Разбиение системы на крупные фрагменты – функции, функции – на подфункции и т.д. до конкретных процедур.

Модель может содержать 4 типа диаграмм:

- контекстную (в каждой модели может быть только 1 контекстная диаграмма);
- декомпозиции;
- дерева узлов;
- только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой общее описание системы и ее взаимодействия с внешней средой.

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов – диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и т.д., до достижения нужного уровня подробности описания.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами.

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

Все диаграммы имеют нумерацию. Контекстная диаграмма имеет номер A-0, декомпозиция контекстной диаграммы – номер A), остальные диаграммы-декомпозиции – номера по соответствующему узлу (например, A1, A2, A21 и т.д.).

3 Методика выполнения практического занятия

В качестве примера рассматривается процесс выполнения студентом курсовой работы (курсового проекта).

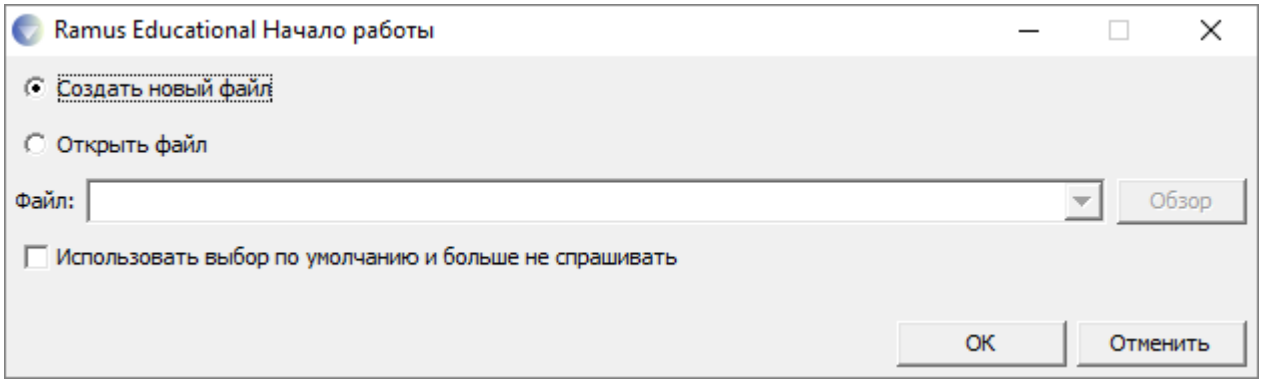
Программное обеспечение «Ramus» предназначено для использования в проектах, в которых необходимо описание бизнес-процессов предприятия. «Ramus» поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD, а также имеет ряд дополнительных возможностей, призванных удовлетворить потребности команд разработчиков систем управления предприятиями.

«Ramus» обладает гибкими возможностями построения отчетности по графическим моделям, позволяющие создавать отчеты в форме документов, регламентирующих деятельность предприятия.

Ramus Educational имеет достаточно интуитивный интерфейс пользователя, позволяющий быстро и просто создавать сложные модели.

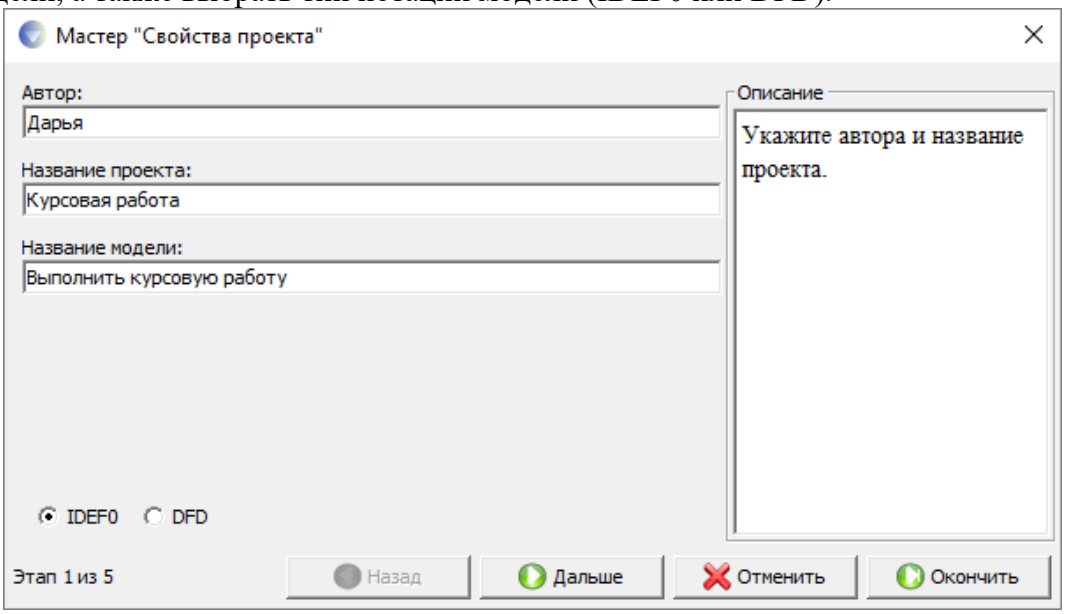
1 Начало работы

1. Запустите программу Ramus Educational. В появившемся окне предлагается создать новый проект или открыть уже существующий.



2. После нажатия на кнопку «ОК» осуществляется запуск мастера проекта.

– На первом шаге (рис. 5) в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD).



– На втором шаге вводится название организации, использующей данный проект.

– На третьем – дается краткое описание будущего проекта.

– Четвертый шаг позволяет создать несколько основных классификаторов (в данном случае можно пропустить этот шаг). Так как модели процессов реальных предприятий могут содержать значительное количество объектов (документы, персонал, функции и т.д.), то в Ramus предусмотрена возможность упорядочено хранить информацию об этих объектах в виде системы классификаторов. Классификация

объектов упрощает поиск и обработку информации об объектах модели, а также и об объектах непосредственно не представленных на диаграммах процессов, но относящихся к процессам предприятия.

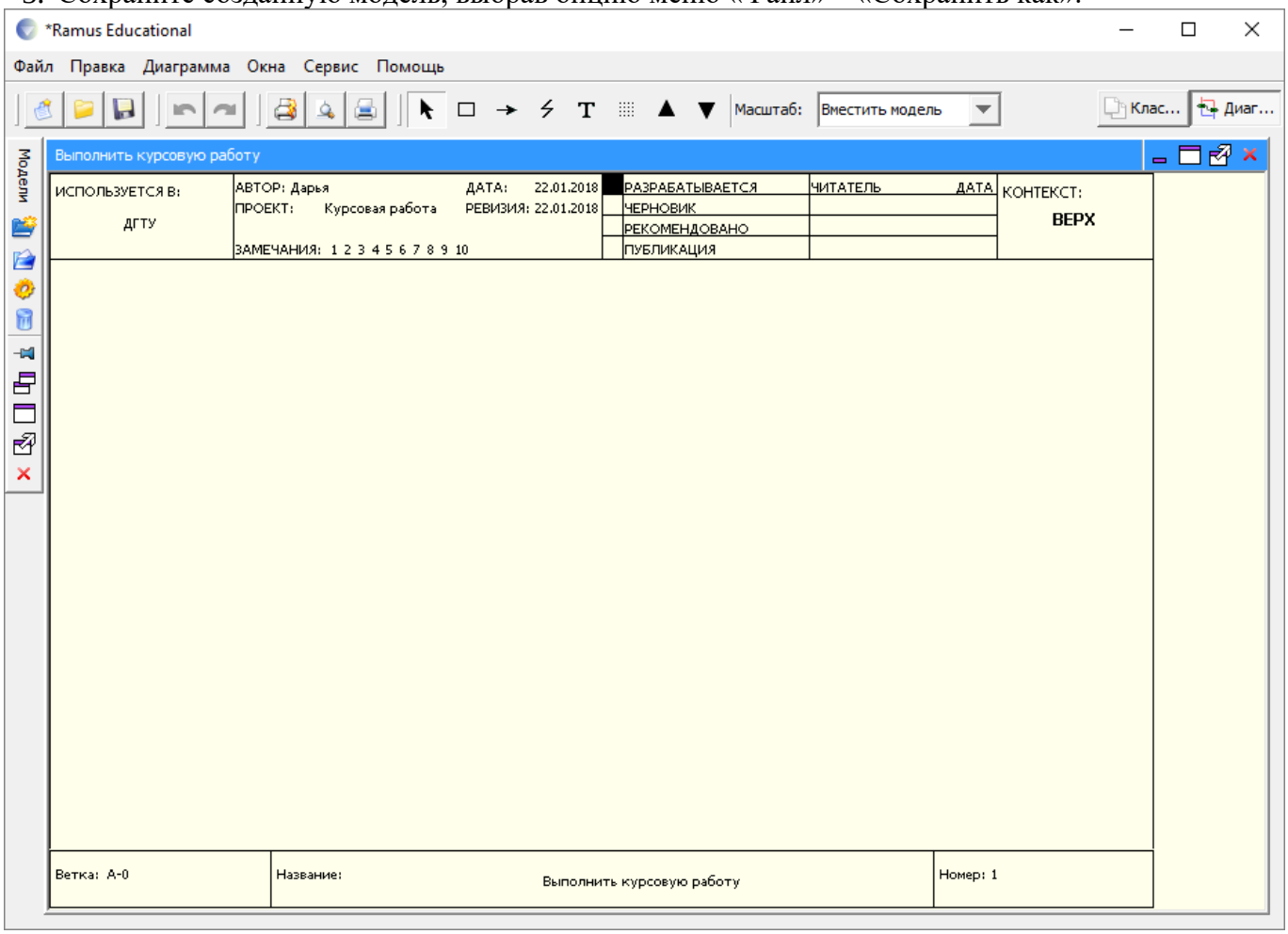
– На пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (пропустить данный шаг).

При необходимости можно завершить работу мастера, нажав кнопку «Окончить».

После завершения работы мастера, откроется рабочее пространство «Диаграммы», в котором можно приступить к рисованию графической модели (рис. 6). В верхней части приводятся сведения о проекте, введенные пользователем посредством мастера диаграмм.

Программа Ramus Educational обладает гибким графическим интерфейсом, который можно настроить под нужды и предпочтения конкретного пользователя: ненужные окна можно закрыть/свернуть; можно менять их размеры и месторасположение; также можно группировать два и более окон в одном, при этом содержимое вложенных окон будет размещено на вкладках общего окна (данный функционал возможен не для всех комбинаций окон).

3. Сохраните созданную модель, выбрав опцию меню «Файл» – «Сохранить как».



2 Создание контекстной диаграммы

1. На панели инструментов выберите пиктограмму функции ( ) и мышью укажите месторасположение на рабочем пространстве.

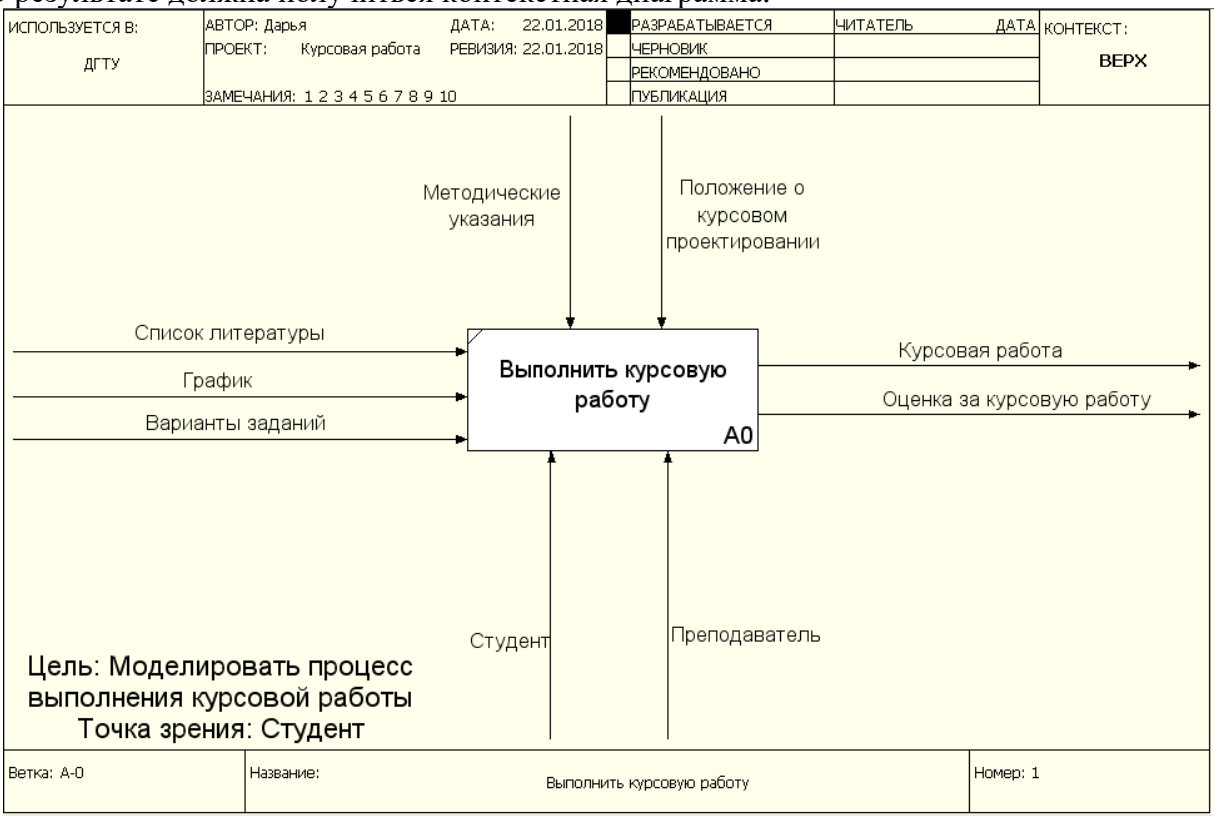
1. Дайте данному функциональному блоку имя «Выполнить курсовую работу». Для этого дважды щелкните внутри блока.

2. Используя пиктограмму панели инструментов , создайте стрелки на контекстной диаграмме согласно Таблица 1.

Таблица 5 – Контекстная диаграмма

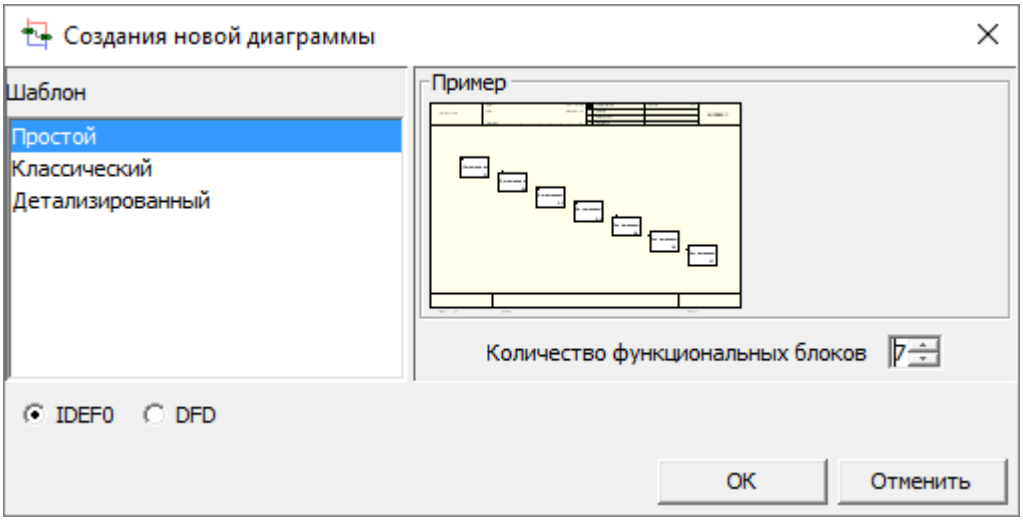
Имя стрелки (Arrow Name)	**Определение стрелки (Arrow Definition)**	**Тип стрелки (Arrow Type)**
График	График консультаций и сроки сдачи	Input
Список литературы	Источники информации для выполнения курсовой работы	Input
Варианты заданий	Список заданий на курсовую работу, подлежащий распределению между студентами	Input
Методические указания	Документ, содержащий указания по выполнению курсовой работы, описывающий содержание ее частей и основные требования	Control
Положение о курсовом проектировании	Документ, отражающий организационные требования по выполнению и сдаче курсовой работы	Control
Курсовая работа	Документ, являющийся основанием для получения оценки	Output
Оценка за курсовую работу	Результат выполнения курсовой работы	Output
Студент	Тот, кто выполняет курсовую работу	Mechanism

3. В результате должна получиться контекстная диаграмма.

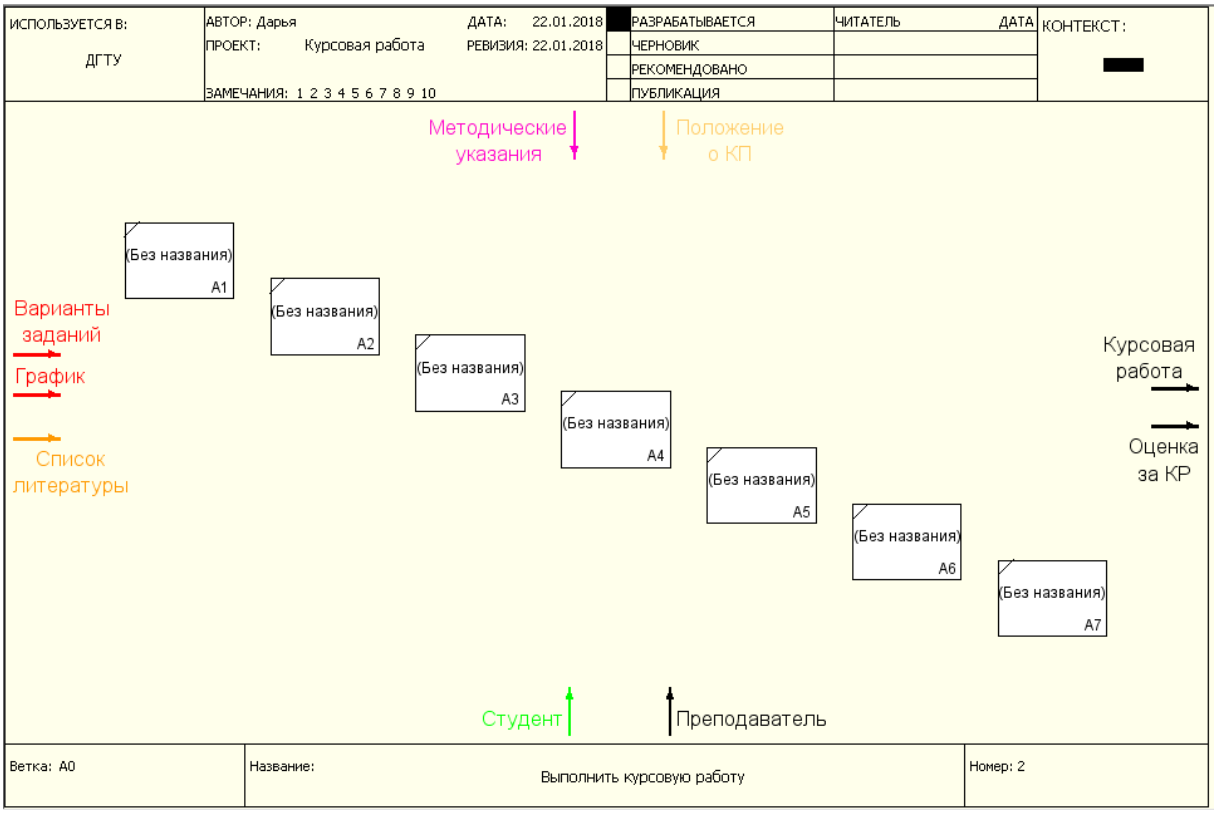


3 Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне «Создание новой диаграммы» установите количество функциональных блоков 7, укажите тип диаграммы (IDEF0) и нажмите кнопку ОК.

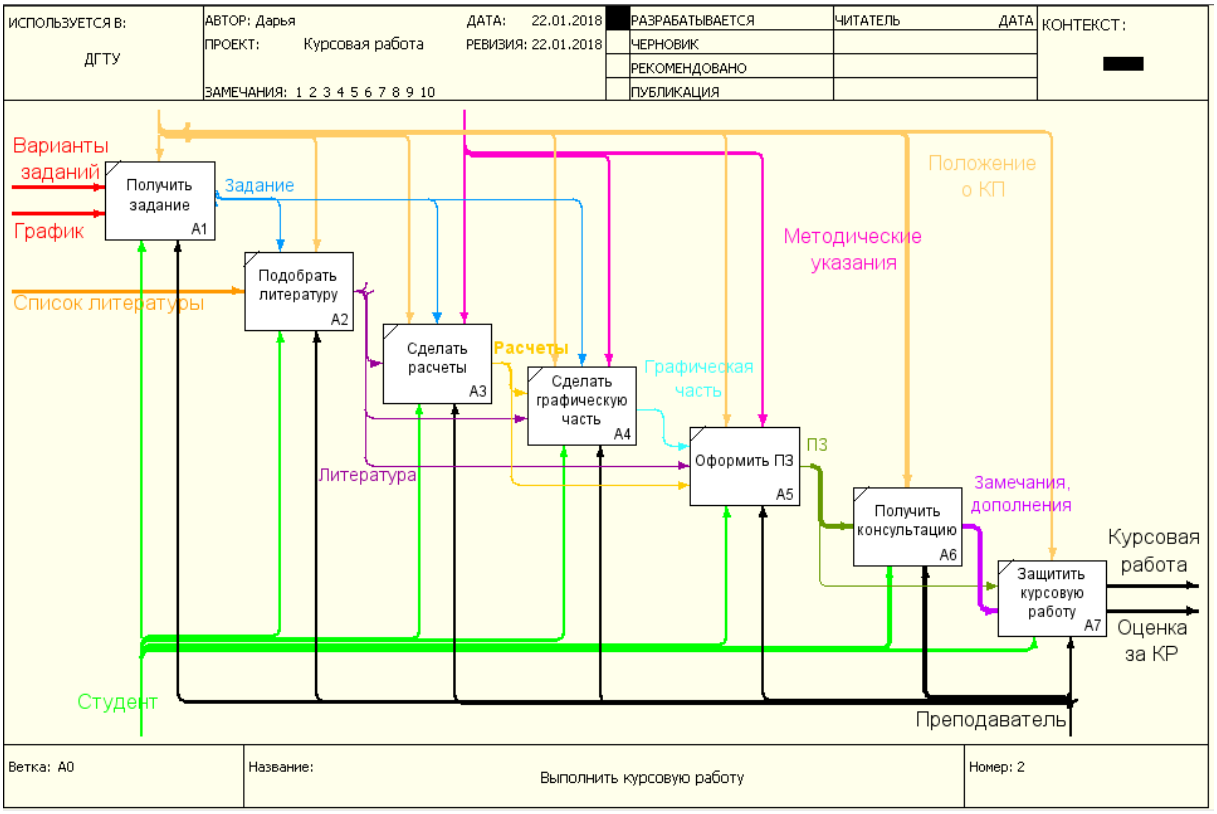


2. Автоматически будет создана диаграмма первого уровня декомпозиции с перенесенными в нее потоками родительской диаграммы.



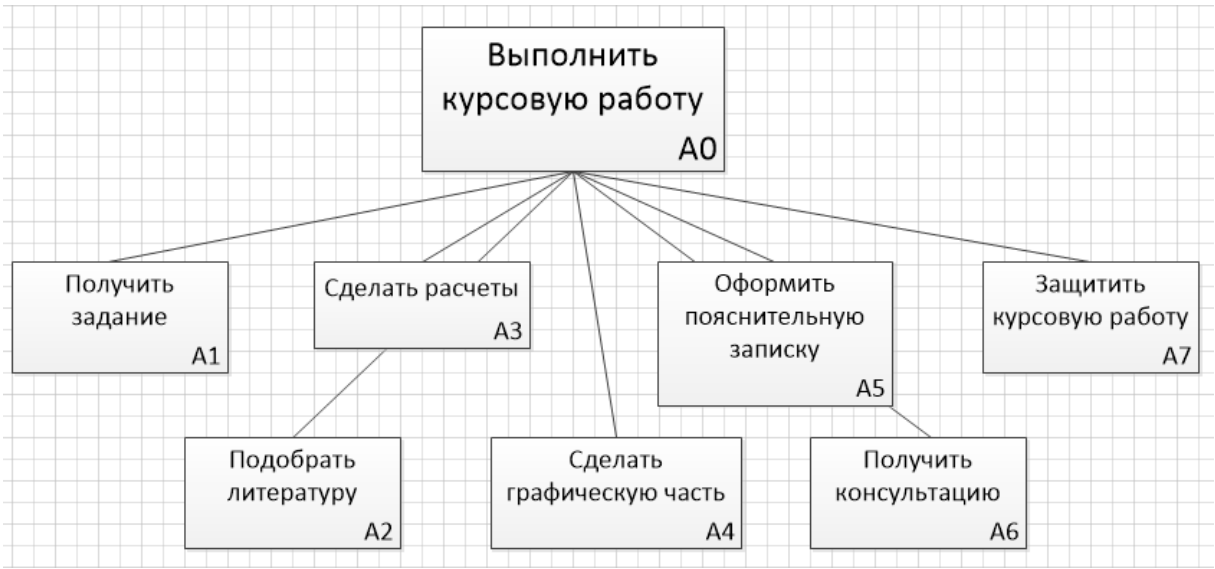
3. Выделите первую работу (функциональный блок), затем двойным щелчком мыши или, выбрав в контекстном меню пункт «Редактировать активный элемент», откройте окно свойств и внесите имя работы. Повторите операцию для оставшихся работ.

4. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом (через вход, управление, механизм или выход) с соответствующим функциональным блоком. В результате должна получиться детализирующая диаграмма.



3. Создание дерева узлов

Дерево узлов – это диаграмма, отображающая иерархию работ процесса. Ее необходимо построить с помощью Visio.



Для построения диаграммы:

– создайте новую страницу;
– присвойте имя странице: дерево узлов;
– постройте дерево узлов, используя фигуры схемы IDEF0.

4. Создание глоссария

Глоссарий – это словарь ключевых слов, повествований, изложений, используемых при описании процесса.

Для построения глоссария:

– создайте документ Microsoft Office Word;
– создайте 2 таблицы: описание работ процесса, описание интерфейсных дуг процесса;
– наименование столбцов таблиц: имя (работы/дуги, описание);
– заполните таблицы в соответствии с ранее разработанной моделью процесса.

Name	Definition
Выполнить курсовую работу	Текущие процессы выполнения курсовой работы
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Подобрать литературу	Выбрать из списка литературы подходящие
Получить задание	Выбрать задание из списка, согласовать его с
Получить консультацию	Получить консультацию у преподавателя перед
Сделать графическую часть	При необходимости сделать графики и чертежи
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой

Name	Definition
Варианты заданий	Список заданий на курсовую работу, подлежа
График	График консультаций и сроки сдачи
Графическая часть	Выполненная графическая часть курсовой раб
Задание	Выдается на консультации преподавателем, чт
Замечания, дополнения	Замечания преподавателя, полученные на кон
Курсовая работа	Документ, являющийся основанием для получ
Литература	Выбранные источники, необходимые для выпо
Методические указания	Документ, содержащий указания по выполнен
Оценка за курсовую раб	Результат выполнения курсовой работы
Положение о курсовом п	Документ, отражающий организационные треб
Пояснительная записка	Теоретическая часть + расчеты + графическая
Преподаватель	Тот, кто оценивает курсовую работу
Расчеты	Выполненная расчетная часть курсовой работ
Список литературы	Источники информации для выполнения курсо
Студент	Тот, кто выполняет курсовую работу

4 Задание

На основе информационной модели предметной области, разработанной в практическом занятии №1, составить функциональную модель в нотации IDEF0.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

5 Варианты

1. Проектирование ИС «Отдел кадров»;
2. Проектирование ИС «Агентство аренды»;
3. Проектирование ИС «Аптека»;
4. Проектирование ИС «Ателье»;
5. Проектирование ИС «Аэропорт»;
6. Проектирование ИС «Библиотека»;
7. Проектирование ИС «Кинотеатр»;
8. Проектирование ИС «Поликлиника»;
9. Проектирование ИС «Автосалон»;
10. Проектирование ИС «Таксопарк»;
11. Проектирование ИС «Издательство»;
12. Проектирование ИС «Прокат велосипедов»;
13. Проектирование ИС «Спортивный клуб».

6 Требования к построению модели

1. На контекстной диаграмме необходимо указать точку зрения и цель моделирования.
2. Количество блоков любой декомпозиции не менее 3-х и не более 9.

3. Количество декомпозиций – 3 уровня декомпозиции.

7 Контрольные вопросы

1. Каковы цели функционального моделирования?
2. Назовите основные компоненты функциональной модели.
3. Какие виды интерфейсных дуг различают в IDEF0?
4. Для чего нужна цель и точка зрения?
5. Что такое функциональный блок?
6. Какие виды диаграмм может содержать функциональная модель?

Практическая работа №5
Моделирование потоков данных DFD с помощью Ramus Educational

Цель: изучение основных характеристик и основ работы с DFD-моделями в графическом редакторе.

1 Задачи

Основными задачами практической работы являются:

– изучение состава диаграмм DFD, назначения элементов каждого вида и способов их размещения на диаграмме;

– изучение операций по созданию DFD-модели.

2 Краткие теоретические сведения

1. Модель потоков данных

DFD – data flow diagrams – диаграммы потоков данных – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

Диаграмма потоков данных – один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML.

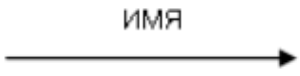
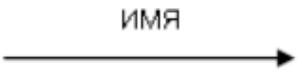
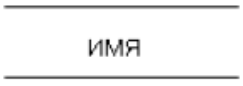
Для описания диаграмм DFD используются две нотации – Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson), отличающиеся синтаксисом.

2. Основные элементы информационной модели логического уровня

Согласно DFD источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации.

При построении диаграмм различают элементы графической нотации, представленные в табл. 1.

Таблица 6 – Элементы графической нотации DFD

Наименование	Нотация Йордана	Нотация Гейна-Сарсона
Поток данных		
Процесс (система, подсистема)		
Накопитель данных		
Внешняя сущность		

Поток данных определяет информацию (материальный объект), передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т. д.

Каждый поток данных имеет имя, отражающее его содержание. Направление стрелки показывает направление потока данных. Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним – двунаправленным.

На диаграммах IDEF0 потоки данных соответствуют входам и выходам, но в отличие от IDEF0 стрелки потоков на DFD могут отображаться входящими и выходящими из любой грани внешней сущности, процесса или накопителя данных.

Процесс (в IDEF0 – функция, работа) представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.

Каждый процесс должен иметь имя в виде предложения с глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

– «Ввести сведения о клиентах»;
– «Рассчитать допускаемую скорость»;
– «Сформировать ведомость допускаемых скоростей».

Номер процесса служит для его идентификации и ставится с учетом декомпозиции. Вложенность процессов обозначается через точку.

Преобразование информации может показываться как с точки зрения процессов, так и с точки зрения систем и подсистем. Если вместо имени процесса «Рассчитать допускаемую скорость» написать «Подсистема расчета допускаемых скоростей», тогда этот блок на диаграмме стоит рассматривать, как подсистему.

Накопитель (хранилище) данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде ящика в картотеке, области в оперативной памяти, файла на магнитном носителе и т.д.

Накопителю обязательно должно даваться уникальное имя и номер в пределах всей модели (всего набора диаграмм). Имя накопителя выбирается из соображения наибольшей информативности для разработчика. Например, если в качестве накопителей выступают таблицы проектируемой базы данных, тогда в качестве имен накопителей рекомендуется использовать имена таблиц. Таким образом, накопитель данных может представлять собой всю базу данных целиком, совокупность таблиц или отдельную таблицу. Такое представление накопителей в дальнейшем облегчит построение информационной модели системы.

Внешняя сущность (терминатор) представляет собой материальный объект или физическое лицо, выступающие как источник или приемник информации (например, заказчики, персонал, программа, склад, инструкция). Внешние сущности на DFD по смыслу соответствуют управлению и механизмам, отображаемым на контекстной диаграмме IDEF0.

Определение некоторого объекта, субъекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ проектируемой информационной системы. В связи с этим внешние сущности, как правило, отображаются только на контекстной диаграмме DFD. В процессе анализа и проектирования некоторые внешние сущности могут быть перенесены на диаграммы декомпозиции, если это необходимо, или, наоборот, часть процессов (подсистем) может быть представлена как внешняя сущность.

3 Рекомендации по выполнению

На основе модели предметной области, разработанной в практических занятиях №1 и 2, составить функциональную модель в нотации DFD.

Отчет по практическому занятию выполняются в формате MS Word, который содержит экранные формы моделей согласно заданию.

4 Методика выполнения практического занятия

В качестве примера рассматривается процесс выполнения студентом курсовой работы (курсового проекта).

Программное обеспечение «Ramus» предназначено для использования в проектах, в которых необходимо описание бизнес-процессов предприятия. «Ramus» поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD, а также имеет ряд дополнительных возможностей, призванных удовлетворить потребности команд разработчиков систем управления предприятиями.

«Ramus» обладает гибкими возможностями построения отчетности по графическим моделям, позволяющие создавать отчеты в форме документов, регламентирующих деятельность предприятия.

Ramus Educational имеет достаточно интуитивный интерфейс пользователя, позволяющий быстро и просто создавать сложные модели.

1 Начало работы

1. Запустите программу Ramus Educational. В появившемся окне (рис. 1) предлагается создать новый проект или открыть уже существующий.

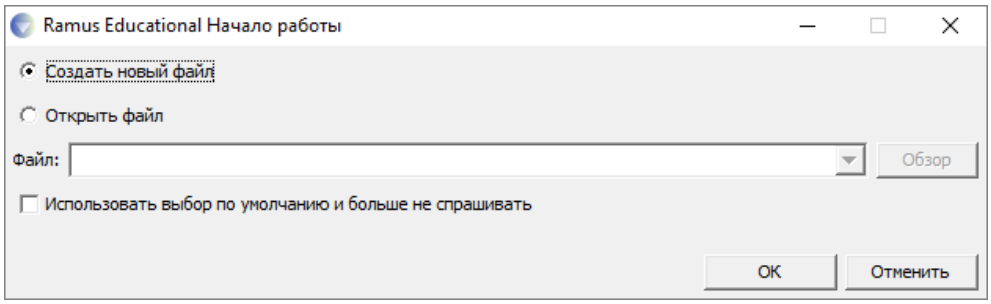


Рисунок 1 – Окно запуска Ramus Educational

2. После нажатия на кнопку «ОК» осуществляется запуск мастера проекта.

– На первом шаге (рис. 2) в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD) – в данном случае – DFD.

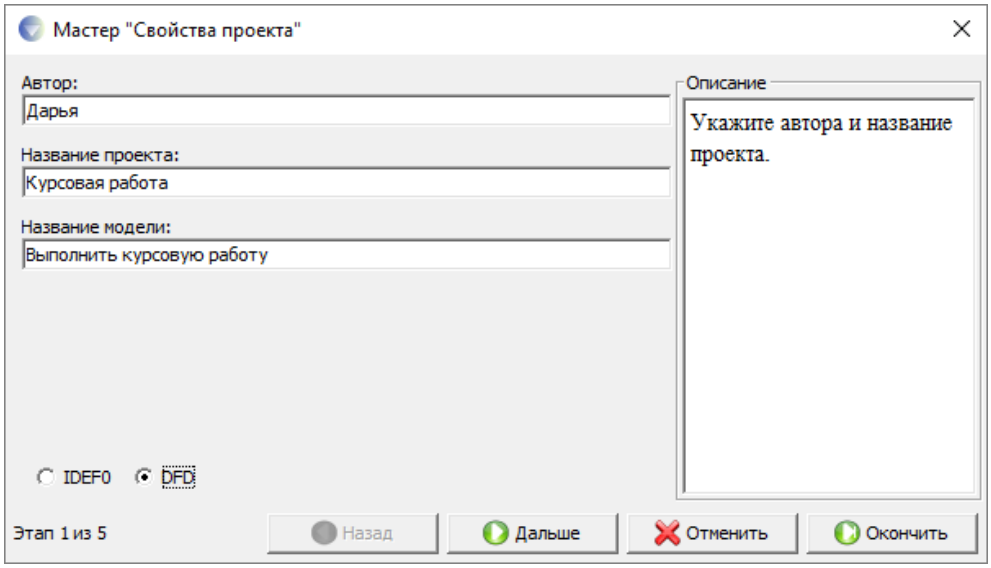


Рисунок 2 – Окно мастера создания проекта. Шаг 1

– На втором шаге вводится название организации, использующей данный проект.
– На третьем – дается краткое описание будущего проекта.
– Четвертый шаг позволяет создать несколько основных классификаторов (рисунок 3).

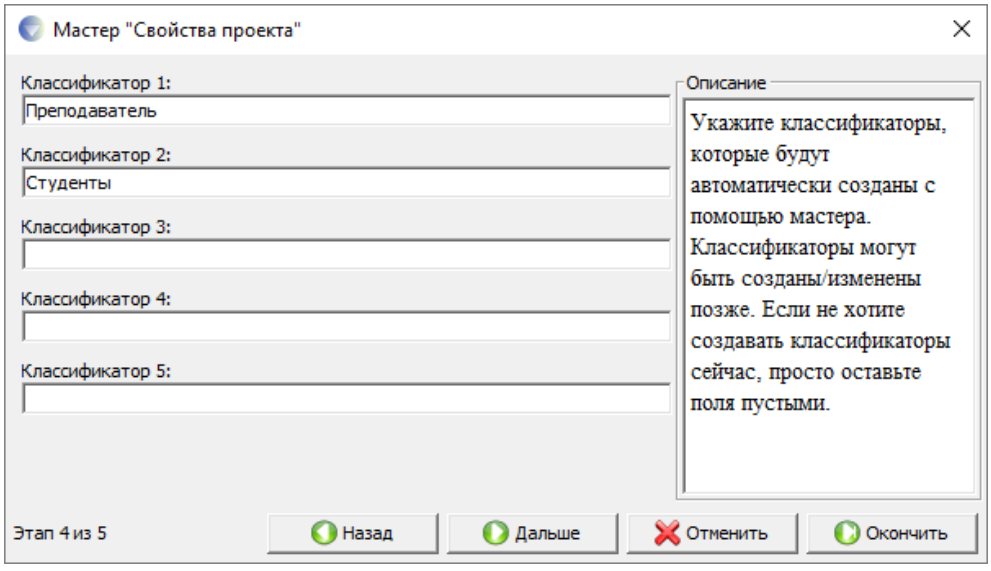


Рисунок 3 – Окно мастера создания проекта. Шаг 4

– На пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (рис. 4).

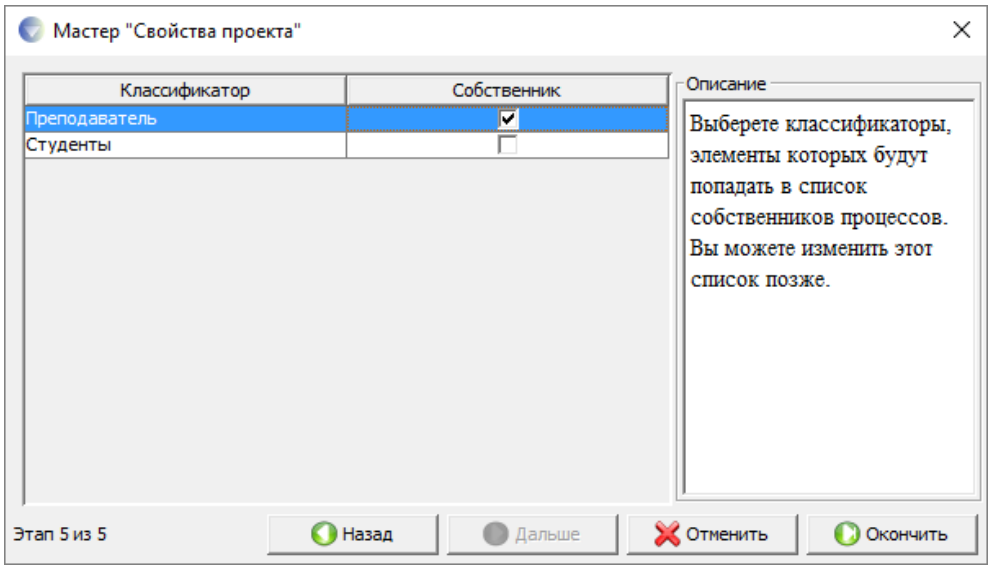


Рисунок 4 – Окно мастера создания проекта. Шаг 5

При необходимости можно завершить работу мастера, нажав кнопку «Окончить».

После завершения работы мастера, откроется рабочее пространство «Диаграммы», в котором можно приступить к построению графической модели. На панели инструментов, в верхней части окна рабочего пространства программы, содержатся элементы диаграммы потоков данных в нотации Gane-Sarson.

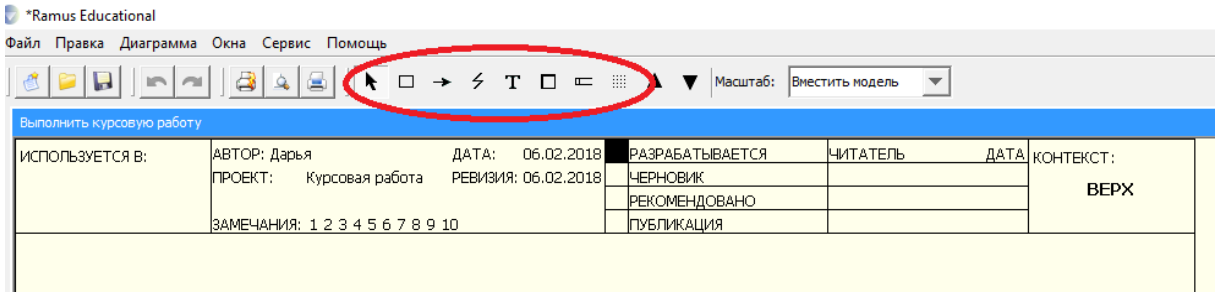


Рисунок 5 – Элементы DFD в нотации Гейна-Сарсона

3. Сохраните созданную модель, выбрав опцию меню «Файл» – «Сохранить как».

2 Создание контекстной диаграммы

2. На панели инструментов выберите инструмент создания процесса ( ) и мышью укажите месторасположение на рабочем пространстве нового процесса.

3. Выделив процесс, выберите в контекстном меню опцию **«Редактировать активный элемент»**. В появившемся диалоговом окне на вкладке **«Название»** присвойте процессу имя «*Выполнить курсовую работу*»; на вкладке «**Тип функционального блока**» укажите тип элемента – «Процесс» (Рис. 6).

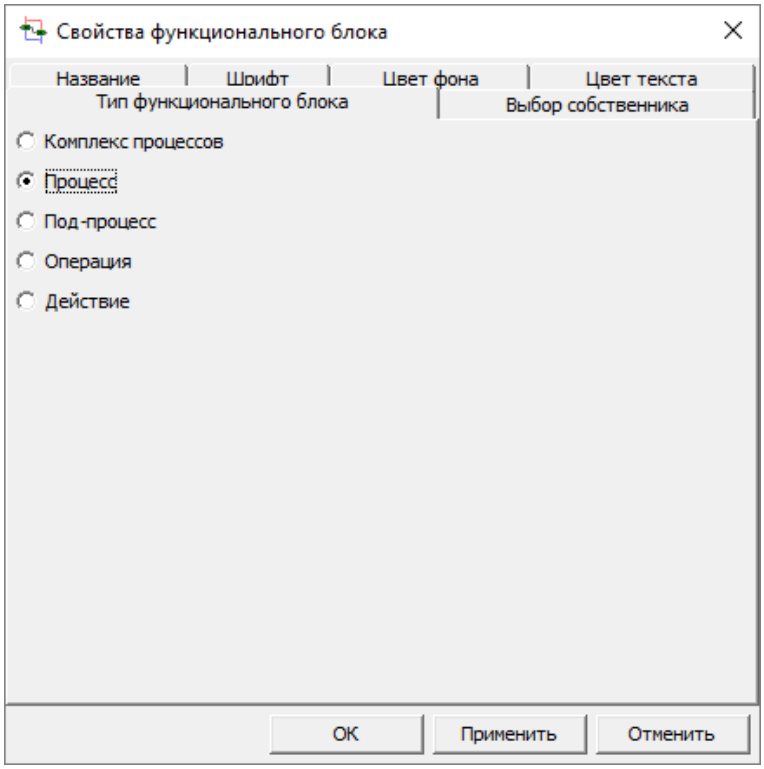


Рисунок 6 – Окно «Свойства функционального блока

4. На панели инструментов выберите инструмент создания внешней сущности ☐ и мышью укажите произвольное ее месторасположение в области построения.
5. В контекстном меню созданной внешней сущности выберите опцию **«Редактировать активный элемент»**, на вкладке **«Объект»** нажмите **«Задать DFD объект»**, после чего, в появившемся окне выделите классификатор *«Преподаватель»* (см. Рис. 7-8) и нажмите «ОК».
6. Повторяя действия предыдущего шага, добавьте внешнюю сущность *«Студенты»*.

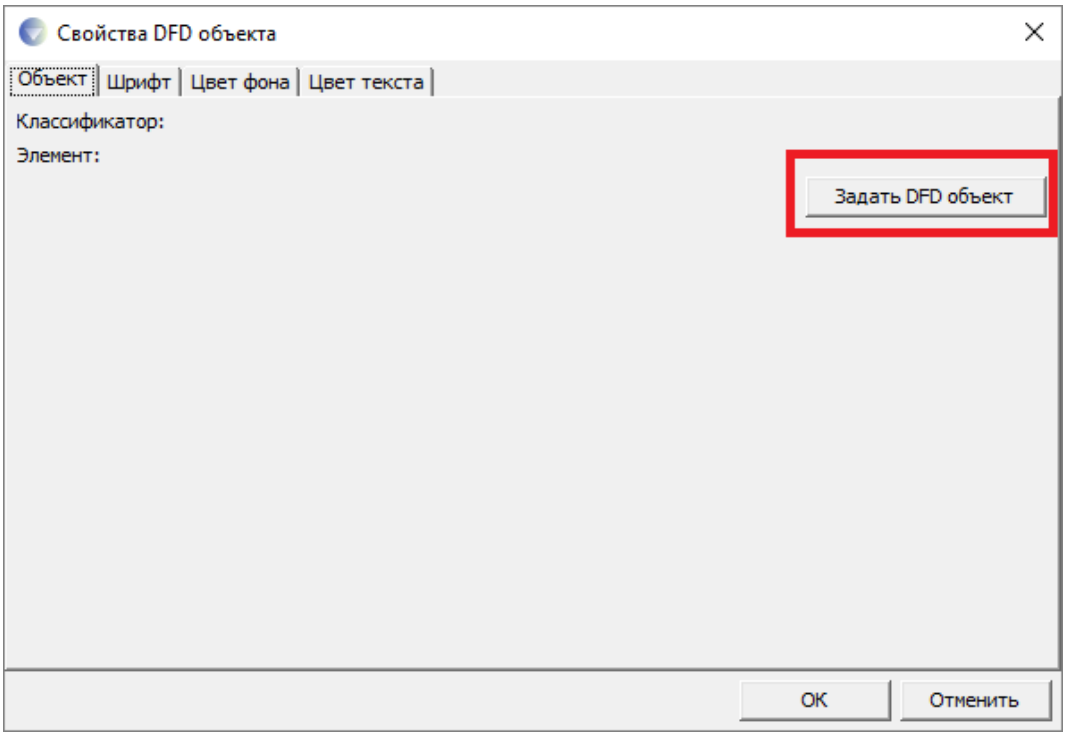


Рисунок 7 – Окно «Свойства DFD объекта»

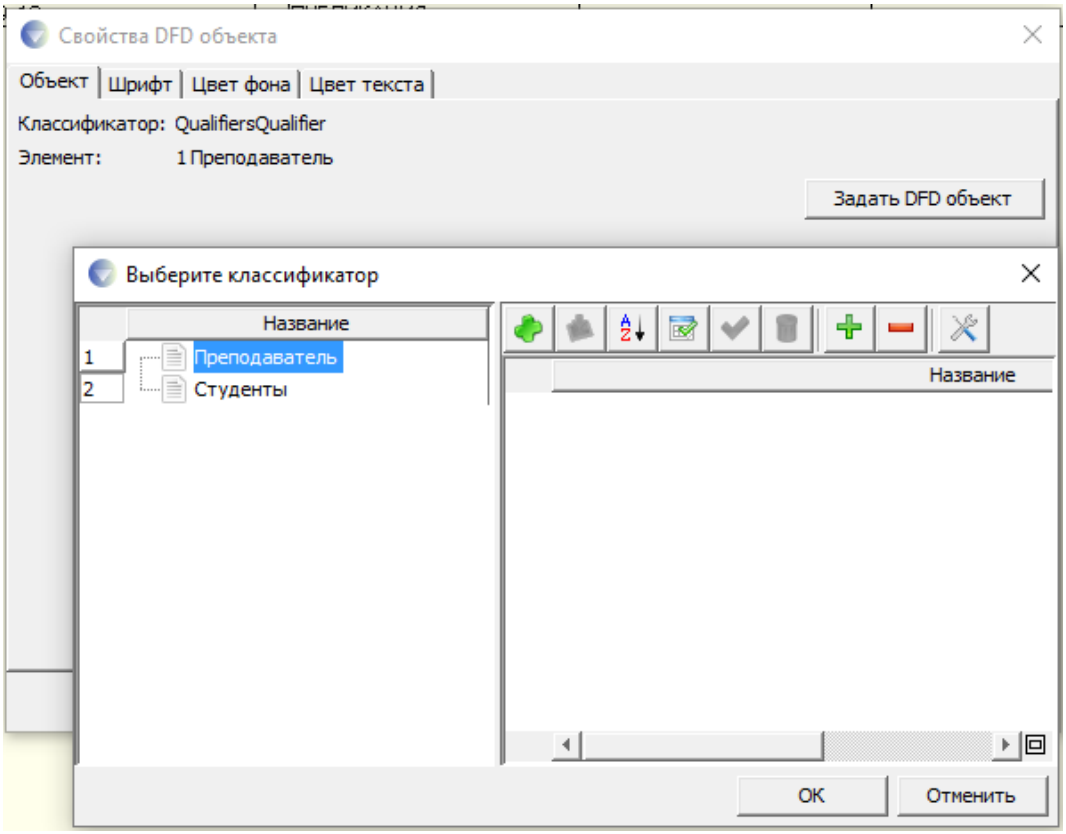


Рисунок 8 – Выбор классификатора

7. Выбрав на панели инструментов элемент →, создайте стрелки на контекстной диаграмме согласно Рис. 9. В результате должна получиться контекстная диаграмма, показанная на Рис. 9.

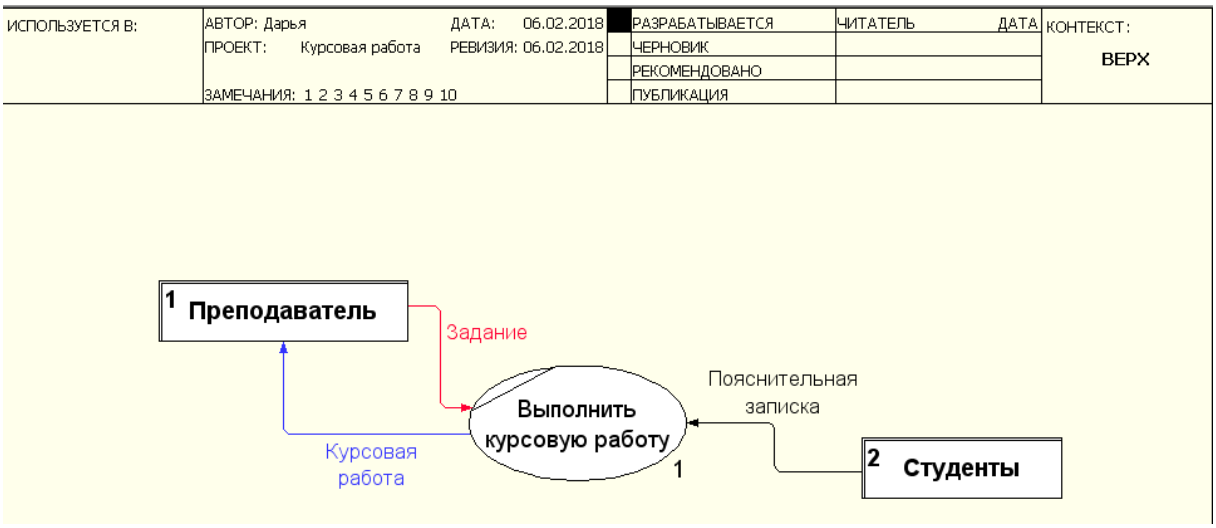


Рисунок 9 – Контекстная диаграмма предметной области «Выполнение курсовой работы»

3 Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне **«Создание новой диаграммы»** установите количество функциональных блоков 3, укажите тип диаграммы (DFD) и нажмите кнопку OK.

2. Автоматически будет создана диаграмма первого уровня декомпозиции (рис. 10) с перенесенными в нее потоками родительской диаграммы.

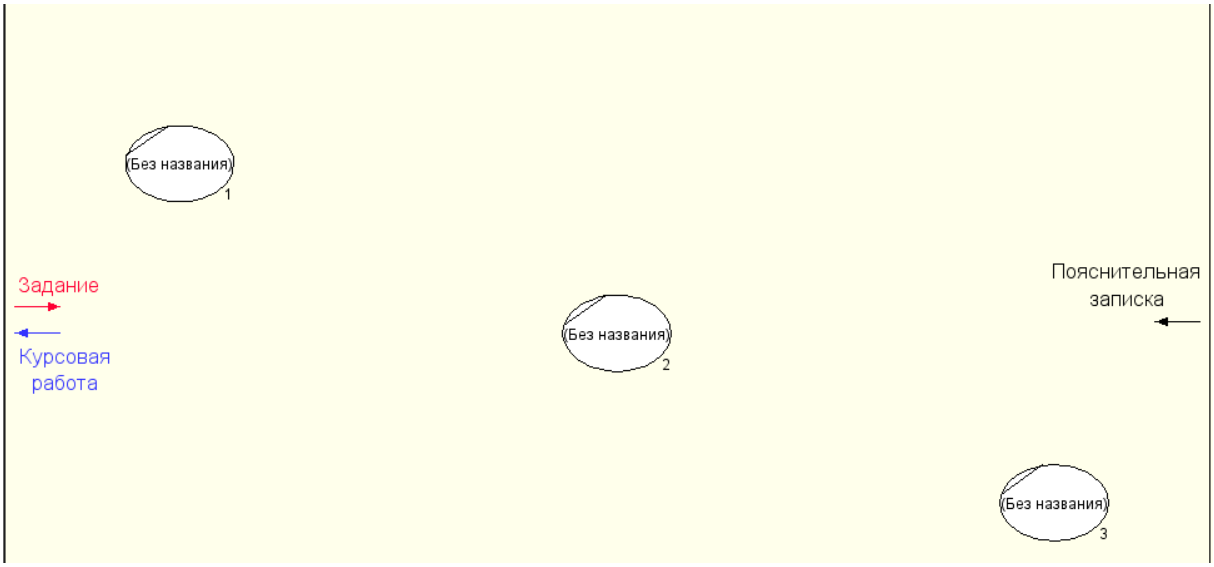


Рисунок 10 – Рабочее пространство детализирующей диаграммы

3. Двойным щелчком мыши на одном из процессов, или, выбрав в контекстном меню процесса пункт **«Редактировать активный элемент»**, откройте окно редактирования свойств и задайте процессу имя. Повторите операцию для оставшихся процессов.

4. Добавьте недостающие классификаторы для задания DFD объектов хранилищам данным. Для этого в меню выберите Окна -> Показать окно -> Классификаторы.

После этого нажмите на кнопку  . Название классификатора можно ввести в созданную строку, дважды, медленно кликнув мышью по строке, или же нажав клавишу F2, предварительно выделив нужную строку мышью.

5. Добавьте хранилища данных, воспользовавшись кнопкой палитры инструментов.

6. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом с соответствующим процессом. В результате должна получиться детализирующая диаграмма, представленная на Рис. 11.

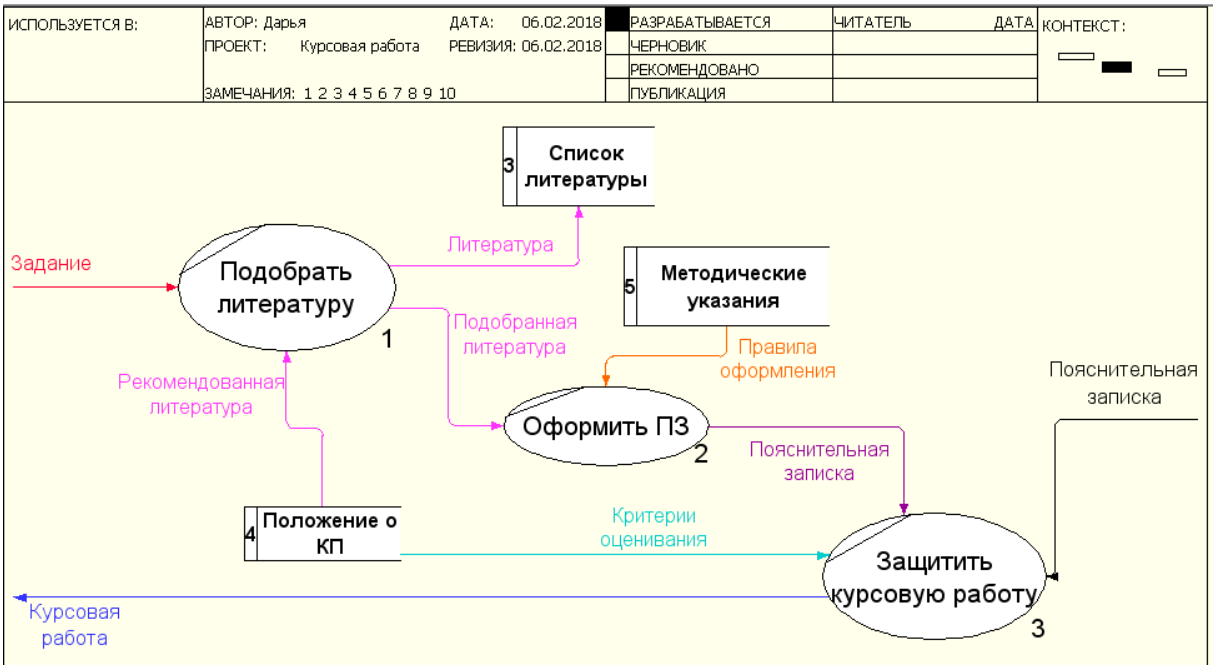


Рисунок 11 – Детализированная диаграмма первого уровня

7. На основе описанных выше действий постройте диаграмму декомпозиций второго уровня для процесса *«Оформить ПЗ»*. Построенная диаграмма должна иметь, изображенный на рисунке 13.

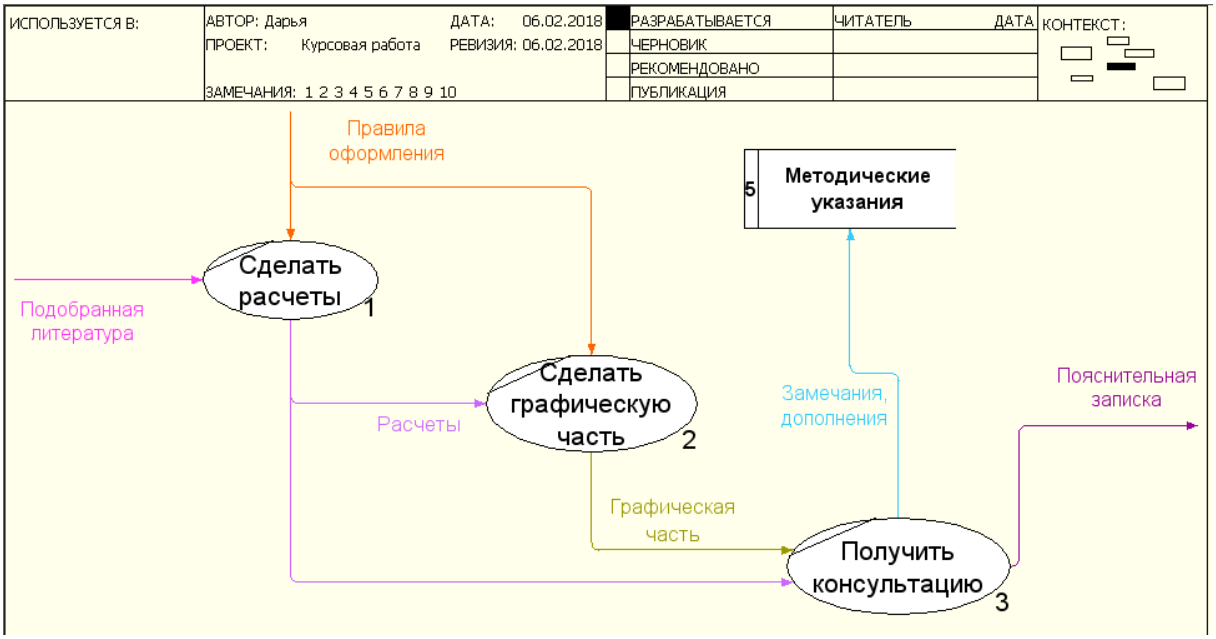


Рисунок 12 – Детализированная диаграмма второго уровня для процесса «Оформить ПЗ»

5 Задание

На основе модели предметной области, разработанной в практических занятиях №1 и 2, составить функциональную модель в нотации DFD.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

6 Варианты

1. Проектирование ИС «Отдел кадров»;
2. Проектирование ИС «Агентство аренды»;
3. Проектирование ИС «Аптека»;
4. Проектирование ИС «Ателье»;
5. Проектирование ИС «Аэропорт»;
6. Проектирование ИС «Библиотека»;
7. Проектирование ИС «Кинотеатр»;
8. Проектирование ИС «Поликлиника»;
9. Проектирование ИС «Автосалон»;
10. Проектирование ИС «Таксопарк»;
11. Проектирование ИС «Издательство»;
12. Проектирование ИС «Прокат велосипедов»;
13. Проектирование ИС «Спортивный клуб».

7 Требования к построению модели

Количество блоков любой декомпозиции не менее 3-х и не более 9.

Количество декомпозиций – 3 уровня декомпозиции.

8 Контрольные вопросы

1. Каково назначение стандарта DFD?
2. В чем основные отличия стандартов IDEF0 и DFD?
3. Каким образом в MS Visio создается схема DFD? Какие для этого используются нотации?
4. Какова роль основных элементов в стандарте DFD?

Практическая работа №6-7
Построение организационных диаграмм с помощью MS Visio

Цель: изучение основ создания организационных диаграмм в ручном и автоматическом режимах.

1 Задачи

Основными задачами практической работы являются:

– ознакомиться с теоретическими вопросами построения организационных диаграмм с помощью MS Visio;
– построить оргдиаграмму стандартными средствами;
– построить оргдиаграмму с помощью мастера диаграмм.

2 Краткие теоретические сведения

1. Общие сведения об организационных диаграммах

Организационная диаграмма – это схема иерархии отчетности, которая используется для отображения отношений между сотрудниками, должностями и группами.

Организационные диаграммы могут быть как простыми схемами, так и большими и сложными на основе сведений из внешнего источника данных. Фигуры организационной диаграммы могут отображать основные сведения, например, имя и должность, или подробные сведения, например, отдел и учетный отдел. К фигурам организационной диаграммы можно даже добавлять рисунки.

Организационная диаграмма – это схематическое представление об иерархии внутри компании, а также распределении полномочий и ответственности между сотрудниками и отделами.

Как показывает опыт, применение руководителями, менеджерами данных схем значительно повышает эффективность управления предприятием.

Использование организационных диаграмм позволяет решить сразу несколько задач:

1. проанализировать состояние дел в компании на текущий момент,
2. вовремя обнаружить какую-то проблему в организации и системе управления,
3. избежать появления новых проблем и ошибок.

В зависимости от поставленных целей организационные диаграммы могут быть простыми или развернутыми.

Простые диаграммы содержат краткую информацию о каждом из сотрудников (имя, должность и место в системе организационной иерархии), а также раскрывают численный состав персонала.

Развернутые диаграммы несут дополнительные сведения о функциях и объектах управления сотрудников компании.

Организационные диаграммы позволяют визуально представить характер взаимоотношений внутри компании, благодаря чему удается своевременно обнаружить и разрешить возникающие в организации проблемы.

Основные **преимущества организационных диаграмм:**

1. Они позволяют выявить главных игроков в организации и проанализировать их отношения с остальным персоналом.

2. Сотрудники компании получают ясное представление о том, кто возьмет на себя ответственность при разрешении тех или иных организационных проблем в соответствии с корпоративными стандартами.

3. Повышают осведомленность сотрудников о функциях и профессиональном статусе каждого субъекта, работающего в компании, тем самым способствуя совершенствованию процесса коммуникации внутри фирмы.

Следует отметить, что организационные диаграммы имеют ряд ограничений. Например, отражая лишь структуру и формальный характер взаимодействия персонала, они не затрагивают личные взаимоотношения людей в коллективе. Кроме того, с помощью этих схем нельзя определить стиль менеджмента, выбранный руководством предприятия.

Организационные диаграммы описывают связи при помощи **трех типов графических элементов:**

1. ***Линия:*** указывает на прямые отношения между руководителями и подчиненными. Для описания взаимосвязей между различными иерархическими уровнями организации линии рисуют слева или справа от диаграммы.

2. ***Ступенька:*** служит для иллюстрации отношений между помощниками менеджера, а также связей между сферами деятельности, в которых помощники могут давать советы руководителю, но при этом их мнение не является авторитетным.

3. ***Функционал:*** показывает связи между должностями специалистов и сферами деятельности, в которых мнение специалистов является авторитетным для руководителя.

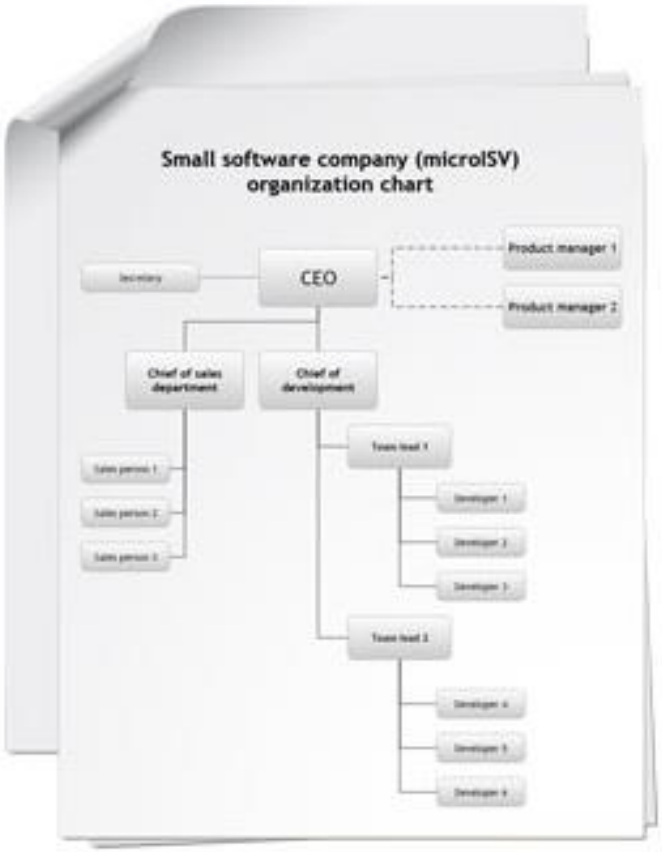


Рисунок 13 – Пример организационной диаграммы

Типы организационных структур

1. ***линейная модель:*** каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности;
2. ***функциональная модель:*** «одно подразделение = одна функция»;
3. ***линейно-функциональная модель:*** ступенчатая иерархическая;
4. ***процессная модель:*** «одно подразделение = один процесс»;
5. ***матричная модель:*** «один процесс или один проект = группа сотрудников из разных функциональных подразделений»;
6. ***множественная (смешанная).***

В ***линейной структуре*** управления каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности. *Достоинство* – простота, экономичность, предельное единоначалие. Основной *недостаток* – высокие требования к квалификации руководителей. Сейчас практически не используется.

Функциональная организационная структура – связь административного управления с осуществлением функционального управления.

Линейно-функциональная структура – линейные руководители являются единоначальниками, а им оказывают помощь функциональные органы. Линейные руководители низших ступеней административно не подчинены функциональным руководителям высших ступеней управления. Она применялась наиболее широко.

Преимущества:

– четкая система взаимных связей внутри функций и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью функций, составляющих деятельность;

– ясно выраженная ответственность;

– быстрая реакция исполнительных функциональных подразделений на прямые указания вышестоящих.

Недостатки:

– в работе руководителей практически всех уровней оперативные проблемы («текучка») доминируют над стратегическими;

– слабые горизонтальные связи между функциональными подразделениями порождают волокиту и перекладывание ответственности при решении проблем, требующих участия нескольких подразделений;

– малая гибкость и приспособляемость к изменению ситуации;

– критерии эффективности и качества работы подразделений и организации в целом разные и часто взаимоисключающие;

– большое число «этажей» или уровней управления между работниками, выпускающими продукцию, и лицом, принимающим решение;

– перегрузка управленцев верхнего уровня;

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств высших управленцев.

Процессная модель. Истоки концепции управления процессами ведут к теориям управления, разработанным еще в XIX веке. В 80-х годах XIX-го века Фредерик Тейлор предложил менеджерам использовать методы процессного управления для наилучшей организации деятельности. В начале 1900-х годов Анри Файоль разработал концепцию реинжиниринга – осуществление деятельности в соответствии с поставленными задачами путем получения оптимального преимущества из всех доступных ресурсов.

Преимущества процессных структур:

– четкая система взаимных связей внутри процессов и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью операций и действий, направленных на достижение поставленной цели и получение заданного результата;

– наделение сотрудников большими полномочиями и увеличение роли каждого из них в работе компании приводит к значительному повышению их отдачи;

– быстрая реакция исполнительных процессных подразделений на изменение внешних условий;

– в работе руководителей стратегические проблемы доминируют над оперативными;

– критерии эффективности и качества работы подразделений и организации в целом согласованы и сонаправлены.

Недостатки процессной структуры:

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств рядовых работников и исполнителей.

– управление смешанными в функциональном смысле рабочими командами – более сложная задача, нежели управление функциональными подразделениями;

– наличие в команде нескольких человек различной функциональной квалификации неизбежно приводит к некоторым задержкам и ошибкам, возникающим при передаче работы между членами команды. Однако потери здесь значительно меньше, чем при традиционной организации работ, когда исполнители подчиняются различным подразделениям компании.

Матричная модель. Матричные структуры совмещают принципы построения функциональных и процессных систем. В этих структурах существуют жестко регламентированные процессы, находящиеся под управлением менеджера процесса. При этом деятельность осуществляется работниками, находящимися в оперативном подчинении менеджера процесса и в административном подчинении руководителя в функциональном «колодце».

Преимущества

– Комплексный подход к реализации проекта, решению проблемы;

– Концентрация усилий на решении одной задачи, на выполнении одного конкретного проекта;

– Большая гибкость структуры;

– Активизация деятельности руководителей проектов и исполнителей в результате формирования проектных групп;

– Усиление личной ответственности конкретного руководителя как за проект в целом, так и за его элементы.

Недостатки

– При наличии нескольких организационных проектов или программ проектные структуры приводят к дроблению ресурсов и заметно усложняют поддержание и развитие производственного и научно-технического потенциала компании как единого целого;

– От руководителя проекта требуется не только управление всеми стадиями жизненного цикла проекта, но и учет места проекта в сети проектов данной компании;

– Формирование проектных групп, не являющихся устойчивыми образованиями, лишает работников осознания своего места в компании;

– При использовании проектной структуры возникают трудности с перспективным использованием специалистов в данной компании;

– Наблюдается частичное дублирование функций.

Смешанные структуры. Если применять различные модели организации деятельности в пределах отдельных бизнес-процессов, то можно использовать преимущества той или иной организационной модели. При этом для организации в целом будет применяться процессная организация основных структурных блоков, а в рамках отдельных блоков могут применяться различные модели.

3 Методика выполнения лабораторной работы

В качестве примера рассматривается процесс создания организационной диаграммы ВУЗа двумя способами:

1. вручную;
2. с помощью мастера.

1 Создание организационной диаграммы вручную

Для построения организационной диаграммы вручную необходимо проделать следующие действия:

1. Запустить *MS Visio*.
2. В разделе *Выберите шаблон* выбрать категорию *Бизнес*, а затем *Организационная диаграмма* и нажать кнопку *Создать*.

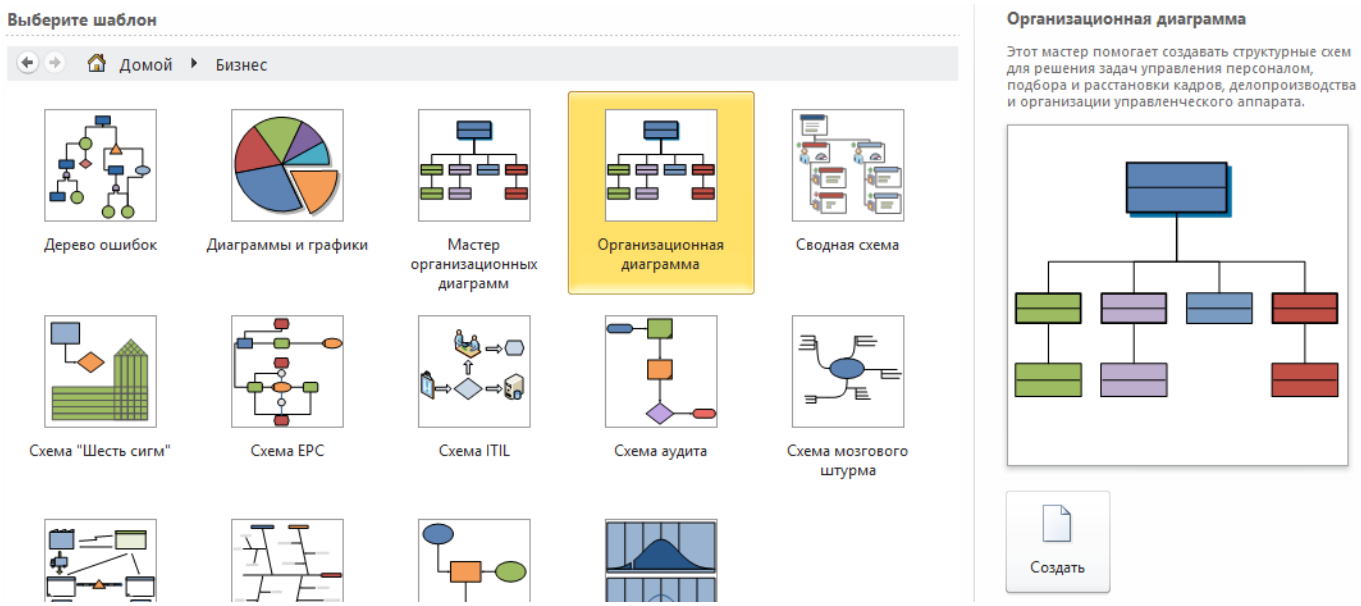


Рисунок 14 – Выбор шаблона

В разделе фигур можно увидеть основные элементы, необходимые для построения организационной диаграммы (рис. 3).

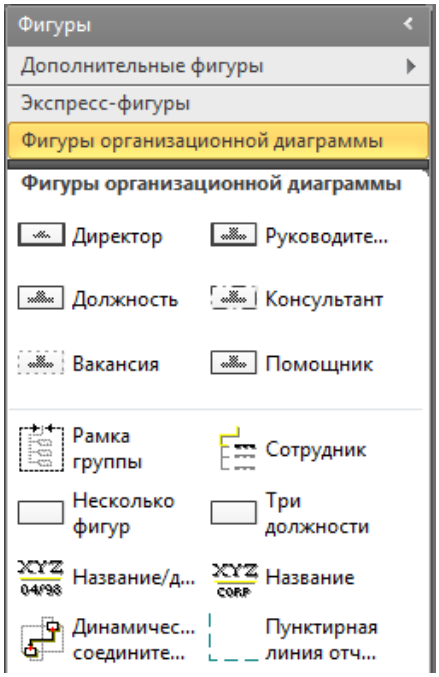


Рисунок 15 – Элементы организационной диаграммы в MS Visio

3. Перетащите фигуру *Директор* из набора фигур *Фигуры организационной диаграммы* в центр вверху страницы документа.

4. Надстройка организационных диаграмм отображает анимированное диалоговое окно (рис. 4), показывающее, как нужно добавлять в схему дополнительные фигуры.

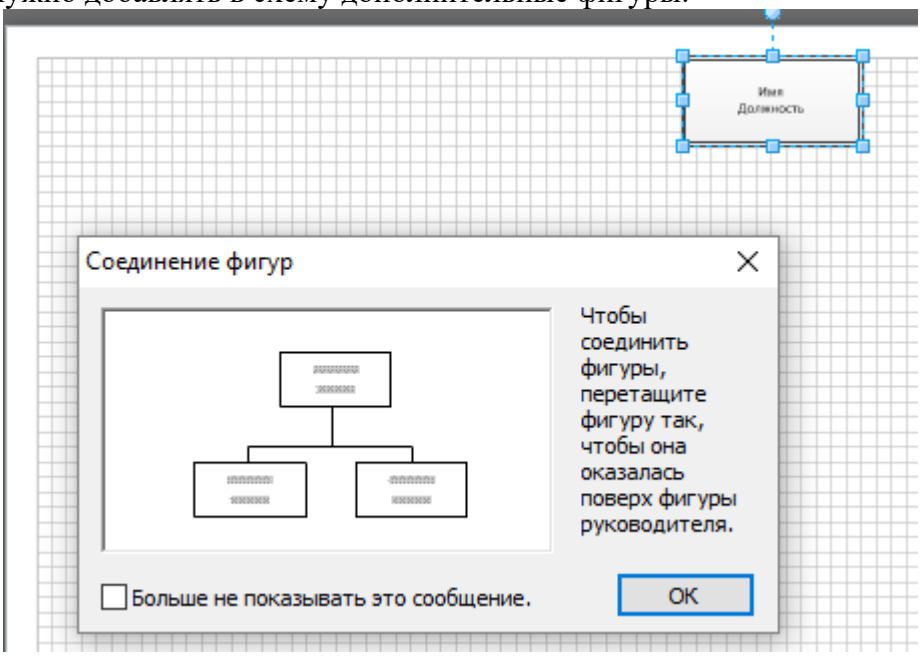


Рисунок 16 – Надстройка организационных диаграмм

5. Не снимая выделения с фигуры, при необходимости введите *ФИО*, затем нажмите клавишу ENTER и во второй строке введите *Должность*. (рис. 5).

При создании организационной диаграммы ВУЗа можно ограничиться только указанием должностей.

Рисунок 17 – Фигура Директор

6. Добавьте еще две фигуры *Директор* и расположите по обе стороны от фигуры *Ректор*. Соедините их используя *Соединительную линию.* В результате должна получиться схема, показанная на рисунке 6.

Рисунок 18 – Схема, полученная при выполнении шага 6

7. Перетащите фигуру *Руководитель* на фигуру *Директор*. Затем, при необходимости, введите ФИО, нажмите клавишу ENTER и введите *Должность*.

Надстройка автоматически размещает новую фигуру под фигурой *Директор*.

8. Повторите предыдущий шаг и обратите внимание, что надстройка разместила вторую фигуру руководителя сбоку от первой. Разместите все необходимые фигуры *Руководитель* в соответствии с рисунком 7.

Рисунок 19 – Фигуры Руководитель

9. Используя фигуры *Руководитель* и *Несколько фигур* добавьте необходимые *Должности,* относящиеся к руководителю **Первый проректор** (рис. 8)**.**

При использовании элемента *Несколько фигур* необходимо выполнить следующие действия:

– перетащить элемент Несколько фигур на необходимую фигуру;
– затем, в появившемся окне задать необходимые параметры, например, как на рисунке 9;
– нажать ОК.

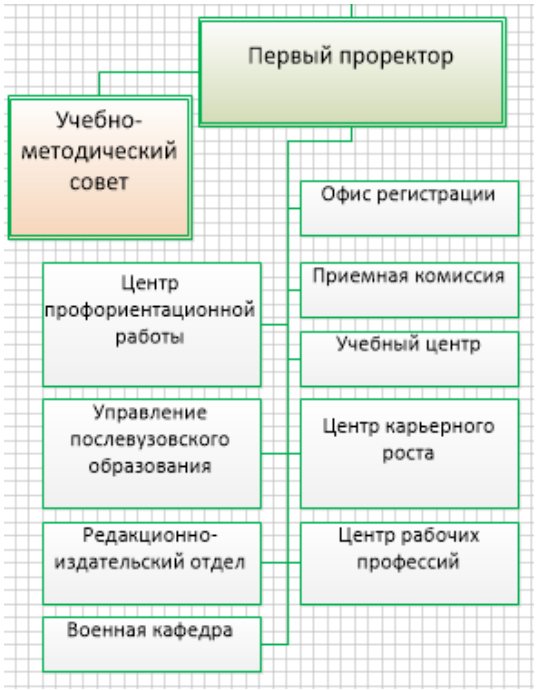


Рисунок 20 – Часть диаграммы «Первый проректор»

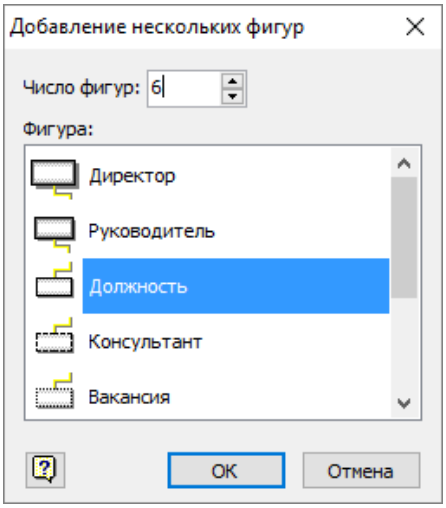


Рисунок 21 – Добавление нескольких фигур

10. Повторите предыдущий шаг и создайте полную организационную диаграмму (рис. 10-13).

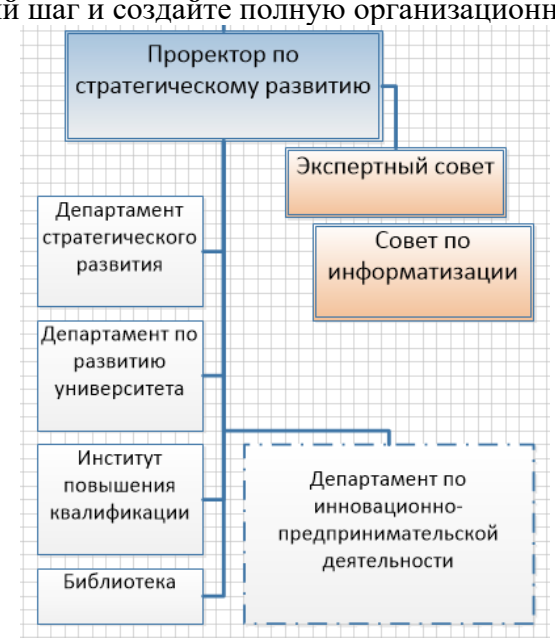


Рисунок 22 – Часть диаграммы «Проректор по стратегическому развитию»

Рисунок 23 – Часть диаграммы «Руководитель аппарата»

Рисунок 24 – Часть диаграммы «Проректор по научной работе»

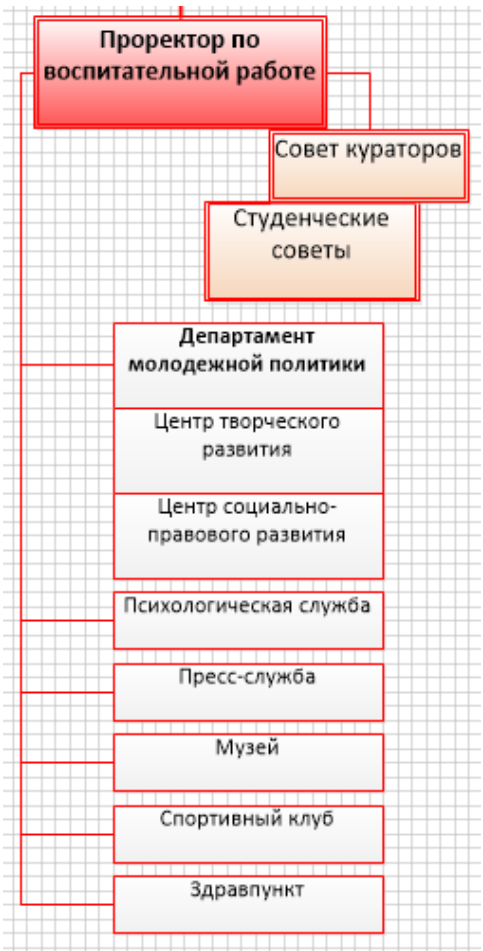


Рисунок 25 – Часть диаграммы «Проректор по ВР»

11. В результате выполнения всех описанных шагов организационная диаграмма ВУЗа должна принять вид, представленный на рисунке 14

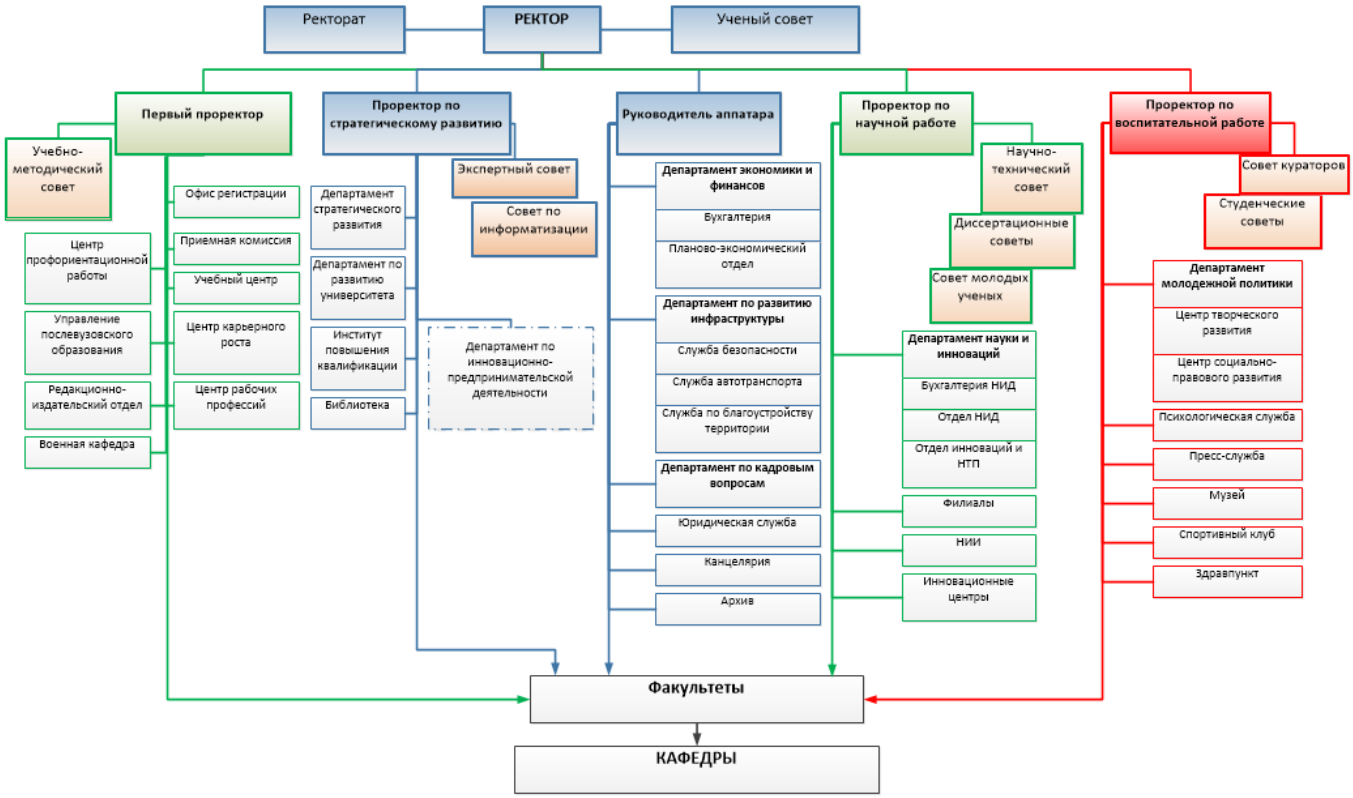


Рисунок 26 – Организационная диаграмма ВУЗа

2 Создание организационной диаграммы с помощью мастера диаграмм на основе данных

Построение организационной диаграммы с помощью мастера рассмотрим также на примере ВУЗа, однако немного упростим структуру.

Для построения организационной диаграммы с помощью мастера диаграмм необходимо проделать следующие действия:

1. Запустить *MS Visio*.
2. В разделе *Выберите шаблон* выбрать категорию *Бизнес*, а затем *Мастер организационных диаграмм* (рис. 15) и нажать кнопку *Создать*.

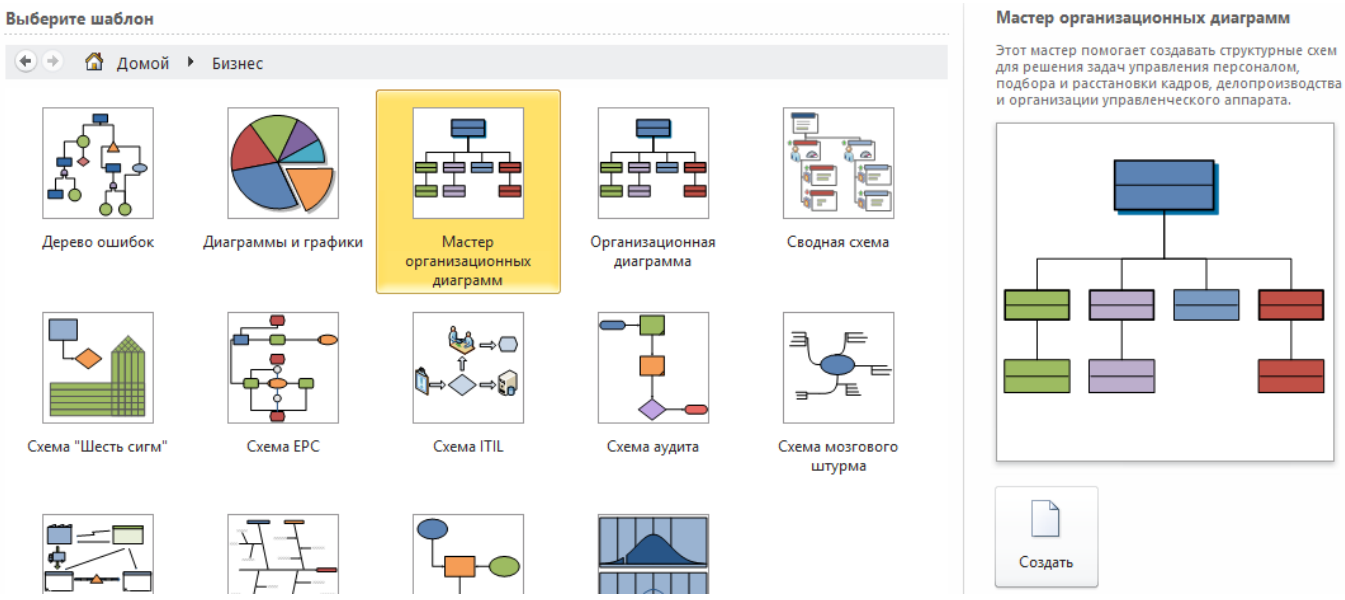


Рисунок 27 – Выбор шаблона «Мастер организационных диаграмм»

3. На первой странице мастера организационных диаграмм выберем переключатель *По данным, введенным с помощью мастера* (рис. 16). Обратите внимание – в описании этого переключателя подтверждается, что будет создан новый источник данных.

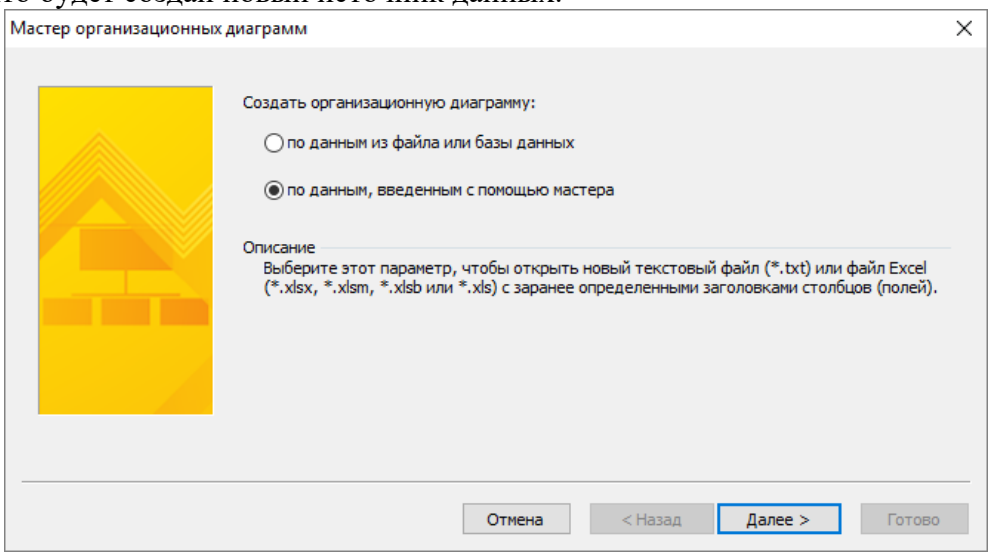


Рисунок 28 – Мастер организационных диаграмм. Шаг 1

4. Далее необходимо нажать кнопку *Далее* и выбрать тип файла.

На странице выбора типа файла выберем переключатель *Excel* и щелкнем на кнопке *Обзор*. В открывшемся диалоговом окне выберем папку, в которую нужно сохранить файл, в поле *Имя файла* введем *Данные оргдиаграммы* и щелкнем на кнопке Сохранить. Имя файла отображается в поле *Имя нового файла*. (рис. 17).

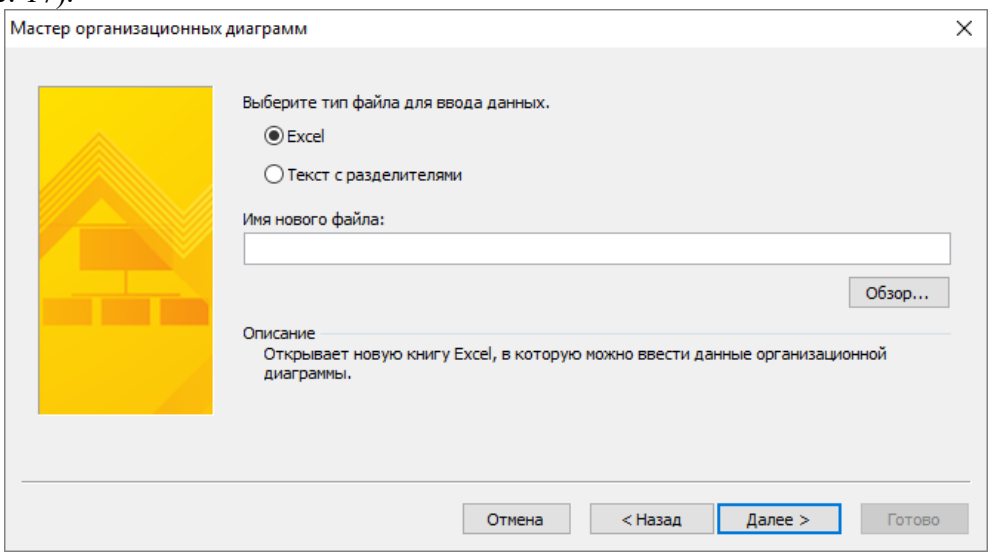


Рисунок 29 – Мастер организационных диаграмм. Шаг 2

5. Щелкнем на кнопке *Далее*. MS Visio показывает на необходимость ввести свои данные поверх ранее введенных (рис. 18).

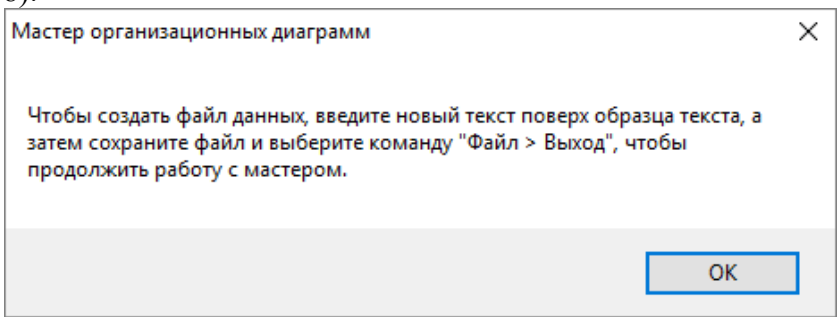


Рисунок 30 – Мастер организационных диаграмм. Шаг 3

6. Щелкнем на кнопке *Ок*. В Excel отображается форматированная книга. ***На рисунке 19 показана книга Excel для ввода данных диаграммы, при этом заголовок каждого столбца включает примечание с инструкциями по вводу данных в этом столбце.***

	A	B	C	D	E
1	**Имя**	**Руководитель**	**Введите фамилию лица, которому подчиняется данный сотрудник, как оно показано в поле** **СТЬ**	**ОТДЕЛ**	**ТЕЛЕФОН**
2					
3	Сергей Белов		Ге	Директор	x5555
4	Ольга Петрова	Сергей Белов	Руководитель разработки	Разработка продукта	x6666
5	Игорь Сергеев	Ольга Петрова	Разработчик программного обеспечения	Разработка продукта	x6667
6					

Рисунок 31 – Форматированная книга Excel по умолчанию

7. Далее вносим свои данные в соответствии с рисунком 20.

	A	B	C	D	E
1	**Имя**	**Руководитель**	**ДОЛЖНОСТЬ**	**ОТДЕЛ**	**ТЕЛЕФОН**
2					
3	Иванов Иван		Ректор	Ректорат	
4	Петров Петр	Иванов Иван	Ректорат	Ректорат	
5	Смирнов Андрей	Иванов Иван	Ученый совет	Ректорат	
6	Сидоров Михаил	Иванов Иван	Первый проректор	Ректорат	
7	Кожемяко Галина	Иванов Иван	Проректор по стратегическому развитию	Ректорат	
8	Куликова Наталья	Иванов Иван	Руководитель аппарата	Ректорат	
9	Лымарь Александра	Иванов Иван	Проректор по научной работе	Ректорат	
10	Голубенко Артем	Иванов Иван	Проректор по воспитательной работе	Ректорат	
11	Евидченко Денис	Сидоров Михаил	Начальник приемной комиссии	Приемная комиссия	
12	Бершатская Светлана	Сидоров Михаил	Начальник военной кафедры	Военная кафедра	
13	Маньшина Кристина	Кожемяко Галина	Начальник департамента развития	Департамент стратегического развития	
14	Шмелева Оксана	Кожемяко Галина	Библиотекарь	Библиотека	
15	Игнатенко Денис	Кожемяко Галина	Начальник департамента экономики	Департамент экономики и финансов	
16	Ненашев Виктор	Куликова Наталья	Начальник службы безопасности	Департамент по развитию инфраструктуры	
17	Ткачев Анатолий	Куликова Наталья	Начальник отдела кадров	Департамент по кадровым вопросам	
18	Сидоренко Александр	Куликова Наталья	Архивариус	Архив	
19	Акинин Павел	Лымарь Александра	Начальник департамента науки	Департамент науки и инноваций	
20	Смирнова Оксана	Голубенко Артем	Начальник департамента молодежной политики	Департамент молодежной политики	
21	Ермаков Дмитрий	Голубенко Артем	Начальник психологической службы	Психологическая служба	
22	Павлова Мария	Голубенко Артем	Врач	Здравпункт	

Рисунок 32 – Данный организационной диаграммы

8. Закрываем Excel. После закрытия Excel происходит возврат в программу MS Visio, где открыта страница импорта фотографий мастера (рис. 21).

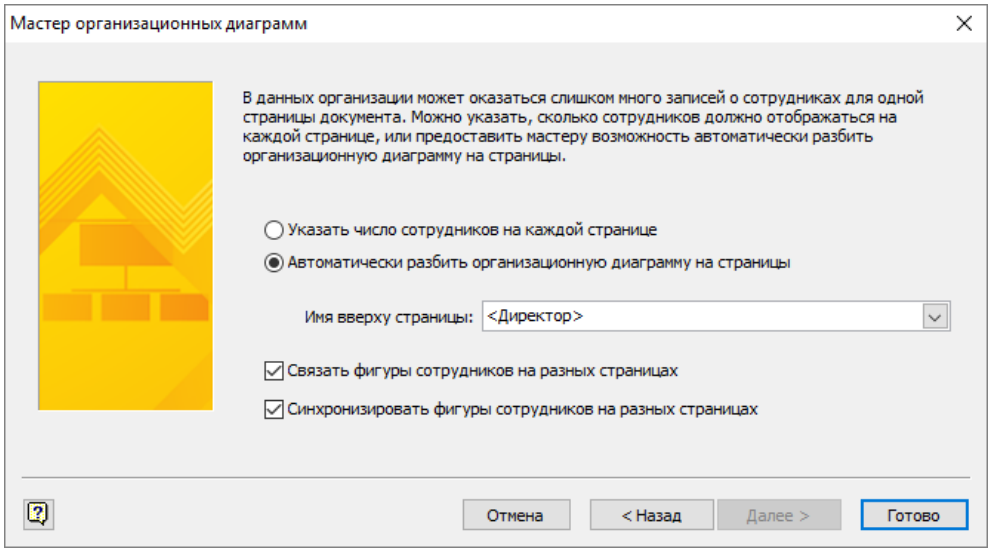


Рисунок 33 – Мастер организационных диаграмм. Шаг 4

9. Щелкнем на кнопке *Готово*, чтобы отобразить организационную диаграмму (рисунок 22).

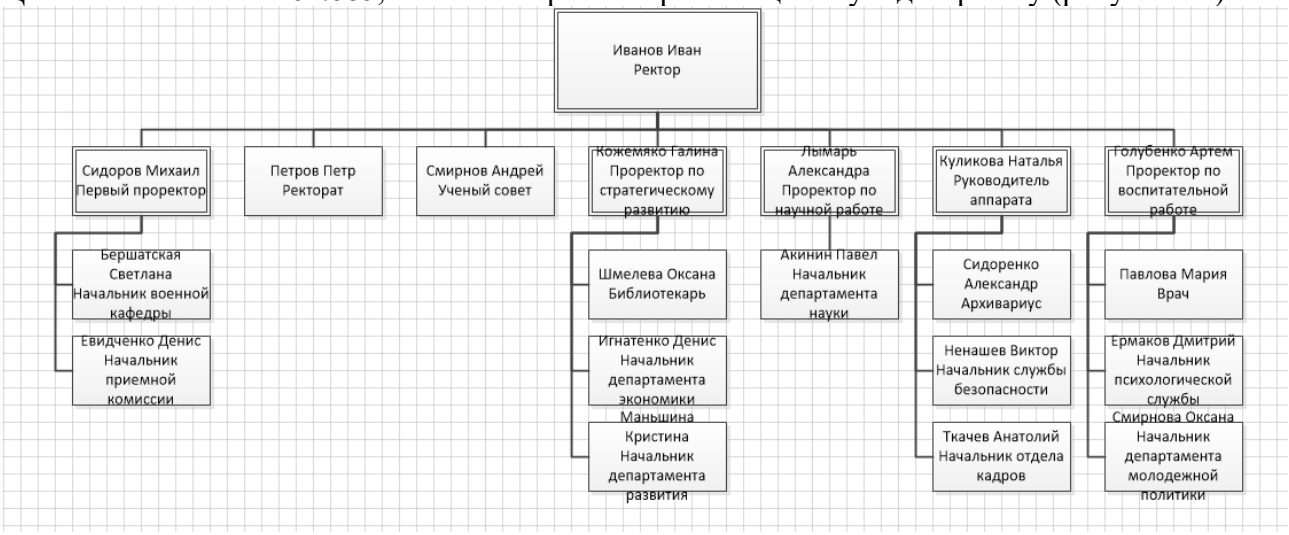


Рисунок 34 – Сформированная оргдиаграмма

10. На последнем шаге необходимо произвести форматирование диаграммы. В результате организационная диаграмма должна принять вид, показанный на рисунке 23.

Рисунок 35 – Итоговый вид диаграммы

4 Задание

Построить организационную диаграмму в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения организационной диаграммы, а также скриншоты результатов согласно заданию.

5 Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. «Таксопарк».
11. «Издательство»;
12. «Прокат велосипедов»;
13. «Спортивный клуб».

6 Контрольные вопросы

1. Что такое организационная диаграмма?
2. Способы построения оргдиаграмм в MS Visio?
3. Каковы принципы создания организационных диаграмм в MS Visio?
4. Какие существуют типы организационных структур? Перечислите их преимущества и недостатки.

Практическая работа №8-9
Построение диаграмм прецедентов на языке UML с помощью MS Visio

Цель: изучение основ создания диаграмм прецедентов (вариантов использования) на языке UML.

1 Задачи

Основными задачами практической работы являются:

– ознакомиться с теоретическими вопросами построения диаграмм прецедентов с помощью MS Visio;

– получить навыки создания диаграмм прецедентов.

2 Краткие теоретические сведения

1. Общие сведения о языке UML

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

В языке UML используется ***четыре основных вида графических конструкций***:

– **Значки или пиктограммы**. Значок представляет собой графическую фигуру фиксированного размера и формы. Примерами значков могут служить окончания связей элементов диаграмм или некоторые другие дополнительные обозначения.

– **Графические символы на плоскости**. Такие двумерные символы изображаются с помощью некоторых геометрических фигур и могут иметь различную высоту и ширину с целью размещения внутри этих фигур других конструкций языка UML.

Наиболее часто внутри таких символов помещаются строки текста, которые уточняют семантику или фиксируют отдельные свойства соответствующих элементов языка UML. Информация, содержащаяся внутри фигур, имеет важное значение для конкретной модели проектируемой системы, поскольку регламентирует реализацию соответствующих элементов в программном коде.

– **Пути,** которые представляют собой последовательности из отрезков линий, соединяющих отдельные графические символы. При этом концевые точки отрезков линий должны обязательно соприкасаться с геометрическими фигурами, служащими для обозначения вершин диаграмм, как принято в теории графов. С концептуальной точки зрения путям в языке UML придается особое значение, поскольку они являются простыми топологическими сущностями.

– **Строки текста.** Служат для представления различных видов информации в некоторой грамматической форме. Предполагается, что каждое использование строки текста должно соответствовать синтаксису в нотации языка UML, посредством которого может быть реализован грамматический разбор этой строки.

При графическом изображении диаграмм следует придерживаться следующих ***основных рекомендаций***:

1. Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области.

2. Все сущности на диаграмме модели должны быть одного концептуального уровня. Здесь имеется в виду согласованность не только имен одинаковых элементов, но и возможность вложения отдельных диаграмм друг в друга для достижения полноты представлений.

3. Вся информация о сущностях должна быть явно представлена на диаграммах. Речь идет о том, что, хотя в языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию (например, в случае неявного указания видимости атрибутов и операций классов), необходимо стремиться к явному указанию свойств всех элементов диаграмм.

4. Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем использовании на практике. Например, наличие замкнутых путей при изображении отношений агрегирования или композиции приводит к ошибкам в программном коде, который будет реализовывать соответствующие классы.

Наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен также приводит к неоднозначной интерпретации и может служить источником проблем.

5. Диаграммы не следует перегружать текстовой информацией. Принято считать, что визуализация модели является наиболее эффективной, если она содержит минимум пояснительного текста.

6. Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов.

7. Количество типов диаграмм для конкретной модели приложения не является строго фиксированным.

2. Диаграмма вариантов использования (usecase diagram)

Разработка диаграммы вариантов использования преследует цели:

1. Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
2. Сформулировать общие требования к функциональному поведению проектируемой системы.
3. Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
4. Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

При этом **актером (actor)** или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

В свою очередь, **вариант использования (usecase)** служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

В самом общем случае, диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно некоторых интерфейсов, и отношений между этими элементами.

Вариант использования (use case) **–** конструкция или стандартный элемент языка UML, который применяется для спецификации общих особенностей поведения системы или любой другой сущности предметной области без рассмотрения внутренней структуры этой сущности. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером.

Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Такой пояснительный текст получил название ***примечания или сценария***.

Отдельный вариант использования обозначается на диаграмме эллипсом (рис. 1), внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами.

Рисунок 36 – Графическое обозначение варианта использования

Актер (actor) представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некая отдельная роль относительно

конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка «человечка» (рис. 2), под которой записывается конкретное имя актера.

Рисунок 37 – Графическое обозначение актера

Интерфейс (interface) служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры. В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов или функциональности для актеров. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации.

На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя (рис. 3).

Рисунок 38 – Графическое обозначение интерфейса

В качестве имени может быть существительное, которое характеризует соответствующую информацию или сервис (например, «датчик», «сирена», «видеокамера»), но чаще строка текста (например, «запрос к базе данных», «форма ввода», «устройство подачи звукового сигнала»).

Если имя записывается на английском, то оно должно начинаться с заглавной буквы I, например, ISecurelnformation, ISensor.

Графический символ отдельного интерфейса может соединяться на диаграмме сплошной линией с тем вариантом использования, который его поддерживает. Сплошная линия в этом случае указывает на тот факт, что связанный с интерфейсом вариант использования должен реализовывать все операции, необходимые для данного интерфейса, а возможно и больше (рис. 4а).

Кроме этого, интерфейсы могут соединяться с вариантами использования пунктирной линией со стрелкой (рис. 4б), означающей, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Рисунок 39 – Графическое обозначение взаимосвязей интерфейсов и вариантов использования

Примечания (notes) в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Применительно к диаграммам вариантов использования примечание может носить самую общую информацию, относящуюся к общему контексту системы.

Графически примечания обозначаются прямоугольником с «загнутым» верхним правым уголком (рис. 5). Внутри прямоугольника содержится текст примечания. Примечание может относиться к любому элементу диаграммы, в этом случае их соединяет пунктирная линия. Если примечание относится к

нескольким элементам, то от него проводятся, соответственно, несколько линий. Разумеется, примечания могут присутствовать не только на диаграмме вариантов использования, но и на других канонических диаграммах.

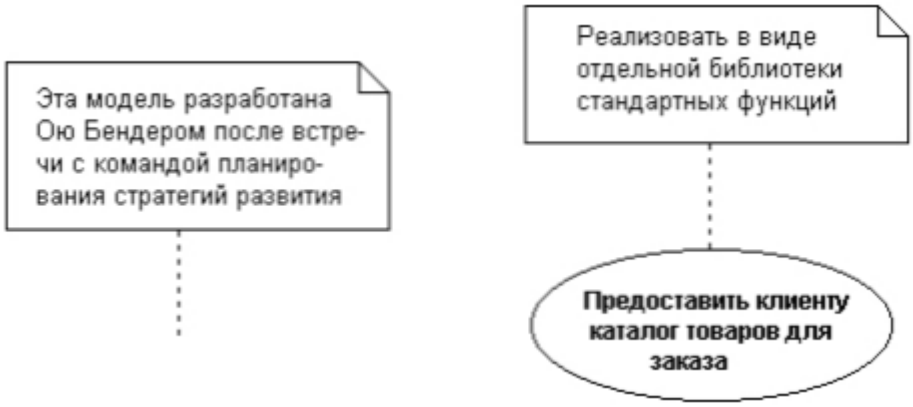


Рисунок 40 – Пример обозначения примечания

3. Отношения на диаграмме вариантов использования

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов.

Один актер может взаимодействовать с несколькими вариантами использования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис. Следует заметить, что два варианта использования, определенные для одной и той же сущности, не могут взаимодействовать друг с другом, поскольку каждый из них самостоятельно описывает законченный вариант использования этой сущности. Более того, варианты использования всегда предусматривают некоторые сигналы или сообщения, когда взаимодействуют с актерами за пределами системы. В то же время могут быть определены другие способы для взаимодействия с элементами внутри системы.

В языке UML имеется несколько стандартных ***видов отношений между актерами и вариантами использования:***

1. Отношение ассоциации (association relationship)

Отношение ассоциации является одним из фундаментальных понятий в языке UML и в той или иной степени используется при построении всех графических моделей систем в форме канонических диаграмм.

Применительно к диаграммам вариантов использования оно служит для обозначения специфической роли актера в отдельном варианте использования. Другими словами, ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы. Таким образом, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования, так же, как и на других диаграммах, отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь дополнительные условные обозначения, такие, например, как имя и кратность (рис. 6).

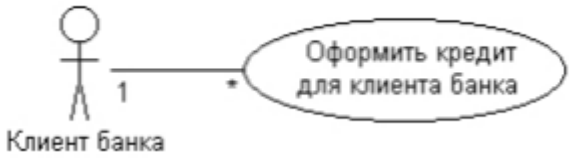


Рисунок 41 – Графическое обозначение отношения ассоциации

2. Отношение расширения (extend relationship)

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров.

Так, если имеет место отношение расширения от варианта использования А к варианту использования В, то это означает, что свойства экземпляра варианта использования В могут быть дополнены благодаря наличию свойств у расширенного варианта использования А.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом «extend» («расширяет»), как показано на рис. 7

Рисунок 42 – Графическое обозначение отношения расширения

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

Один из вариантов использования может быть расширением для нескольких базовых вариантов, а также иметь в качестве собственных расширений несколько других вариантов. Базовый вариант использования может дополнительно никак не зависеть от своих расширений.

3. Отношение обобщения (generalization relationship)

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В.

В этом случае вариант А будет являться специализацией варианта В. При этом В называется предком или родителем по отношению А, а вариант А – потомком по отношению к варианту использования В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения.

Графически данное отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования (рис. 8). Эта линия со стрелкой имеет специальное название – стрелка «обобщение».

Рисунок 43 – Графическое обозначение отношения обобщения

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Например, отношение обобщения от актера А к актеру В отмечает тот факт, что каждый экземпляр актера А является одновременно экземпляром актера В и обладает всеми его свойствами. В этом случае актер В

является родителем по отношению к актеру А, а актер А, соответственно, потомком актера В. При этом актер А обладает способностью играть такое же множество ролей, что и актер В.

Графически данное отношение также обозначается стрелкой обобщения, т. е. сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительского актера (рис. 9).

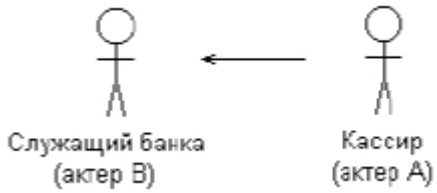


Рисунок 44 – Графическое обозначение отношения обобщения между актерами

4. Отношение включения (include relationship)

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

Отношение включения, направленное от варианта использования А к варианту использования В, указывает, что каждый экземпляр варианта А включает в себя функциональные свойства, заданные для варианта В. Эти свойства специализируют поведение соответствующего варианта А на данной диаграмме. Графически данное отношение обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом «include» («включает»), как показано на рис. 10

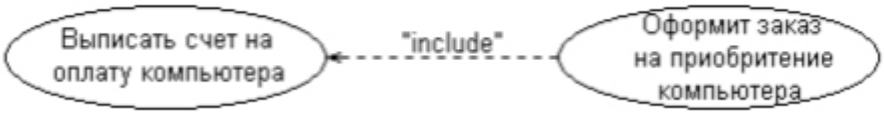


Рисунок 45 – Графическое обозначение отношения включения

3 Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите MS Visio.
2. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
3. Окно программы примет вид, подобный рис. 11.

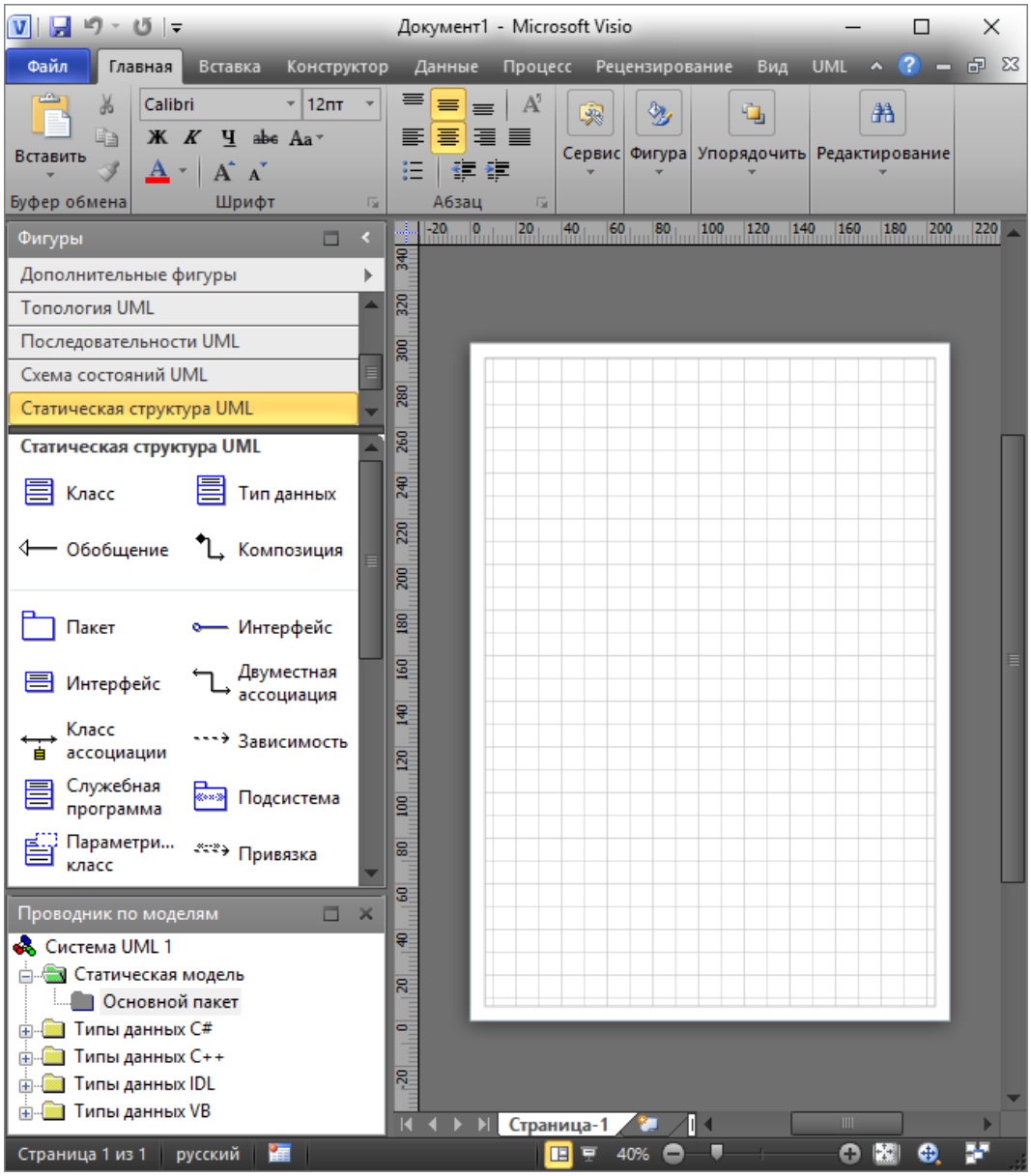


Рисунок 46 – Схема модели UML в MS Visio

4. Далее необходимо открыть все фигуры, необходимые для построения UML-диаграмм. Для этого в левой части экрана необходимо нажать кнопку *Дополнительные фигуры.* В открывшемся вспомогательном меню выбрать *Программы и БД -> Программное обеспечение* и выбрать все доступные фигуры для построения UML (рис. 12).

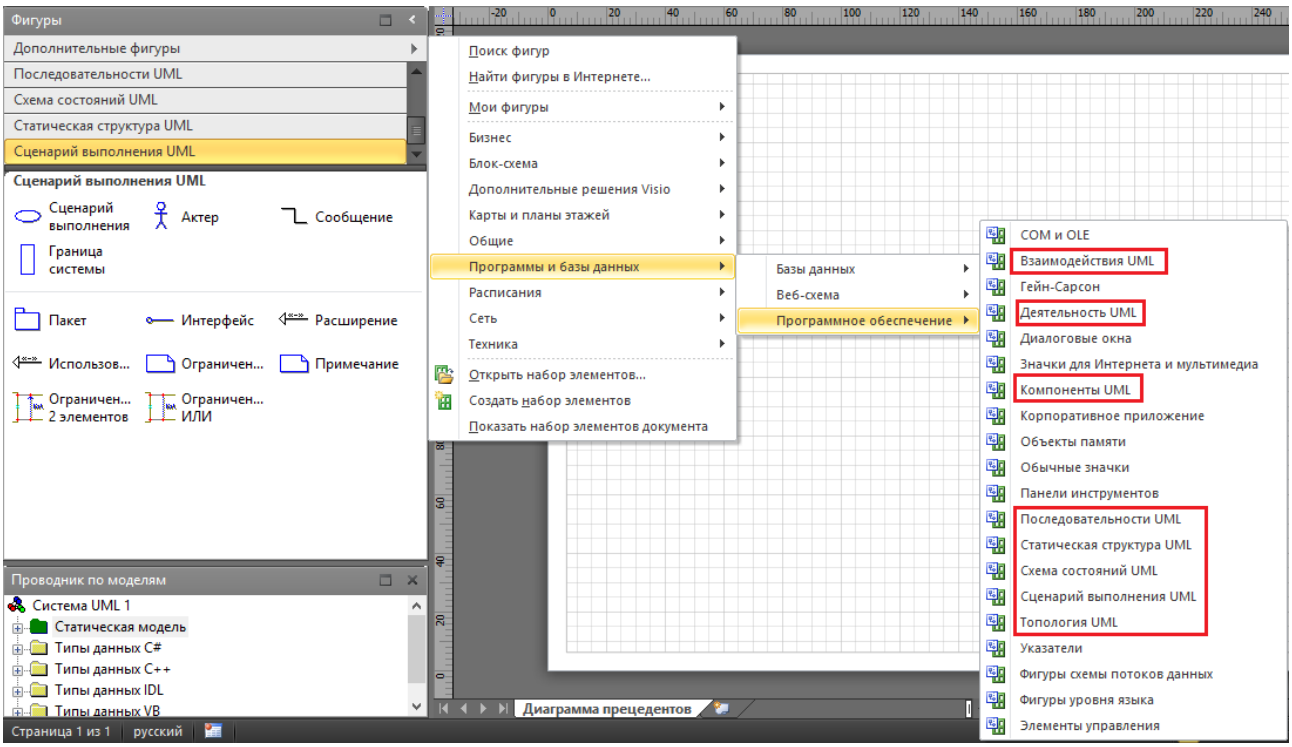


Рисунок 47 – Добавление фигур UML

5. После этого необходимо провести следующие этапы моделирования.

5.1. Выбор актеров.

В качестве актеров данной системы могут выступать два субъекта, один из которых является продавцом, а другой – покупателем. Каждый из этих актеров взаимодействует с рассматриваемой системой продажи товаров по каталогу и является ее пользователем, т. е. они оба обращаются к соответствующему сервису «Оформить заказ на покупку товара». Как следует из существа выдвигаемых к системе требований, этот сервис выступает в качестве варианта использования разрабатываемой диаграммы, первоначальная структура которой может включать в себя только двух указанных актеров и единственный вариант использования (рис. 14).

– В группе фигур *Сценарий выполнения UML* выбрать блок *Граница системы* и добавить его на лист.

– Внутрь границы системы добавить блок *Сценарий выполнения* и добавить к нему название, дважды щелкнув внутри блока.

– Добавить два блока *Актер* – покупатель и продавец.

– С помощью блока *Сообщение* установите связь актеров и варианта использования. Двойным щелчком правой кнопки мыши по блоку Сообщение откройте окно *Свойств ассоциации UML*, проведите настройки в соответствии с рисунком 13.

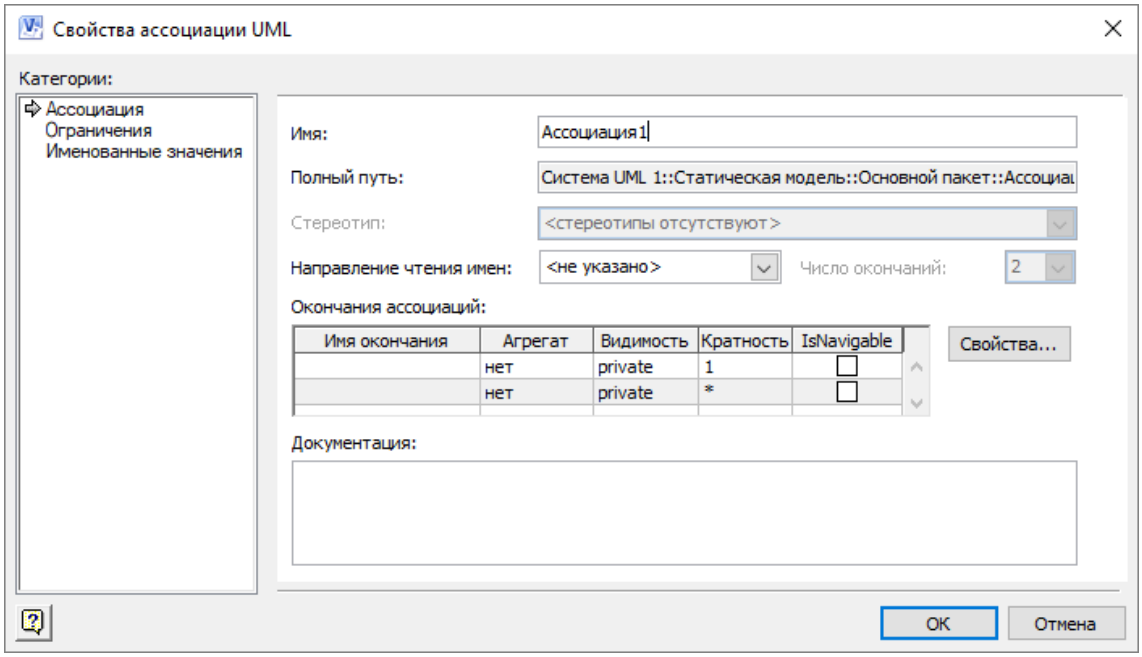


Рисунок 48 – Свойства ассоциации UML

Рисунок 49 – Исходная диаграмма вариантов использования системы по продаже товаров

5.2. Выделение дополнительных вариантов использования.

Детализировать вариант использования «Оформить заказ на продажу товара» можно выделив следующие дополнительные варианты использования:

- обеспечить покупателя информацией – является отношением включения;
- согласовать условия оплаты – является отношением включения;
- заказать товар со склада – является отношением включения;
- запросить каталог товаров – является отношением расширения.

Так как в MS Visio отсутствует отношение включения, его необходимо добавить самостоятельно. Для этого перейти на вкладку *UML* -> в группе *Модель* выбрать пункт *Стереотипы.* В открывшемся окне нажать кнопку *Создать* и настроить стереотип в соответствии с рисунком 15.

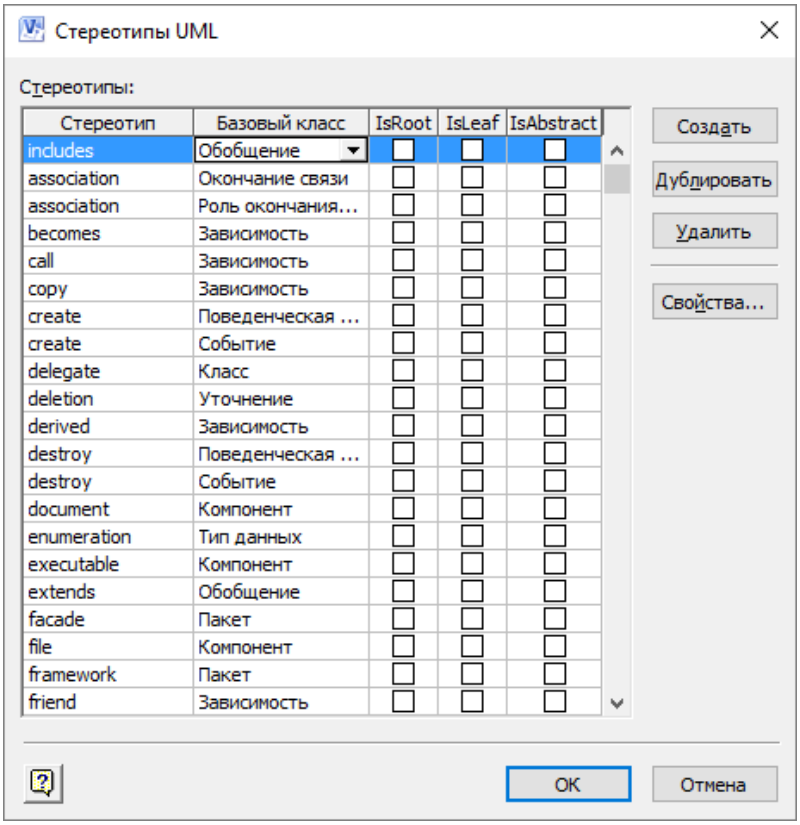


Рисунок 50 – Создание стереотипа

Далее на новом листе необходимо добавить границу системы и все варианты использования. После чего соединить варианты использования с помощью блока *Расширение.*

Для того, чтобы изменить тип отношения дважды щелкните по стрелке и окне свойств задайте необходимые параметры.

Дополненная диаграмма вариантов использования примет вид, показанный на рисунке 16.

Рисунок 51 – Дополненная диаграмма вариантов использования

5.3. Написание описательной спецификации для каждого варианта использования.

Спецификация для варианта использования «Оформить заказ на покупку компьютера» приведена в таблице 1.

Таблица 7 – Спецификация варианта использования

Раздел	**Описание**
Краткое описание	Покупатель желает оформить заказ на покупку компьютера, который он выбрал в каталоге товаров. При условии, что клиент зарегистрирован и выбранный компьютер есть в наличии оформляется заказ. Если клиент не зарегистрирован, то предлагается ему пройти регистрацию, и после этого заказать выбранный компьютер. Если компьютера нет в наличии, то предлагается заказать товар со склада в течении заданного срока поставки.
Субъекты	Продавец, Покупатель
Предусловия	В каталоге товаров имеются компьютеры, которые можно заказать. У покупателей есть доступ к системе для регистрации. Продавцы умеют пользоваться рассматриваемой системой продажи. У покупателя есть бонусы.
Основной поток	Зарегистрированный покупатель имеет возможность заказать любой компьютер из каталога товаров. В случае наличия выбранного компьютера оформляется заказ с присвоением ему уникального номера. После этого покупателю предлагается выбрать способ оплаты и способ получения компьютера. В случае отсутствия компьютера в наличии предлагается оформить заказ со склада и ожидания его поставки в рамках указанного срока или выбрать другой компьютер.
Альтернативный поток	Покупатель не зарегистрирован. В этом случае, прежде чем оформить заказ на компьютер, ему предлагается пройти регистрацию. Попытка заказать товар, который отсутствует на складе Начисление бонусов
Постусловия	Заказ оформлен и определен срок поставки компьютера и место его получения

На рисунках 17-19 приведены примеры диаграмм вариантов использования для различных систем.

Рисунок 52 – Пример 1

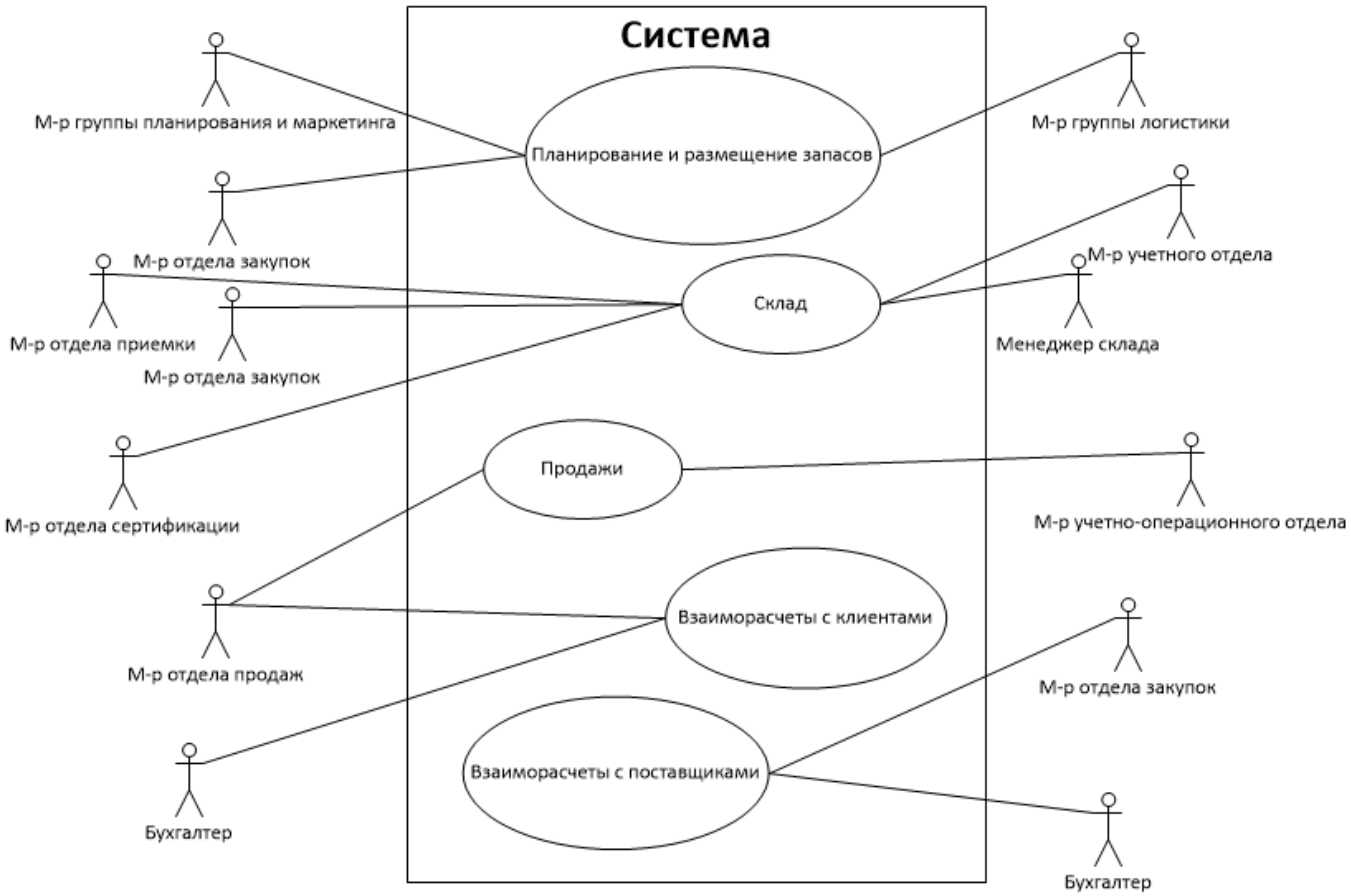


Рисунок 53 – Пример 2

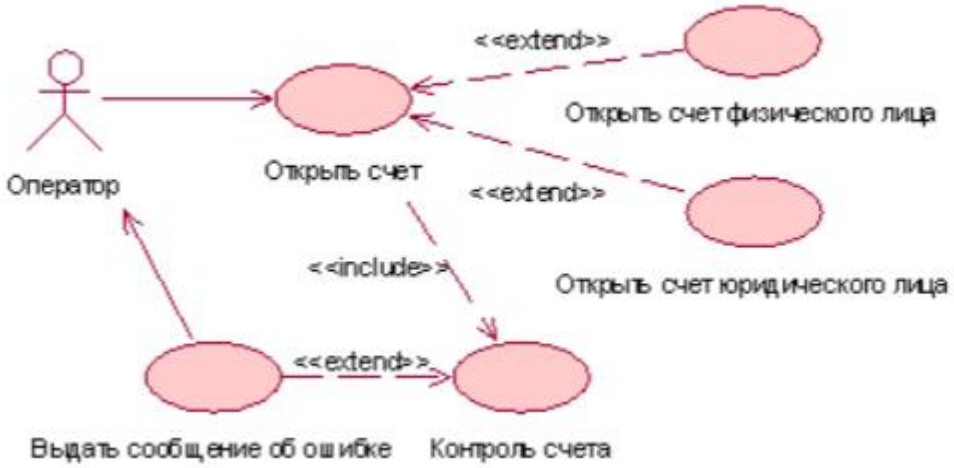


Рисунок 54 – Пример 3

4 Задание

Построить диаграмму прецедентов (вариантов использования) в соответствии с вариантом. Составить спецификацию.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения организационной диаграммы, а также скриншоты результатов согласно заданию.

5 Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. «Таксопарк».
11. «Издательство»;
12. «Прокат велосипедов»;
13. «Спортивный клуб».

6 Контрольные вопросы

1. Для чего используется язык UML?
2. Назначение диаграммы вариантов использования?
3. Что такое «актер»?
4. Что такое «вариант использования»?
5. Что такое «интерфейс»?
6. Что такое «примечание»?
7. Перечислить виды отношений между актерами и вариантами использования, охарактеризовать каждое из них?

Практическая работа №10-11
Построение диаграмм классов на языке UML с помощью MS Visio

Цель: изучение основ создания диаграмм классов на языке UML, получение навыков построения диаграмм классов, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

1 Задачи

Основными задачами практической работы являются:
- ознакомиться с теоретическими вопросами построения диаграмм классов;
- ознакомиться с теоретическими вопросами построения диаграмм классов с помощью MS Visio;
- получить навыки создания диаграмм классов.

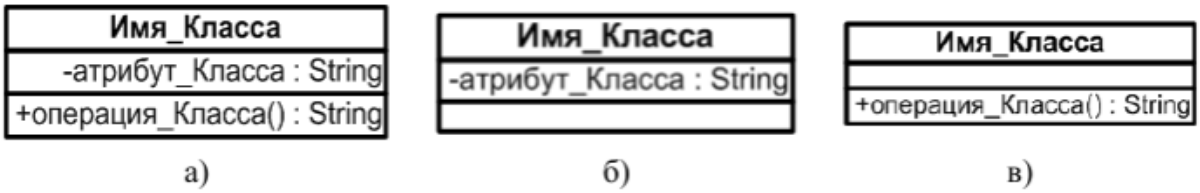
2 Краткие теоретические сведения

Диаграмма классов является одной из канонических диаграмм UML, создаваемой для визуализации структурированной статической модели предметной области. Этот вид диаграмм представляет собой графическое изображение объектов – классов с присущими им атрибутами, операциями и различных отношений между классами.

1. Классы

Класс (class) служит для обозначения множества объектов, обладающих функциональным набором одинаково описывающих параметров (атрибутов), реализуемых операций и однотипными отношениями с объектами других классов.

На диаграмме класс изображается габаритной прямоугольной рамкой, которая дополнительно может быть разделена горизонтальными линиями на секции, каждая из которых предназначена для указания имени, атрибутов (свойств) и реализуемых операций объектов данного класса (рис. 1 а, б, в).



а) б) в)

Рисунок 55 – а) Обозначение класса, б) Обозначение атрибутов, в) Обозначение операций

Имя класса является обязательным элементом в его обозначении и должно быть уникальным (хотя бы в пределах пакета), имеющее непосредственное отношение к контексту моделируемой предметной области.

В соответствии с принятым в языке UML общим соглашением в качестве имени класса используются существительные и прилагательные, каждое из которых начинается с заглавной буквы, записанные без пробелов. Например, в качестве имен классов могут быть использованы профессиональные термины: «Сотрудник», «Компания», «Руководитель», «Клиент», «Продавец», «Менеджер», «Офис», «Покупатель», «Датчик_Температуры» и др. Такое имена классов являются простыми.

Иногда возникает необходимость в явном указании пакета, к которому относится данный класс. С этой целью в условном обозначении перед именем класса указывается имя пакета и специальный символ разделитель – двойное двоеточие "::". Такое имя класса является квалифицированным. Текстовая строка имени класса в этом случае записывается в формате <Имя_пакета>::<Имя_класса>. Например, если определен пакет с именем «Банк», то имя класса «Счет» может быть записано так, как показано на рис. 2.

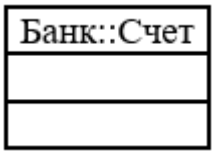


Рисунок 56 – Обозначение квалифицированного имени класса

2. Атрибуты классов

Содержательной характеристикой класса является атрибут, содержащий множество значений, которые могут принимать отдельные объекты этого класса. При этом, класс может иметь любое число атрибутов или не иметь ни одного. Так, например, атрибутами класса «Усилитель» являются частотный диапазон, выходная мощность, коэффициент нелинейных искажений, уровень шума и т. д.

Запись атрибута также представляет собой отдельную строку текста, содержащую обязательное имя, в котором обычно каждое слово пишется с заглавной буквы, за исключением первого, например, name (имя) или birth_Date (дата_Рождения).

Например, на рис. 3 указаны атрибуты класса Контейнер, в качестве которых выступают атрибут тип_Контейнера, атрибут регистрационный_Номер_Контейнера, регистрационный_Номер_После_Перерегистрации, рабочее_Состояние.

Рисунок 57 – Указание атрибутов класса Контейнер

Качественной характеристикой описания элементов класса является квантор видимости атрибута – потенциальная возможность других объектов модели оказывать влияние на отдельные аспекты поведения данного класса.

Эта характеристика может принимать одно из трех возможных значений и, соответственно, отображается при помощи специальных символов: символ "+" (public) обозначает атрибут с областью видимости типа общедоступный; атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма; например, для класса Class_1 указан атрибут общедоступного типа (рис. 4а); символ "#" (protected) обозначает атрибут с областью видимости типа защищенный; атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса; например, для класса Class_2 указан атрибут защищенного типа (рис. 4б); символ "-" (private) обозначает атрибут с областью видимости типа закрытый; атрибут с этой областью видимости недоступен или невиден для всех классов без исключения; например, для класса Class_3 указан атрибут защищенного типа (рис. 4в);

Квантор видимости при описании атрибутов может быть опущен, что будет означать тот факт, что видимость атрибута не указывается.

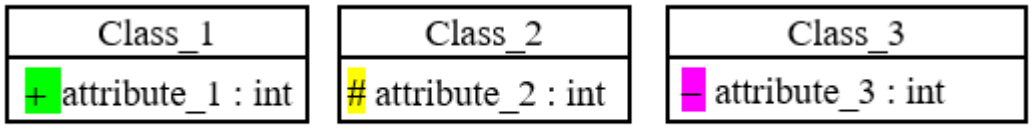


Рисунок 58 – Обозначение видимости атрибутов класса

3. Операции классов

Операция (operation) класса – это реализация услуги, которая может быть запрошена у любого объекта данного класса, чтобы вызвать определенное его поведение. Класс может иметь любое число операций либо не иметь ни одной. Так автомобиль может перемещаться по грунту, корабль – перемещаться по воде, компьютер – производить вычисления.

Представление полного синтаксиса записи операций класса также подчиняется определенным синтаксическим правилам: каждой операции класса соответствует отдельная строка, которая состоит из

квантора видимости операции, обязательного имени операции, выражения типа возвращаемого операцией значения и, возможно, строки-свойства данной операции:

< квантор видимости >< имя операции > (список параметров) : < выражение типа возвращаемого значения >{строка-свойство}

Квантор видимости, как и в случае атрибутов класса, может принимать одно из трех возможных значений и, соответственно, также отображается при помощи специального символа.

Для именования операции используются короткие глагольные конструкции, описывающие некоторое поведение класса, которому принадлежит операция. Обычно каждое слово в имени операции пишется с заглавной буквы, за исключением первого.

Например, запись +создать() – может обозначать абстрактную операцию по созданию отдельного объекта класса, которая является общедоступной и не содержит формальных параметров, запись +нарисовать(форма: Многоугольник = прямоугольник, цвет_заливки: Color = (O, O, 255)) – может обозначать операцию по изображению на экране монитора прямоугольной области синего цвета, если не указываются другие значения в качестве аргументов данной операции.

(список–параметров) содержит необязательные аргументы, синтаксис которых совпадает с синтаксисом атрибутов;

< выражение типа возвращаемого значения > является необязательной спецификацией и зависит от конкретного языка программирования;

{строка-свойство} показывает значения свойств, которые применяются к данной операции.

Например, запись запросить_Счет_Клиента(номер_счета:Integer) – обозначает операцию по установлению наличия средств на текущем счете клиента банка. При этом аргументом данной операции является номер счета клиента, который записывается в виде целого числа (например, «123456»).

Рисунок 59 – Обозначение операции «запросить_Счет_Клиента»

Квантор видимости для операции может быть опущен. В этом случае его отсутствие означает, что видимость операции не указывается.

4. Отношения между классами

Классы на диаграмме связываются различными типами отношений. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена их семантикой.

4.1. Отношение зависимости

Отношением зависимости (dependency relationship) называют связь по использованию, когда изменение в спецификации одного класса может повлиять на поведение другого. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели. Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов, направленной к той сущности, от которой зависит данная сущность. Например, некая сущность Класс_2 использует другую сущность Класс_1 (рис. 6).

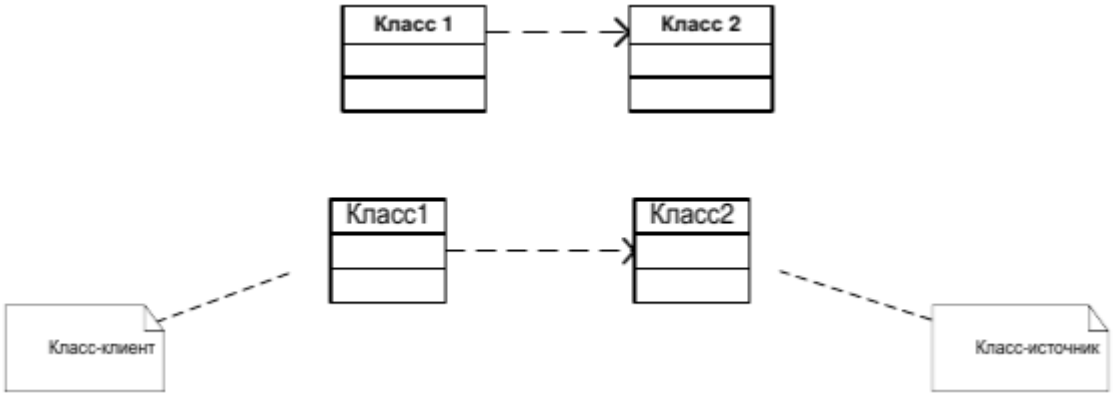


Рисунок 60 – Обозначение отношения зависимости

В качестве класса-клиента и класса-источника зависимости может выступать множество элементов модели. В этом случае одна линия со стрелкой, выходящая от источника зависимости, расщепляется в некоторой точке на несколько отдельных линий, каждая из которых имеет отдельную стрелку для класса-клиента.

Например, если функционирование сущности Класс_С зависит от особенностей реализации сущностей Класс_А и Класс_Б, то данная зависимость может быть изображена следующим образом (рис. 7).

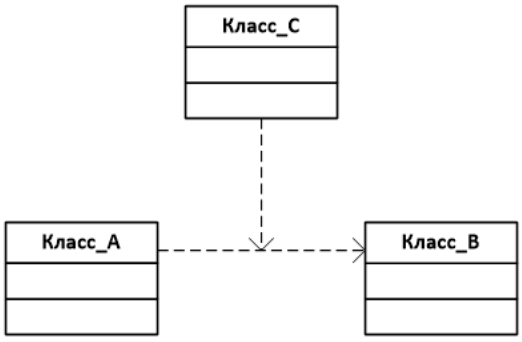


Рисунок 61 – Обозначение зависимости между классом-клиентом (Класс_С) и классами-источниками (Класс_А и Класс_В)

4.2. Отношение ассоциации

Ассоциацией (association relationship) называется структурная связь, показывающая, что объекты одного класса некоторым образом связаны с объектами другого или того же самого класса.

Ассоциация может отображаться графически линией со стрелкой (маркером в виде треугольника), показывающей направление следования классов и кратность – количество объектов, связанных отношением. Отсутствие стрелки рядом с именем ассоциации означает, что порядок следования классов в рассматриваемом отношении не определен (рис. 8).

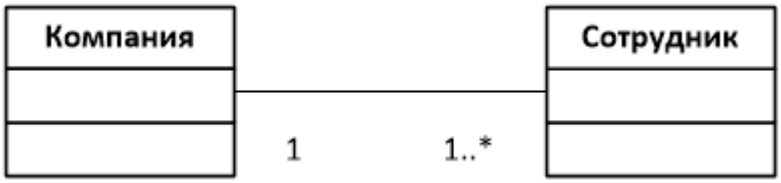


Рисунок 62 – Обозначение отношения ассоциации

Так, в примере на рис. 8 кратность «1» для класса «Компания» означает, что каждый сотрудник может работать только в одной компании. Кратность «1..*» для класса «Сотрудник» означает, что в каждой компании могут работать несколько сотрудников, общее число которых заранее неизвестно и ничем не ограничено.

Специальной формой или частным случаем отношения ассоциации является отношение агрегации, которое, в свою очередь, тоже имеет специальную форму – отношение композиции (см. пункт 3.4.4).

4.3. Отношение агрегации

Отношение агрегации (generalization relationship) имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа «часть – целое».

Это отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты декомпозиции, если в этом возникнет необходимость в последующем.

Так, автомобиль состоит из кузова, двигателя, трансмиссии и т.п., а в состав приемопередающего устройства входят передатчик, приемник и антенно-фидерное устройство.

Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой геометрическую фигуру – ромб. Этот ромб указывает на тот из классов, который представляет собой «целое» (рис. 9).

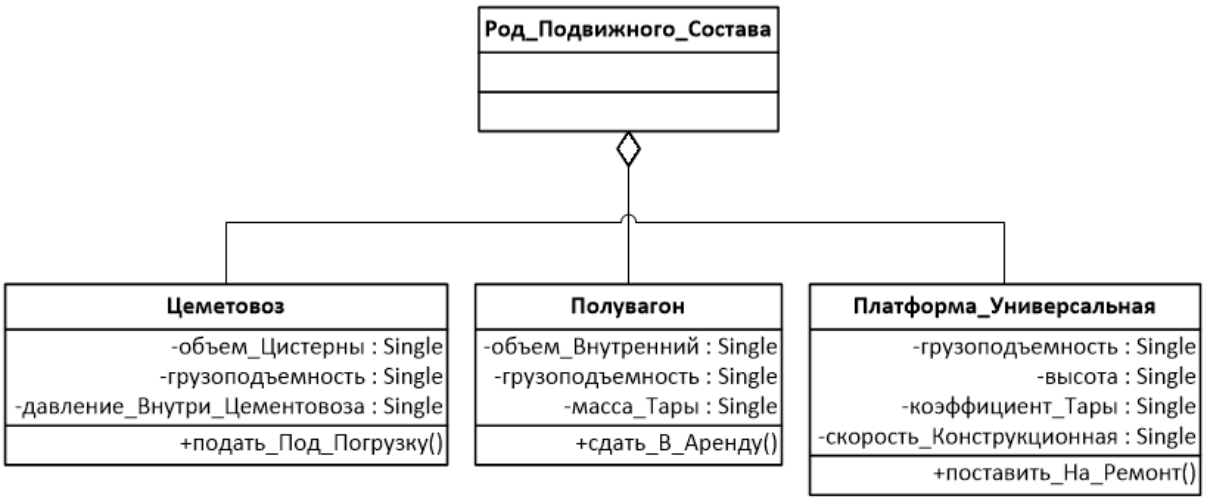


Рисунок 63 – Обозначение отношения агрегации

Примером отношения агрегации может служить деление класса Аналитическая_информация на составные части: Отчет_по_грузу, Отчет_по_контейнерам, Отчет_по_тарифам (рис. 10).

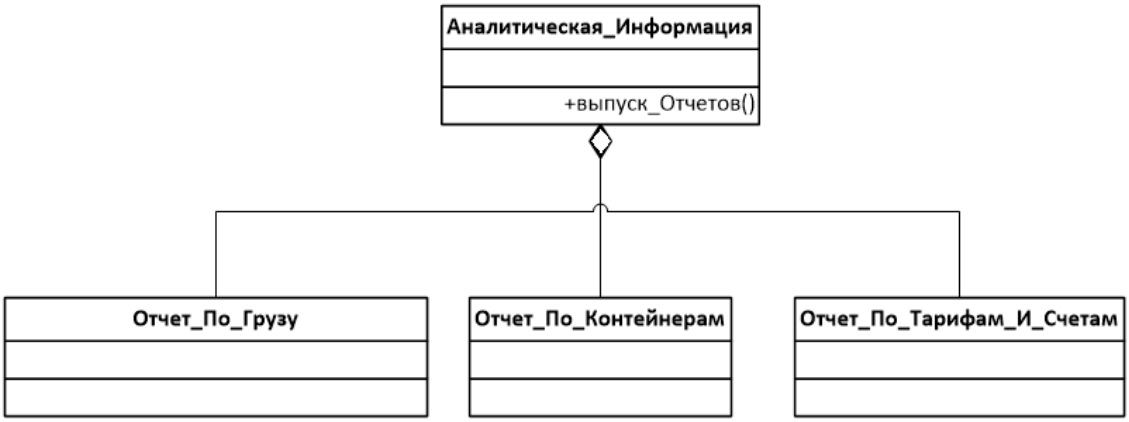


Рисунок 64 – Пример отношения агрегации

Отношение агрегации обладает кратностью. Так, класс Система_обеспечения_безопасности_объектов может содержать содержит одну подсистему Сопровождение_грузов, которая в свою очередь может содержать, например, четыре класса Охрана_вооруженная, каждый из которых может принадлежать лишь одному классу Сопровождение_грузов (рис. 11).

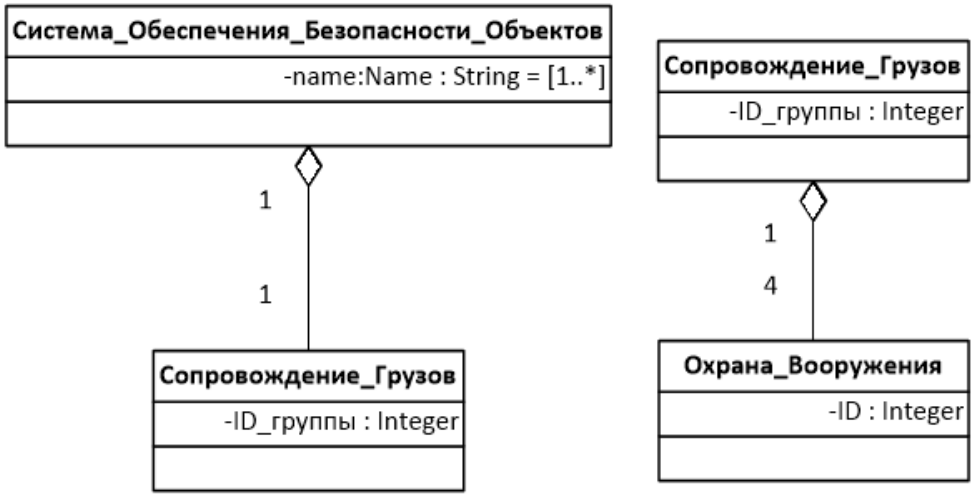


Рисунок 65 – Пример обозначения кратности отношения агрегации

4.4. Отношение композиции

Отношение композиции (realization relationship) служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого.

Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой класс-композицию или «целое» (рис. 12).

Рисунок 66 – Обозначение отношения композиции

Применительно к классу Заказ_на_перевозку_грузов отношение композиции может иметь следующий вид (рис. 13).

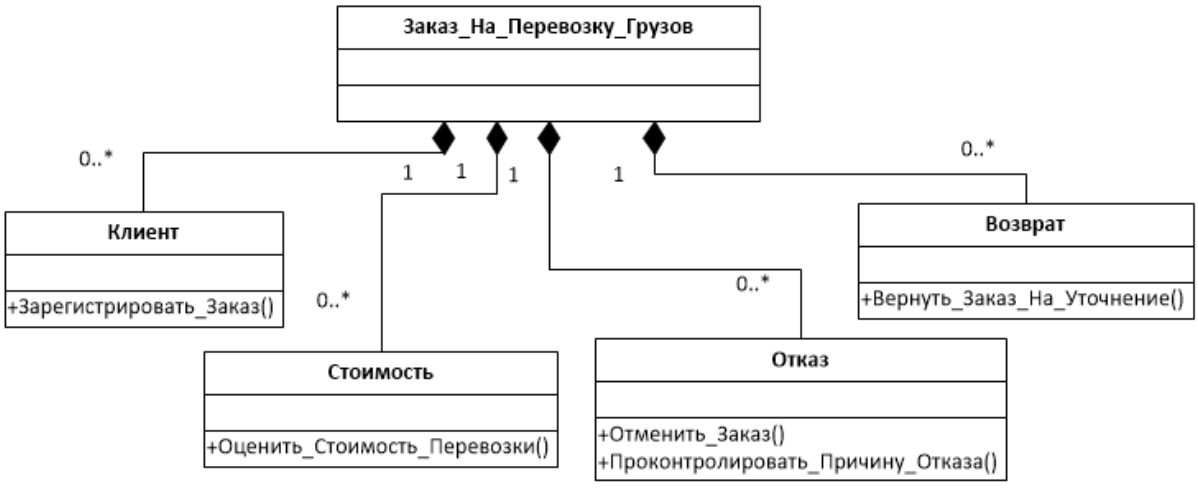


Рисунок 67 – Обозначение на диаграмме отношения композиции

4.5. Отношение обобщения

Отношение обобщения (генерализация) является обычным таксономическим отношением между более общим элементом (класс-предок) и более частным или специальным элементом (класс-потомок).

Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.

На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов, направленной на более общий класс (класс-предок или суперкласс) от более специального класса (класса-потомка или подкласса) (рис. 14).

Как правило, на диаграмме может указываться несколько линий для одного отношения обобщения, что отражает его таксономический характер. В этом случае более общий класс разбивается на подклассы одним отношением обобщения, например, так, как показано на рис. 14.

В этом случае данные отдельные линии изображаются сходящимися к единственной стрелке, имеющей с ними общую точку пересечения. Родительский Класс Отчет_По_Заказам_На_Перевозку имеет три потомка Отчет_По_Количеству_Заказов, Отчет_По_Клиентам, Отчет_За_Период, которые наследуют структуру и поведение родительского класса.

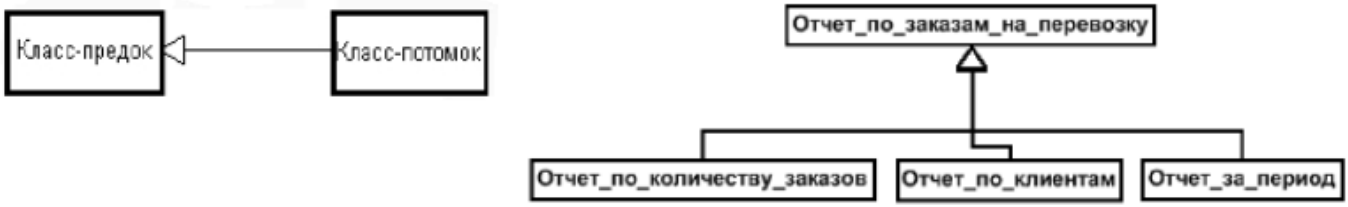


Рисунок 68 – Обозначение на диаграмме отношения обобщения

Для связей обобщения язык UML содержит ограничения. В большинстве случаев ограничение размещается рядом с элементом и заключается в фигурные скобки, например {complete}.

В качестве ограничений могут быть использованы следующие ключевые слова языка UML:

1. {complete} означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.

Например, класс Клиент_банка является предком для двух классов: Физическое_лицо и Компания, и других классов-потомков он не имеет.

На соответствующей диаграмме классов это можно указать явно, записав рядом с линией обобщения данную строку-ограничение (рис. 15).

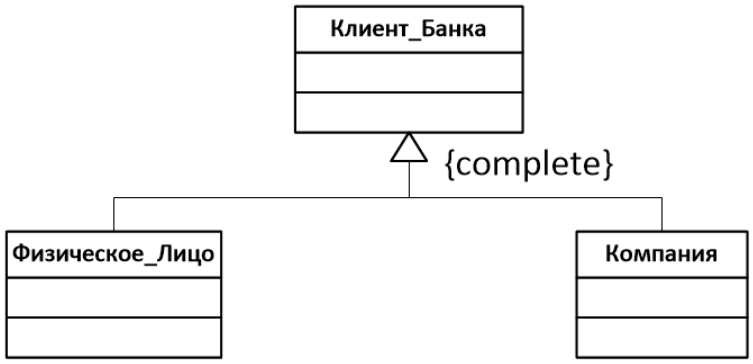


Рисунок 69 – Обозначение ограничения {complete} отношения обобщения

2. {incomplete} означает тот факт, что на диаграмме указаны в обобщении не все классы-потомки. В последующем, возможно, восполнить их перечень, не изменяя уже построенную диаграмму.

3. {disjoint} означает тот факт, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов.

В приведенном выше примере это условие также выполняется, поскольку предполагается, что никакое конкретное физическое лицо не может являться одновременно и конкретной компанией. В этом случае рядом с линией обобщения можно записать данную строку-ограничение.

4. {overlapping} означает, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

Например, класс Транспорт может быть специализирован путем создания подклассов Наземный_Транспорт и Водный_Транспорт, автомобиль – амфибия относится к обоим классам.

3 Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите MS Visio.
2. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
3. Окно программы примет вид, подобный рис. 16.

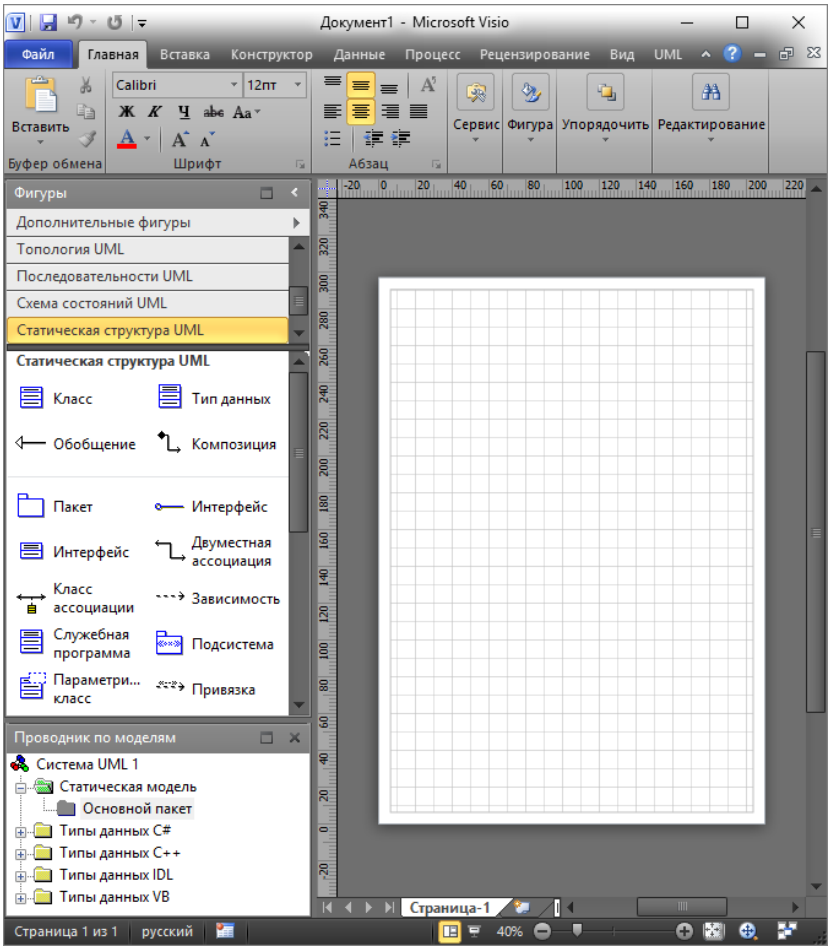


Рисунок 70 – Схема модели UML в MS Visio

4. Ознакомьтесь с элементами графического интерфейса и найдите обязательные панели инструментов **Фигуры**, содержащие категории **Деятельность UML, Взаимодействия UML, Компоненты UML, Топология UML, Последовательности UML, Схема состояний UML, Статическая структура UML, Сценарий выполнения UML, Проводник по моделям**, содержащий иерархическую структуру объектов Системы UML1, Рабочую область, ярлык Страница_1, горизонтальную и вертикальную линейки. (рис. 17).

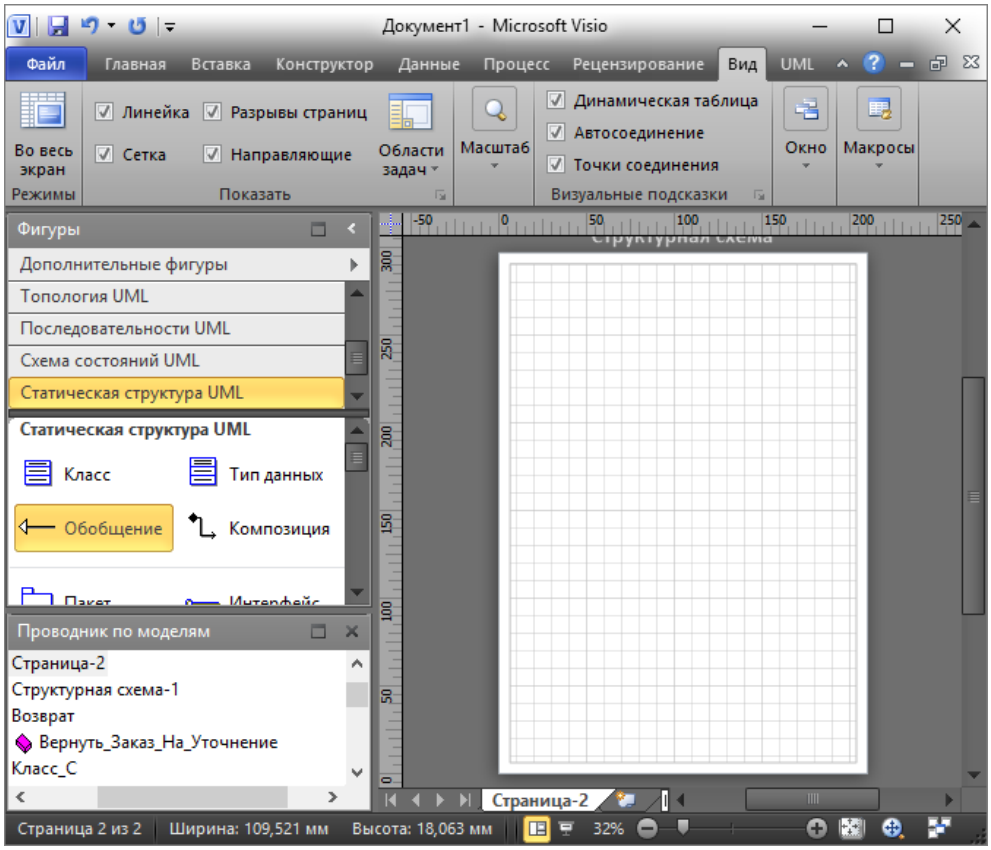


Рисунок 71 – Добавление фигур UML

5. Установите следующие параметры страницы: **Ориентация** – Альбомная, **Автоподбор размера** – выключен, **Имя страницы** – *Диаграмма классов для системы продажи товаров по каталогу.*

6. Перейдите в категорию **Статическая структура UML**, ознакомьтесь с содержимым этой категории и найдите элементы: Класс, Пакет, Подсистема, Интерфейс, Метакласс, Двусторонняя ассоциация, Обобщение, Композиция, Примечание, Ограничение и др.

7. Создайте поэтапно статическую структуру классов UML, с помощью которой может быть сформирована некоторая функциональная часть системы, например, *Система продажи товаров по каталогу*. Для чего:

– Выберите структурные элементы (идентифицируйте классы), участвующие в организации продаж, например, *Продавец, Товар, Заказ, Заказ_Оплата, Клиент, Корпоративный_Клиент, Частный_Клиент* и создайте предварительный вариант совокупности классов с указанием имен (один из возможных вариантов представлен на рис. 18).

– Установите для каждого класса атрибуты в соответствии с перечнем и содержательным описанием бизнес-процессов:

например, для класса *Продавец* в качестве атрибутов могут выступать данные: *фамилия, имя, отчество, телефон*. В данном случае все атрибуты видимы, принадлежат основному пакету *Продавец* (рис. 19).

для класса *Товар* в качестве атрибутов могут выступать данные: тип, марка, артикул (рис. 20).

Рисунок 72 – Формирование статической структуры объектов диаграммы

Рисунок 73 – Описание атрибутов класса Продавец

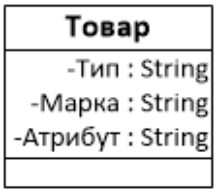


Рисунок 74 – Описание атрибутов класса Товар

для класса *Заказ* в качестве атрибутов могут выступать данные: количество, цена, статус, а в качестве операций – сформировать заказ.

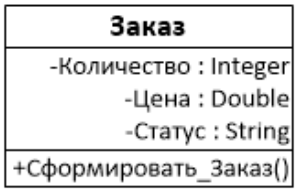


Рисунок 75 – Описание атрибутов и операций класса Заказ

для класса *Заказ_Оплата* в качестве атрибутов могут выступать данные: дата получения, проплачен, номер, цена, а в качестве операций – отправить, закрыть.

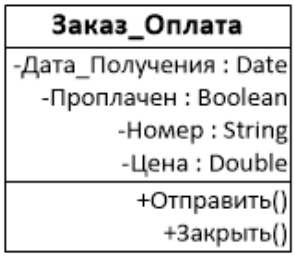


Рисунок 76 – Описание атрибутов и операций класса Заказ_Оплата

для класса *Клиент* в качестве атрибутов могут выступать данные: имя, адрес, а в качестве операций – кредитный рейтинг.

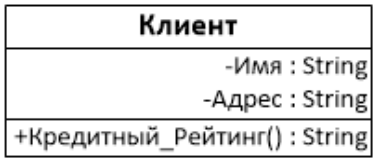


Рисунок 77 – Описание атрибутов и операций класса Клиент

для класса *Корпоративный_Клиент* в качестве атрибутов могут выступать данные: контактное имя, кредитный рейтинг, кредитный лимит, а в качестве операций – сделать, напоминание, счет за месяц.

Рисунок 78 – Описание атрибутов и операций класса Корпоративный_Клиент

для класса *Частный_Клиент* в качестве атрибутов могут выступать данные: номер кредитной карты.

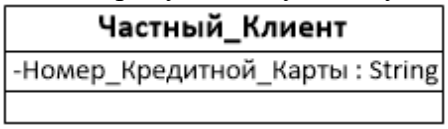


Рисунок 79 – Описание атрибутов класса Частный_Клиент

для класса *Вариант_Оплаты* в качестве атрибутов могут выступать данные: тип оплаты, а в качестве операций – выбор варианта оплаты.

Рисунок 80 – Описание атрибутов и операций класса Вариант_Оплаты

для класса *Каталог_Товаров* в качестве атрибутов могут выступать данные: тип, марка, артикул, а в качестве операций – проверить наличие.

Рисунок 81 – Описание атрибутов и операций класса Каталог_товаров

для класса *Склад* в качестве атрибутов могут выступать данные: товар, наличие, количество, а в качестве операций – Проверить наличие.

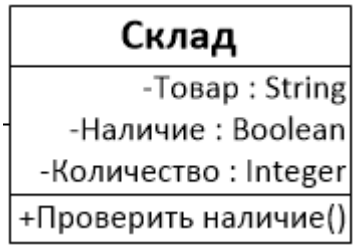


Рисунок 82 – Описание атрибутов и операций класса Склад

– Убедитесь, что все элементы наполнены адекватным содержанием и расположите все структурные элементы диаграммы наиболее оптимально на странице для установления отношений между ними.

В качестве примера на рис. 26 показан набор классов, описывающих реализацию системы продаж товаров по каталогу. Акцент сделан на классе *Клиент*, с которым связан класс *Заказ_Оплата*

посредством двусторонней ассоциации «один-ко-многим», *Вариант_Оплаты* – двусторонней ассоциацией «один-к-одному» и классы *Корпоративный_Клиент* и *Частный_Клиент* посредством отношения обобщения. Классы *Заказ_Оплата* и *Товар* связаны с классом *Заказ* посредством двусторонней ассоциации «один-ко-многим». Класс *Товар* связан с классом *Продавец* двусторонней ассоциацией «многие-ко-многим» и классом *Каталог_Товаров* двусторонней ассоциацией «один-ко-многим». Класс *Каталог_Товаров* связан посредством двусторонней ассоциации «многие-ко-многим» с классом *Склад.*

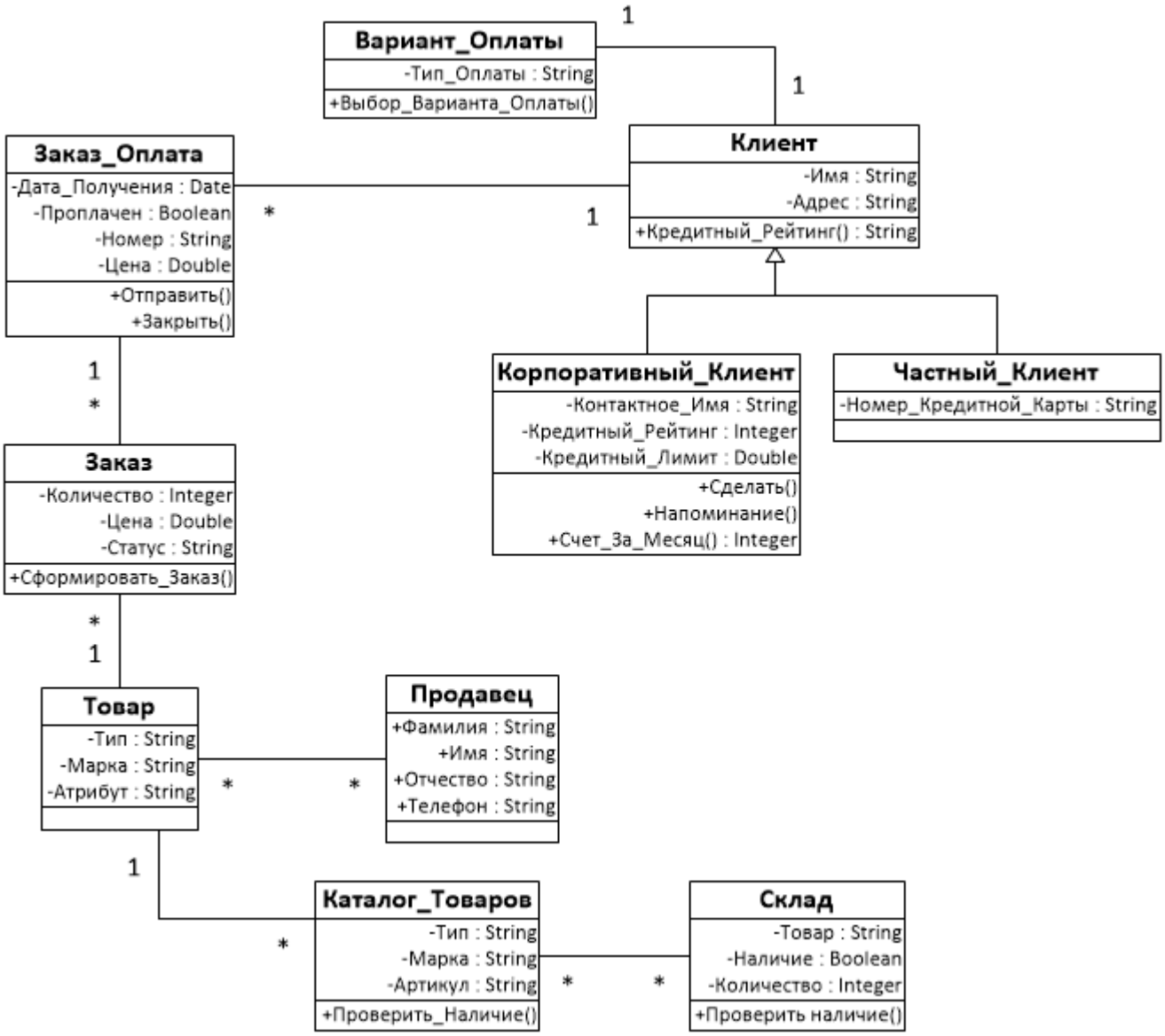


Рисунок 83 – Фрагмент диаграммы классов, описывающей реализацию систем продаж товаров по каталогу

8. Создайте новую страницу с именем *Диаграмма классов учета клиентов*, и установите следующие опции: **Ориентация** – Альбомная, **Автоподбор размера** – выключен.

9. Идентифицируйте классы учета клиентов, осуществляющих заказы и создайте диаграмму классов с указанием их имен, атрибутов, операций, например, так как показано на рис. 30.

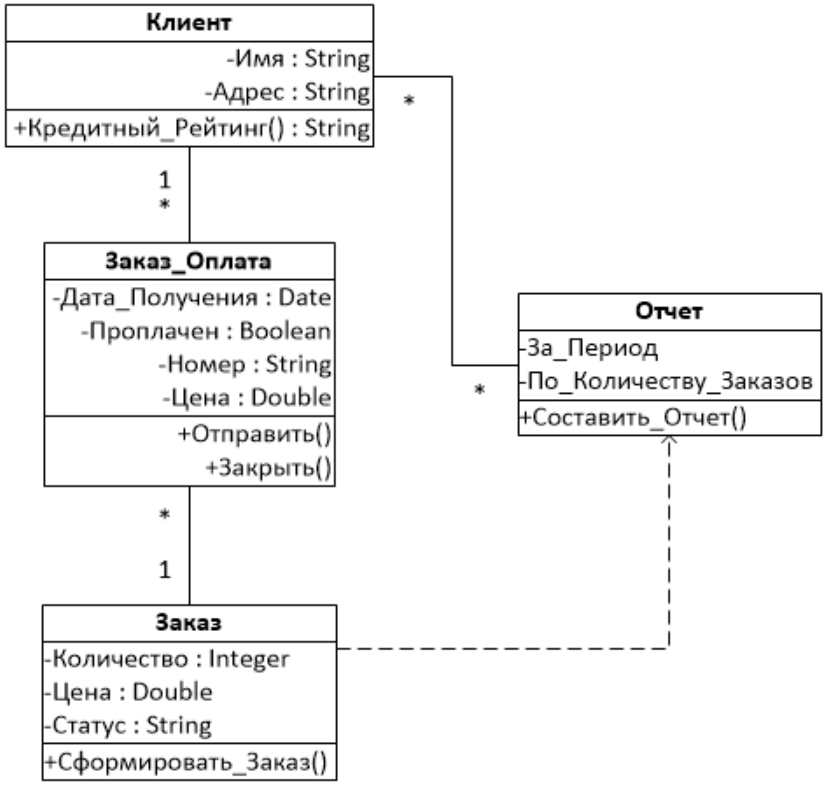


Рисунок 84 – Фрагмент диаграммы классов, описывающей процесс формирования отчетности

4 Задание

Построить диаграмму классов в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

5 Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. «Таксопарк».
11. «Издательство»;
12. «Прокат велосипедов»;
13. «Спортивный клуб».

6 Контрольные вопросы

1. Каково назначение диаграммы классов?
2. Назовите основные элементы диаграммы классов.
3. Какие виды связей доступны в диаграмме классов?
4. Для чего используется каждый вид связи?
5. Как создать диаграмму классов в VISIO?