МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«12» 09 2025г.

Управление проектированием информационных систем

Методические указания по выполнению
лабораторных работ по дисциплине «Управление проектированием информационных систем» для обучающихся по направлению подготовки 09.04.01 Информатика и вычислительная техника направленность (профиль, специализация) "Элементы и устройства цифровой техники и информационных систем"

*Дисциплина реализуется по модели «перевернутого обучения»

Курск 2025

УДК 004.82 (075.8)
Составитель Т.И.Лапина
Рецензент
Кандидат технических наук, доцент *Е.А.Петрик*

Управление проектированием информационных систем

методические указания по выполнению лабораторных работ по дисциплине «Управление проектированием информационных систем» / Юго-Зап. гос. ун-т; сост.: Т. И. Лапина, Курск, 2025. 89с. ил. 20, табл.3, Билиогр.: с89.

Содержат краткие теоретические сведения о разработке требований к проекту ИС, методологиях проектирования и подходах к организации работ над проектом информационных систем. Большое внимание уделено стандартам проектирования информационных систем. Рассмотрены различные методологические подходы к проектированию ИС и соответствующие инструментальные средства.

Предназначены для студентов направления подготовки 09.04.01 Информатика и вычислительная техника направленность (профиль, специализация) "Элементы и устройства цифровой техники и информационных систем".

Текст печатается в авторской редакции
Подписано в печать 7.09.25 Формат 60x84 1/16.
Усл. печ. л.0,9 . Уч. – изд. л.0.8 .Тираж 100 экз. Заказ.
Бесплатно.
Юго - Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1

Разработка требований к проекту ИС

1. Цель работы

Получить навыки описания и анализа требований к проектируемой информационной системе, формализации и кодирования информации. Приобретение навыков анализа полученных материалов для последующего моделирования.

2. Основные теоретические положения

Предпроектное обследование включает в себя два этапа: сбор информации, первичная ее обработка.

Первичное обследование затрагивает два основных момента, характеризующих предприятие:

- решаемые задачи и выполняемые функции – на основе технологии анкетирования;
- информационные потоки и связи, как внутренние, так и внешние – с помощью реестров.
- Согласно предложенной методологии построения информационной модели, обследование всех аспектов деятельности предприятий проводится на различных уровнях управления:
- предприятие (директор);
- службы и отделы (начальники отделов и служб);
- рабочие группы и сектора (начальники групп и секторов в рамках отделов и служб);
- отдельные рабочие места.

Таким образом, для полного охвата деятельности подразделений предприятия необходимо участие в обследовании всех перечисленных сотрудников.

Среди универсальных методов, пригодных для обследования всех функциональных звеньев предприятия, наиболее важными являются следующие методы:

- наблюдения;
- опрос исполнителей (метод интервью);
- анализа материала (анкетирование);

– личного участия.

Обследование предприятия – очень важный этап для проведения анализа. От полноты данных, полученных при проведении обследования, зависит адекватность информационной модели предприятия, а, следовательно, в дальнейшем, архитектура проектируемой системы.

Для построения информационной модели по выбранной методологии при обследовании необходимо собрать информацию, которая наиболее полно характеризует бизнес-процессы, происходящие в организации.

Для этого согласно предложенной организационной модели на каждом уровне управления необходимо сформировать перечень целей, задач и функций.

Для формализации функциональных и информационных связей между предприятиями и подсистемами необходимо сформировать и обработать перечень документов, данных в электронном виде, циркулирующих на предприятии.

Таким образом, необходимо провести обследование и формализацию:

– организационных структур подразделений предприятия;
– управленческих процедур (цели, задачи, функции управления);
– информационных потоков.

Для эффективного проведения обследования необходимо, чтобы сотрудники предприятия понимали цель и методологию проводимых работ. Для этого организуются обучающие семинары, на которых сотрудникам объясняется перечень, порядок и содержание проводимых работ. Подробно объясняется методика заполнения раздаточного материала.

В дальнейшем, по мере проведения работ, организовываются рабочие встречи с сотрудниками, для проведения консультаций и выдачи дополнительных рекомендаций.

Информационный анализ собранной документации.

Собранная информация позволяет получить представление о принципах функционирования каждого из предприятий, участвовавших в обследовании. Таким образом, от того, насколько полно были отображены процессы, протекающие на предприятии, в

собранной документации, зависит полнота и достоверность создаваемой модели.

Первым критерием при оценке собранного материала является количественная характеристика. Целью такого анализа может быть определение необходимых технических средств для построения автоматизированной системы управления (АСУ): каналов передачи данных, средств вычислительной техники. Так же на основании такого анализа можно оценить информационную загруженность подразделения в целом, отдельных отделов и служб.

Качественная оценка представленного материала может быть дана на основании следующих параметров: качество заполнения и систематизированность материала.

По этим данным можно оценить объем информационных потоков, циркулирующих на предприятии. По полученным материалам можно судить о том, что сотрудники предприятия перегружены работой с документацией. Необходимо при проектировании АСУ учесть необходимость снижения нагрузки на работников, увеличения эффективности их работы.

Большинство документов может быть либо внутренними, либо исходящими, предназначенными другим филиалам, т.е. могут быть унифицированы и автоматизированы в рамках создаваемой АСУ. Это позволит значительно увеличить оперативность и эффективность работы персонала.

При обобщении данных по направлению передачи (получения) информации может быть получена следующая картина:

Тип информации	Процент от общего числа, %
Внутренняя информация подразделения	59,15
Другие филиалы	9,52
Управление предприятием	8,86
Внешние организации	22,47

Таким образом, основная часть информации (59%) является внутренней и не покидает пределов предприятия. Это означает, что большая часть информационных потоков может быть унифицирована

и, в дальнейшем, автоматизирована. Большую часть внешней информации составляет документооборот с организациями, не относящимися к предприятию. Эта часть документооборота практически не поддается стандартизации. Однако наиболее важная часть с точки зрения управления предприятия – это информация, которой обмениваются подразделения между собой и с управлением. Именно эта часть документооборота может быть автоматизирована в первую очередь.

Например по составу информации, участвующему в документообороте, можно выделить следующие наибольшие потоки:

Тип информации	Количество	Процент от общего числа, %
Оперативные данные	436 034	41,07
Отчетная документация	221 850	22,42
Служебная информация	220 503	20,22
Управляющая информация	111 080	11,23
Прочая информация	49 871	5,04

Анализ показывает, что основную часть информации составляют оперативные данные, которые в большинстве своем (95,53%) передаются в бумажном виде, что не может сказаться на скорости прохождения и трудоемкости их обработки. Так же большую часть составляет отчетная документация (отчеты, своды, аналитические данные, калькуляции), автоматизированное формирование и передача этой документации может значительно повысить эффективность работы предприятия.

При этом время оперативного использования характеризуется следующим образом:

Время оперативного использования	Процент от общего числа, %
Сутки	3,27
Неделя и меньше	14,31
Месяц и меньше	51,56

Год и меньше	22,42
Больше года	8,44

Возможно проведения анализа загруженности отдельных отделов и служб в целом на предприятии:

Наименование отделов, служб	Процент от общего числа документов, %
Отдел бухучета (ОБУ)	32,57
Административно-хозяйственный отдел (АХО)	7,88
Оперативно-диспетчерская служба (ОДС)	4,33
Служба механизации и транспорта (СМиТ)	4,03
Отдел договорной работы (ОДР)	3,17

На основании приведенных данных можно сделать вывод о том, что наиболее загруженными работой с документацией являются отделы бухучета и административно-хозяйственные. Таким образом, данные отделы в наибольшей степени нуждаются в автоматизации.

Также можно произвести оценку средней загруженности отделов и служб на предприятии. Данный показатель характеризует среднее количество обрабатываемой информации, приходящееся на каждый отдел или службу предприятия. Чем выше этот показатель, тем более сконцентрирован документооборот на предприятии, тем больше нагрузка на отделы. Низкое его значение показывает, что процессы, протекающие на предприятии, не документируются достаточным образом.

Другим показателем эффективности документооборота может являться коэффициент дублирования информационных потоков. Чем выше этот коэффициент – тем ниже полезная нагрузка на сотрудников.

Наименование подразделений предприятия	Средняя загрузка отделов и служб	Коэффициент дублирования, %
Подразделение №1	12 177	18,24

Наименование подразделений предприятия	Средняя загрузка отделов и служб	Коэффициент дублирования, %
Подразделение №2	22 994	24,99
Подразделение №3	23 695	25,85
Подразделение №4	13 342	18,42

Можно отметить значительную среднюю загрузку всех подразделений, которая составляет в среднем около 1 500 документов в месяц, что говорит о необходимости перехода на автоматизированную обработку и передачу информации.

Также можно отметить высокий процент дублирования документооборота, что при использовании бумажных носителей значительно снижает эффективность работы. Дублирование возникает при большом количестве однородных документов. Дублирование возникает и в случаях, когда документация составляется неоднократно, в нескольких копиях и передается в различные пункты назначения. Это может бы вызвано как производственной необходимостью, так и недостатками организационной структуры. Высокий коэффициент дублирования (более 30%) дает основание говорить о необходимости оптимизации документооборота и оргструктуры при построении АСУ.

На основании собранных данных можно рассчитать и ряд параметров, характеризующих силу внешних связей структурного подразделения.

Коэффициент информационного выхода показывает отношение выходящей информации к общему объему, обрабатываемому в структурном подразделении.

Коэффициент связи показывает, насколько сильно связан данный структурный элемент с прочими элементами оргструктуры. Чем он выше, тем более сильна такая связь, и следовательно – зависимость от внешних факторов.

Коэффициент передачи характеризует отношение исходящей и входящей информации. На его основании можно сделать вывод о том, является ли структурный элемент «генерирующим» информацию или он предназначен для её обработки.

Например:

Наименование филиала, ДЗО ОАО	Коэффициент информации выхода, %	Коэффициент связи, %	Коэффициент передачи, %
Подразделение №1	13,85	25,74	1,08
Подразделение №2	14,68	27,97	1,11
Подразделение №3	16,34	27,04	1,21

Коэффициент информационного выхода характеризует насколько полно информация (поступающая и внутренняя) отражается в выходных данных данного подразделения. Низкое значение данного параметра говорит о том, что некоторые процессы, протекающие на предприятии, могут не отражаться в отчетной документации должным образом. Нормальное значение данного коэффициента ~ 15%, значительное превышение данного параметра свидетельствует о перегруженности отчетной документацией.

Степень взаимодействия с внешней средой характеризует коэффициент связи. Чем он выше, тем более зависимо, с информационной точки зрения, предприятие от внешних воздействий.

Коэффициент передачи показывает, является ли предприятие «производителем» информации или ее потребителем. Значения большие единицы говорят о том, что из предприятия выходит информации больше чем поступает, т.е. оно осуществляет какие-то производственные процессы, отражаемые в документации. В случае значения меньше единицы структурный элемент в основном «пользуется» поступающей информацией.

Собранные материалы позволяют проведение анализа документации, которая исходит из подразделений в управление и другие филиалы.

Этой информация наиболее важна с точки зрения управления, так как она характеризует работу предприятия.

Например:

Название отдела, службы	Процент от общего числа документов, %
Отдел бухучета (ОБУ)	18,35
Административно-хозяйственный отдел (АХО)	11,98
Руководство	10,91
Оперативно-диспетчерская служба (ОДС)	9,05
Оперативно-режимный отдел(ОРО)	8,35

Наибольшая доля таких документов исходит из отдела бухгалтерского учета, административно-хозяйственного отдела и руководителей, что говорит о том, что наиболее сильная связь между предприятиями заключена в административно–финансовой сфере, т.е. первоочередное внимание при автоматизации необходимо уделить именно этой части документооборота.

Так же значительную часть информационных потоков составляют оперативно-диспетчерские данные.

Проанализируем состав этих информационных потоков по типам документов:

Название документа	Процент от общего числа документов, %
Письма и приказы	13,94
Акты	10,29
Заявки	8,62
Отчеты	8,06
Ведомости	4,91

Можно отметить, что наибольшее количество в процентном отношении составляют письма и приказы, т.е. при автоматизации этих потоков не встает задача реализации расчетных задач, а, главным образом, тщательной организации документооборота между предприятиями.

Если провести аналогичный анализ внутренней информации, то можно заметить большой процент, приходящийся на долю бухгалтерского и административного блока, а также возросшую долю оперативных данных, касающихся производственной деятельности

предприятия. При построении частей АСУ внутри каждого предприятия необходимо реализовать подсистемы оперативного управления производственными процессами.

3. Задание на практическую работу

Рассмотрим процесс выполнения лабораторной работы на примере нулевого варианта.

Нам необходимо закодировать документы, входящие в реестры. Для этого проанализируем каждый документ и выберем те критерии, которые будут подлежать кодированию:

- задачи и функции структурных элементов;
- предприятия/отделы/сектора/рабочие места, в которые входит и исходит информация;
- тип документа;
- направление передачи (входящий, исходящий, внутренний);
- характеристики его использования (время оперативного использования, срок хранения).

Далее создаем базу данных, в которую будем заносить кодированную информацию. База данных должна включать связанные таблицы, в которых помимо базы, включающей перечень документов, кодированная информация должна представляется разграниченной по критериям.

В состав базы данных входят следующие таблицы:

- справочник кодов задач;
- справочник кодов функций;
- справочник предприятий;
- справочник отделов;
- справочник секторов;
- справочник рабочих мест;
- справочник типов документов;
- основная база по документам.

В справочники заносятся коды и их значения. В справочник предприятий занесены названия предприятий и их коды (табл. 1.1). Справочник отделов содержит названия отделов предприятия и их коды (табл. 1.2). Справочник секторов содержит названия секторов предприятия и их коды (табл. 1.3). Справочник рабочих мест содержит названия рабочих мест предприятия и их коды (табл. 1.4). В

справочнике кодов задач перечислены задачи подразделений и коды задач (табл. 1.5). В справочнике кодов функций перечислены функции подразделений и коды функций (табл. 1.6). Справочник типов документов содержит перечень типов документов и их коды (табл. 1.7).

Основная база по документам (табл. 1.8) включает полное кодированное описание всех документов подразделения.

В эту таблицу последовательно заносятся следующие параметры:

1. Закодированные данные о виде документа заносятся в колонку под названием “тип кода документа”.
2. В колонку “название документа” заносится полное имя кодируемого документа.
3. В “принадлежность” заносится информация о принадлежности кодируемого документа к определенному подразделению.
4. В колонку “направление документа” заносится информация о том, является ли документ входящим, исходящим или внутренним для данного подразделения.
5. В колонку “куда откуда” заносится информация о том, в какое подразделение направляется кодируемый документ или из какого подразделения он пришел.
6. Информация о количестве данных документов в год заносится в колонку “количество”.
7. В колонку “физ. представление” заносится информация о том, в каком виде приходит документ: электронное письмо, бумажный документ и т.д.
8. В колонку “время использования” заносят информацию о количестве времени, в течение которого кодируемый документ будет использоваться.
9. В колонку “срок хранения ” заносят информацию о времени, в течение которого документ будет храниться в архиве.
10. В колонку “код задачи и функции ” заносят информацию о задаче и функции данного отдела, к выполнению которой имеет отношение кодируемый документ.
11. В колонку “путь прохождения документов” заносят информацию о сотрудниках, имеющих доступ к кодируемому документу.

Таблица 1.1– Справочник предприятий

Код предприятия	Название предприятия
1	Западные электросети (ЗЭС)
2	Светлоградские электросети (СЭС)
3	Центральные электросети (ЦЭС)
4	Новотроицкие электросети (НЭС)

Таблица 1.2 – Справочник отделов

Код отдела	Название отдела
1	Отдел бухучета (ОБУ)
2	Планово-экономический отдел (ПЭО)
3	Производственно-технический отдел (ПТО)
4	Отдел управления персоналом (ОУП) или отдел кадров (ОК)

Таблица 1.3 –Справочник сектора

Код сектора	Названия сектора
56	Налогового и финансового учета
57	Учета заработной платы
58	Электротехническая лаборатория
59	Учета материалов и ОС, расчетов с поставщиками
60	Учета реализации электрической и тепловой энергии

Таблица 1.4– Справочник рабочего места

Код	Название РМ
35	Долтова
36	Тикунова
37	Кузнецова
38	Евланова (инженер-технолог)
39	Подстанция №1
40	Подстанция №2

Таблица 1.5 –Справочник задач

Код	Название задачи	Код-название отдела
УМ	Учет материалов	ОБУ СПЭР
УС	Учет основных средств	ОБУ ЗЭС
БУ	Документальный учет и анализ всех хозяйственных операций	ОБУ СЭС

Таблица 1.6 – Справочник функций

Код	Название функции	Код- название отдела
84	Начисление заработной платы	ОБУ ЗЭС
85	Учет первичных документов	ОБУ ЗЭС
86	Учет налоговых отчислений	ОБУ ЗЭС
87	Расчет с внешними организациями	ОБУ СПЭР

Таблица 1.7– Справочник типов документов

Код	Название типа документа
зп01	Докладная записка
дк01	Документ
дк03	Документ нормативный
рц01	Расценки сметные
рч01	Расчет
ув01	Уведомления
уд02	Удержание

Таблица 1.8 – Основная база по документам

Код типа документа	Название документа	Принадлежность	Направление	Куда-откуда	Кол-ичество	Физ. предста-вление	Время исполь-зования	Срок хране-ниея	Код задачи и функции	Путь прохожде-ния документов
фо01	Форма №2 НДФЛ	001.001.057.036	Ис.	049.000.000.000	1	Б+Д	3 года	З	УС86	Ф - бух., У-дир./гл. бух
ср01	Справки для исчисления пенсий	001.001.057.036	Ис.	001.042.000.000	180	Б	1 год	З	УС86	Ф-бух., У-дир./гл. бух
ср01	Справки для получения ссуд и др.	001.001.057.036	Ис.	001.042.000.000	180	Б	1 год	З	УС86	Ф-бух., У-дир./гл. бух
ли01	Больничный лист	001.001.057.036	Вх.	013.004.000.000	240	Б	1 год	З	УС85	Ф-бух., У-дир./гл. бух
та01	Табель	001.001.057.037	Ис.	001.000.035.000	540	Б	5 дней	З	УС85	О-бух.
та01	Табель	001.001.057.037	Ис.	001.000.036.000	540	Б	5 дней	З	УС85	О-бух.

3. Контрольные вопросы

1. Какие вопросы включает методика описания проектируемой ИС?
2. На каких уровнях проводится обследование аспектов деятельности предприятий?
3. Какие существуют универсальные методы, пригодные для обследования всех функциональных звеньев предприятия?
4. Какие существуют документы для описания ИС?
5. В каких направлениях выполняется информационный?
6. В чем заключается методика предпроектного обследования?
7. Какие существуют универсальные методы, пригодные для обследования всех функциональных звеньев предприятия?
8. Какие существуют характеристики документа?
9. Каким образом производится кодирование полученной документации?
10. На каких уровнях проводится обследование аспектов деятельности предприятий?
11. Что включает информационная база данных?
12. В каких направлениях выполняется информационный анализ предметной области?
13. Какие существуют типы реестров?
14. Что указывает анкетируемый, заполняя реестры?
15. В чем заключается методика сбора документации при обследовании?
16. Перечислите основные виды анализа.
17. Цель анализа полученной информации.

4. Содержание отчета

В отчете следует указать:

1. Цель работы

2. Введение. Краткое описание целей проекта и проектных ограничен
(бюджетных, временных и т.д.), которые важны для управления проектом

3. Описание проекта информационной системы.

4. Наличие заключения о возможности реализации проект
содержащего рекомендации относительно разработки системы, базов
предложения по объёму требуемого бюджета, числу разработчиков, время
и требуемому программному обеспечению

5. Анализ осуществимости (согласно требованиям к результата
выполнения лабораторного практикума п.2), указать возможные проблемы
пути их решения.

6. Роли участников группы разработки ПО.

7. Программно-аппаратные средства, используемые при выполнен
работы.

8. Заключение (выводы) и список используемой литературы

Лабораторная работа №2

Разработка модели бизнес-процессов проекта ИС

1. Цель работы

Приобретение навыков использования методологии IDEF0 пр разработке бизнес-модели компании, освоение приемов работы инструментальной средой бизнес-моделирования в нотации IDEF0 помощью MS Visio.

2.Основные теоретические положения

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм.

Каждая диаграмма является единицей описания системы и располагает на отдельном листе.

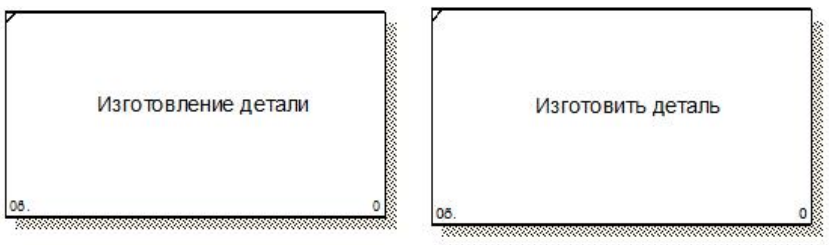
Цель моделирования. Модель не может быть построена без четко сформулированной цели. Пример цели:

«Описать функциональность предприятия с целью написан спецификаций ИС».

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 (рис. 1):



а) б)

Рисунок 1 – Примеры работ

Все функциональные блоки модели нумеруются.

Номер состоит из префикса и числа.

Может использоваться префикс любой длины, но обычно использует префикс А.

Контекстная (корневая) работа (функциональный блок) имеет номер А0.

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 обозначают существительными (например, «Заготовка», «Изделие») или именуемым сочетаниями (например, «Готовое изделие»).

В IDEF0 различают 4 типа стрелок (рис.2).

Каждая стрелка имеет свое расположение относитель функционального блока.

Рисунок 2 – Типы стрелок

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

1. Создание контекстной диаграммы
Запустите MS Visio.

На закладке выбора шаблона выберите категорию Блок-схема и в ней элемент Схема IDEF0.
Нажмите кнопку Создать в правой части экрана.
Окно программы примет вид, подобный рис. 3.

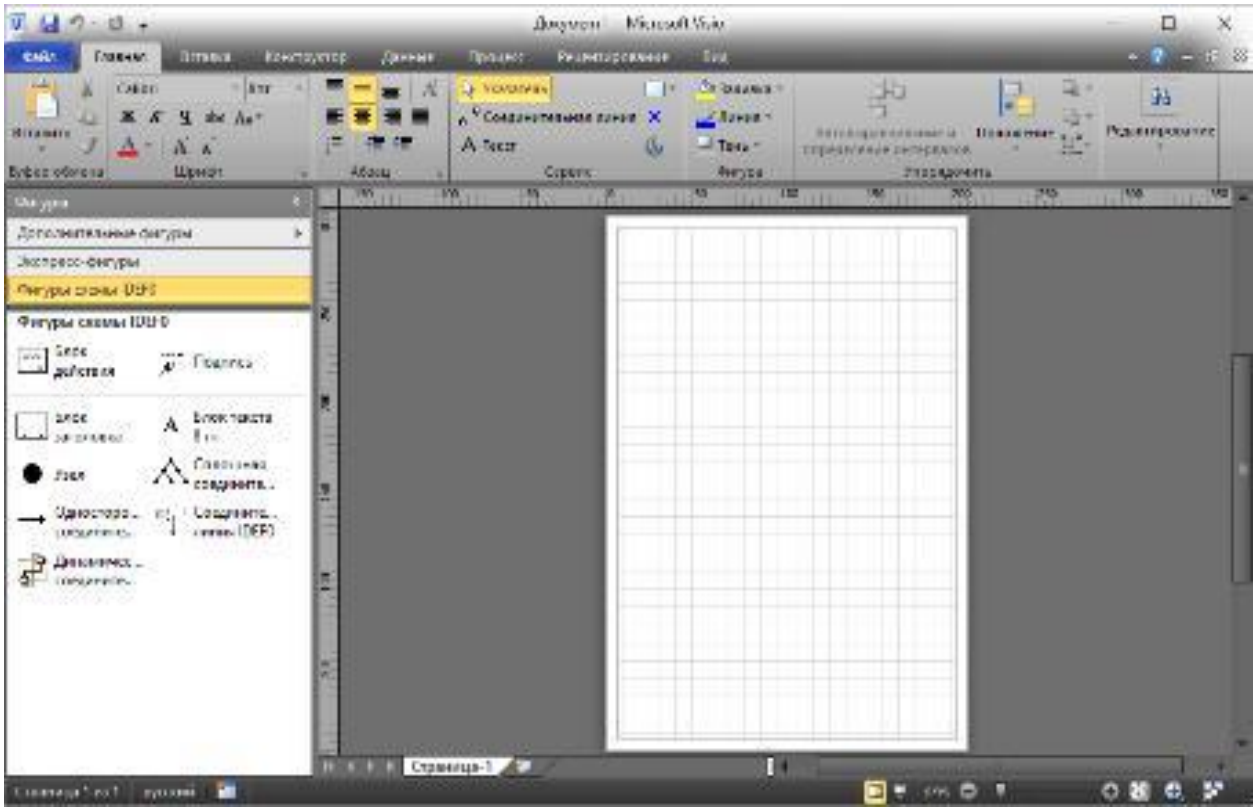


Рисунок 3 – Окно программы с выбранным шаблоном

В область диаграммы (поле Блока заголовка) внесите Блок действия. В открывшемся окне «Данные фигуры» внесите имя процесса и идентификатор процесса (рис. 6).

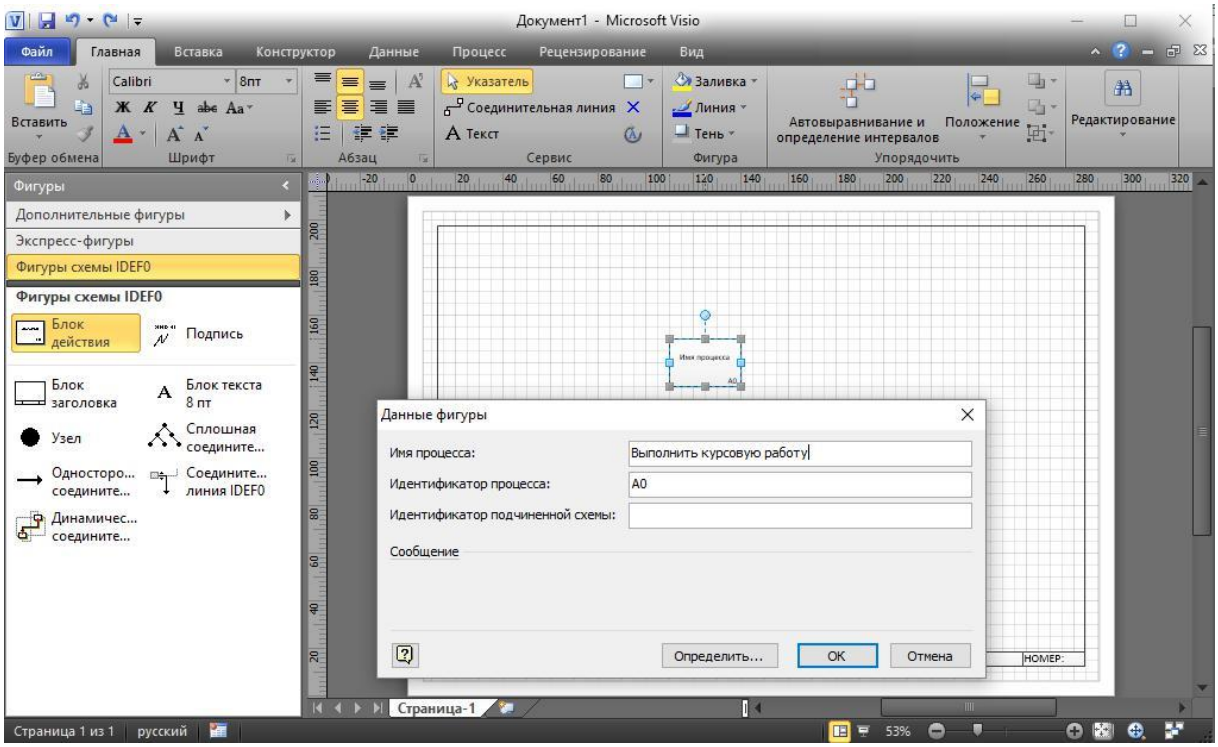


Рисунок 6 – Блок действия

С использованием блока Односторонняя соединительная линия создайте стрелки на контекстной диаграмме (табл. 1). Чтобы добавить текст необходимо дважды щелкнуть по стрелке.

Таблица 5 – Стрелки контекстной диаграммы

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

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

Результат выполнения предыдущих пунктов представлен на рис. 7.

Рисунок 7 – Контекстная диаграмма

2. Создание диаграммы декомпозиции

Для построения декомпозиции диаграммы создайте новую страницу путем нажатия правой кнопкой мыши нижнем левом углу окна на ярлык Страница 1.

Переименуйте страницы в соответствии с уровнем декомпозиции, например: А-0, А1 и т.д.

Распределите работы диаграммы декомпозиции в области Блока заголовка в соответствии с табл. 2.

5

Таблица 2 – Работы диаграммы декомпозиции А0

Имя работы (Activity Name)	Определение (Definition)
Получить задание	Выбрать задание из списка, согласовать его с преподавателем
Подобрать литературу	Выбрать из списка литературы подходящие источники
Сделать расчеты	Выполнить (если необходимо) расчетную часть работы согласно заданию
Сделать гграфическую часть	При необходимости сделать графики и чертежи
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Получить консультацию	Получить консультацию у преподавателя выявить неточности и недостатки
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы Преподавателя

4. Распределите стрелки для диаграммы декомпозиции в соответствии с контекстной диаграммой.

5. Для этого «перенесите» входные и выходные стрелки, связанные с декомпозируемой работой, в поле декомпозиции.

6. Итог выполнения вышеописанных шагов представлен на рис. 9.

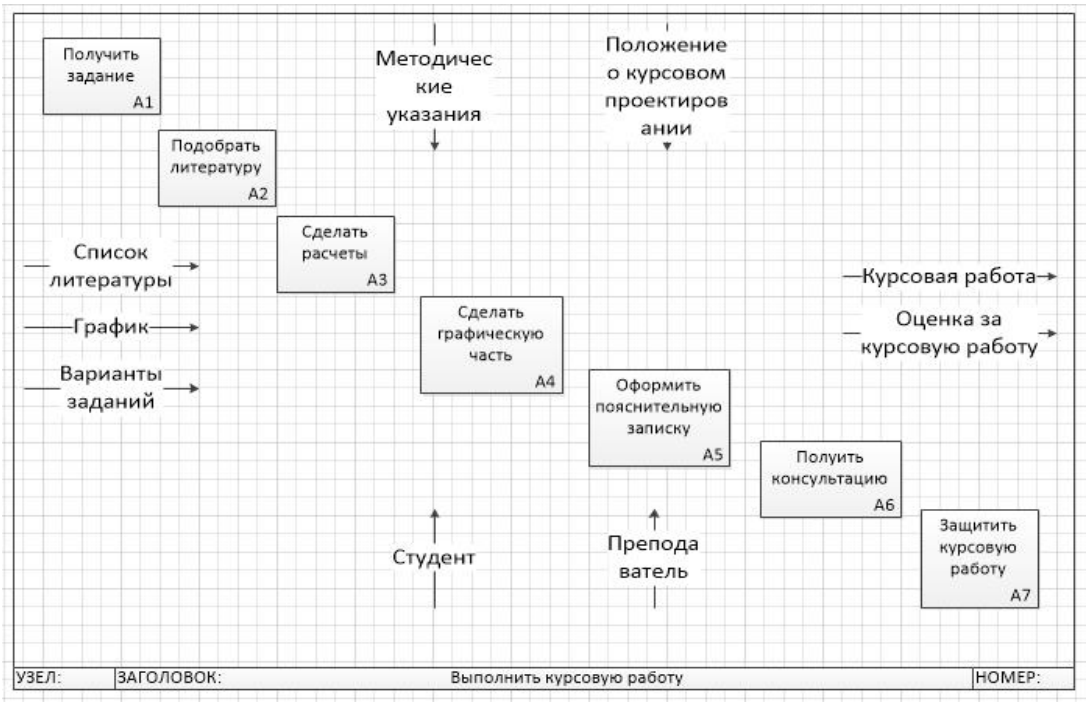


Рисунок 9 – Диаграмма декомпозиции

Разветвление стрелок. График (расписание) необходимо для того, чтобы прийти на консультацию и на защиту, т.е. необходимо подвести одноименную стрелку к 2 работам.
Для разветвления стрелки необходимо от фрагмента стрелки до сегмента работы провести стрелку, состоящую из нескольких блоков
Однонаправленное соединение.
Слияние стрелок. Для слияния двух стрелок выхода необходимо провести работы аналогичные разветвлению.
ICOM-метки. Используя блок текста, расставьте ICOM метки.

Результат выполнения предыдущих пунктов представлен на рисунке (рис. 10).

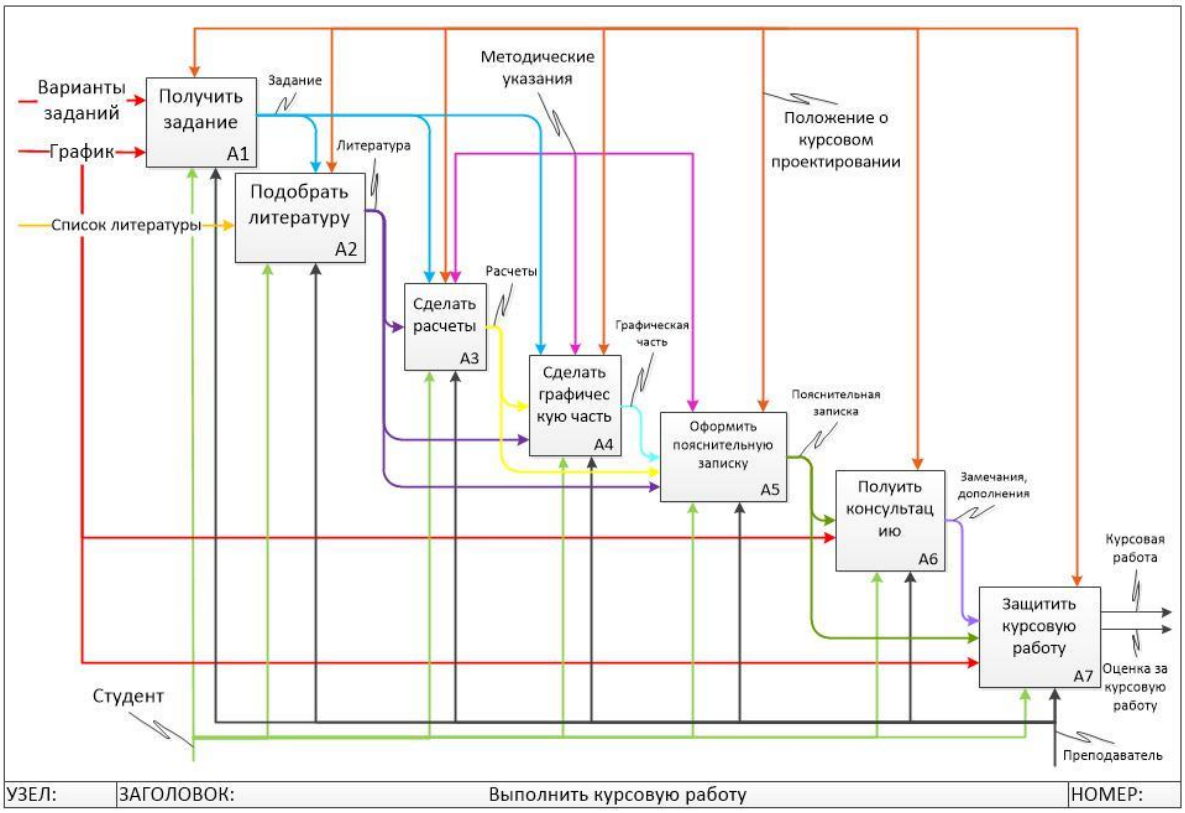


Рисунок 10 – Диаграмма декомпозиции блока A0

3.Создание дерева узлов

Дерево узлов – это диаграмма, отображающая иерархию работ процесса (рис. 11).

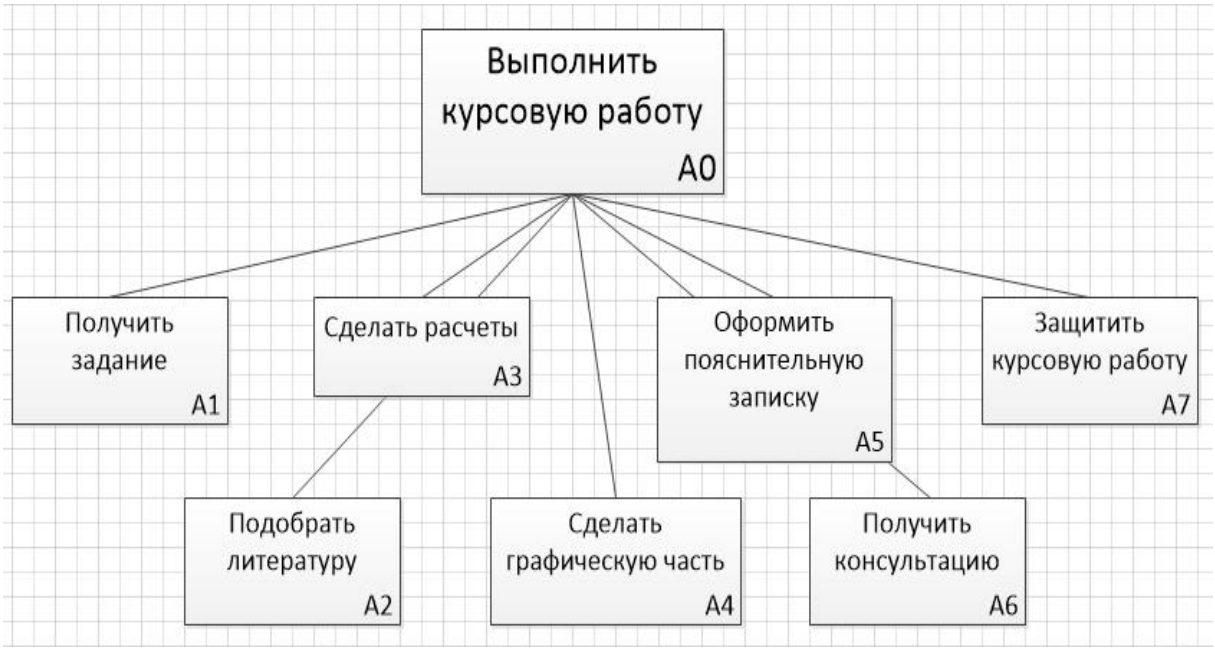


Рисунок 11 – Диаграмма узлов

Для построения диаграммы:

– создайте новую страницу;

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

3.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?

4. Содержание отчета

В отчете следует указать:
1. Цель работы
2. Введение. Краткое описание целей и назначения работы.
3. Графическое представление бизнес-модели проекта ИС.
4. Описание элементов бизнес-модели проекта.
5. Анализ возможные проблемы и пути их решения при реализации
6. Заключение (выводы).
7. Список используемой литературы.

Лабораторная работа №3

Разработка информационной модели проекта ИС

1.Цель работы

Целью работы является освоение технологии построения информационной модели логического и физического уровней в нотации IDEF1X с использованием пакета Microsoft Office Visio..

2.Основные теоретические положения

1. 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 три 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).

2. Основные элементы информационной модели логического уровня

2.1. 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**) (рис. 3.1).

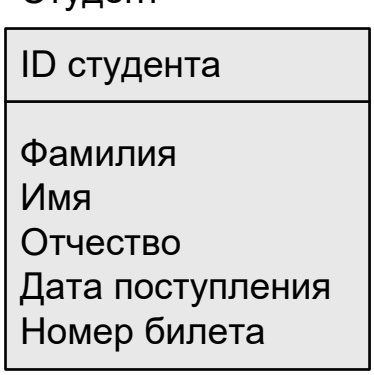


Рисунок 1 – Графическое представление сущности «Студент» в MS Office Visio

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

Атрибут сущности – характеристика сущности, то есть свойство реального объекта. Например, на рис. 3.1 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.

2.2. 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.

3. 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).

4. Методика выполнения лабораторной работы

Методика проектирования базы данных рассмотрена на примере

«Выполнение курсовой работы»

Построение логической информационной модели уровня «сущность-связь»

Информационная модель может быть построена на основе функциональной модели (в нотации IDEF0). В этом случае названия всех интерфейсных дуг IDEF0-модели заносятся в список потенциальных сущностей.

Список потенциальных сущностей для рассматриваемого примера будет представлен таблицей вида (рис. 5.1):

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

Рисунок 2 – Список потенциальных сущностей

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.

Создание логической модели «сущность-связь»

1. Запустите MS Office 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 Office Visio 2007 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* (рис. 5.2).

В закладке *Отношение* окна *Параметры документа базы данных* в меню *Показывать* нужно отметить галочкой пункт *Мощность*, в меню *Отображение вида* выбрать пункт *Показывать вербальную фразу*, снять галочку в пункте *Обратный текст* (рис. 5.3). 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*.

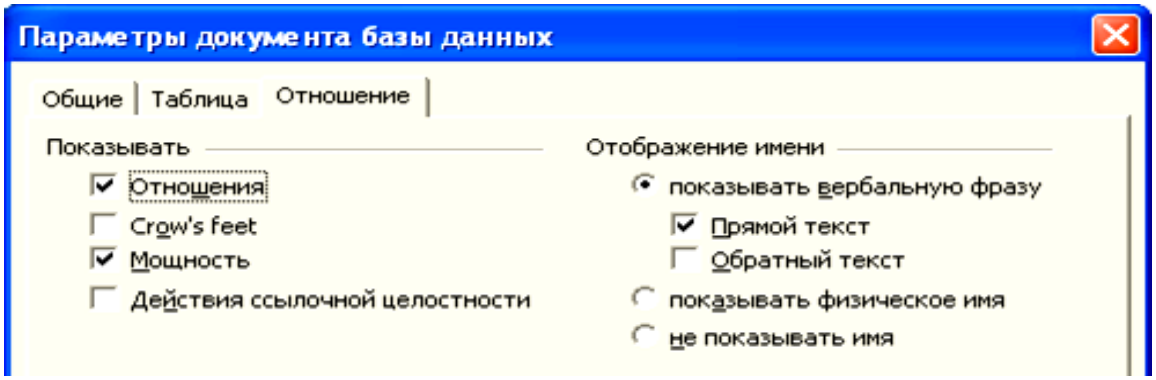


Рисунок 3 – Настройка вида отношений информационной модели

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 1.

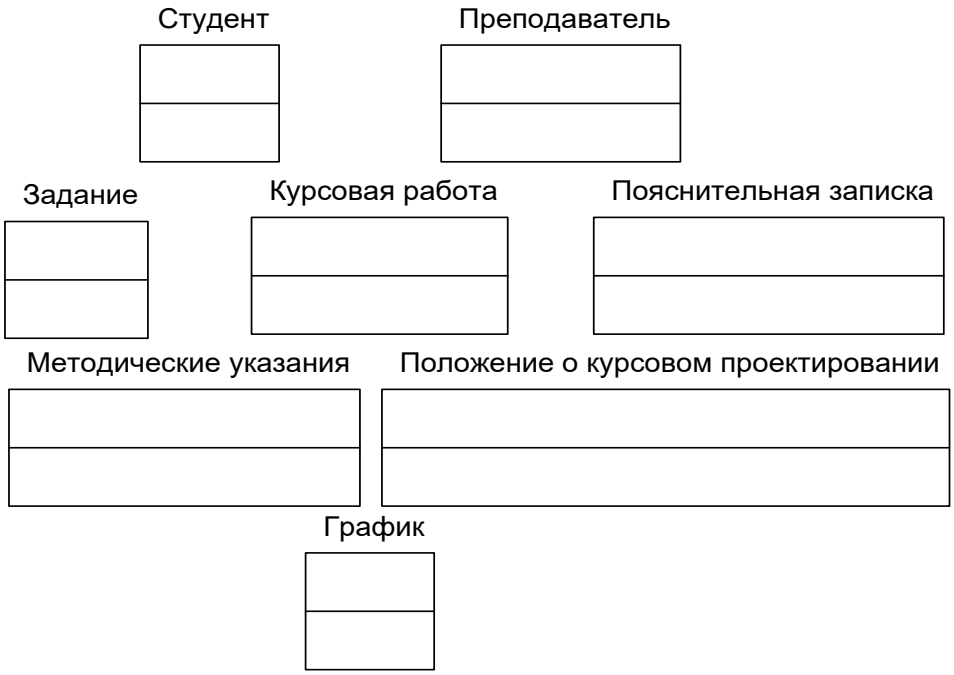


Рисунок 4 – Сущности информационной модели логического уровня

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. 5.7). 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.

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 (см. рис. 5).

Рисунок 5 – Определение типа связи и мощности

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. 6). 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.

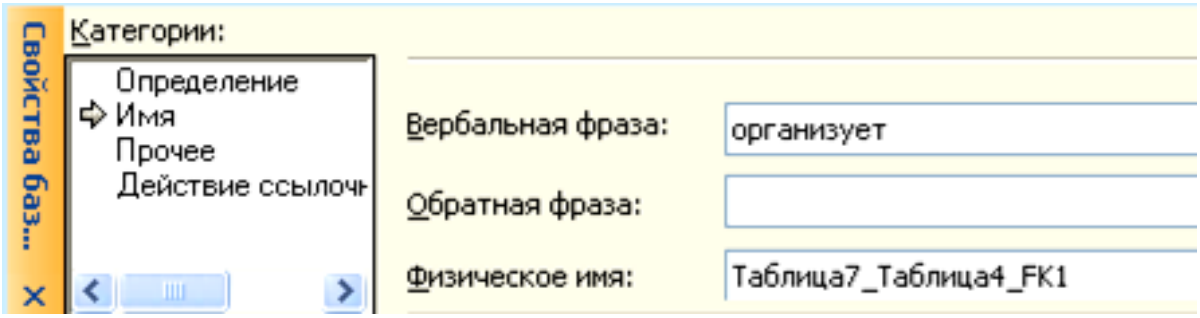


Рисунок 6 – Определение имени отношения

После определения имен, типов связей и задания мощностей получим информационную модель, представленную на рис. 7.

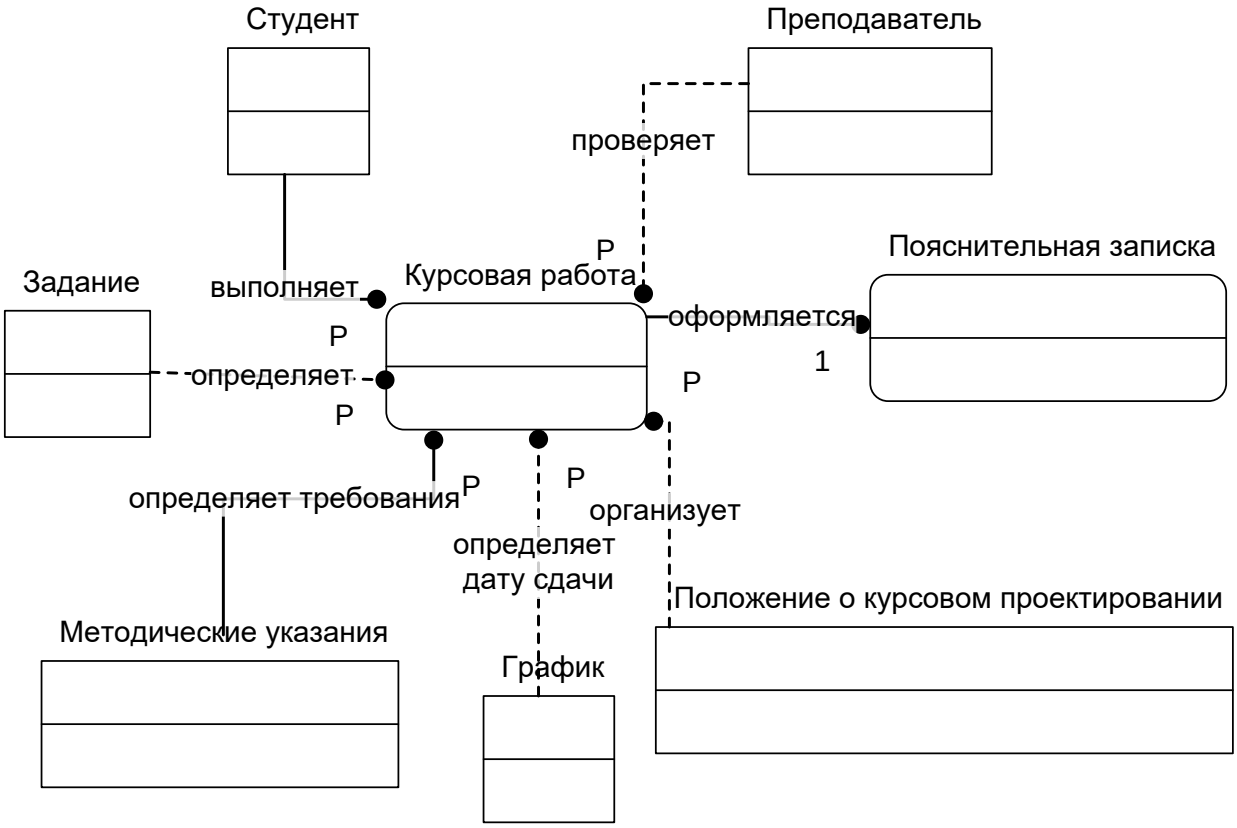


Рисунок 7 – Информационная модель уровня «сущность-связь»

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 (см. рис. 5.9).

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 *PK* (рис. 8). Данное поле становится обязательным автоматически.

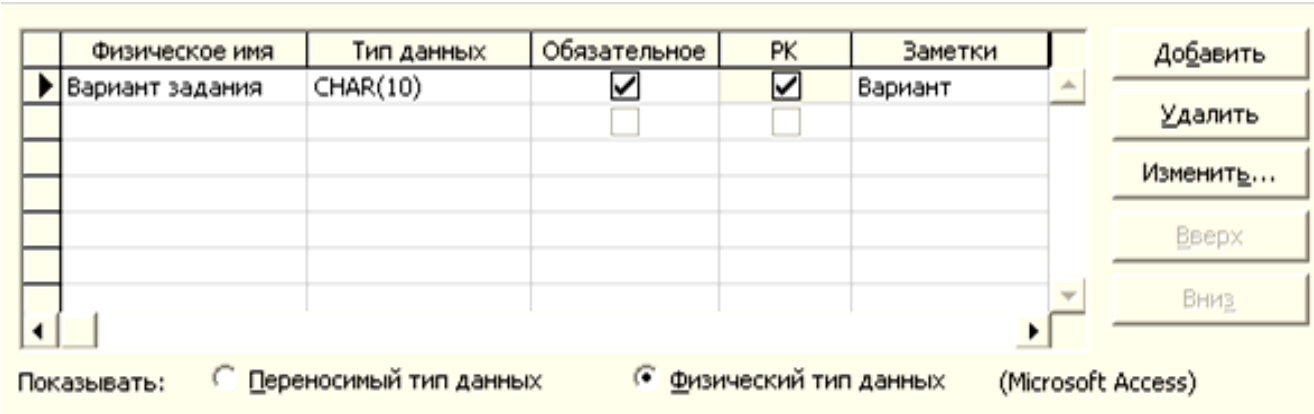


Рисунок 8 – Определение ключевого атрибута

Аналогичным образом зададим ключевые атрибуты для всех сущностей информационной модели. Результат представлен на рис.5.9.

Как видно из рис. 9 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 (рис. 10).

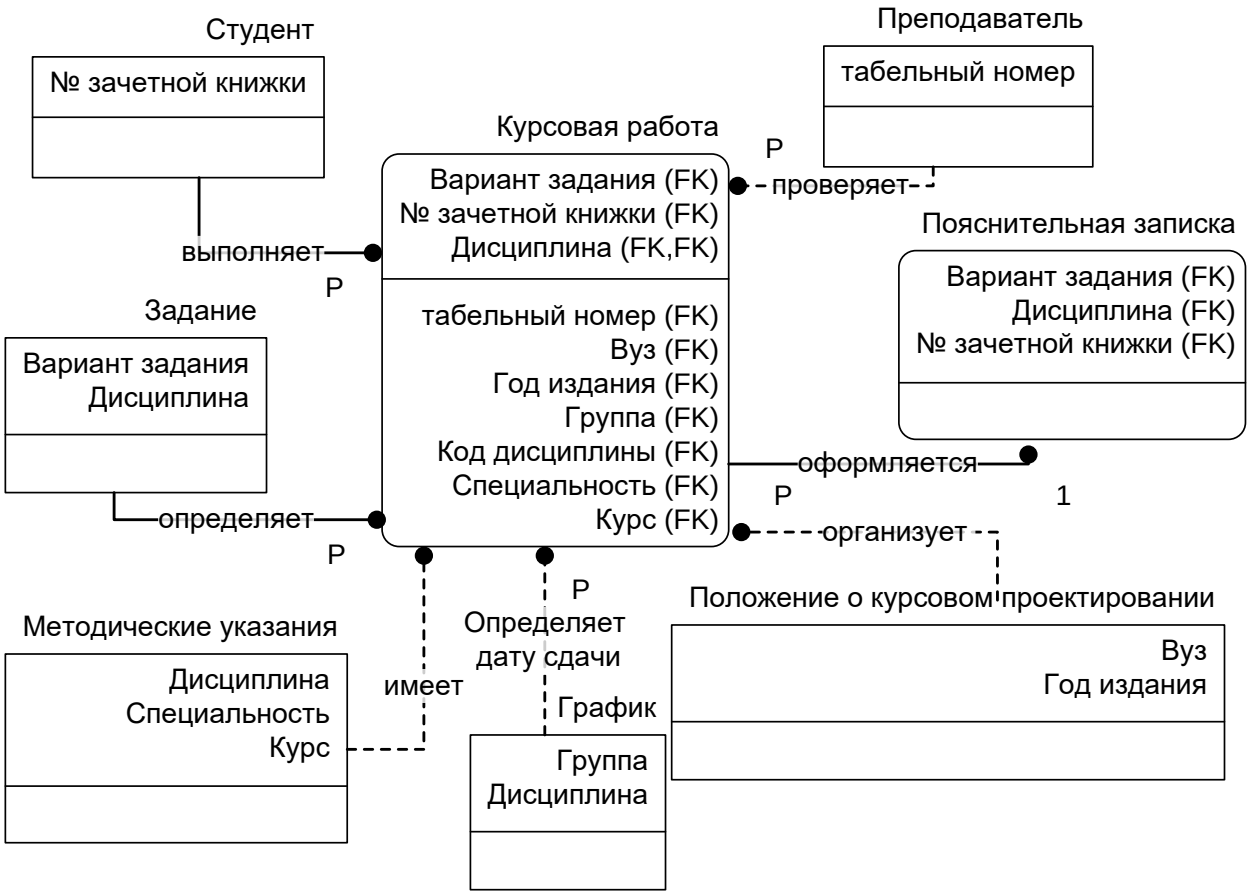


Рисунок 9 – Информационная модель с ключевыми атрибутами

Создание полной атрибутивной модели

Для того чтобы получить полную атрибутивную модель, необходимо дополнить сущности не ключевыми атрибутами. Дополненная модель представлена на рис. 5.11.

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)*.

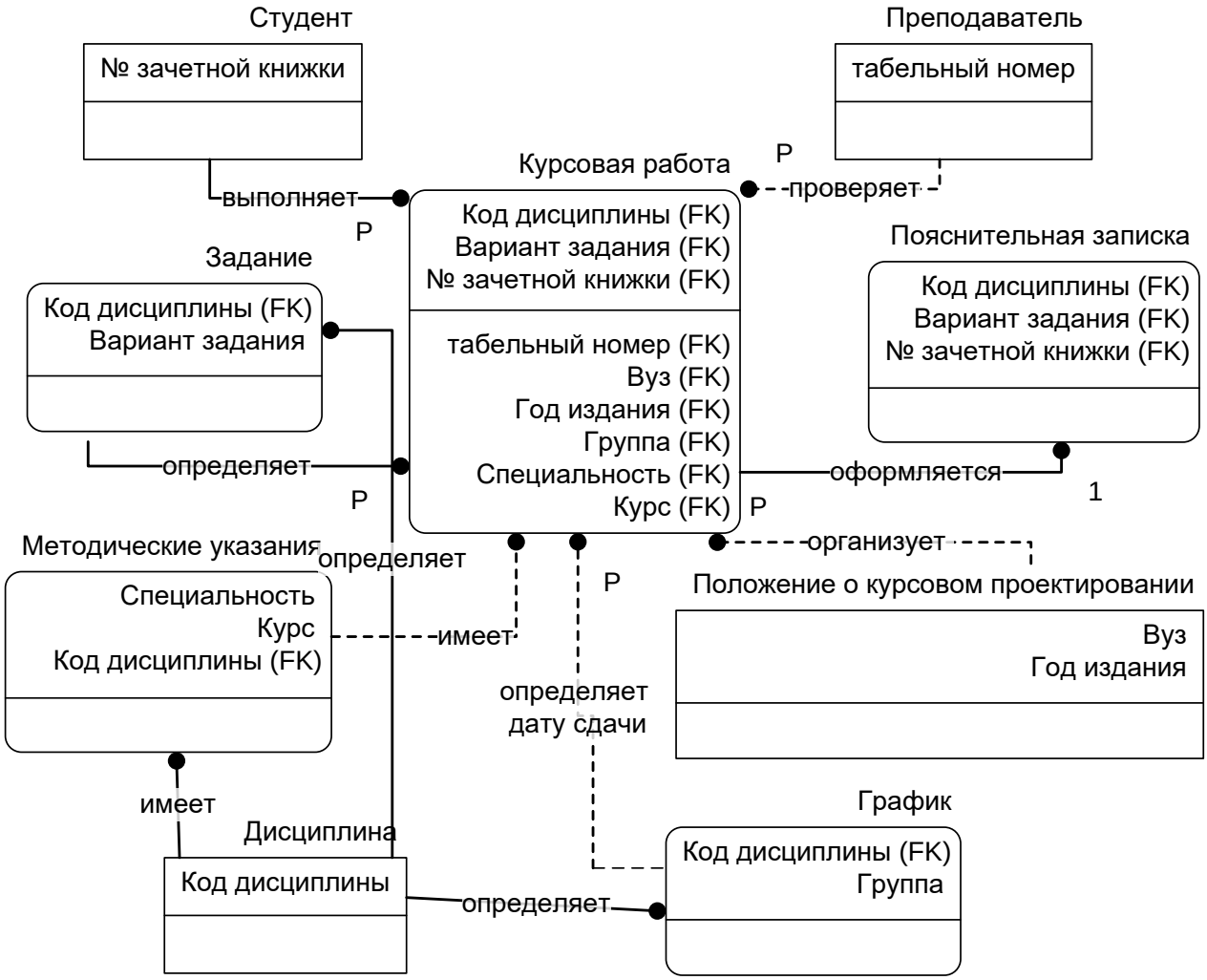


Рисунок 10 – Скорректированная информационная модель, основанная на ключах

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, представлена на рис. 12.

Рисунок 11 – Полная атрибутивная модель

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 12, приведена к 3 НФ.

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

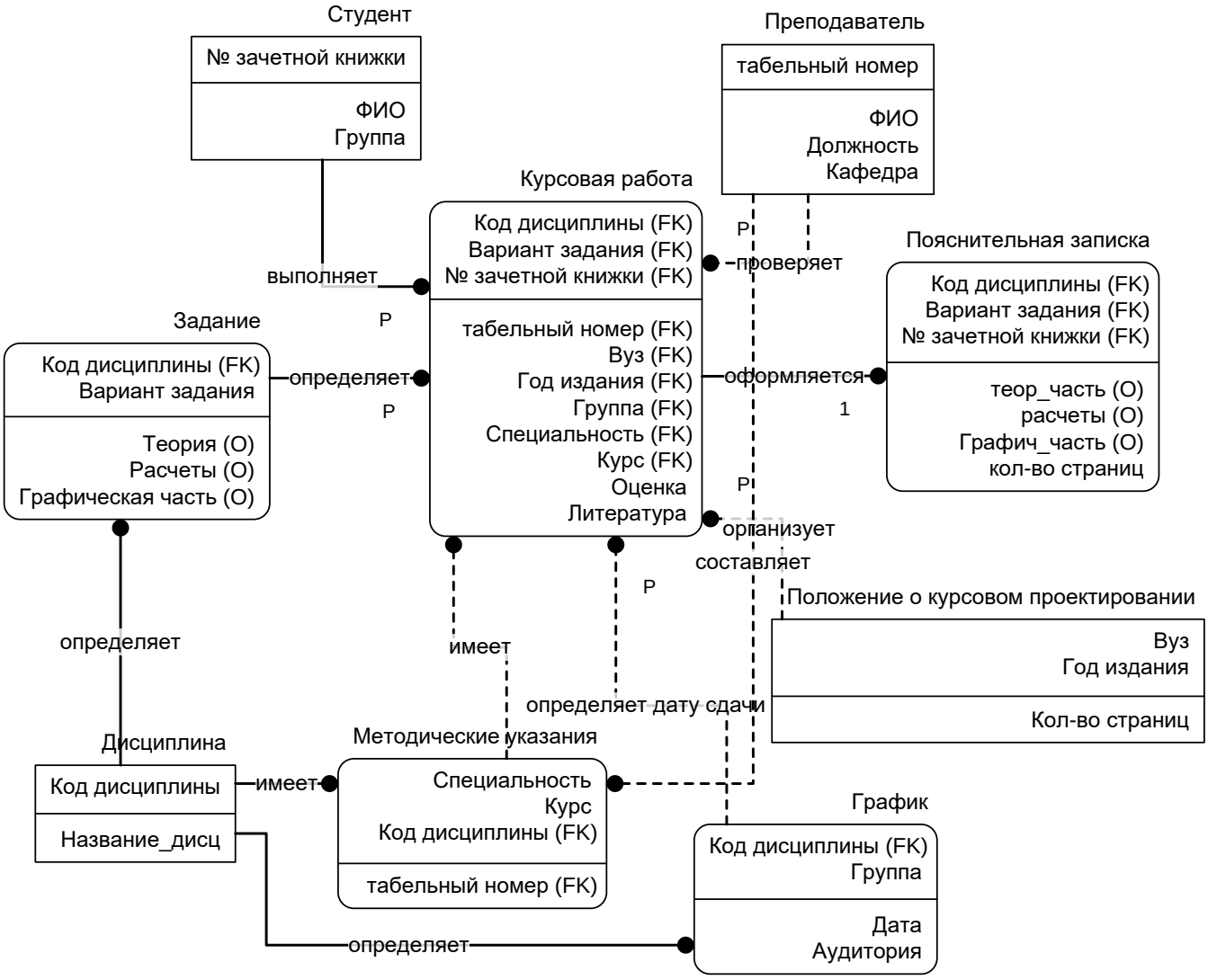


Рисунок 12 – Информационная модель, приведенная к 1 НФ

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* (рис. 13).

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* (рис. 5.14).

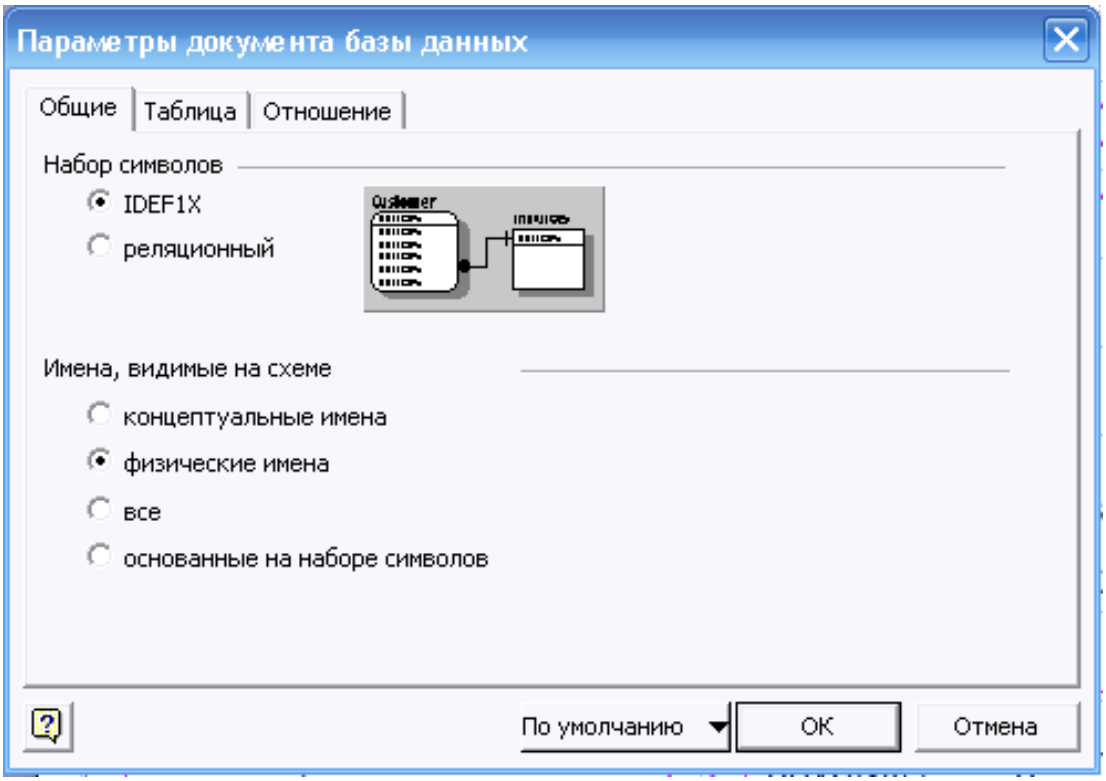


Рисунок 13 – Настройка параметров модели

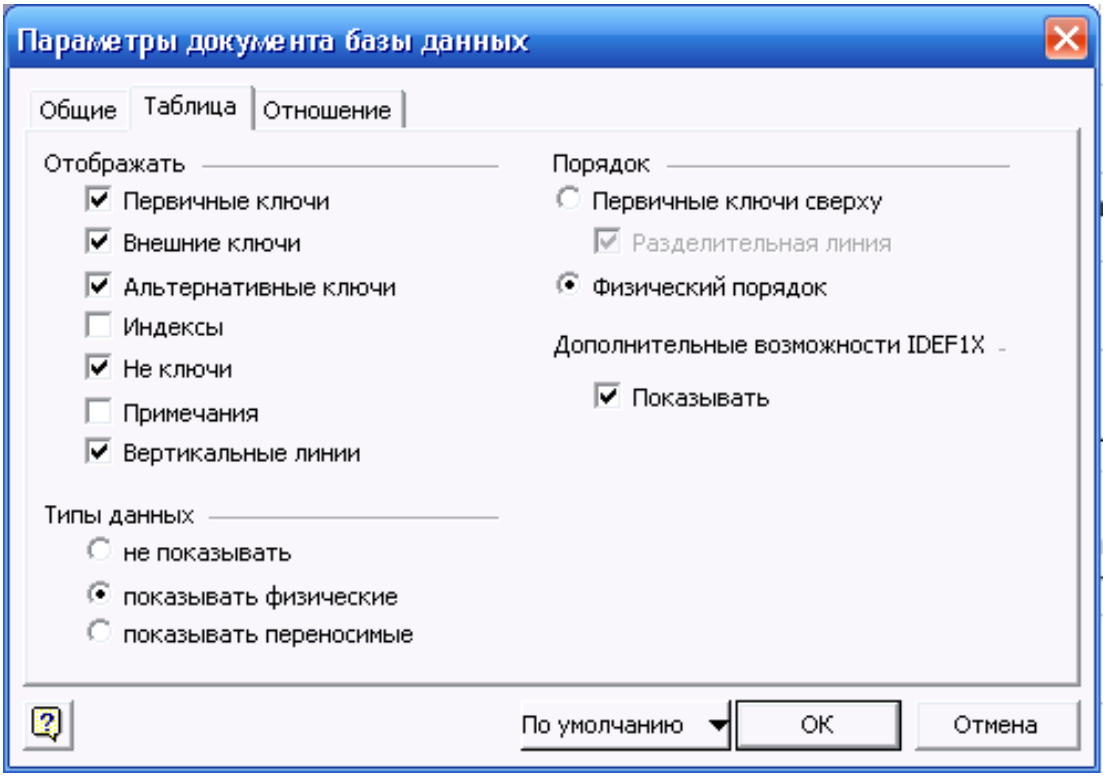


Рисунок 14 – Настройка параметров отображения сущности

3. В закладке *Отношение* окна *Параметры документа базы данных* в меню *Отображение вида* выбрать пункт *Показывать физическое имя* (рис. 5.15).

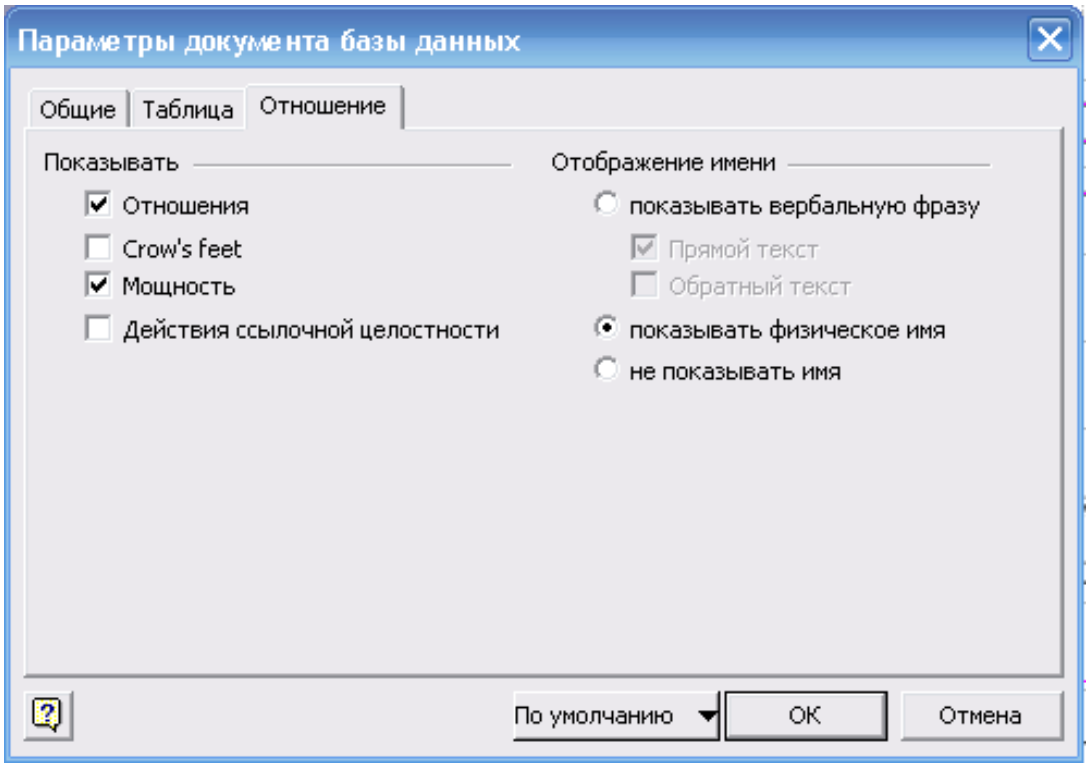


Рисунок 15 – Настройка вида отношений информационной модели

По окончании настройки документа информационная модель будет выглядеть, как представленная на рис. 5.16.

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

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*.

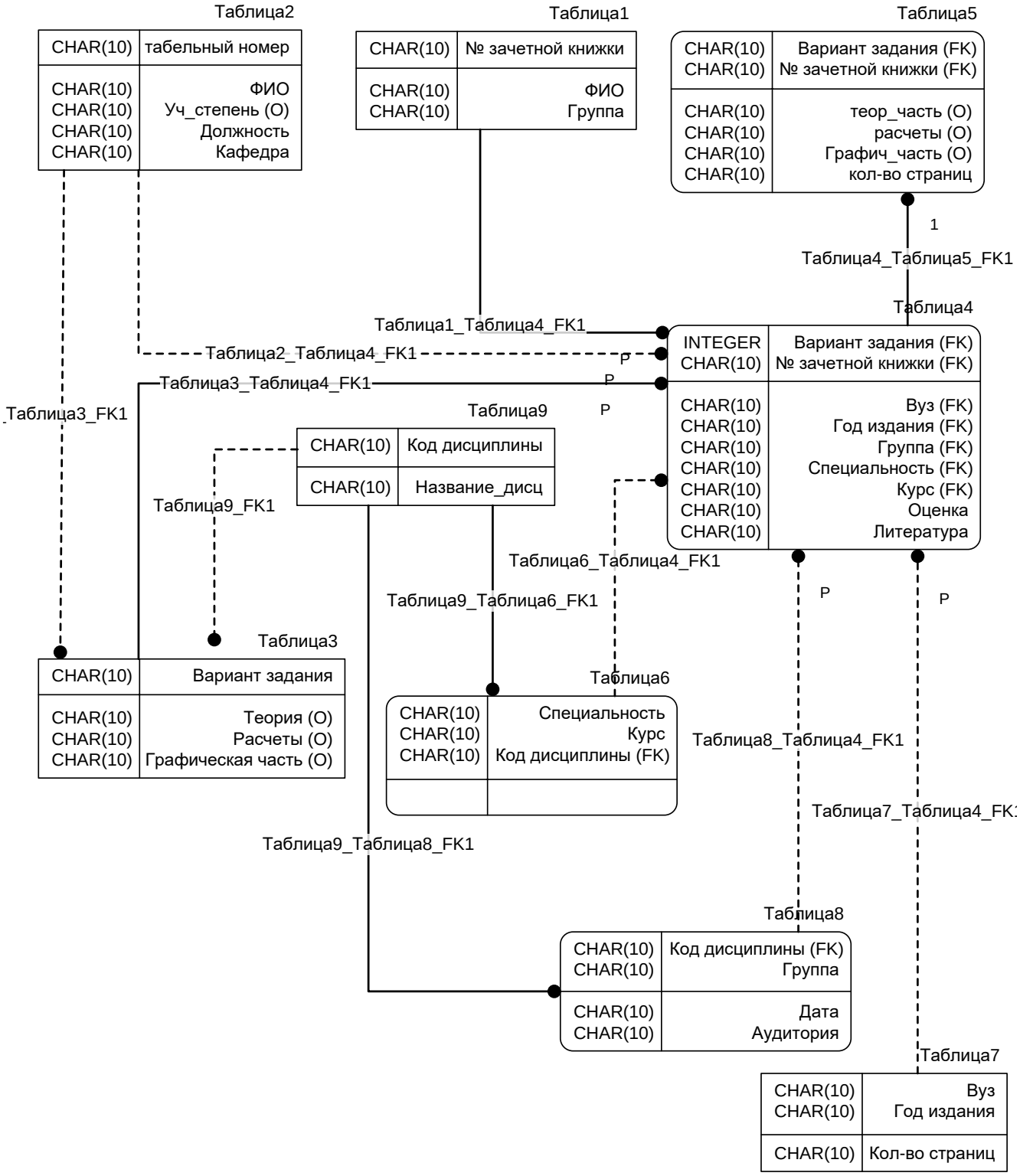


Рисунок 16 – Вид физической модели

В группе *Тип данных* для каждого атрибута выберите необходимый вариант из множества альтернатив (рис. 5.17). Описание типов данных приведено в *Приложении Б*.

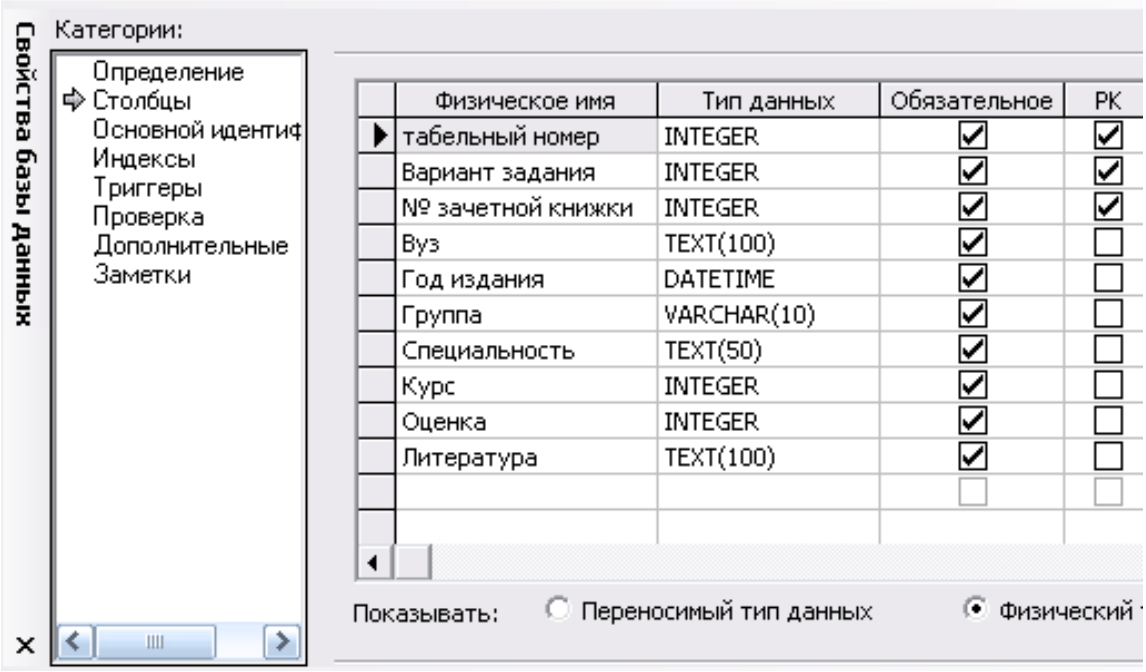


Рисунок 5.17 – Определение типа данных атрибутов сущности

После того, как будут выполнены все действия, физическая модель будет выглядеть, как показано на рис. 5.18.

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).

Задание

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

1) создать информационную модель логического уровня (выполнить пункты 1 –3).

Минимальное количество сущностей – 4-6 (в зависимости от предметной области);

2) провести нормализацию полученной модели;

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

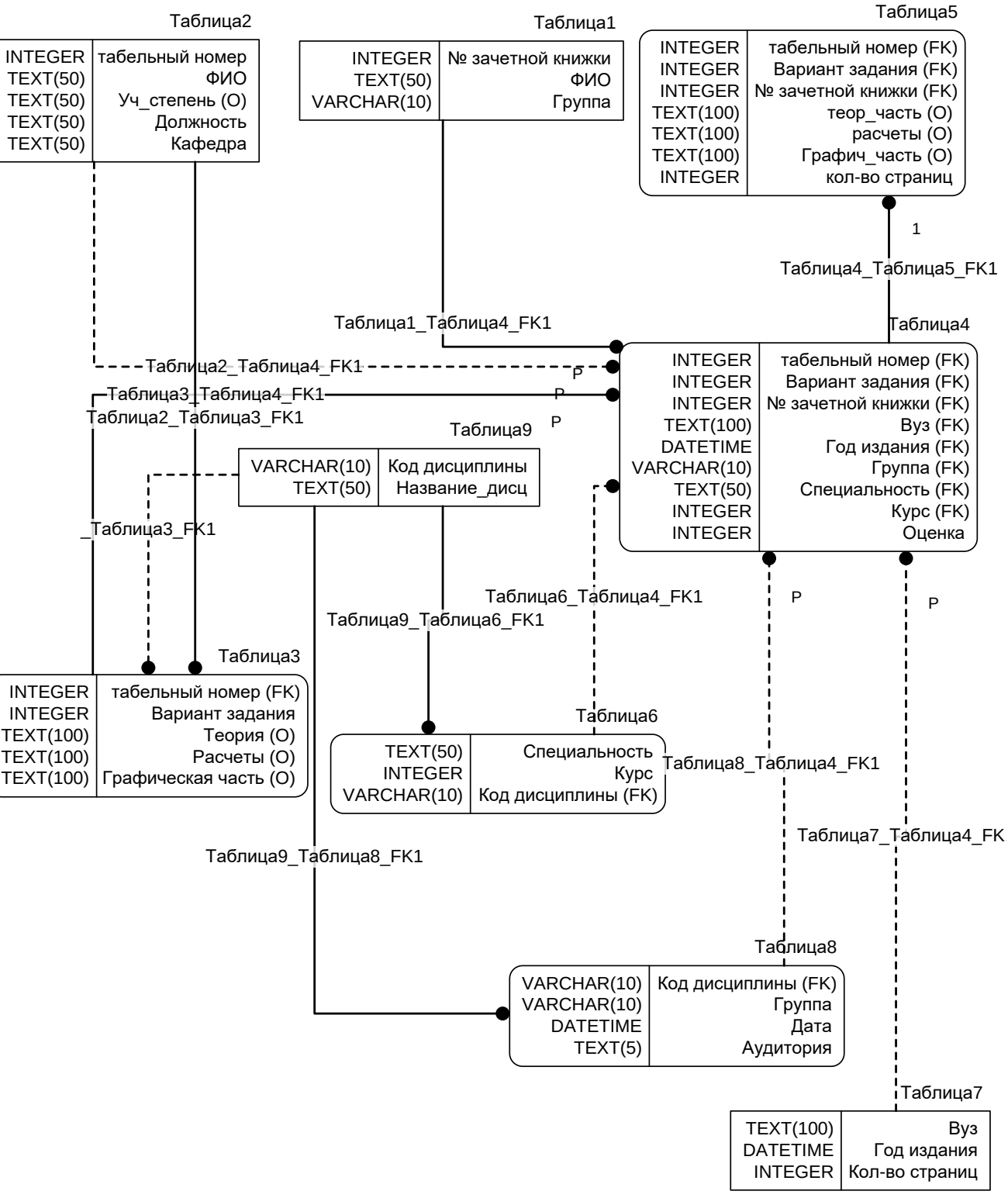
Порядок выполнения работы

Для выполнения работы необходимо:

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

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

– оформить отчет по лабораторной работе.



18 – Физическая модель базы данных

5.Контрольные вопросы

1. Для чего предназначена диаграмма «сущность-связь»?
2. Чем отличается полная атрибутивная модель от диаграммы «сущность-связь»?
3. Какие виды отношений существуют и чем они отличаются?
4. Приведите пример идентифицирующего отношения.
5. Приведите пример отношения полной категоризации.
6. Чем отличаются отношения полной и неполной категоризации?
7. Что представляет собой нормализация?
8. В чем разница между логическим уровнем модели данных и физическим?

6. Содержание отчета

В отчете следует указать:
1. Цель работы
2. Введение. Краткое описание целей и назначения работы.
3. Графическое представление модели данных проекта ИС.
4. Описание элементов модели.
5. Анализ возможные проблемы и пути их решения при реализации
6. Заключение (выводы)
7. Список используемой литературы

Лабораторная работа №4

Разработка диаграмм вариантов и использования

1.Цель работы

Целью работы является освоение технологии построения модели функционирования проекта ИС на основе диаграмм вариантов использования с использованием пакета Microsoft Office Visio.

2.Основные теоретические положения

Диаграмма (***use case***) представляет собой последовательность действий (функция ИС или прецедентов), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом или актером) и представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно некоторых интерфейсов, и отношений между этими элементами

Разработка **диаграммы использования (прецедентов)** преследует цели:

– Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.

– Сформулировать общие требования к функциональному поведению проектируемой системы.

– Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

– Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграммы прецедентов относятся к той группе диаграмм, которые представляют динамические или поведенческие аспекты системы.

3.Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

Запустите MS Visio.

На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.

Окно программы примет вид, подобный рис. 1.

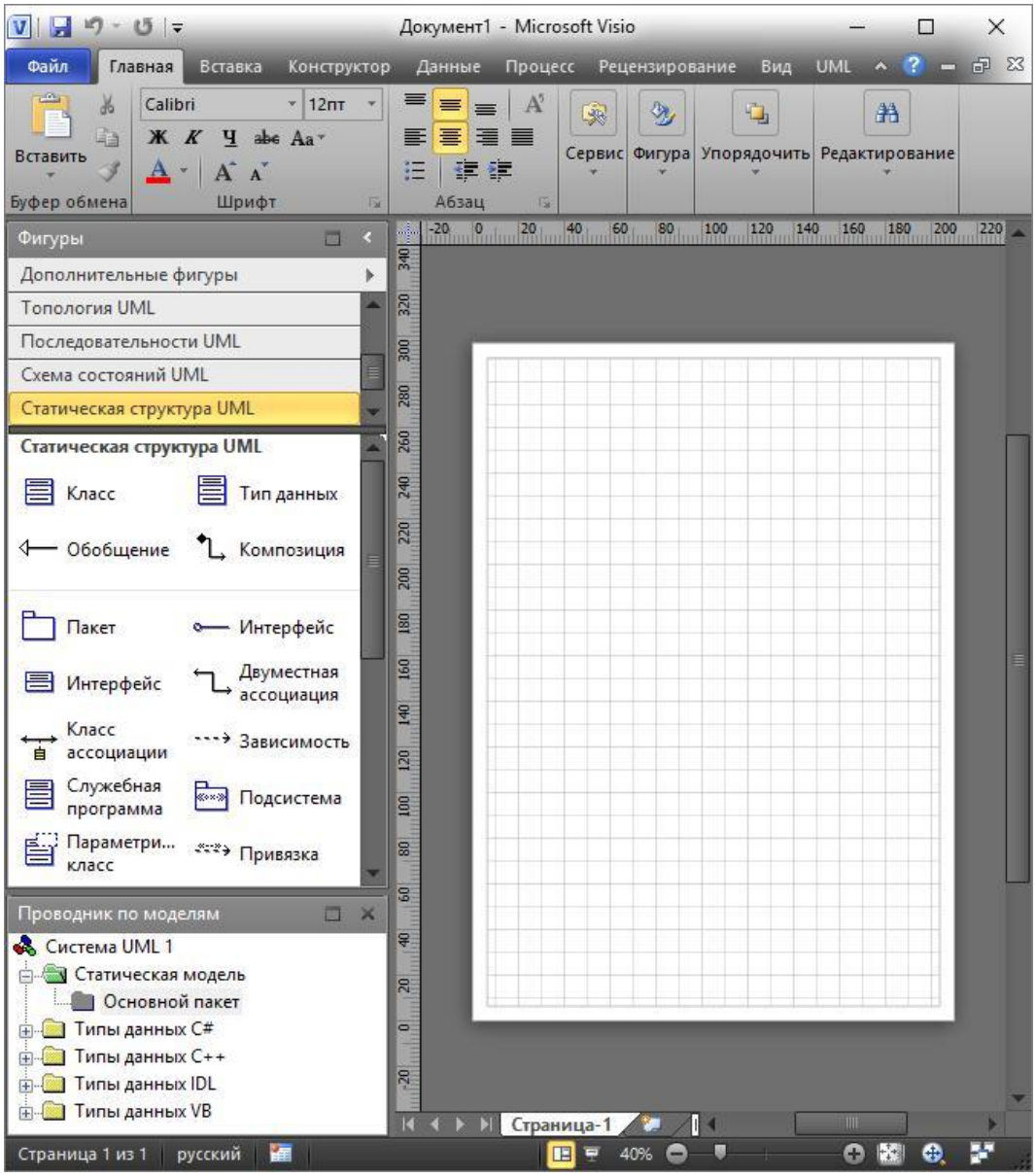


Рисунок 1– Схема модели UML в MS Visio

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. 2).

После этого необходимо провести следующие этапы моделирования.

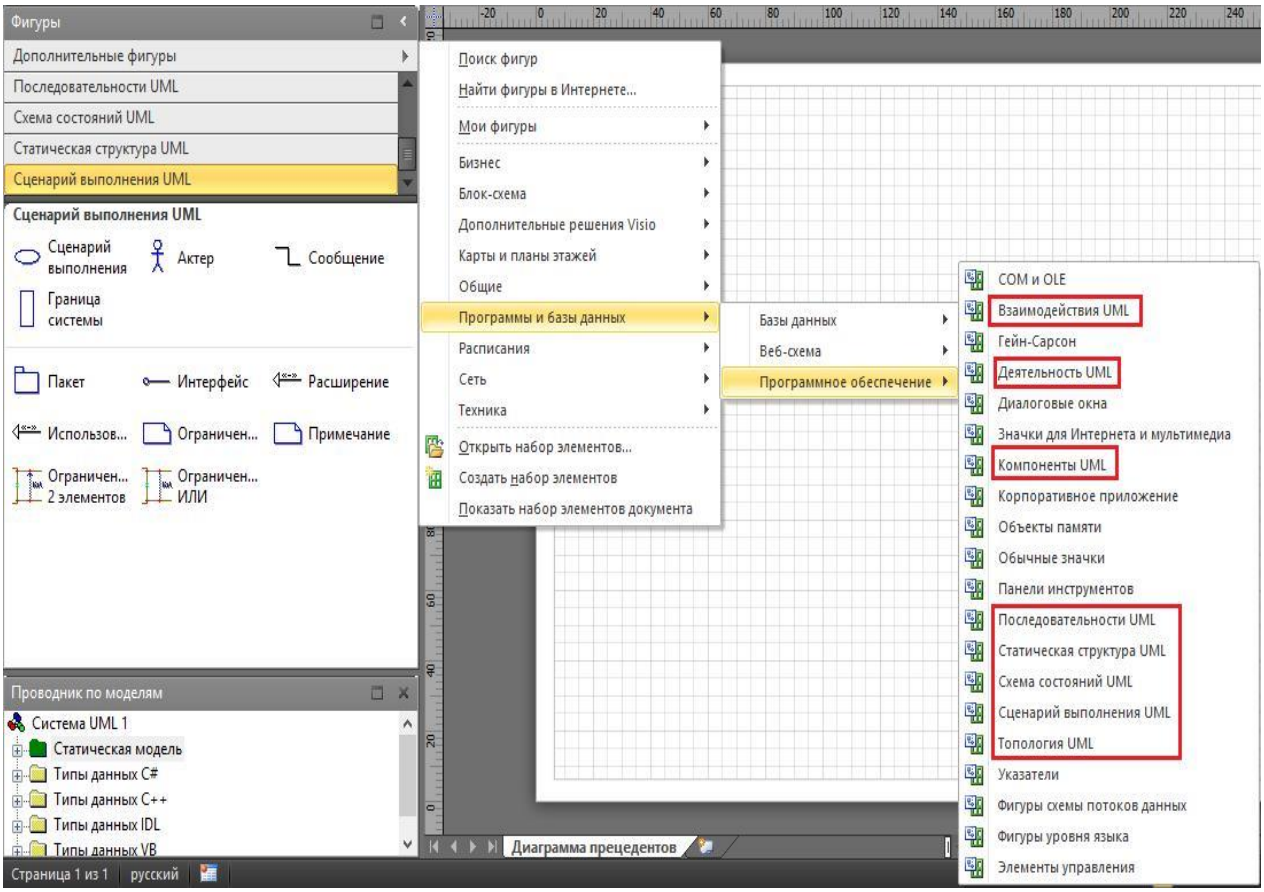


Рисунок 2 – Добавление фигур UML

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 3.

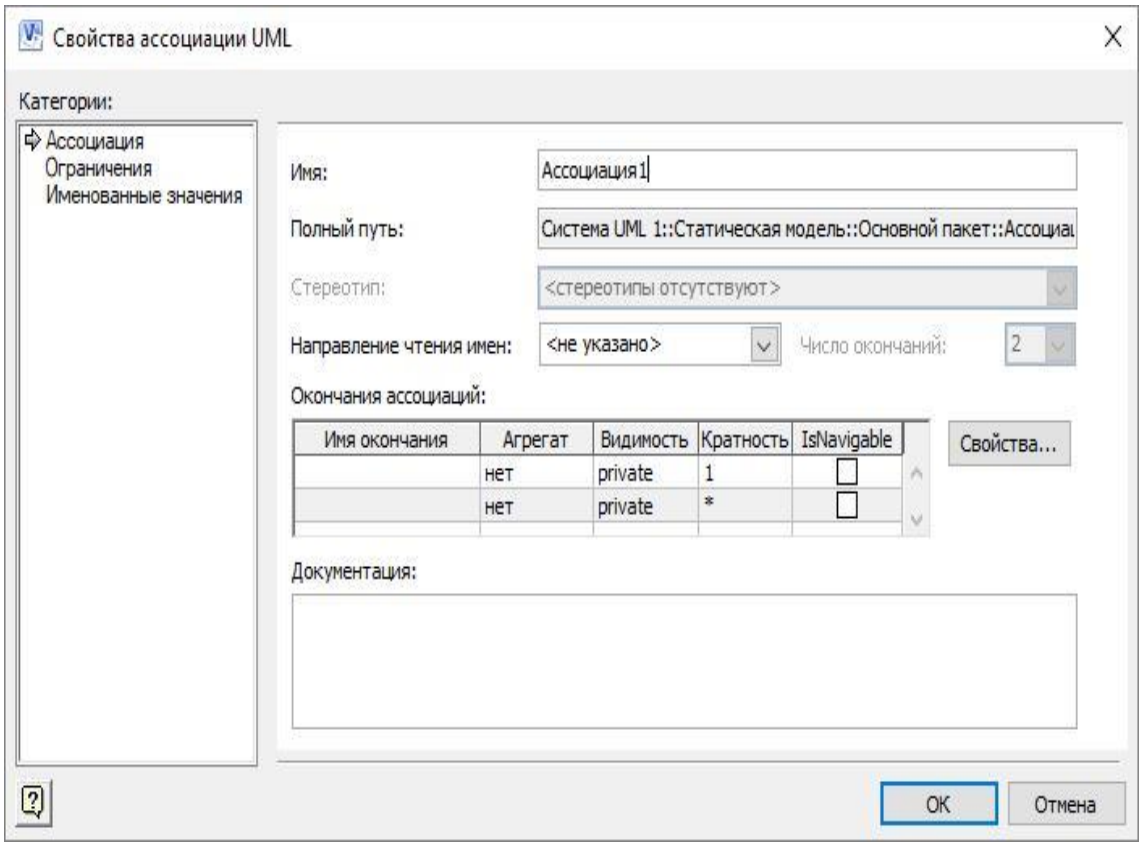


Рисунок 3 – Свойства ассоциации UML

Рисунок 4– Исходная диаграмма вариантов использования системы по продаже товаров

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 5.

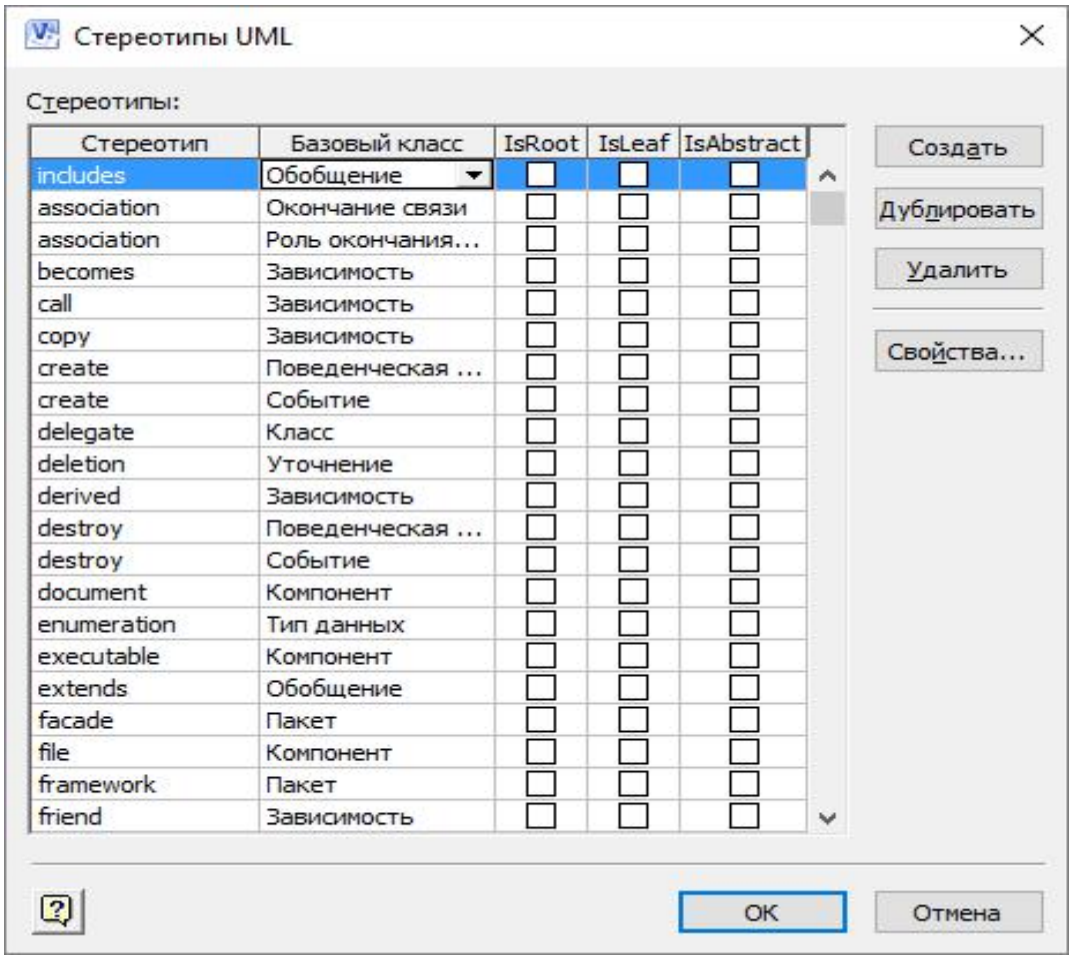


Рисунок 5 – Создание стереотипа

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 6.

Рисунок 6 – Дополненная диаграмма вариантов использования

Суть построения диаграммы заключается в представлении проектируемой системы в виде множества сущностей (актёров), взаимодействующих с системой при помощи вариантов использования. Связи между ними называются отношениями.

Пример: Оформление заказов по каталогу

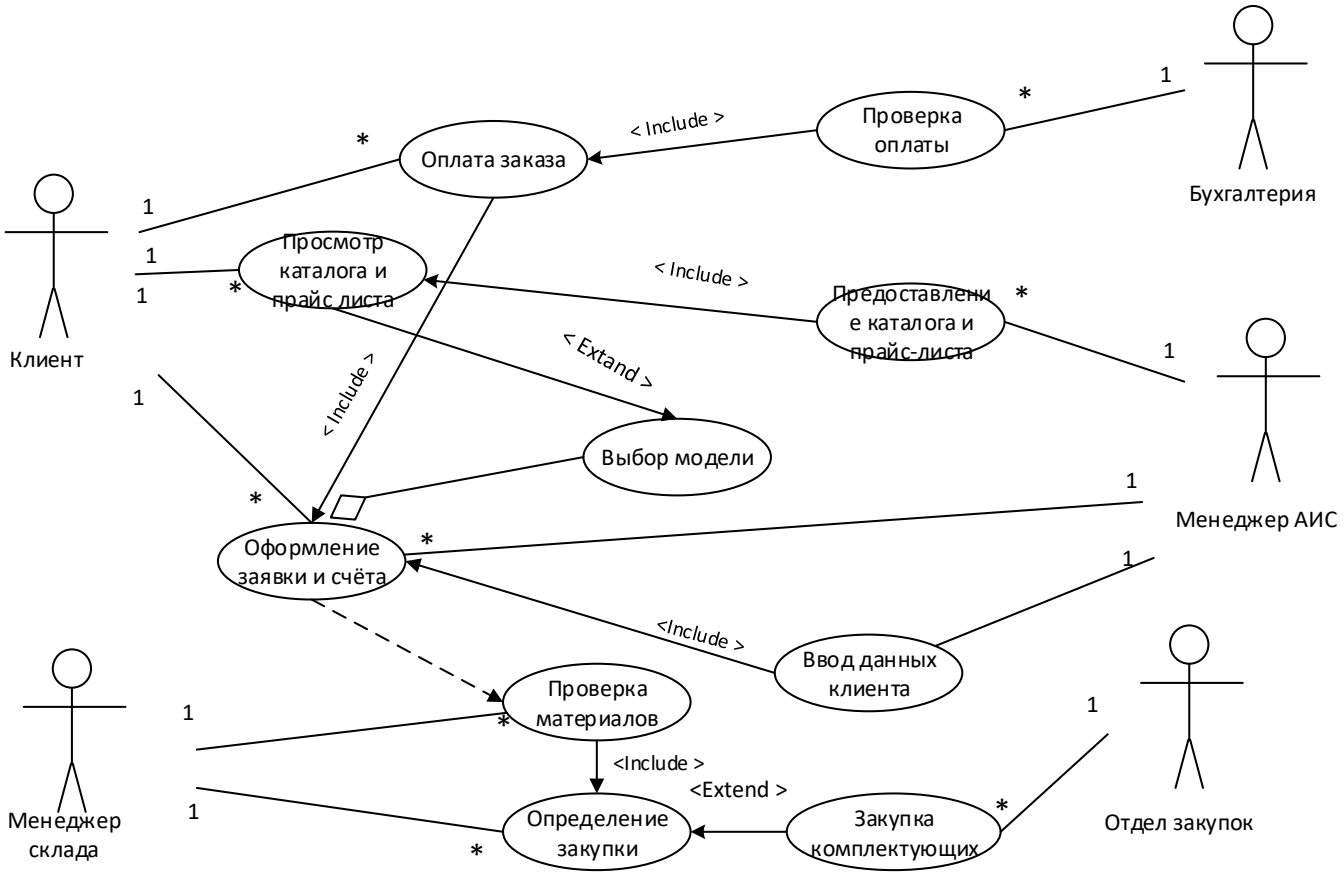


Рисунок 7– Диаграмма вариантов использования

Опишем некоторые основные варианты использования.

Вариант использования «просмотр каталога» и «выбор модели».

Краткое описание: данный вариант использования выполняется, когда клиент просматривает продукцию и совершает выбор (таблица 1).

Таблица 1 – Вариант использования «просмотр каталога», «выбор модели».

Действия актёра	**Действия (отклик) системы**
Основной поток	

1) Пользователь системы выбирает пункт меню «Просмотреть каталог изделий»	2) Система выводит каталог изделий
3) Пользователь выбирает модель	4) Система фиксирует выбранную модель
Альтернативный поток А1: модель не реализуется	
	5) Система выдаёт сообщение «модель не реализуется»
	6) Система передаёт управление на пункт 3
Вариант использования завершается	

Вариант использования «оформление заявки».

Краткое описание: данный вариант использования начинает выполняться, когда пользователь оформляет заявку клиента (таблица 2).

Таблица 2 – Вариант использования «оформление заявки»

Действия актёра	**Действия (отклик) системы**
1) Пользователь выбирает пункт меню «оформить заказ»	2) Система предоставляет сведения о выбранной модели
4) Клиент вводит ФИО	3) Система запрашивает ФИО клиента
	5) Система проверяет наличие данных клиента в базе
7) Клиент вводит данные для оформления заявки	6) Система предоставляет шаблон для ввода данных клиента
	8) Система сохраняет данные в БД
Альтернативный поток А1: данные клиента имеются в базе	
	9) Система пересохраняет сведения о клиенте

Альтернативный поток завершается	
	10) Система предоставляет бланк заявки клиента
	11) Печать заявки
Вариант использования завершается	

4.Задание

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

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?

Лабораторная работа №5

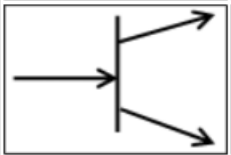
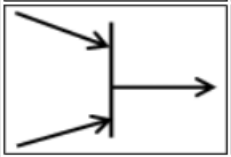
Разработка информационной модели проекта ИС

1. Цель работы

Целью работы является освоение технологии построения диаграмм активности (деятельности) унифицированного языка моделирования UML с использованием пакета Microsoft Office Visio.

2. Основные теоретические положения

Диаграмма активности **(Activity)** – позволяет моделировать **жизненный цикл объекта** в **виде** переходами из одного состояния (деятельности) в другое, т.е. отображают алгоритмы по преобразованию классов. Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов с применением обозначений:

Обозначение	Наименование	
	на диаграмме автоматов	на диаграмме деятельности
	Начальное псевдосостояние (англ. initial pseudostate)	Начальный узел (англ. initial node)
	Конечное состояние (англ. final state)	Завершение деятельности (англ. activity final)
	Точка выхода (англ. exit point pseudostate)	Завершение потока (англ. flow final)
	Ветвление (англ. fork pseudostate)	Ветвление (англ. fork node)
	Соединение (англ. join pseudostate)	Соединение (англ. join node)
	Выбор (англ. choice pseudostate)	Слияние / решение (англ. merge / decision node)

Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз или слева направо. В этом случае начальное состояние будет изображаться в верхней или левой части диаграммы, а конечное – в ее нижней или правой части. В интересах удобства визуального представления на диаграмме деятельности допускается изображать несколько конечных состояний. В этом случае все их принято считать эквивалентными друг другу.

Переход на диаграмме деятельности аналогичен переходу на диаграмме состояний. При построении диаграммы деятельности используются только нетриггерные переходы, т.е. такие, которые происходят сразу после завершения деятельности или выполнения

соответствующего действия. Такой переход передает управление в последующее состояние сразу, как только закончится действие или деятельность в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит единственный переход, то его можно никак не помечать. Если же таких переходов несколько, то при моделировании последовательной деятельности запускается только один из них. В этом случае для каждого из таких переходов должно быть явно записано собственное сторожевое условие в прямых скобках.

При этом для всех выходящих из некоторого состояния деятельности переходов должно выполняться требование истинности только одного из них. Подобный случай встречается тогда, когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви в зависимости от значения промежуточного результата. Такая ситуация получила название ветвления, а для ее обозначения применяется специальный символ решения.

Графически ветвление на диаграмме деятельности обозначается символом решения (decision), изображаемого форме небольшого ромба, внутри которого нет никакого текста (рис. 4). В ромб может входить только одна стрелка от того состояния действия, после выполнения, которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей. Принято входящую стрелку присоединять к верхней или левой вершине символа решения. Выходящих стрелок может быть две или более. Для каждой из них явно указывается соответствующее сторожевое условие в форме булевского выражения.

Для графического объединения альтернативных ветвей на диаграмме деятельности рекомендуется использовать аналогичный символ в форме ромба, который в этом случае называют соединением (merge). Наличие этого символа, внутри которого также не записывается никакого текста, упрощает визуальный контроль логики выполнения процедурных действий на диаграмме деятельности (рис. 4 внизу). Входящих стрелок у символа соединения может быть несколько, они исходят от состояний действия, принадлежащих к одной из взаимно исключающих ветвей. Выходить из ромба соединения может только одна стрелка, при этом ни входящие, ни выходящая стрелки не должны содержать сторожевых условий. Исключением является ситуация, когда с целью сокращения диаграммы объединяют символ решения с символом соединения.

Пример: моделируется ситуация, возникающая в супермаркетах при оплате товаров. Как правило, заплатить за покупки можно либо наличными, либо по кредитной карточке. Если покупателем выбран вариант оплаты по кредитной карточке, то проверяется сумма баланса предъявленной к оплате кредитной карточки. При этом оплата

происходит только в том случае, если общая стоимость приобретаемых товаров не превышает суммы баланса этой карточки. В противном случае оплаты не происходит, и товар остается у продавца (рисунок 1).

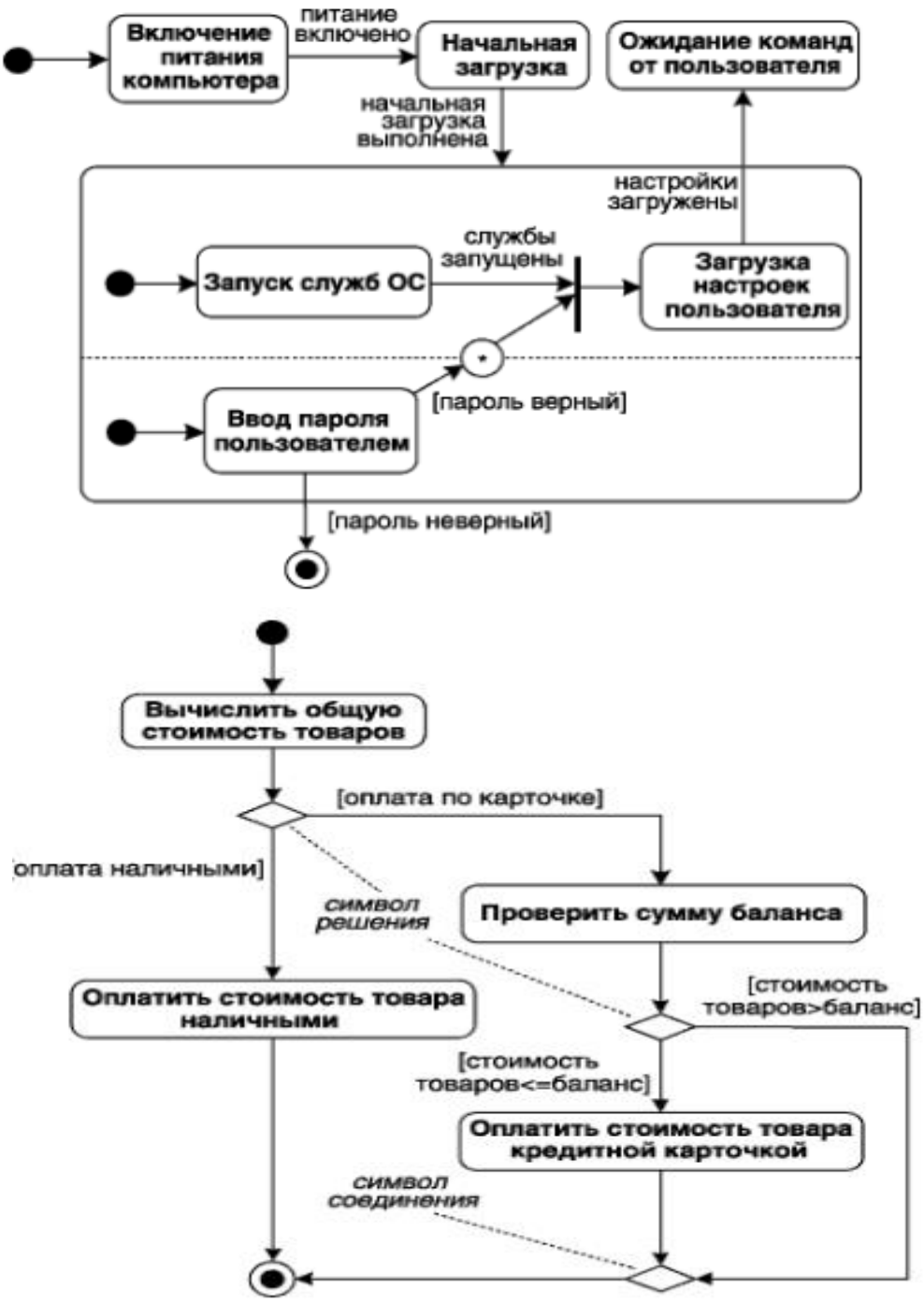


Рисунок 1 – Примеры диаграммы деятельности (активности)

Диаграмма активности позволяет разделить функции по исполнителям с использованием разделов диаграмм.

Разделы группируют действия относительно какой-либо характеристики, например:

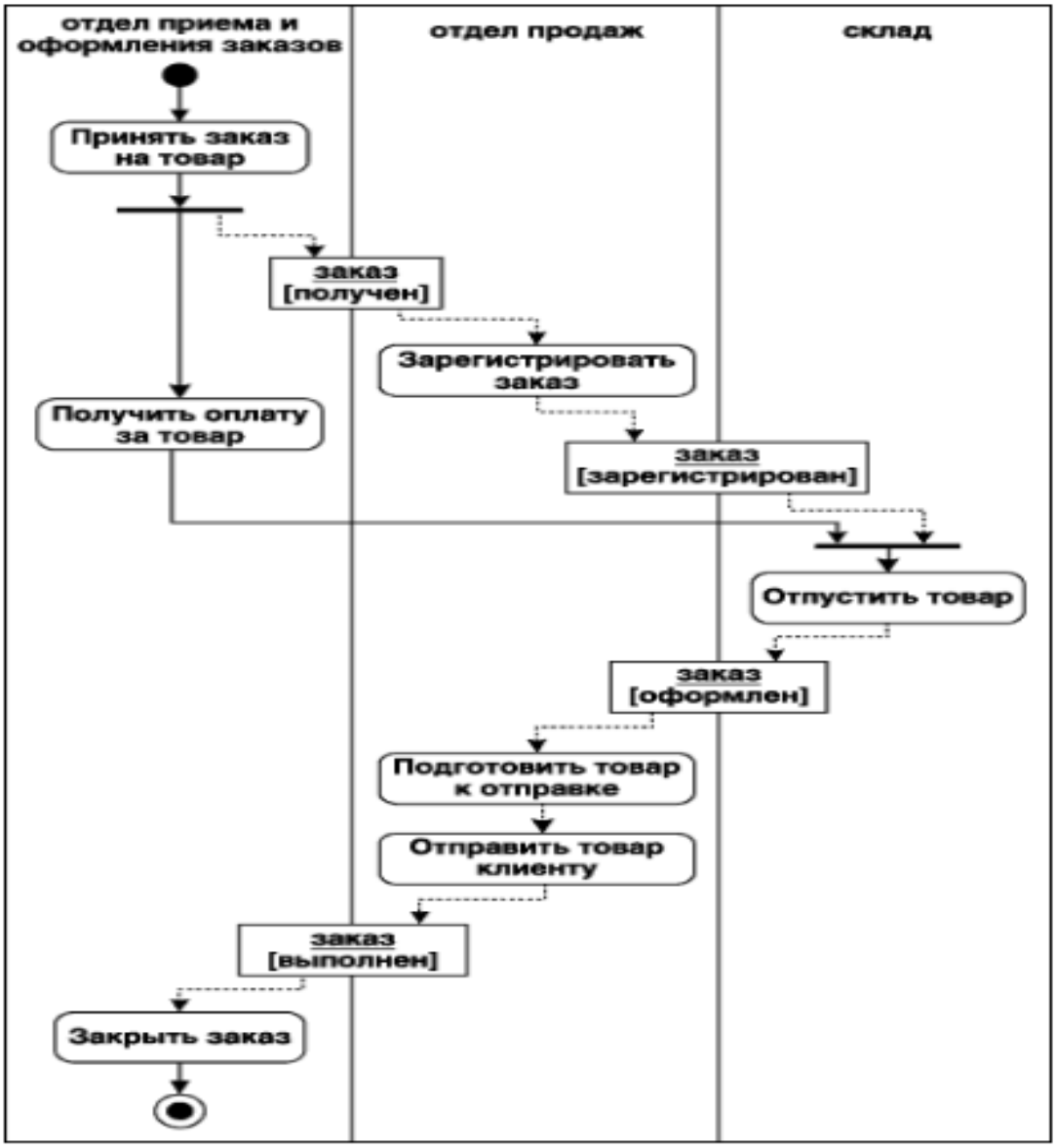


Рисунок 2 – Диаграммы деятельности (активности)

3.Контрольные вопросы

1. Для чего предназначена диаграмма «сущность-связь»?
2. Чем отличается полная атрибутивная модель от диаграммы «сущность-связь»?
3. Какие виды отношений существуют и чем они отличаются?
4. Приведите пример идентифицирующего отношения.
5. Приведите пример отношения полной категоризации.
6. Чем отличаются отношения полной и неполной категоризации?
7. Что представляет собой нормализация?
8. В чем разница между логическим уровнем модели данных и физическим?

4. Содержание отчета

В отчете следует указать:

1. Цель работы
2. Введение. Краткое описание целей и назначения работы.
3. Графическое представление модели деятельности проекта ИС.
4. Описание элементов модели.
5. Анализ возможные проблемы и пути их решения при реализации
6. Заключение (выводы)
7. Список используемой литературы

Лабораторная работа №6

Разработка модели взаимодействия объектов проекта ИС

1.Цель работы

Целью работы является освоение технологии построения диаграмм активности и последовательностей унифицированного языка моделирования UML с использованием пакета Microsoft Office Visio..

2.Основные теоретические положения

Взаимодействие между объектами в системе представляются *диаграммами взаимодействия (interaction diagrams)*. Диаграммы взаимодействия подразделяются на два основных типа диаграмм: *диаграммы последовательности (sequence diagrams)* и *диаграммы деятельности (activity)*.

Как правило, диаграмма взаимодействия используется для описания поведения в рамках одного варианта использования. На такой диаграмме изображается ряд объектов и те сообщения, которыми они обмениваются в рамках этого варианта использования.

Диаграммы последовательности несут в себе одну информацию, но выраженную разными способами. Диаграммы последовательности показывают взаимодействие объектов во времени и отражают последовательность происходящих событий.

Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Оси могут меняться местами, так что ось времени может располагаться горизонтально, слева направо, а список объектов располагаться вертикально.

Объект на диаграмме изображается в виде прямоугольника на вершине вертикальной пунктирной линии, называемой *линией жизни объекта (lifeline)* (рис.2.20). Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "X". При сообщении, вызывающем уничтожение объекта (или самоуничтожение), в конце возвращается сообщение об уничтожении объекта. Линия жизни может разветвляться в две (и более) параллельные линии, показываемые условно. Каждая

ответвляющаяся линия соответствует переходу в потоке сообщений. Линии жизни могут объединяться в некоторой последующей отметке.

Рисунок 2 –Диаграмма последовательности

Сообщения (message) связывают объекты между собой и передают информацию о выполняемом действии.

Сообщения могут быть следующих типов:

Сообщения могут быть следующих видов:

– ***синхронное*** сообщение (англ. synchronous message). Клиент посылает сообщение серверу и ждет, пока тот примет и обработает сообщение. Как правило, один объект передает синхронное сообщение второму, второй – третьему и т.д., образуя вложенный поток сообщений. В любом случае клиент, инициирующий поток сообщений, должен дождаться его завершения, т.е. возврата управления. Это самый распространенный тип сообщений;

– ***асинхронное*** сообщение (англ. asynchronous message). Клиент посылает сообщение серверу и, не дожидаясь ответа, продолжает выполнять следующие операции;

– ***возвращающее*** сообщение (англ. reply message), обозначающее возврат значения или управления от сервера обратно клиенту.

Сообщения, получаемые от внешнего источника (англ. found message) и передаваемые внешнему приемнику (англ. lost message), должны начинаться и заканчиваться закрашенным кружком.

Диаграммы последовательности полезны для представления параллельных процессов. Для этого в диаграммах последовательности вводятся *активации (activation).*

Активация Ф***окус управления*** (англ. focus of control) показывает период времени, в течение которого объект выполняет действия

непосредственно или через зависимую процедуру. Пример диаграммы последовательбности на рис.3.

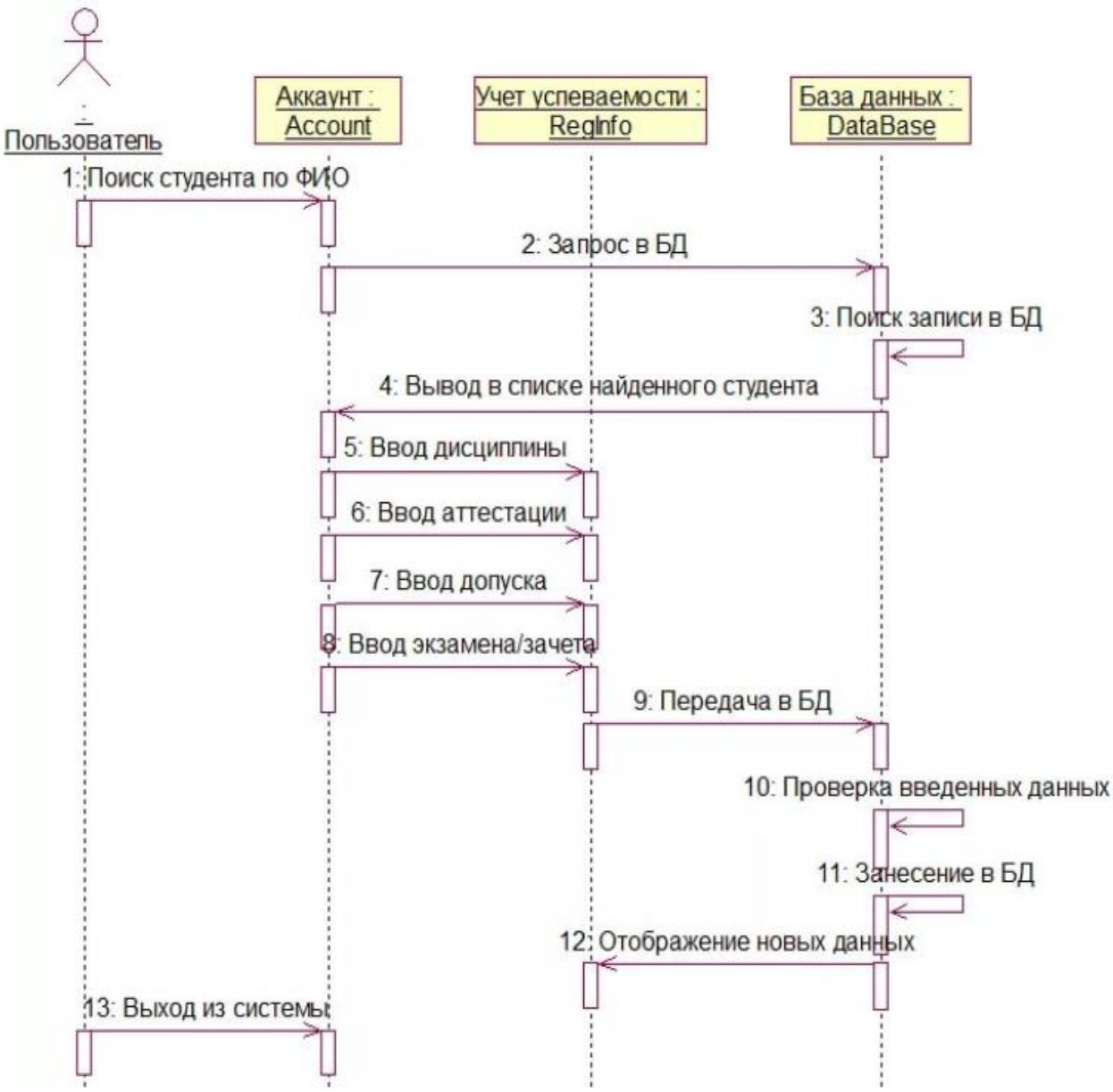


Рисунок 3 - Пример диаграммы последовательбности

3.Выполнение работы

1. Изучить раздел 1 и 2.
2. Разработать модель взаимодействия объектов системы в виде и последовательности в соответствии с вариантом задания.

3. Создать разработанную модель в среде MSVisio.

3.Контрольные вопросы

1. Каково назначение диаграмм активности и последовательности?
2. Для чего используется диаграмма активности на стадии анализа?
3. Для чего используется диаграмма последовательности на стадии проектирования?
4. Назовите основные компоненты диаграммы последовательности.
5. Что представляет собой сообщение?
6. Что представляет собой диаграмма активности (деятельности)?
7. Назовите основные компоненты диаграммы деятельности ?
8. Каково назначение диаграммы кооперации?

4. Содержание отчета

В качестве отчета о выполненной работе предъявите преподавателю:

1. Модель поведения системы в виде диаграмм активности и последовательности по индивидуальному заданию;
2. отчет в печатном виде содержащий:
   - задание,
   - диаграммы активности и последовательности.

Лабораторная работа №7

Разработка модели реализации проекта ИС

1.Цель работы

Целью работы является освоение технологии построения диаграмм компонентов и размещения унифицированного языка моделирования UML с использованием пакета Microsoft Office Visio..

2.Основные теоретические положения

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

- визуализация общей организации структуры исходного кода программной системы;
- спецификация исполнимого варианта программной системы;
- обеспечение многократного использования отдельных фрагментов программного кода;
- представление концептуальной и физической схем баз данных.

В разработке диаграмм компонентов участвуют как системные аналитики и архитекторы, так и программисты. Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие — на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

Примечание

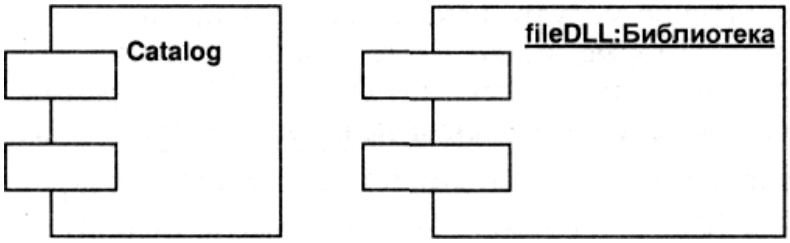
Применительно к бизнес-системам программные компоненты следует понимать в более широком контексте, чтобы иметь возможность моделировать бизнес-процессы. В этом случае в качестве компонентов можно рассматривать отдельные организационные

подразделения (отделы, службы), должности сотрудников (менеджер, кассир) или документы (счет, заказ, накладная), которые реально существуют в системе.

Компоненты

Для представления физических сущностей в языке UML применяется специальный термин - *компонент* (*component*). В метамодели языка UML компонент является потомком классификатора. Он предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты - собственное графическое представление.

Компонент служит для общего обозначения элементов физического представления модели и может реализовывать некоторый набор интерфейсов. Для графического представления компонента используется специальный символ — прямоугольник со вставленными слева двумя прямоугольниками меньшего размера (рис. 10.1). Внутри большого прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация. Этот символ является базовым обозначением компонента в языке UML.



а б

Рис. 1. Графическое изображение компонента в языке UML

Графическое изображение компонента ведет свое происхождение от обозначения модуля программы, применявшегося некоторое время для отображения особенностей инкапсуляции данных и процедур. Так, верхний маленький прямоугольник концептуально ассоциировался с данными, которые реализует этот компонент (ранее он изображался в форме овала). Нижний маленький прямоугольник ассоциировался с операциями или методами, реализуемыми компонентом. В простых случаях имена данных и методов записывались явно в этих маленьких прямоугольниках, однако в языке UML они не указываются.

Имя компонента

Имя компонента подчиняется общим правилам именования элементов модели в языке UML и может состоять из любого числа букв, цифр и некоторых знаков препинания. Отдельный компонент может быть представлен на уровне типа или на уровне экземпляра. Хотя его

графическое изображение в обоих случаях одинаковое, правила записи имени компонента несколько отличаются.

Если компонент представляется на уровне типов, то в качестве его имени записывается только имя типа с заглавной буквы в форме

<имя типа>.

Если же компонент представляется на уровне примеров, то в качестве его имени записывается в форме

<имя компонента> : <имя типа>.

При этом вся строка имени подчеркивается.

Так, в первом случае (рис. 10.1, а) для компонента уровня типов указывается имя типа, а во втором (рис. 10.1, б) для компонента уровня примеров – собственное имя компонента и имя типа.

Примечание

Хотя правила именования объектов в языке UML требуют подчеркивания имени отдельных экземпляров, применительно к компонентам в литературе подчеркивание их имени часто опускают. В этом случае запись имени компонента со строчной буквы характеризует компонент уровня примеров.

В качестве собственных имен компонентов принято использовать имена исполняемых файлов (с расширением EXE), имена динамических библиотек (расширение DLL), имена Web-страниц (расширение HTML), имена текстовых файлов (расширения TXT или DOC) или файлов справки (HLP), имена файлов баз данных (DB) или имена файлов с исходными текстами программ (расширения H, CPP для языка C++, расширение JAVA для языка Java), скрипты (PL, ASP) и другие.

Поскольку конкретная реализация логического представления модели системы зависит от используемого программного инструментария, то и имена компонентов будут определяться особенностями синтаксиса соответствующего языка программирования.

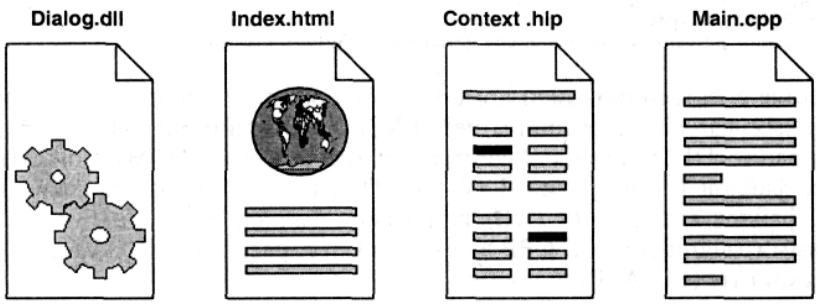
В отдельных случаях к простому имени компонента может быть добавлена информация об имени объемлющего пакета и о конкретной версии реализации данного компонента. Необходимо заметить, что в этом случае номер версии записывается как помеченное значение в фигурных скобках. В других случаях символ компонента может быть разделен на секции, чтобы явно указать имена реализованных в нем интерфейсов. Такое обозначение компонента называется *расширенным* и рассматривается далее в этой главе.

Виды компонентов

Поскольку компонент как элемент модели может иметь различную физическую реализацию, то иногда его изображают в форме специального графического символа, иллюстрирующего конкретные особенности реализации. Строго говоря, эти дополнительные обозначения не специфицированы в нотации языка UML. Однако,

удовлетворяя общим механизмам расширения языка UML, они упрощают понимание диаграммы компонентов, существенно повышая наглядность графического представления.

1) Для более наглядного изображения компонентов были предложены и стали общепринятыми следующие графические стереотипы: во-первых, стереотипы для компонентов развертывания, которые обеспечивают непосредственное выполнение системой своих функций. Такими компонентами могут быть динамически подключаемые библиотеки с расширением DLL (рис. 10.2, а), Web-страницы на языке разметки гипертекста с расширением HTML (рис. 10.2, б) и файлы справки с расширением HLP (рис. 2, в);



а б в г

Рис. 2. Варианты графического изображения компонентов

во-вторых, стереотипы для компонентов в форме рабочих продуктов. Как правило — это файлы с исходными текстами программ, как например, с расширениями H или CPP для языка C++.

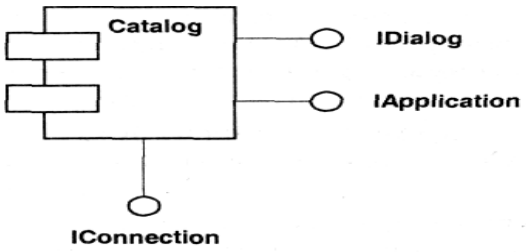
Интерфейсы

Следующим графическим элементом диаграммы компонентов являются интерфейсы. Последние уже рассматривались, поэтому здесь будут отмечены те их особенности, которые характерны для их представления на диаграммах компонентов.

В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис. 10.3, а). При этом имя интерфейса, которое рекомендуется начинать с заглавной буквы "I", записывается рядом с окружностью. Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом "интерфейс" и возможными секциями атрибутов и операций (рис. 10.3, б). Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

а



б

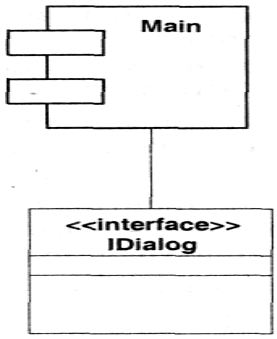


Рис. 3. Графическое изображение интерфейсов на диаграмме компонентов

Зависимости

В общем случае отношение зависимости также было рассмотрено ранее. Напомним, что отношение зависимости служит для представления факта наличия специальной формы связи между двумя элементами модели, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели. Отношение зависимости на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу).

Так, например, изображенный ниже фрагмент диаграммы компонентов (рис. 10.4) представляет информацию о том, что компонент с именем Main зависит от импортируемого интерфейса iDialog, который, в свою очередь, реализуется компонентом с именем Library. При этом для второго компонента этот интерфейс является экспортируемым.

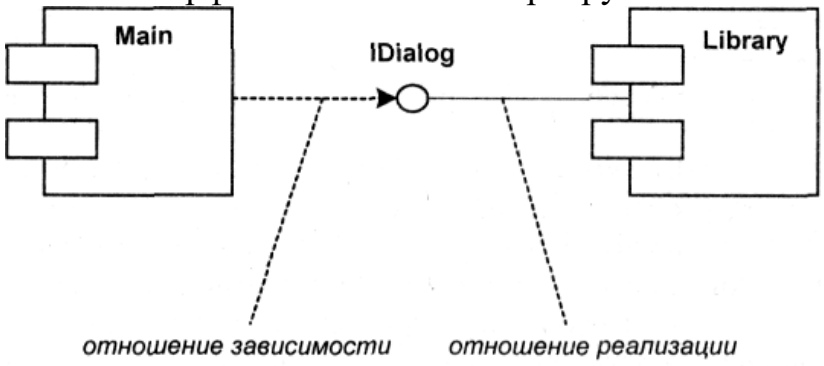


Рис. 4. Фрагмент диаграммы компонентов с отношениями зависимости

Диаграмма развертывания (размещения) (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы.

Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства, в большинстве случаев — часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единственной для системы в целом, поскольку должна всецело отражать особенности ее реализации. Эта диаграмма, по сути, завершает процесс ООАП для конкретной программной системы, и ее разработка, как правило, является последним этапом спецификации модели.

Цели, преследуемые при разработке диаграммы развертывания, следующие:

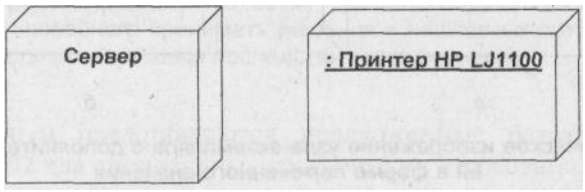
- указать размещение исполнимых компонентов программной системы по ее физическим узлам;
- показать физические связи между всеми узлами реализации системы на этапе ее исполнения;
- выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Для обеспечения этих требований диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками. Далее рассмотрим отдельные элементы, из которых состоят диаграммы развертывания.

Узел (*node*) представляет собой некоторый физически существующий элемент системы, который может обладать некоторым вычислительным ресурсом. В качестве вычислительного ресурса узла может рассматриваться наличие одного или нескольких процессоров, а также некоторого объема оперативной памяти. В языке UML понятие узла расширено — оно может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Графически узел на диаграмме развертывания изображается в форме трехмерного куба (строго говоря, псевдотрехмерного прямоугольного параллелепипеда). Узел имеет имя, которое

указывается внутри этого графического символа. Сами узлы могут представляться как в качестве типов (рис. 1, а), так и в качестве экземпляров (рис. б).



а б

Рис. 11. Графическое изображение узла на диаграмме развертывания

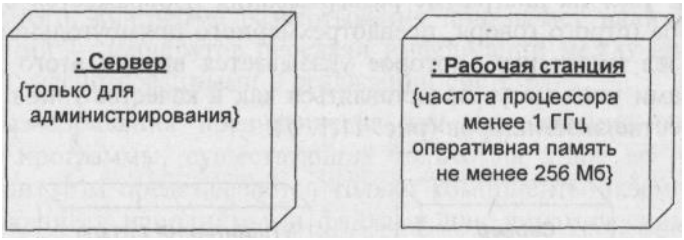
В первом случае имя узла записывается в форме: <имя типа узла> без подчеркивания и начинается с заглавной буквы. Во втором, имя узла-экземпляра записывается в виде <имя узла> : <имя типа узла>, а вся запись подчеркивается. Имя типа узла указывает на некоторую разновидность узлов, присутствующих в модели системы.

Например, аппаратная часть системы может состоять из нескольких персональных компьютеров, часть из которых выполняют функции сервера и соответствуют отдельным узлам-экземплярам в модели. Однако все эти узлы-экземпляры относятся к одному типу узлов, а именно — узлу с именем типа Сервер. Так, на представленном выше рисунке (рис. 1, а) узел с именем сервер относится к общему типу и никак не конкретизируется. Второй узел (рис. 1, б) является анонимным узлом-экземпляром конкретной модели принтера.

Так же, как и на диаграмме компонентов, изображения узлов могут расширяться, чтобы включить некоторую дополнительную информацию о спецификации узла. Если дополнительная информация относится к имени узла, то она записывается под этим именем в форме помеченного значения (рис. 2).

Если необходимо явно указать компоненты, которые размещаются или выполняются на отдельном узле, то это можно сделать двумя способами. Первый из них позволяет разделить графический символ узла на две секции горизонтальной линией. В верхней секции записывают имя узла, а в нижней секции — размещенные на этом узле компоненты (рис. 3, а).

Второй способ разрешает показывать на диаграмме развертывания узлы с вложенными изображениями компонентов (рис. 3, б). Важно помнить что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты и динамические библиотеки.



а б

Рис. 2. Графическое изображение узла-экземпляра с дополнительной информацией в форме помеченного значения

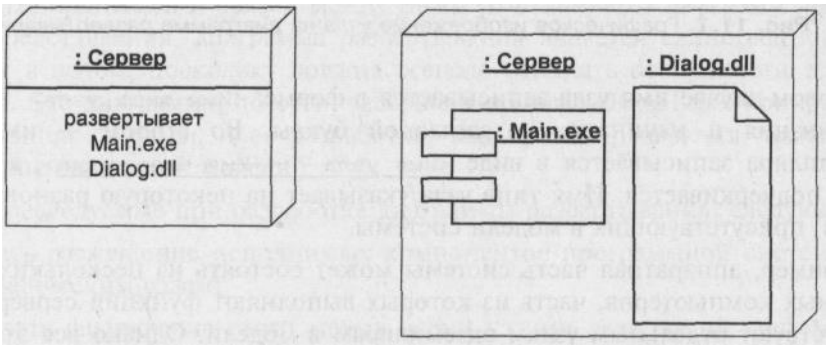


Рис. 3. Варианты графического изображения узлов-экземпляров с размещаемыми на них компонентами

В качестве дополнения к имени узла могут использоваться различные текстовые стереотипы, которые явно специфицируют назначение этого узла. Хотя в языке UML стереотипы для узлов не определены, разработчики предложили следующие текстовые стереотипы: «processor» (процессор), «sensor» (датчик), «modem» (модем), «net» (сеть), «printer» (принтер) и другие, смысл которых достаточно очевиден.

На диаграммах развертывания допускаются специальные условные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции. Однако пользоваться этой возможностью следует весьма осторожно, помня, что одно из основных достоинств языка UML следует из его названия -унификация графических элементов визуализации моделей.

3.Выполнение работы

1. Изучить раздел 1.
2. Изучить раздел 2. Выполнить упражнение 1.
3. Создать логическую модель системы в виде диаграммы компонентов.
4. Создать разработанную в п. 3 модель в среде Visio.

4. Содержание отчета

В качестве отчета о выполненной работе предъявите преподавателю:

3. на экране в среде Visio.: упражнение 1 (раздел 2) и модель диаграммы компонентов по индивидуальному заданию;

4. отчет в печатном виде содержащий:

- задание,
- модель реализации проекта.

5.Контрольные вопросы

1. Из каких основных элементов состоят диаграммы компонентов?
2. Из каких основных элементов состоят диаграммы размещения?
3. В каком случае используются диаграммы компонентов?
4. Сколько компонентов рекомендуется в проекте?
5. Какие обозначения на диаграмме размещения?
6. Какие стереотипы используются при оформлении диаграмм?
7. Чем отличаются диаграммы компонентов от диаграмм размещения?

Лабораторная работа №8

Управление проектами на основе гибкой методологии AGILE

1. Цель работы

Получить навыки управление проектами СИИ на основе гибкой методологии AGILE.

2.Теоретические сведения

Гибкая методология разработки (от англ. - Agile software development) - манифест, определяющий способ мышления и содержащий основные ценности и принципы, на которых базируется несколько подходов (фреймворков, от англ. framework — каркас, структура) к разработке программного обеспечения (хотя в последнее время идет тенденция и попытки применения гибкой методологии разработки к иным направлениям деятельности, не только в части информационных технологий), подразумевающих под собой интерактивную разработку, периодического (динамического) предоставления (обновления) требований от Заказчика и их реализацию посредством самоорганизующихся рабочих групп, сформированных из

экспертов различного профиля (разработчики, тестировщики, внедренцы и т.д.). Такой перевод Agile, как "гибкая методология разработки" не совсем корректен т.к. обычно Agile не называют методологией, а вот подходы на основе данного манифеста и есть методологии, но с точки зрения Agile их называют - фреймворки. На данный момент существует множество фреймворков (методологий),

подходы которых базируются на гибкой методологии разработки, например такие, как: Scrum, Extreme programming, FDD, DSDM и т.д.

Это набор итеративных подходов к разработке программного обеспечения, в которых требования и решения возникают в результате сотрудничества между самоорганизующимися межфункциональными командами. Agile-методы или Agile-процедуры часто поддерживают дисциплинированную методологию управления проектами, которая

поощряет регулярные проверки и адаптацию, а также философию лидерства, поощряющую командную работу, самоорганизацию и подотчетность. Также требуется набор передовых инженерных методов для обеспечения быстрой доставки высококачественного программного обеспечения и бизнес-стратегии, связывающей разработку с

потребностями клиентов и целями компании. Гибкая разработка требует, чтобы все процессы разработки были согласованы с концепциями **Agile Manifesto.**

Agile можно определить как сотрудничество заинтересованных лиц (stakeholders) для поставки ценности заказчику с частыми инкрементами, при постоянном размышлении (reflection) и адаптации. Это определение фокусируется на характеристиках, присущих любому agile окружению, а именно:

- Сотрудничество – как люди работают вместе, включая ко-

манду, занимающуюся разработкой, и заинтересованных лиц (stakeholders)

- Поставка ценности – цель прилагаемых усилий – это постав-

ка ценности заказчикам, что бы это ни было: программное обеспечение, более эффективные процессы или новые продукты.

- Частые инкременты – команда поставляет ценность каждые несколько дней, недель или месяцев, а не единожды, в конце проекта.
- Постоянное размышление и адаптация – проектная команда размышляет над подходами и проектом на регулярной основе, и настраивает свою работу в соответствии со сделанными выводами.

Agile – одна из методологий итеративной и пошаговой разработки ПО, в противоположность традиционной линейной методологии «водопад». Методология гибкой разработки определяет систему методов проектирования, разработки и тестирования на протяжении всего жизненного цикла ПО. Методы гибкой разработки (например, SCRUM) основаны на оперативном реагировании на изменения за счет применения адаптивного планирования, совместной выработки требований, рационализации самоорганизующихся кросс-функциональных групп разработчиков, а также пошаговой разработки ПО с четкими временными рамками. Этот подход используется во многих современных проектах разработки коммерческого ПО. В основе гибкой методологии разработки лежит либерально-демократический подход к управлению и организации труда команд, члены которой сконцентрированы на разработке конкретного программного обеспечения.

За счет того, что разработка программного обеспечения с применением гибкой методологии определяет серии коротких циклов (итераций), с длительностью 2-3 недели, достигается минимизация

рисков т.к. по завершению каждой итерации Заказчик принимает результаты и выдает новые или корректирующие требования т.е. контролирует разработку и может на неё сразу влиять. Каждая итерация включает в себя этапы планирования, анализа требований, проектирование, разработку, тестирование и документирование. Обычно одной итерации не достаточно для выпуска полноценного программного продукта, но при этом по окончании каждого этапа разработки должен появляться "осязаемый" продукт или часть функционала, которую можно посмотреть, протестировать и выдать дополнительные или корректирующие меры. На основе проделанной работы, после каждого этапа, команда подводит итоги и собирает новые требования, на основании чего вносит корректировки в план разработки программного обеспечения.

Одной из основных идей Agile, является взаимодействие внутри команды и с заказчиком лицом к лицу, что позволяет быстро принимать решения и минимизирует риски разработки программного обеспечения, поэтому команду размещают в одном месте, с географической точки зрения. Причем в команду входит представитель заказчика (англ. product owner - полномочный представитель заказчика или сам заказчик, представляющий требования к продукту; такую роль выполняет менеджер проекта от заказчика или бизнес-аналитик).

Agile-манифест разработки программного обеспечения

«Манифест гибкой методологии разработки программного обеспечения» был выпущен и принят в феврале 2001 года (штат ЮТА США, лыжный курорт The Lodge at Snowbird) группой экспертов. Данный манифест определяет 4 основные ценности и 12 принципов для методологий, базирующихся на нем, а также дает альтернативное видение подхода к разработке программного обеспечения в отличие от крупных и известных методов и методологий, но не является сам по себе методологией. Обычно Agile сравнивают в первую очередь с "методом водопада" ("waterfall"), т.к. на момент выхода манифеста, именно "метод водопада" являлся основным при планировании разработки программного обеспечения. В разработке и выпуске Agile манифеста принимали участие представители следующих методологий: Adaptive software development (ASD)

Crystal Clear

Dynamic Systems Development Method (DSDM)

Extreme Programming (XP)
Feature driven development (FDD)
Pragmatic Programming
Scrum

Собственно, данные методологии гибкой разработки существовали и до выпуска манифеста. Сам же выпуск манифеста дал новый толчок к развитию гибких методологий, заложил основы, можно сказать конституцию гибкого подхода к разработке программного обеспечения.

Основной метрикой agile-методов является рабочий продукт. Отдавая предпочтение непосредственному общению, agile-методы уменьшают объём письменной документации по сравнению с другими методами.

Это привело к критике этих методов как недисциплинированных. Мы постоянно открываем для себя более совершенные методы

разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

- Люди и взаимодействие важнее процессов и инструментов
- Работающий продукт важнее исчерпывающей документации
- Сотрудничество с заказчиком важнее согласования условий контракта
- Готовность к изменениям важнее следования первоначальному плану.

То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.

Основополагающие принципы Agile-манифеста

1) Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.

2) Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

3) Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

4) На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

5) Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте под-держку и полностью доверьтесь им.

6) Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

7) Работающий продукт — основной показатель прогресса.

8) Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.

9) Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.

10) Простота — искусство минимизации лишней работы — крайне необходима.

11) Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

12) Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

В agile подходах фокус делается на командную работу и взаимодействие, поэтому роли там более общие по своей природе, чем те, что выделяют в традиционных подходах. В agile окружении не описаны задачи, специфичные для бизнес аналитика, и поэтому практикующему бизнес аналитику нужно знать, какие методики бизнес анализа нужно применять в таких проектах. Вот четыре основные роли, присутствующие в agile проектах.

Product Owner основной специалист, принимающий решения по проекту. Эта роль отвечает за видение продукта (product vision), приоритезацию фич, в соответствии с бизнес ценностью, и за ответы на вопросы команды.

Scrum Master отвечает за процесс. Эта роль отвечает за окружение, подходящее для достижения успеха, за устранение препятствий и за получение тесного сотрудничества между всеми ролями.

Team – это группа из 5-9 человек, выделенных для проекта на полное время, они отвечают за самоорганизацию и поставку ценности заказчику в каждой итерации. Команда определяет, как разрабатывать продукт и как распределить работы в условиях выделенного на проект времени.

Stakeholder (заинтересованное лицо) – это любой, кто может повлиять на проект и внести вклад в определение бизнес целей продукта. Заинтересованные лица, активно вовлеченные в проект, это часть команды. Заинтересованные лица, которые не вовлечены активно в проект, могут взаимодействовать с product owner-ом, участвуя таким образом в пополнении баклога (backlog) продукта.

Гибкая модель — это тип инкрементной модели, которая разрабатывается в виде быстрых циклов. Гибкий процесс разбивает задачи на более мелкие итерации или добавочные выпуски, которые не требуют долгосрочного планирования. Каждая итерация обычно длится от одной до четырех недель и проверяется на качество.

Планирование и масштаб проекта заранее включают количество итераций и продолжительность каждой итерации. Каждую итерацию обрабатывает команда профессионалов, которые управляют всем циклом разработки и обеспечивают своевременную реализацию проекта.

Основные этапы гибкой модели включают:

Этап сбора требований. Этот этап включает в себя определение требований для надлежащего планирования проекта заблаговременно.

Осуществимость проекта и технические требования могут быть оценены на основе этих знаний.

Этап проектирования. После определения требований к проекту поработайте с заинтересованными сторонами, чтобы показать им объем проекта и новые функции с помощью блок-схемы высокого уровня.

Этап итерации. Также известный как этап строительства, этап итерации означает начало проекта, который включает в себя различные

этапы разработки. Разработчики начинают работать над развертыванием работающего проекта.

Этап тестирования. Этап тестирования включает в себя проверку качества продукта и его производительности.

Этап развертывания. На этом этапе команда развертывает продукт для рабочей среды пользователя.

Фаза обратной связи. Это последняя фаза процесса гибкой модели, на которой команда получает обратную связь. Руководитель группы гарантирует, что они работают с этой обратной связью.

Четыре идеала применяются по-разному в каждом подходе Agile, но все разработчики программного обеспечения полагаются на них при создании и поставке высококачественного работающего программного обеспечения.

Люди и взаимодействия важнее процессов и инструментов

Легко понять, почему люди ценятся выше, чем процессы или инструменты, потому что именно люди реагируют на потребности компании и управляют процессом разработки. Когда разработка управляется процедурами или инструментами, команда менее приспособлена к изменениям и с меньшей вероятностью оправдает ожидания клиента. Различие между оценкой людей и процессов можно ясно увидеть в общении. Индивидуальное общение является гибким и может происходить всякий раз, когда есть необходимость, тогда как общение должно быть запланировано в случае процессов и требует определенного контента.

Рабочее программное обеспечение важнее полной документации

Исторически сложилось так, что значительное количество времени уходило на документирование продукта при подготовке к разработке и конечной поставке. Для каждого из них требуется собственный набор технических спецификаций, технических требований, технических проспектов, документации по дизайну интерфейса, планов тестирования, планов документации и утверждений. Список был обширным, что часто приводило к длительным задержкам разработки. Agile не устраняет документацию; скорее, это упрощает ее, чтобы у разработчика было все необходимое для выполнения задачи, не увязая в деталях. Agile использует пользовательские истории для документирования требований, и их достаточно, чтобы разработчик программного обеспечения начал работать над новой функцией.

Agile Manifesto уделяет большое внимание документации, но еще большее значение придается работающему программному обеспечению.

Сотрудничество с заказчиком при заключении контракта Переговоры — это когда заказчик и продакт-менеджер прорабатывают детали доставки, с возможностью повторного согласования деталей по маршруту. Сотрудничество — это совсем другая игра. Клиенты обсуждают требования к продукту, часто в мельчайших деталях, до начала какой-либо работы с помощью методологий разработки, таких как Waterfall. Это означало, что заказчик участвовал в процессе разработки до его начала и после его завершения, но не на протяжении большей его части. Манифест Agile описывает клиента, который участвует и взаимодействует с командой разработчиков на протяжении всего процесса. Это значительно облегчает разработчикам выполнение требований заказчика. Хотя гибкие методологии могут включать клиента через регулярные промежутки времени для периодических демонстраций, проект может так же легко включать конечного пользователя, как и члена команды, который ежедневно посещает все встречи и обеспечивает соответствие продукта бизнес-потребностям клиента.

Реакция на изменение вместо следования плану

Это изменение было расценено как расходы в традиционной разработке программного обеспечения. Таким образом удалось избежать. Цель состояла в том, чтобы создать точные, сложные планы с определенным набором функций и всем, имеющим тот же приоритет, что и все остальное, и огромным количеством различных зависимостей

от доставки в определенном порядке, чтобы команда могла перейти к следующему кусочку головоломки.

Поскольку agile-итерации короткие, приоритеты можно корректировать от итерации к итерации, а новые функции можно добавлять на следующей итерации. Изменения, согласно Agile, всегда улучшают проект, т. е. придают новую ценность.

Концепция адаптации метода, возможно, является лучшим примером конструктивного подхода Agile к изменениям. Agile-подходы позволяют Agile-команде изменять процесс так, чтобы он подходил команде, а не наоборот.

Частая поставка рабочего программного обеспечения — Scrum допускает это, поскольку команда работает в спринтах или итерациях программного обеспечения, что обеспечивает постоянную доставку рабочего программного обеспечения.

Сотрудничество между заинтересованными сторонами бизнеса и разработчиками на протяжении всего проекта. Более эффективные решения принимаются, когда бизнес и техническая команда согласованы. Это также уменьшает количество изменений в программном обеспечении.

Поддерживайте, доверяйте и мотивируйте вовлеченных людей. Недовольные команды с меньшей вероятностью сделают свою работу лучше, чем мотивированные.

Обеспечьте взаимодействие лицом к лицу. Эффективное общение может быть достигнуто, когда команды разработчиков находятся рядом друг с другом.

Работающее программное обеспечение является основным мерилом прогресса. Предоставление функционального программного обеспечения заказчику является основным фактором, измеряющим прогресс и успех.

Agile-процессы для поддержки постоянного темпа разработки. Команды устанавливают темп, в котором они могут создавать функционирующее программное обеспечение, которое можно

повторять и поддерживать, и они придерживаются его с каждым выпуском.

Внимание к техническим деталям и дизайну повышает гибкость. Надлежащие таланты и продуманный дизайн гарантируют, что команда

сможет идти в ногу со временем, регулярно развивать продукт и адаптироваться к изменениям.

Простота — разрабатывайте ровно столько, чтобы выполнить работу прямо сейчас, чтобы ее можно было выполнить быстро и эффективно.

Самоорганизующиеся команды поощряют отличные архитектуры, требования и проекты. Качественные товары производятся квалифицированными и мотивированными членами команды, которые имеют право принимать решения, берут на себя ответственность, регулярно взаимодействуют с другими членами команды и обмениваются идеями.

Регулярные размышления о том, как стать более эффективными. Члены команды могут работать более эффективно, улучшая свое самосознание, улучшая свои процессы и изучая новые навыки и методы.

Цель Agile — связать разработку с потребностями бизнеса, и она доказала свою эффективность. Клиенты находятся в центре гибких инициатив, которые способствуют вкладу и вовлечению потребителей.

В результате Agile превратился в всеобъемлющее видение разработки программного обеспечения для всей индустрии программного обеспечения и отдельного бизнеса.

Поскольку Agile — это итеративный подход к разработке программного обеспечения, в отличие от модели линейного водопада, гибкие проекты состоят из небольших циклов, называемых спринтами.

Каждый спринт представляет собой мини-проект с бэклогом и состоит из всех этапов Agile, таких как определение, проектирование, разработка, демонстрация и доставка. После завершения цикла спринта небольшой модуль всего программного продукта готов к доставке, и описанный выше процесс повторяется, что приводит к постепенному росту продукта. Следование этому подходу в значительной степени снижает вероятность сбоя продукта, поэтому большинство организаций по всему миру используют agile-подход в своей практике.

Ниже приведены следующие аспекты гибкого процесса:

а.) Гибкость: требования и объем работ меняются в зависимости от потребностей бизнеса.

б.) Разбивка работы: продукт разрабатывается по модулям небольшими циклами (спринтами).

в) Улучшения: изучение прошлых ошибок для улучшения конечного продукта.

d.) Сотрудничество с клиентами: предоставление подробной информации о любых изменениях в требованиях или принятие предложений команды на протяжении спринтов.

1. Задание

Изучить методику управления проекта разработки СИИ **Agile.**

Подготовить ртеферат и презентацию окладв об оснолвных особенностях методологии **Agile.**

2. Контрольные вопросы

1. Что такое гибкая разработка программного обеспечения ?
2. Включает ли в себя разработка проекта как организации планируют, разрабатывают, тестируют и выпускают программное обеспечение?
3. Что включает agile-подход в своей практике реализации проектов СИИ?

Лабораторная работа №9

Управление проектами СИИ на базе методологии SCRUM

Цель работы: **О**свреение приемов использования методологии SCRUM для азрабатываемой системы искусственного интеллекта и разработка требований к проекту.

1. Теоретические сведения

Scrum — это набор правил, благодаря которым команда налаживает гибкий рабочий процесс, разработка ведется итерациями, четко обозначаются цели каждой итерации и задачи каждого члена команды. Благодаря фреймворку компании могут применять принципы и ценности методологии управления проектами по Agile [**Ошибка! Источник ссылки не найден.**].

Scrum (как, собственно, и Agile) зародился для упрощения рабочих процессов в компаниях, которые занимаются разработкой программного обеспечения и управлением продуктов. В наше время методика Scrum используется в сферах маркетинга, брендинга, дизайна и многих других. Это отличный фреймворк для работы над динамично развивающимися проектами. Scrum направлен на самостоятельную работу над проектом, а не на решение данных «сверху» задач.

Эти два понятия регулярно путают, считая, что Agile и Scrum одно и то же. Обе методологии фокусируются на постоянном совершенствовании продукта, а не на его выпуске. Это гибкие структуры, суть которых в постоянном изменении, адаптивности, направленности на самостоятельную работу участников, нестандартных подходах к работе.

Разница кроется в масштабе двух подходов.

Agile — это особый образ мышления. Идея, стоящая за тем, к чему вы стремитесь — например, к адаптивности, самоконтролю или скорости выполнения заданий.

Scrum — это инструкция по применению. Четкий план, описывающий каждый шаг по внедрению Agile в разработку продукта. Можно сказать, что Scrum — это методология управления проектами с конкретными этапами, в которой четко определены роли и события.

Система управления проектами Scrum основана на пяти ценностях:

- Преданность (Commitment);
- Сфокусированность (Focus);
- Открытость (Openness);
- Уважение (Respect);
- Смелость (Courage).

В контексте Scrum все, что делают работники должно быть направлено на усиление этих ценностей, и ни в коем случае не подрывать их.

И это рабочая методика, так как 58% Agile-команд используют фреймворк Скрам. Благодаря ему члены Scrum-команды могут учитывать нужды клиентов на протяжении всей работы над проектом.

В Scrum нет стандартов идеального долгосрочного планирования, на которую опираются в традиционных рабочих подходах. Фреймворк сосредотачивается на выполнении задач на короткой дистанции.

Состав Scrum команды.

Прежде чем говорить о структуре фреймворка, рассмотрим, кто обычно входит в состав Scrum-команды.

Владелец продукта — тот, кто налаживает связь между командой и заинтересованными лицами. Он понимает, что нужно клиентам, контролирует общее видение проекта и его цели.

Scrum-мастер — один из членов команды, в задачи которого входит внедрение и укрепление ценностей Scrum на командных митингах и поддержка участников во время выполнения задач.

Члены команды — остальные участники Scrum-команды. Все они равноправны и каждый выполняют свою задачу.

Заинтересованные лица, упомянутые выше — не члены команды. Это все те, кто инвестирует в результат проекта. Например, особые клиенты, внутренние пользователи продукта, руководители высшего звена и прочие. Ключевые заинтересованные лица присутствуют на важных встречах и рассматривают ключевые решения по модернизации продукта, а также предоставляют обратную связь после каждой итерации.

Scrum-команда обязательно кросс-функциональна. Например, в команде по созданию мобильных приложений должны быть UX-дизайнеры, разработчики, специалисты по API и прочие. Каждый

участник обязан располагать соответствующими инструментами для завершения итерации. Поэтому у них не должно возникать необходимости передавать часть работы на аутсорс. Это один из основных принципов управления проектами по Scrum.

Этапы Scrum.

В фреймворке Scrum можно выделить пять основных этапов:

1. Предварительное планирование.

Постановка целей, определение видения продукта. Лидер проекта обозначает задачи, намечает дорожную карту проекта. Создание и доработка бэклога продукта — списка функций, требований и исправлений ошибок, где для команды прописываются все этапы работы над продуктом. Обычно к этапу предварительного планирования объема работы присоединяются заинтересованные лица.

2. Планирование.

На этом этапе участники команды вместе занимаются планированием спринта и выбором функций для включения в его бэклог. Поскольку их обычно определяет точка зрения пользователя, они называются пользовательскими историями. Необходимо разбить большие требования (которые обычно называют «эпиками») на простые задачи с приблизительной оценкой времени выполнения. Стоит убедиться, что бэклог спринта достаточно небольшой, его получится выполнить в рамках планируемого времени, распределить задачи и назначить ответственных за пользовательские истории.

3. Спринт, этап реализации.

Работа идет над итерацией или инкрементом продукта (ощутимый результат работы одного спринта), который реализуется в конце спринта. Необходимо проводить ежедневные митинги или Scrum-собрания, на которых будет обсуждаться прогресс, задачи, потенциальные трудности.

4. Тестирование и проверка.

По окончанию спринта клиенты и пользователи продукта (заинтересованные лица) тестируют новые функции или улучшения продукта. Если все работает как надо, итерация считается завершенной.

5. Ретроспектива.

Анализ итогов спринта вместе со Scrum-командой, во время которого разбираются ошибки и выдвигаются предложения по

улучшению работы. Общий бэклог продукта актуализируется в зависимости от результатов работы над обновлениями и смены приоритетов у заинтересованных лиц.

Ценность фреймворка Scrum для управления проектами

Scrum популярен за счет ряда преимуществ для команд, которые решили использовать его для организации работы [**Ошибка! Источник ссылки не найден.**]:

- Наглядность процесса. Намеченные задачи к выполнению можно представить в удобном виде на доске в таск-трекере. Визуализируя задачи на Канбан-доске, Scrum-команда всегда видит, как продвигается работа, к кому обращаться по тем или иным вопросам и какие задачи в работе сегодня.

- Концентрация на важном. Предварительное планирование целей спринта помогает не распыляться на другие задачи, оставаться сосредоточенным и собранным.

- Конкретные результаты. Итогом итерации по Scrum всегда является какое-то улучшение, определенное достижение. Его можно оценить и однозначно ответить, достигла ли команда поставленных целей в полной мере или нет. Для быстрой и удобной оценки используются разнообразные Agile-метрики: упомянутый выше график сгорания задач (Burndown chart), накопительная диаграмма потока (Cumulative Flow Diagram) и другие.

- Все участники процесса поддерживают связь. Все участники команды, Scrum-мастер, владелец продукта, заказчик и заинтересованные лица всегда поддерживают коммуникацию. Любые уточнения всегда можно получить быстро, чтобы не только в короткие сроки выпускать продукт, но и поддерживать его актуальность, вовремя реагируя на изменения рынка и нужды клиентов.

- Высокая гибкость и адаптивность. Несмотря на точное планирование целей, методика Scrum все же не подразумевает обязательного следования одним и тем же правилам. Вы можете адаптировать подход к потребностям именно вашей команды и для достижения ваших целей, организовав работу максимально удобно.

Отличия Scrum как подхода для организации работы

Среди множества различных подходов и методологий организации работы, Scrum выделяется следующими особенностями:

- Четко зафиксированные роли сотрудников, цели и этапы спринтов. Во время итерации Scrum-команда всегда знает, кто, над чем и для чего работает в любой момент времени.

- Кросс-функциональность команды. Команда включает в себя разных специалистов, которые работают в связке. Например, для создания видеоигры необходима команда из разработчиков, графических дизайнеров, тестировщиков, сценаристов и др. Полностью укомплектованная команда для проекта самодостаточна и не требует сторонних экспертов для выполнения задач.

- Отсутствие долгосрочного планирования. Scrum не подходит для построения долгосрочных планов. При данном подходе приоритеты и цели постоянно меняются между итерациями, гибко адаптируясь к текущим требованиям к продукту. Краткосрочные спринты помогают Scrum-команде единым рывком выполнять поставленные цели, при этом держится фокус на обозначенных задачах без траты ресурсов на остальные дела.

- Предварительное планирование задач для краткосрочных спринтов. Scrum требует обязательного составления подробного бэклога и выделения целей на каждый цикл.

- Есть только одно лицо для коммуникации между командой и заинтересованными лицами — владелец продукта. Так устраняется риск противоречивости полученной информации по задачам, все запросы и ответы исходят от одного человека.

- Регулярное общение с командой. Участники проекта говорят о прогрессе и проблемах в работе на ежедневных собраниях, встречах по пополнению очереди задач и других видах собраний для обмена информацией и получением обратной связи. Обсуждение текущих сложностей и способов их решения — важная часть работы по Scrum.

- Обязательная оценка результата и получение обратной связи от заинтересованных лиц после окончания спринта через владельца продукта. Без получения одобрения от заинтересованных лиц результата работы цель не может считаться достигнутой и следующий спринт не может быть начат.

3. Задание

Изучить методику управления проекта разработки СИИ **Scrum.**

Подготовить ртеферат и презентацию окладв об оснолвных особенностях методологии **Scrum.**

4. Контрольные вопросы

4. Что такое гибкая разработка программного обеспечения ?
5. Включает ли в себя разработка проекта как организации планируют, разрабатывают, тестируют и выпускают программное обеспечение?
6. Что включает Scrum- подход в своей практике реализации проектов?

Список литературы

1. Управление проектами : фундаментальный курс : учебник / А. В. Алешин, В. М. Аньшин, К. А. Багратиони [и др.] ; под ред. В. М. Аньшина, О. Н. Ильиной. - Москва : Издательский дом Высшей школы экономики, 2022. - 800 с. - (Учебники Высшей школы экономики). - URL: https://biblioclub.ru/index.php?page=book&id=699578 (дата обращения 16.05.2025) . - Режим доступа: по подписке. - Текст : электронный.
2. Ипатова, Э. Р. Методологии и технологии системного проектирования информационных систем : учебник / Э. Р. Ипатова, Ю. В. Ипатов. – 3-е изд., стер. – Москва : ФЛИНТА, 2021. – 256 с. – (Информационные технологии). – URL: https://biblioclub.ru/index.php?page=book&id=79551 (дата обращения: 16.05.2025). – Режим доступа: по подписке. – Текст : электронный.
3. Кугаевских, А. В. Проектирование информационных систем. Системная и бизнес-аналитика : учебное пособие / А. В. Кугаевских ; Новосибирский государственный технический университет. – Новосибирск : Новосибирский государственный технический университет, 2018. – 256 с. – URL: https://biblioclub.ru/index.php?page=book&id=573827 (дата обращения: 16.05.2025). – Режим доступа: по подписке. – Текст : электронный.
4. Шуваев, А. В. Программная инженерия : учебное пособие для магистрантов направления подготовки 09.04.02 – Информационные системы и технологии / А. В. Шуваев ; Ставропольский государственный аграрный университет, Кафедра информационных систем. – Ставрополь : Ветеран, 2020. – 84 с. - URL:https://biblioclub.ru/index.php?page=book&id=700960 (дата обращения: 16.05.2025). – Режим доступа: по подписке. – Текст : электронный.
5. Антонов, В. Ф. Методы и средства проектирования информационных систем : учебное пособие / В. Ф. Антонов, А. А. Москвитин ; Северо-Кавказский федеральный университет. – Ставрополь : Северо-Кавказский Федеральный университет (СКФУ), 2016. – 342 с. - URL: http://biblioclub.ru/index.php?page=book&id=458663 (дата обращения: 16.05.2025). - Режим доступа: по подписке. - Текст : электронный.
6. 6. Влацкая, И. В. Проектирование и реализация прикладного программного обеспечения : учебное пособие / И. В. Влацкая ; Н. А. Заельская ; Н. С. Надточий. - Оренбург : ОГУ, 2015. - 119 с. - URL: http://biblioclub.ru/index.php?page=book&id=439107 (дата обращения 16.05.2025) . - Режим доступа: по подписке. - Текст : электронный.

Золотов, С. Ю. Проектирование информационных систем : учебное пособие / С. Ю. Золотов. - Томск : Эль Контент, 2013. - 88 с. – URL: https://biblioclub.ru/index.php?page=book_red&id=208706 (дата обращения 16.05.2025) . - Режим доступа: по подписке. - Текст : электронный