МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ
ПИ (филиал) ДГТУ в г. Таганроге

ЦМК «Прикладная информатика»

.

Практикум

По выполнению практических работ

по дисциплине

«Устройство и функционирование информационной системы»

Таганрог

2023

Составители: А.В. Ганциевский

Практикум по выполнению практических работ по «Устройство и функционирование информационной системы» по специальности СПО, ПИ (филиала) ДГТУ в г. Таганроге, 2023г.

В практикуме кратко изложены теоретические вопросы, необходимые для успешного выполнения практической работы, рабочее задание и контрольные вопросы для самопроверки.

Предназначено для обучающихся по специальности 09.02.04 Информационные системы (по отраслям)

Ответственный за выпуск:

Председатель ЦМК: _____ О.В. Андриян

Введение

В учебно-методическом пособии к практикуму по курсу «Устройство и функционирование информационной системы» изложены сведения, необходимые для успешного выполнения практических занятий по данному курсу. Описан процесс работы с инструментарием, применяемым на практических занятиях, представлен ряд типичных задач и подходы к их решению. Практические занятия посвящены углубленному знакомству обучающихся с текстовыми редакторами, графическими редакторами, электронными таблицами, базами данных, архиваторами, виртуальными средами, операционными системами, способами безопасности, основы работы с реестром и облачными технологиями. Цель настоящего пособия – помочь обучающимся при выполнении практических работ, выполняемых для закрепления знаний по теоретическим основам и получения практических навыков работы на компьютерах.

Обучающийся должен знать: выделять жизненные циклы проектирования информационной системы; использовать методы и критерии оценивания предметной области и методы определения стратегии развития бизнес-процессов организации; использовать и рассчитывать показатели и критерии оценивания информационной системы, осуществлять необходимые измерения. Обучающийся должен уметь: типы организационных структур; реинжиниринг бизнес-процессов; требования к проектируемой системе, классификацию информационных систем, структуру информационной системы, понятие жизненного цикла информационной системы; модели жизненного цикла информационной системы, методы проектирования информационных систем; технологии проектирования информационных систем, оценку и управление качеством информационных систем; организацию труда при разработке информационных систем; оценку необходимых ресурсов для реализации проекта. Данное учебно-методическое пособие предназначено для обучающихся 3-4 курса.

Правила выполнения практических занятий

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Практическое занятие № 1.
Выделение жизненных циклов ИС

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Жизненный цикл ИС –это непрерывный процесс с момента принятия решения о необходимости принятия решения о необходимости ее создания до полного завершения ее эксплуатации.

Процесс проектирования АИС регламентирован следующей документацией (стандартами, методологиями, моделями):

· ГОСТ 34.601-90. Введен в действие 01.01.1992. Устанавливает стадии и этапы создания автоматизированных систем и дает содержание работ на каждой стадии. Стадии и этапы работы, закрепленные в стандарте, соответствуют каскадной модели жизненного цикла.

· ISO/IEC 12207:1995. Международный стандарт, описывающий процессы жизненного цикла программного обеспечения. Содержит описание более, чем 220 базовых работ, выполнение которых может потребоваться в процессе создания ИС. Все процессы ЖЦ ПО подразделяются на три большие группы:

o Основные процессы (приобретение, поставка, разработка, эксплуатация, сопровождение);

o Вспомогательные процессы (документирование, управление конфигурацией, обеспечение качества, разрешение проблем, аудит, аттестация, совместная оценка, верификация);

o Организационные процессы (создание инфраструктуры, управление, обучение, усовершенствование).

Для реализации положений стандарта должны быть выбраны инструментальные средства, совместно образующие взаимосвязанный комплекс технологической поддержки и

автоматизации жизненного цикла программного обеспечения и не противоречащие предварительно скомпонованному набору нормативных документов. Чтобы облегчить практическое применение стандарта, международной организацией по стандартизации были разработаны и утверждены следующие документы:

o ISO/IEC TR 15271:1998 – руководство по применению ISO/IEC 12207;

o ISO/IEC TR 16326:1999 – руководство по управлению проектами при использовании ISO/IEC 12207.

· ISO/IEC 15288:2002. Международный стандарт, описывающий возможные процессы жизненного цикла систем, созданных человеком. Был создан с учетом опыта проектирования автоматизированных информационных систем, а также с привлечением специалистов различных областей: системной инженерии, программирования, администрирования, управления качеством, безопасностью и т.д. Предполагается, что стандарт содержит полное множество процессов, которые могут протекать в ходе жизненного цикла системы. Таким образом, задача разработчика ИС заключается в формировании необходимого ему множества – среды процессов. В обзоре стандарта отмечается, что в нем не содержится описания методов и процедур, необходимых для обеспечения выполнения целей, задач и результатов указанных процессов. В 2003 году выпущено руководство по применению стандарта (ISO/IEC TR 19760:2003). В настоящее время продолжается работа над подготовкой новой редакции стандарта серии 15288.

Модели ЖЦ ИС.

Под моделью ЖЦ ИС понимается структура определяющая последовательность выполнения и взаимосвязи процессов действий и задач на протяжении жизненного цикла.

Модель жизненного цикла ИС— это структура, описывающая процессы, действия и задачи, которые осуществляются и ходе разработки, функционирования и сопровождения в течение всего жизненного цикла системы.

I. Каскадная модель описывает классический подход к разработке систем в любых предметных областях; широко использовалась в 1970—80-х гг.

Каскадная модель предусматривает последовательную организацию работ, причем основной особенностью модели является разбиение всей работы на этапы. Переход от предыдущего этапа к последующему происходит только после полного завершения всех работ предыдущего.

II. Итерационная модель заключается в серии коротких циклов (шагов) по планированию, реализации, изучению, действию. Разработка ИС ведется *итерациями* с

циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.

III. Спиральная модель, в отличие от каскадной, но аналогично предыдущей предполагает итерационный процесс разработки ИС. При этом возрастает значение начальных этапов, таких как анализ и проектирование, на которых проверяется и обосновывается реализуемость технических решений путем создания прототипов.

ЗАДАНИЕ НА РАБОТУ

Задание.

1. Ознакомиться с теоретическими сведениями по лабораторной работе

2. Определить достоинства и недостатки моделей ЖЦ ИС

3. Выбрать и обосновать выбор модели ЖЦ ИС для выполнения индивидуального проектного задания.

4. Сформировать план построения ИС индивидуального проектного задания, с использованием программных средств.

Содержание отчета.

Отчет оформляется на листах формата А4 и содержит

1. Название работы

2. Цель работы

3. Заполненную таблицу «Достоинства и недостатки моделей ЖЦ ИС»

Модель ЖЦ ИС Достоинства Недостатки

Каскадная

Итерационная

Спиральная

4. Выбранную модель ЖЦ ИС и обоснование выбора для персонального проектного задания на разработку ИС.

5. План построения ИС персонального проектного задания в форме:

№ п/п	Название стадии (этапа) работ	Содержание работ	Результат работ	Применяемые программные средства

Контрольные вопросы к практическому занятию №1

1.Что отражает модель ЖЦ ИС?

2. Укажите свойства каскадной модели ЖЦ.

3. Укажите свойства каскадной модели ЖЦ

4. Укажите свойства итерационной модели ЖЦ

5. Какую модель ЖЦ следует использовать при создании простых ИС?

6. Какая модель ЖЦ наиболее объективно отражает реальный процесс создания сложных систем?

7. Какие процессы относятся к группе основных в соответствии со стандартом ISO/IEC 12207?

Практическое занятие № 2.
Оценка предметной области и уровня автоматизации, построение схемы бизнес-процессов

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

В случае создания ПО для информационной системы управления предприятием (совокупность средств, методов и персонала для обработки, хранения и выдачи информации) структурный анализ начинается с исследования того, как организована система управления предприятием, с обследования функциональной и информационной структуры системы управления, чтобы понять, как работает организация, которую собираются автоматизировать.

Для описания работы предприятия необходимо построить модель. Такая модель должна быть адекватна предметной области; следовательно, она должна содержать в себе знания всех участников бизнес-процессов организации.

По результатам обследования аналитик строит модель «как есть»: обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется. Далее создают модель «как надо»: усовершенствованную обобщенную логическую модель, отображающую реорганизованную предметную область или ее часть, которая подлежит автоматизации. Эта стадия анализа содержит элементы проектирования. Структурные, функциональные модели созданные на ранних этапах проектирования программной систем, помогут проектировщику выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки. Функциональные диаграммы предметной области помогут понять, как выполняются отдельные операции организации, которые собираются автоматизировать. На этом уровне определяются все функции, которые выполняет объект, и процессы , протекающие в объекте (например, подразделениях предприятия) для выполнения функций, определяются связи между функциями, между процессами. Функция - преобразование входных потоков в выходные, осуществляемое в соответствии с некоторыми внутренними правилами. Выполнение функции обеспечивает процесс. Процесс - совокупность взаимосвязанных действий (работ), преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами. Для описания работы организации необходимо построить модель и выделить те процессы, которые должны быть автоматизированы.

Наиболее удобным языком моделирования бизнес-процессов является методология функционального моделирования - IDEF0, предложенный более 20 лет назад Дугласом Россом (SoftTech, Inc.) и называвшийся первоначально SADT - Structured Analysis and Design Technique.

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

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

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

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

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами; другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования - вопросы, на которые построенная модель должна дать ответ; другими словами, первоначально необходимо определить область (Scope) моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента - широту и глубину. Широта подразумевает определение границ модели - мы определяем, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени - трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции.

После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему; поскольку все объекты модели взаимосвязаны, внесение нового объекта может быть не просто арифметической добавкой, но в состоянии изменить

существующие взаимосвязи. Внесение таких изменений в готовую модель является, как правило, очень трудоемким процессом (так называемая проблема "плавающей области").

Модели AS-IS и TO-BE. Целью построения функциональных моделей обычно является выявление наиболее слабых и уязвимых мест деятельности организации, анализ преимуществ новых бизнес-процессов и степени изменения существующей структуры организации бизнеса. Анализ недостатков и "узких мест" начинают с построения модели AS-IS (Как есть), т. е. модели существующей организации работы. Модель AS-IS может строиться на основе изучения документации (должностных инструкций, положений о предприятии, приказов, отчетов и т. п.), анкетирования и опроса служащих предприятия, создания фотографии рабочего дня и других источников. Полученная модель AS-IS служит для выявления неуправляемых работ, работ не обеспеченных ресурсами, ненужных и неэффективных работ, дублирующийся работ и других недостатков в организации деятельности предприятия. Исправление недостатков, перенаправление информационных и материальных потоков приводит к созданию модели TO-BE (Как будет) - модели идеальной организации бизнес-процессов. Как правило, строится несколько моделей TO-BE, среди которых определяют наилучший вариант.

ЗАДАНИЕ НА РАБОТУ

1. Опишите процесс учета посещения студентов учебных занятий и успеваемости студентов с точки зрения работника деканата.

2. Разработать программный модуль «Учет успеваемости студентов». Программный модуль предназначен для оперативного учета успеваемости студентов в сессию деканом, заместителями декана и сотрудниками деканата. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому.

3. Опишите процесс учета студентов, обучающихся в институте от процесса зачисления студента до получения диплома с точки зрения работника деканата. Разработать программный модуль «Личные дела студентов». Программный модуль предназначен для получения сведений о студентах сотрудниками деканата, профкома и отдела кадров. Сведения должны храниться в течение всего срока обучения студентов и использоваться при составлении справок и отчетов.

4. Опишите процесс организации рабочего дня руководителя с точки зрения его секретаря. Разработать приложение «Органайзер». Приложение предназначено для записи,

хранения и поиска адресов и телефонов физических лиц, и организаций, а также расписания, встреч и др. Приложение предназначено для организации рабочего дня руководителя.

5. Опишите процесс работы кафедры вуза с точки зрения преподавателя.

6. Разработать программный модуль «Кафедра», содержащий сведения о сотрудниках кафедры (ФИО, должность, ученая степень, дисциплины, нагрузка, общественная работа, совместительство и др.). Модуль предназначен для использования сотрудниками отдела кадров и деканата.

Контрольные вопросы к практическому занятию №2

1. Что такое жизненный цикл программного продукта?

2. Дайте определение модели жизненного цикла ПО.

3. Приведите этапы разработки программного средства.

4. Что представляет собой структурный подход к разработке ПС?

5. Что включает в себя этап пред проектные исследования?

6. В чем преимущество построения модели предметной области при разработке ПС?

7. Перечислите особенности методологии SADT?

8. Для чего строят модели AS-IS и TO-BE?

9. Что такое бизнес-процесс?

10. В каких отношениях находятся заказчик и разработчик при выработке требований к программному средству?

Практическое занятие № 3.
Построение организационно-функциональной модели

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 6 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

ЗАДАНИЕ НА РАБОТУ

Задание 1.

1.Разработать и составить организационную структуру торгового комплекса, в котором имеются директор ТК, три отдела: бакалейно-гастрономический, мясной, овощной.

2. Определить тип организационной структуры управления.

3. Определите уровни управления в данной организации.

4.Перечислите преимущества и недостатки данной структуры.

Задание 2.

1. Разработать и составить организационную структуру управления акционерного общества, в котором имеются:
   - общее собрание акционеров;
   - Совет директоров;
   - Генеральный директор;
   - Директора: директор по коммерции; директор по общим вопроса; директор по экономике.
   - Торговый отдел, главный товаровед;
   - Служба главного инженера, Служба инженера по технике безопасности, Служба транспортного отдела, Служба начальника отдела снабжения.
   - Планово-экономический отдел, Отдел организации торговли, Бухгалтерия,
2. Определить тип организационной структуры управления.
3. Перечислите преимущества и недостатки данной структуры

Задание 3.

1. Разработать и составить организационную структуру фирмы «Стандарт», в которой имеются:
   - Руководитель фирмы «Стандарт»;
   - Отдел химических продуктов;
   - Функциональное обеспечение проекта: производственные мощности, исследования и разработки, материально-техническое обеспечение, кадры, контроль и бухучет.
   - проекты: Проект «Продукт XXI», Проект «Здорово!», Проект «Эковзгляд»
   - последовательность осуществления операций каждого проекта: производственная группа, группа конструкторов-технологов, группа снабжения, кадровая группа, бухгалтерская группа.
2. Определить тип организационной структуры управления
3. Определите уровни управления в данной организации.

Контрольные вопросы к практическому занятию №3

1. Что такое организационная структура предприятия? Для чего необходимо разрабатывать организационную структуру?
2. Какие существуют типы организационных структур? Перечислите их преимущества и недостатки.

3. Проведите сравнительный анализ линейно-функциональной структуры управления и проектной? В каких видах бизнеса используются эти структуры?
4. Проведите сравнительный анализ матричной структуры управления и процессной? В каких видах бизнеса используются эти структуры?
5. Проведите сравнительный анализ процессов построения организационных структур в различных программных системах.
6. Какие особенности построения организационных структур в системах моделирования CA ERwin Process Modeler, ARIS и Business Studio?

Практическое занятие № 4.
BPwin- средство функционального моделирования (IDEF0), BPwin-средство моделирования потоков данных (DFD)

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 6 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

BPwin - CASE-средство верхнего уровня, помогающее проводить анализ и реорганизацию бизнес-процессов. Поддерживается методология IDEF0 (функциональная модель), IDEF3 (Work Flow Diagram), DFD (Data Flow Diagram). Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии (так называемая модель AS-IS) и идеального положения вещей – того, к чему надо стремиться (модель TO-BE). Процесс построения информационной модели в BPwin состоит из следующих шагов:• построение контекстной диаграммы; • проводится функциональная декомпозиция; • после каждого сеанса декомпозиции проводится сеанс экспертизы. На основе модели BPwin можно построить модель данных. В программе поддерживается связь с ERwin

ЗАДАНИЕ НА РАБОТУ

1. Согласно варианту, создайте контекстную диаграмму. Определите цель, точку зрения модели. Опишите свойства в соответствующих закладках диалога Model Properties.

2. Задайте входы, выходы, механизмы и управление.

3. Создайте декомпозицию контекстной диаграммы, состоящую из 2-3 блоков. Задайте автоматическую нумерацию блоков и ICOM-кодов.

4. Установите связи между блоками. Задайте имена дуг.

5. Сохраните проект в отдельный файл.

Вариант 1 Система должна описывать порядок подготовки к экзамену, предполагающий получение отличной оценки.

Вариант 2 Система должна описывать порядок выполнения практической работы по дисциплине «Проектирование ИС».

Вариант 3 Система должна описывать порядок получения водительских прав.

Вариант 4 Система должна описывать порядок организации городского спортивного соревнования.

Вариант 5 Система должна описывать порядок организации общеинститутского студенческого мероприятия.

Вариант 6 Система составления учебного графика дисциплин, изучаемых на факультете

Вариант 7 Система должна описывать порядок поставок товара в систему розничных киосков.

Вариант 8 Система должна описывать порядок обработки заказов в службе быта.

Вариант 9 Система должна описывать работу одного из участков автосалона.

Вариант 10 Система должна описывать работу приемного покоя в больнице.

Вариант 11 Система должна описывать порядок приема заявки на поставку продукции на хлебокомбинате.

Вариант 12 Система должна описывать процесс поставки сезонных товаров в оптовой фирме.

Вариант 13 Система должна описывать процесс работы торгового отдела.

Вариант 15 Система учета в видеопрокате.

Вариант 16 Система учета проката на лыжной базе

Контрольные вопросы к практическому занятию №4

1. Перечислите основные возможности BPwin.
2. Охарактеризуйте основные элементы рабочего интерфейса BPwin.
3. Какую методологию поддерживает BPwin?
4. Назовите основные этапы построения модели.
5. Какой процесс можно назвать функциональной декомпозицией?
6. Перечислите элементы контекстной диаграммы.
7. При помощи какого инструмента строятся дуги на диаграмме?

Практическое занятие № 5.
BPwin- средство моделирования процессов (IDEF3), Erwin – средство информационного моделирования (IDEF1X)

Цель работы: знакомство с инструментарием функционального моделирования BPWin 40.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы);
- протокол выполнения заданий;
- заключение.

Время работы: 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

CASE-средства для моделирования предметной области

Моделирование предметной области, как правило, выполняется с помощью CASE-средств. К таким средствам относятся BPwin (PLATINUM technology), Silverrun (Silverrun technology), Oracle Designer (Oracle), Rational Rose (Rational Software) и др. Функциональные возможности инструментальных средств структурного моделирования деловых процессов будут рассмотрены на примере CASE-средства BPwin.

BPwin поддерживает три методологии моделирования: функциональное моделирование (IDEF0); описание бизнес-процессов (IDEF3); диаграммы потоков данных (DFD). Чаще всего применяется для создания функциональной модели предметной области на начальных этапах проектирования информационной системы, а также для анализа существующей или проектируемой ИС.

Функциональная модель включает в себя:

- поименованные процессы, функции или задачи, которые должны выполняться в системе;
- взаимодействия этих процессов, функций, задач с внешним миром и между собой.

BPWin с использованием IDEF0 методология позволяет наглядно представить выбранную систему как совокупность взаимо-действующих функций и задач. Функции и задачи системы анали-зируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Описание интерфейса программы BPWin

После запуска программы BPWin на экране появится окно программы (Рис. 1).

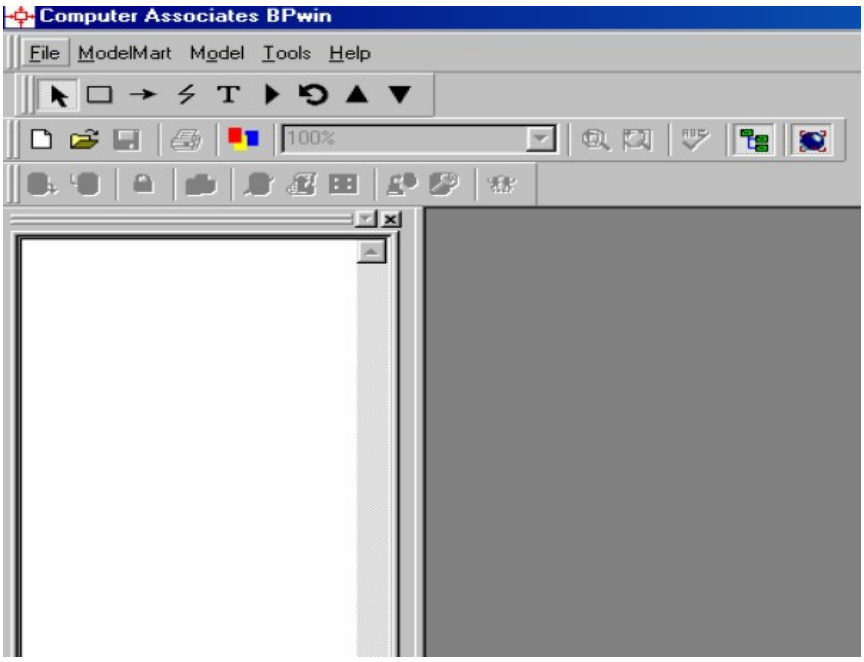


Рис. 1. Окно программы

Для создания новой модели необходимо вызвать диалог File/New или нажать на соответствующий значок на панели инструментов. После этого возникнет диалоговое окно, в котором следует указать название модели, выбрать методологию моделирова-ния Business Process (IDEF0) и нажать OK (Рис. 2).

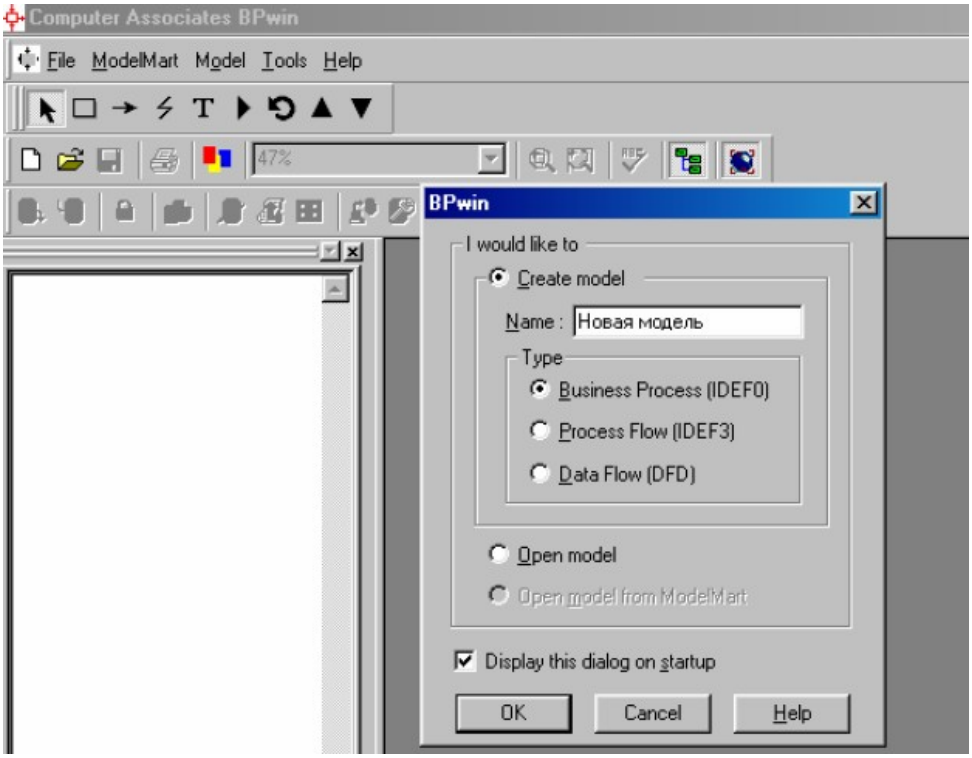


Рис.2. Окно создания новой модели

Далее появится окно, где следует указать свойства создаваемой модели (Рис.3). На первой вкладке следует указать Фамилию и имя автора модели, а также его инициалы. Остальные вкладки, определяющие такие свойства модели как: нумерация и положение функциональных блоков, высота и ширина страницы рекомендуется оставить без изменения.

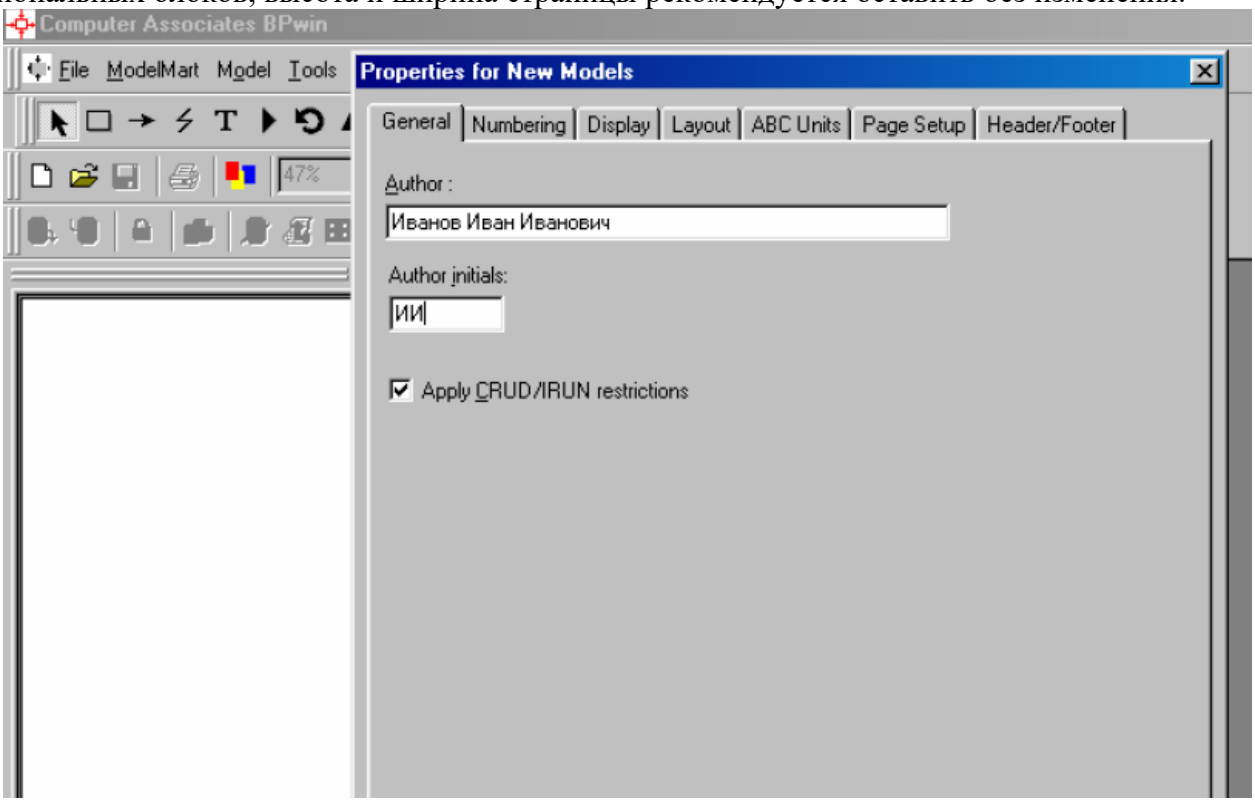


Рис.3. Окно свойств для новой модели

На появившейся странице верхнего уровня модели находится первый функциональный блок модели (Рис.4).

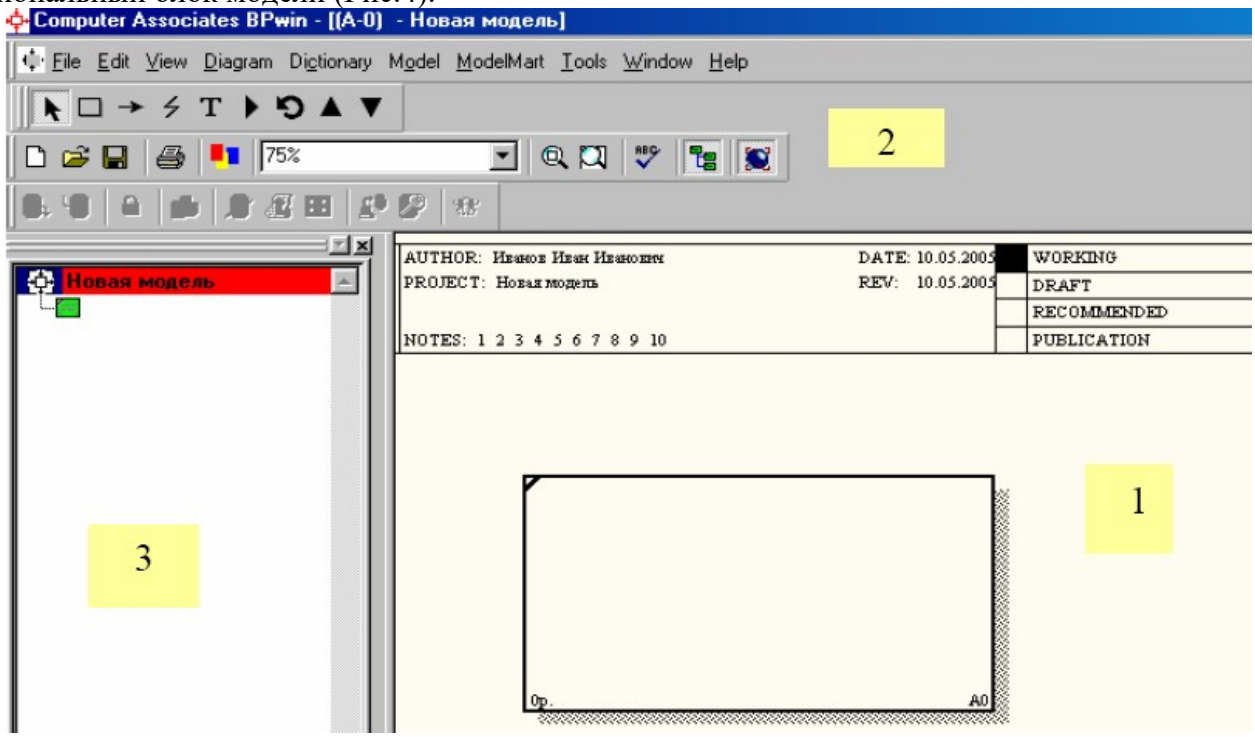


Рис.4. Основное окно BPWin

Основное окно программы содержит следующие части:
1. Рабочая область
2. Панели инструментов
3. Область модели

Рассмотрим подробнее содержимое каждой из частей программы.

Рабочая область – содержит собственно разрабатываемую модель. На каждой странице отображается соответствующий уровень декомпозиции функциональной модели.

Панели инструментов: эти панели содержат практически все используемые при работе элементы. По умолчанию все панели отображаются на экране. При необходимости пользователь может отключить или, наоборот, включить требуемые модели, используя меню «View». Имеются следующие панели инструментов:

- Standard toolbar – содержит кнопки для управления файлами (новый, открыть, сохранить, печать), кнопки отображения области свойств модели, кнопки управления масштабом изображения.

Рис.5. Стандартная панель инструментов

- BPWin Toolbox for Business Process Diagrams (IDEF0) – инструментальные кнопки создания элементов модели: функциональ-ных блоков и связей (стрелок) (Рис.6). Содержит кнопки: стрелка – выбор объекта, создание функционального блока, создание стрелки для связи функциональных блоков с внешним миром и между собой, создание текста, редактор модели, переходы к родительской и дочерней моделям (диаграммам).

Рис.6. Панель BPWin Toolbox for Business Process Diagrams (IDEF0)

- ModelMart – панель кнопок специального инструментального средства, предназначенного для связывания пакета BPWin и пакета ERWin.

Область модели содержит название модели, все уровни де-композиции разрабатываемой функциональной модели, а также на-звания всех функций, выполняемых на каждом уровне декомпози-ции.

Основные приемы работы с пакетом BPWin

1. Работа с функциональными блоками

Для того, чтобы написать название процесса, функции или задачи, выполняемой проектируемой системой нужно два раза щелкнуть мышкой по первому функциональному блоку с номером A0, поя-вившемуся после создания новой модели. После чего появится диа-

логовое окно Activity Properties, где будет предложено написать на-звание функционального блока (Рис.7).

При выборе названия функционального блока следует использовать глагол, либо глагольное существительное, обозначающее действие, так как разработка функциональной модели, заключается в описании функций, которые должна выполнять проектируемая система, и связей между ними.

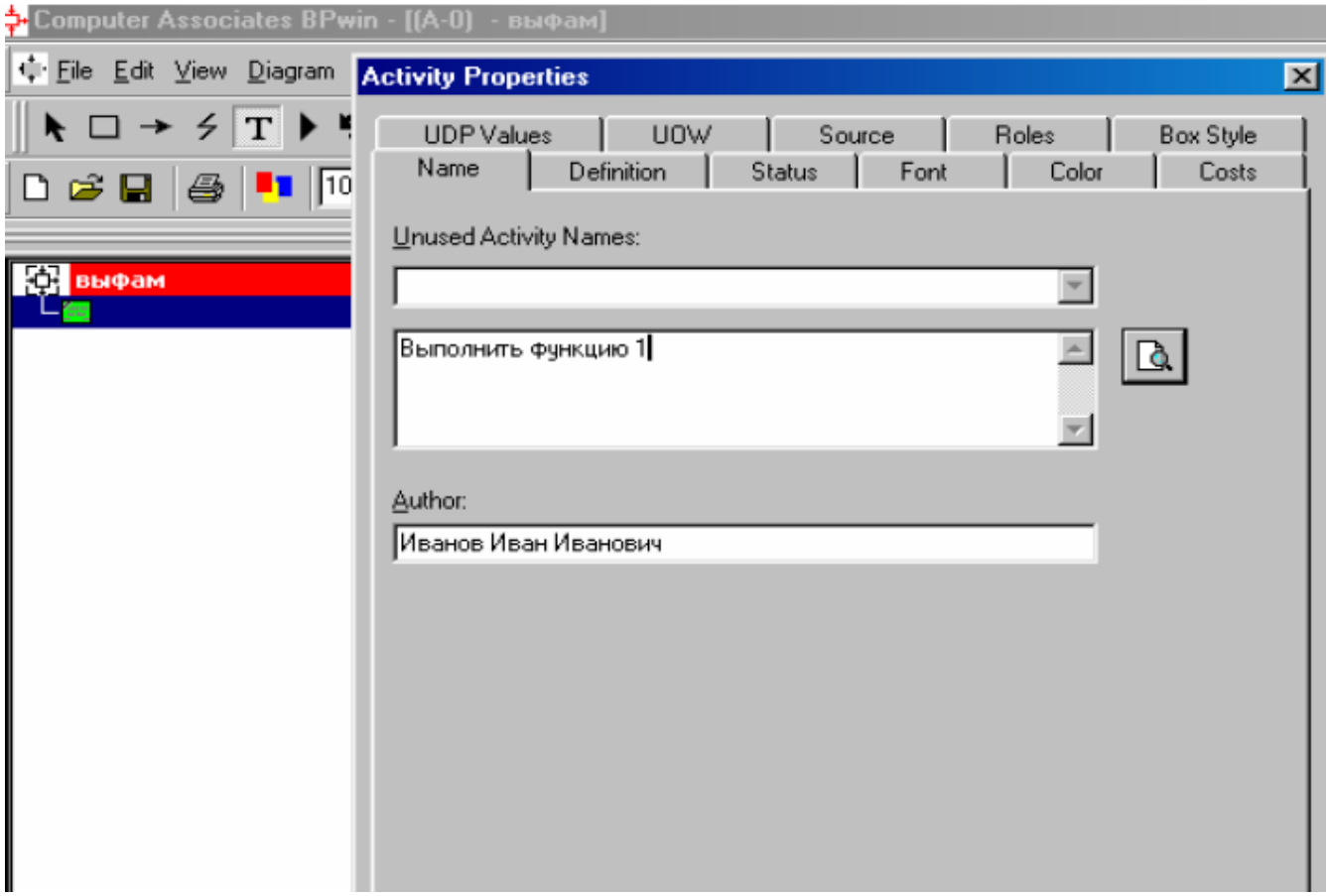


Рис.7.Окно свойств функционального блока 9

Также в этом диалоговом окне можно установить вид, стиль и размер шрифта надписи функционального блока, используя вкладку Font. Для функциональной декомпозиции этого блока следует перейти в область модели, встать на появившееся название первого функционального блока и вызвать меню по нажатию правой кнопки мышки (Рис.8).

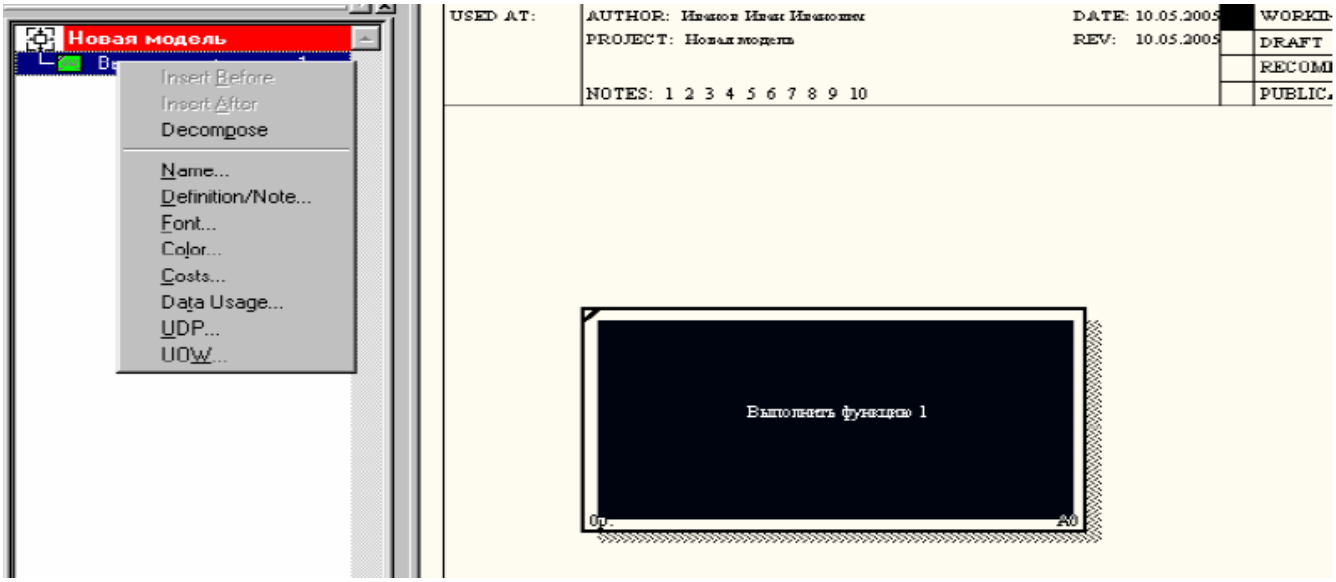
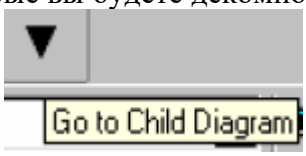


Рис.8. Декомпозиция функционального блока

Выбрать пункт меню «Decompose», после чего появится диалоговое окно с предложением выбрать количество блоков, на которые вы будете декомпозировать данный функциональный блок (Рис.9), либо нажать на значок 

Допустимый интервал числа функций 3-6. Число блоков на диаграмме в дальнейшем можно будет изменить (добавить недостающие или удалить лишние).

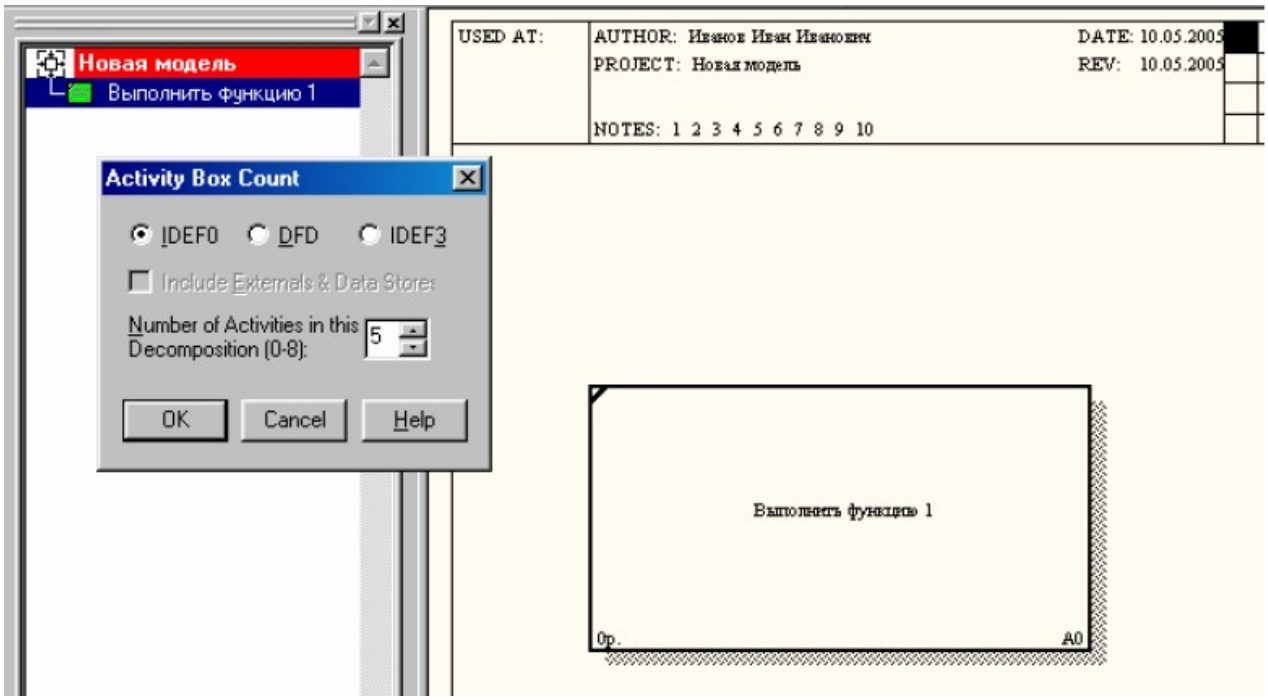


Рис.9. Выбор количества функциональных блоков 10

После ввода количества блоков на экране появится следующая страница, отражающая второй уровень декомпозиции проектируемой системы (Рис.10).

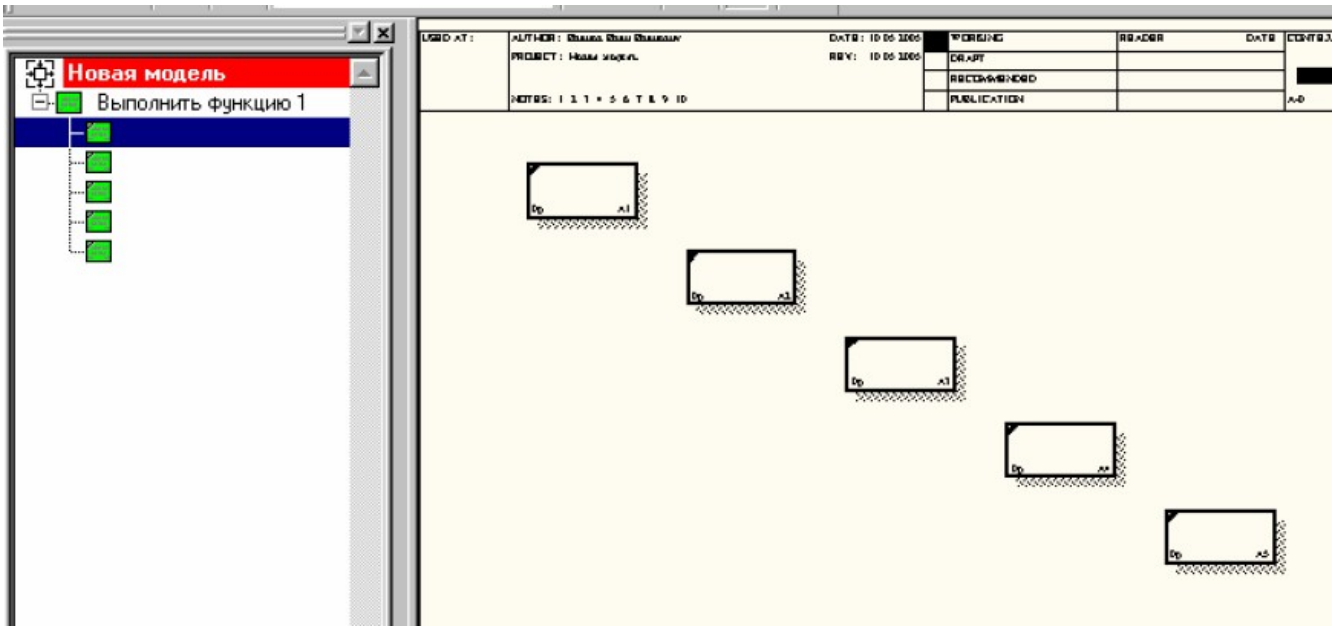


Рис.10. Первый уровень декомпозиции

Функциональные блоки второго уровня декомпозиции имеют номера A1, A2, A3, A4, A5. При декомпозиции каждого из них бло-ки на следующем уровне декомпозиции будут иметь номера соот-ветственно A11, A12, A21, A22, A31, A32 и т.д. в зависимости от номера декомпозируемого функционального блока.

2. Работа со стрелками

Стрелки представляют собой некую информацию и именуются существительными.

В основе методологии IDEF0 лежат следующие правила:

- функциональный блок (функция) преобразует Входы в Выходы (т.е. входную информацию в выходную), Управление определяет, когда и как это преобразование может или должно произойти Исполнители (механизм) непосредственно осуществляют это преобразование (Рис.11);
- с дугами связаны надписи (или метки) на естественном языке, описывающие данные, которые они представляют;
- дуги показывают, как функции между собой взаимосвязаны, как они обмениваются данными и осуществляют управление друг другом;
- выходы одной функции могут быть Входами, Управлением или Исполнителями для другой;
- дуги могут разветвляться и соединяться.

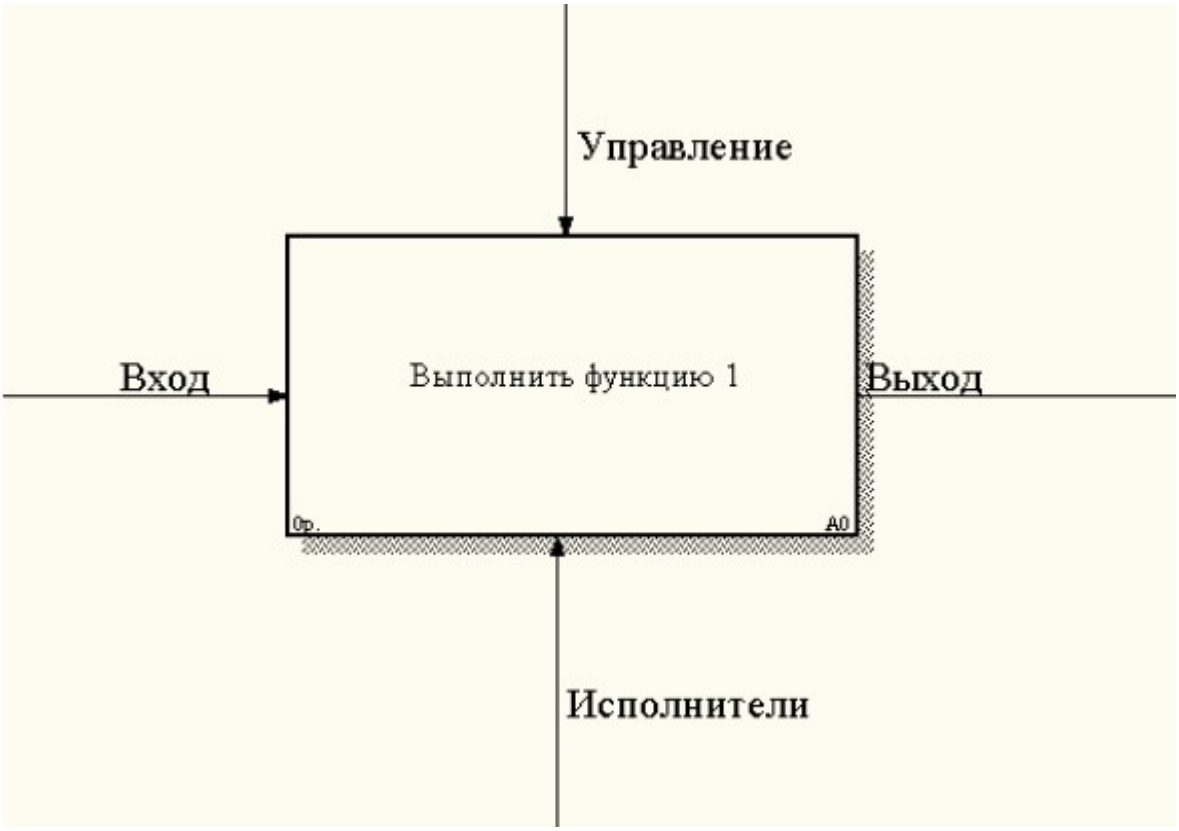


Рис.11. Виды связей в функциональной модели

Для создания стрелки следует щелкнуть по кнопке с символом стрелки  на панели инструментов.

Чтобы создать стрелку входа следует подвести курсор к левой стороне рабочей области, пока не появится черная полоса, затем щелкнуть по этой полосе и подвести курсор к функциональному блоку с правой стороны пока не появится черная стрелка затем щелкнуть по ней (Рис.12).

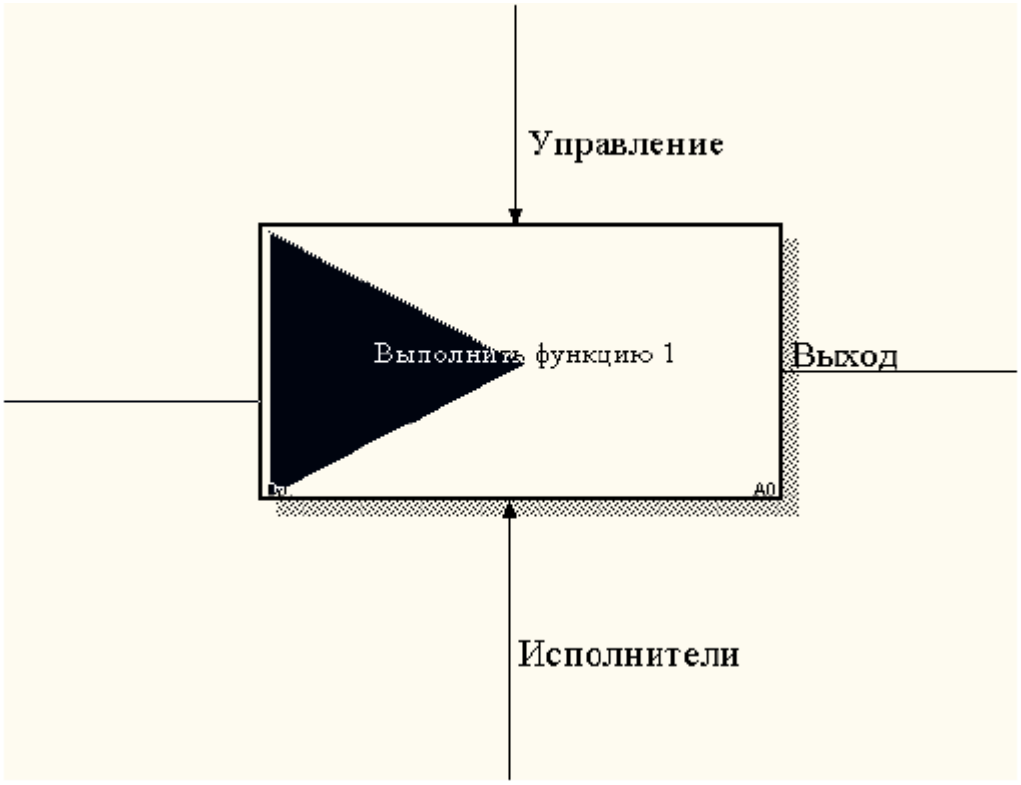


Рис.12. Пример создания стрелок

Аналогично рисуются стрелки выхода, управления и исполнения функции. Для того, чтобы назвать стрелку необходимо дважды по ней щелкнуть, в результате появится диалоговое окно Arrow Properties, где в поле Arrow Name следует ввести название стрелки (Рис.13).

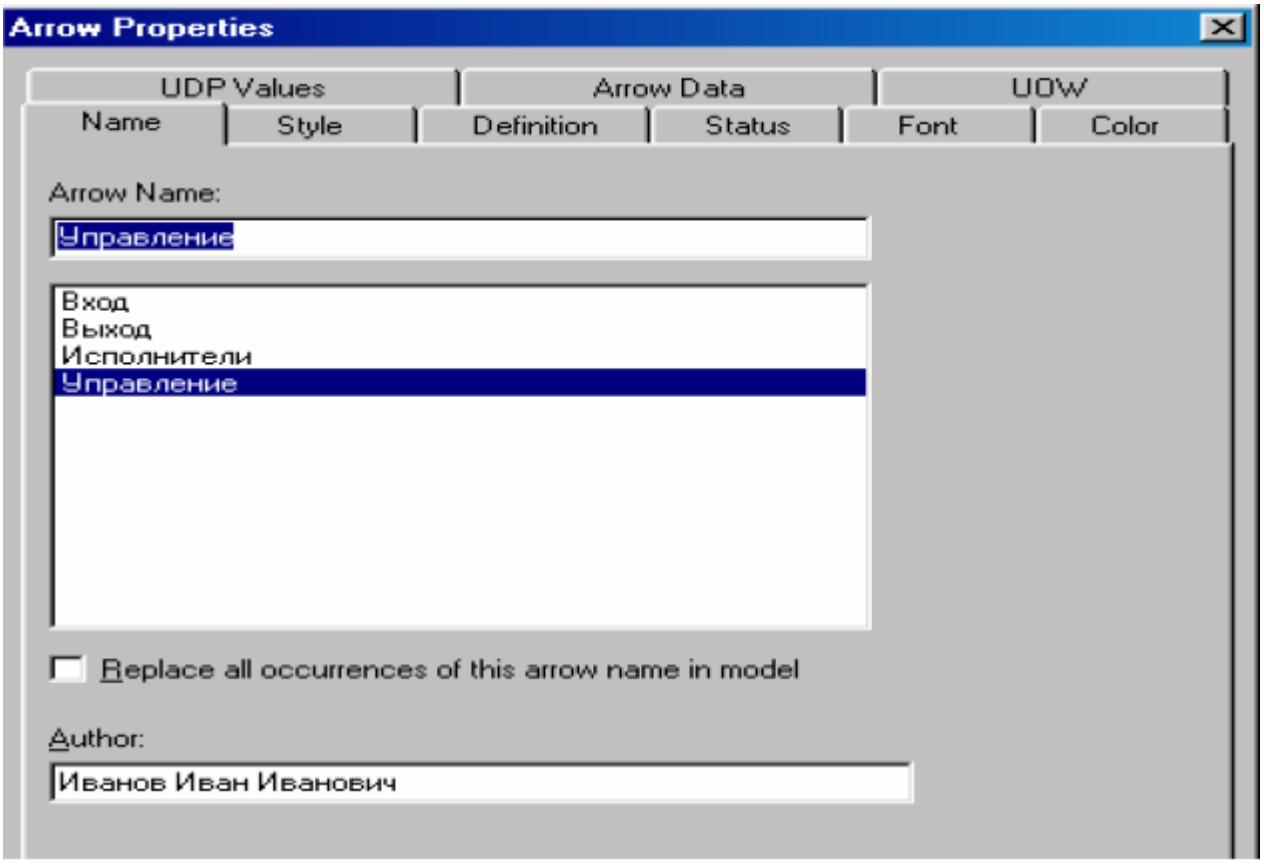


Рис.13. Окно создания названий стрелок

Также в этом диалоговом окне можно установить вид, стиль и размер шрифта надписей стрелок, используя вкладку Font. При декомпозиции функционального блока входящие и исходящие из него стрелки автоматически появляются на следующем уровне декомпозиции, но не касаются функциональных блоков нового уровня модели (Рис.14). Такие стрелки называются несвязными и воспринимаются как синтаксическая ошибка.

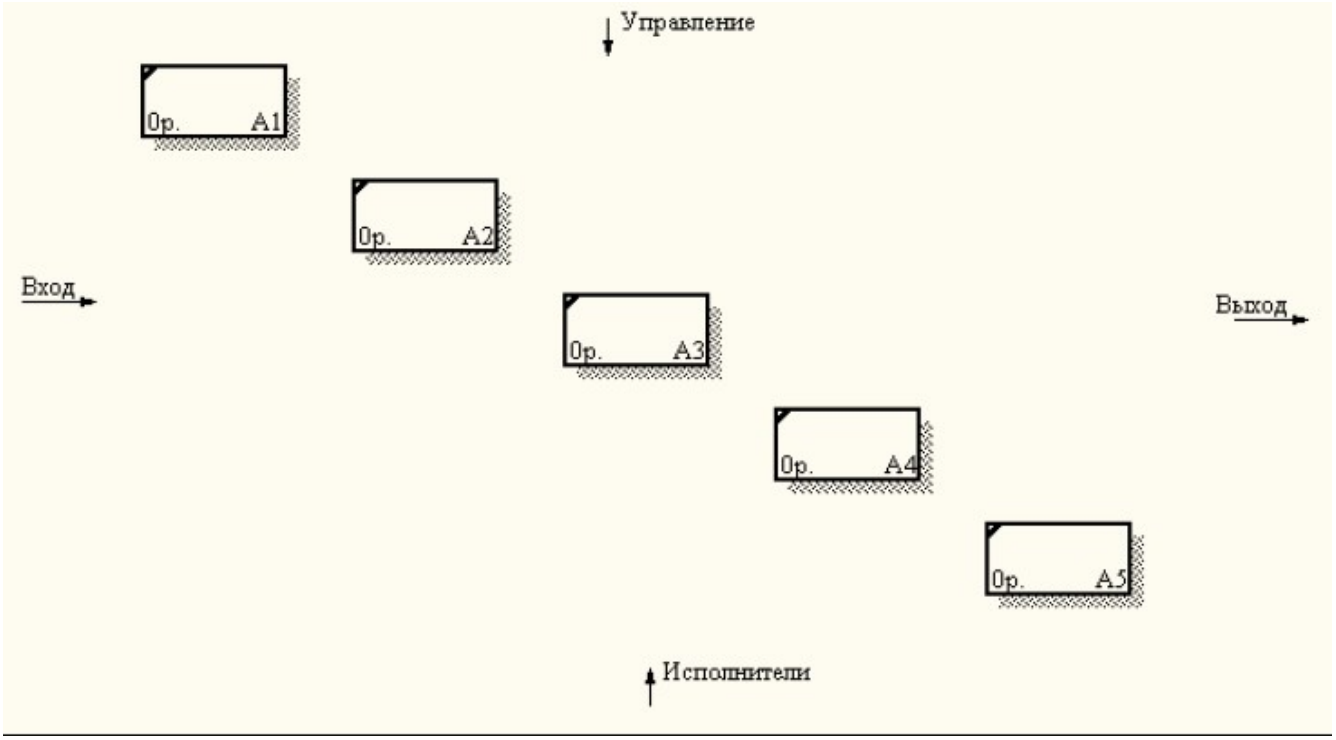


Рис.14. Пример несвязных стрелок

Для связывания стрелок с функциональными блоками следует сначала щелкнуть по наконечнику стрелки, а затем по соответст-вующему сегменту функционального блока (сверху – если связывается стрелка, обозначающая «управление», снизу – если связывается стрелка исполнения (механизма) функции и т.д.). Стрелки, появляющиеся на каком-то определенном уровне де-композиции и служащие для связи между функциональными блоками называются внутренними. Для того, чтобы нарисовать внутреннюю стрелку следует щелкнуть по выходу одного, а затем по входу другого функционального блока (Рис.15).

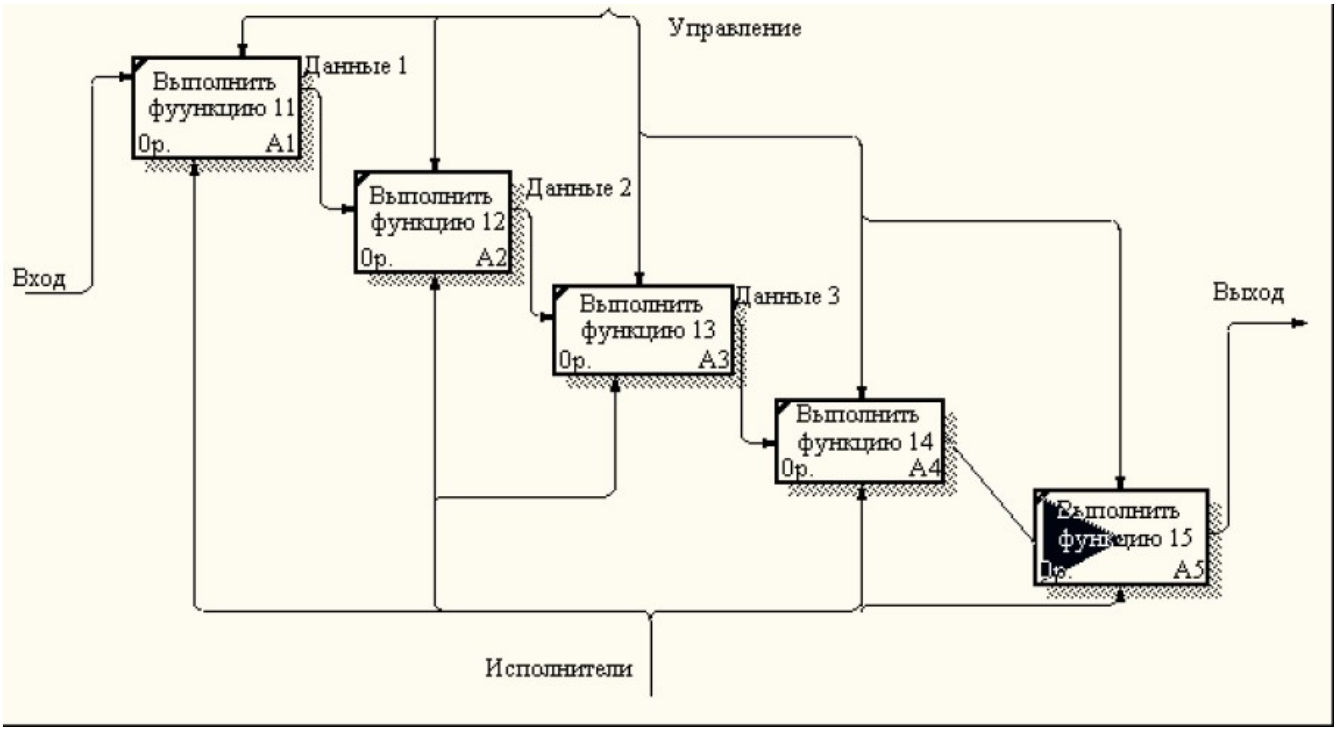


Рис.15. Пример внутренних стрелок

Одна стрелка может соединяться (разветвляться) с различными функциональными блоками. Для разветвления стрелки нужно перейти в режим редактирования стрелки, затем щелкнуть по той стрелке, которую вы хотите разветвить, а затем по соответствующему сегменту функционального блока.

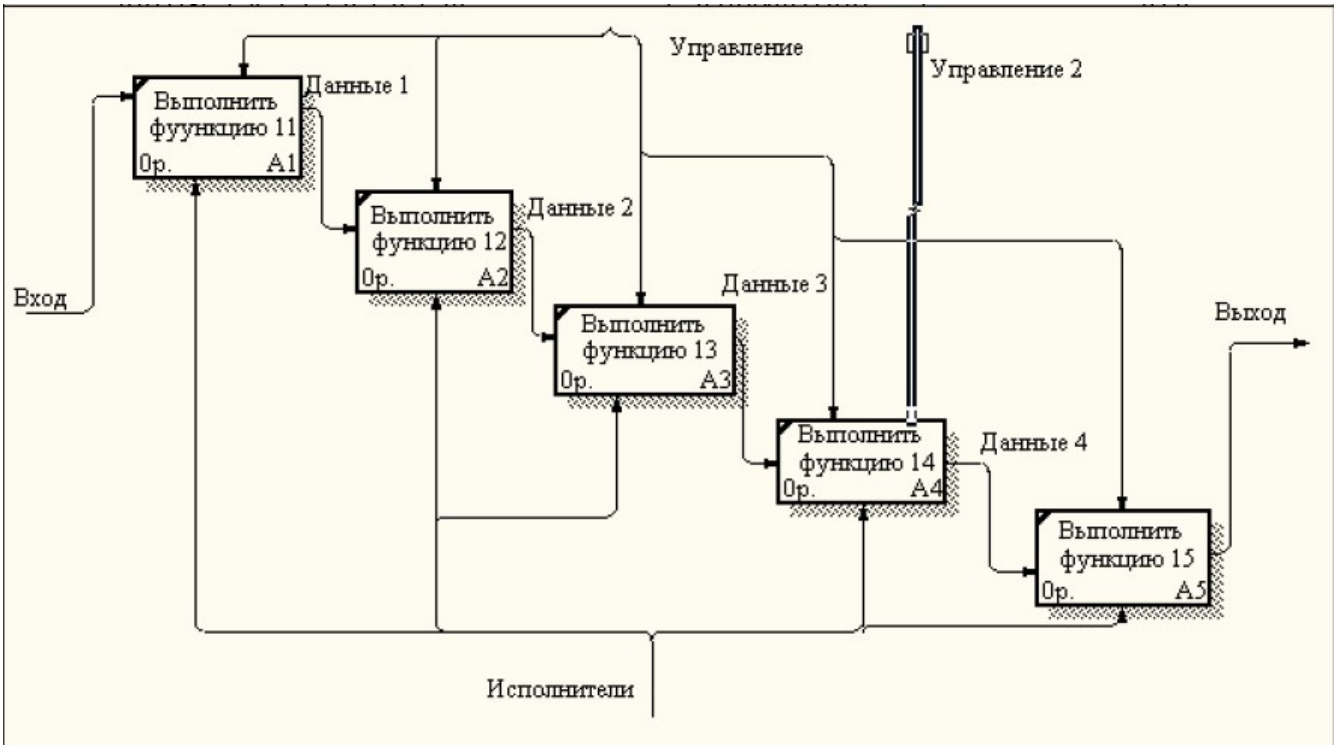


Рис.16. Пример неразрешенных стрелок

Стрелки, нарисованные на диаграмме декомпозиции нижнего уровня изображаются в квадратных скобках (Рис.16) и не появляются на диаграмме верхнего уровня. Такие стрелки называются неразрешенными и воспринимаются программой как синтаксическая ошибка. Следует либо перетащить эту стрелку на верхний уровень диаграммы, либо затоннелировать стрелку и тогда она не будет восприниматься как ошибка и не попадет на другую диаграмму. Для этого следует щелкнуть по квадратным скобкам неразрешен-ной стрелки правой кнопкой мышки и вызвать диалоговое окно, показанное на рисунке 17 и выбрать пункт Arrow Tunnel.

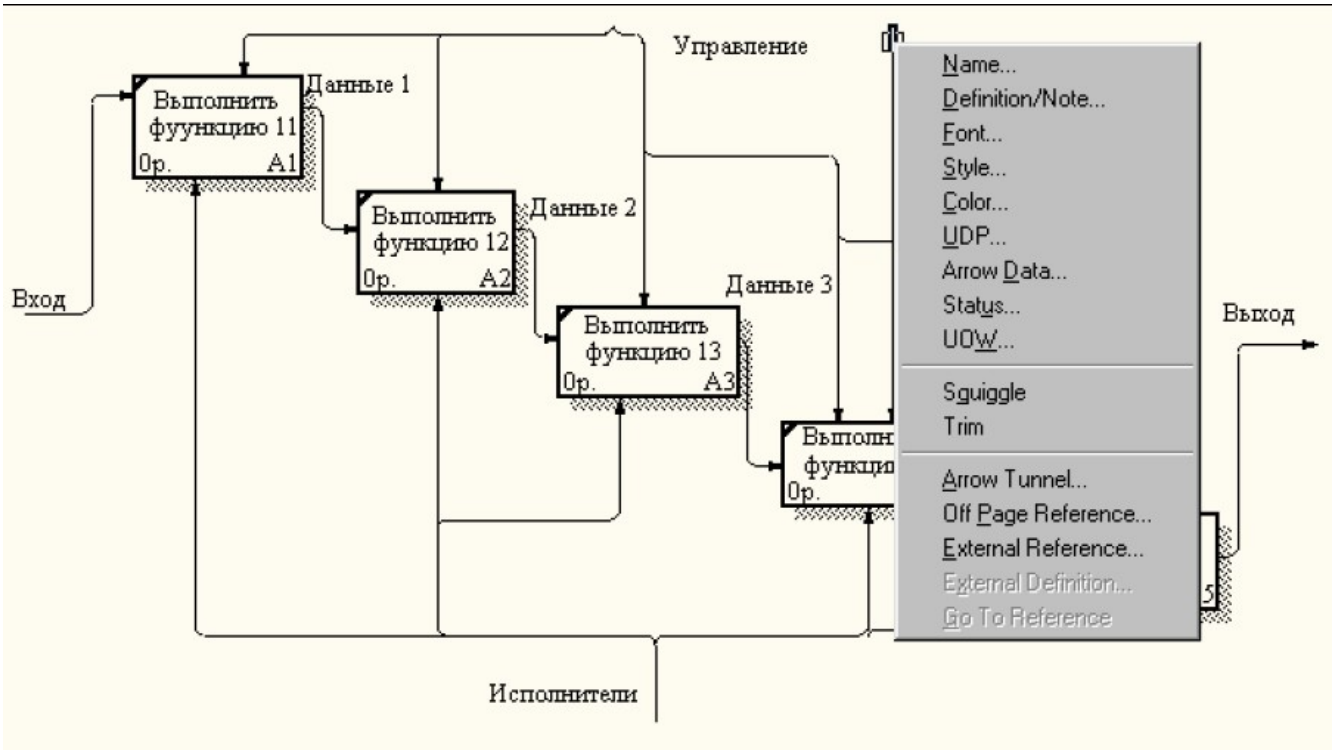


Рис.17. Окно редактирования стрелок

Если Вы хотите, чтобы новая стрелка попала «наверх», следу-ет выбрать пункт меню Resolve it to border arrow. Если же Вы хоти-те оставить стрелку только на текущем уровне диаграммы, тогда следует выбрать пункт меню Change it to resolved rounded tunnel. Тоннельная стрелка изображается с круглыми скобками.

3. Проверка синтаксиса модели

Для проверки синтаксиса модели следует вызвать диалог Tools/Reports/Model Consistency Report. После чего появится диалоговое окно (Рис.18).

Затем следует выбрать пункт Preview для предварительного просмотра списка синтаксических ошибок модели. Список синтак-сических ошибок может включать: неименованные функциональ-ные блоки и стрелки (unnamed arrows, unnamed activities), несвязан-ные стрелки (unconnected border arrow), неразрешенные стрелки (unresolved (square tunneled) arrow connection), блоки, не имеющие по крайней мере одной стрелки выхода и одной стрелки управления (activity «Наименование функционального блока» has no Control) и т.д.

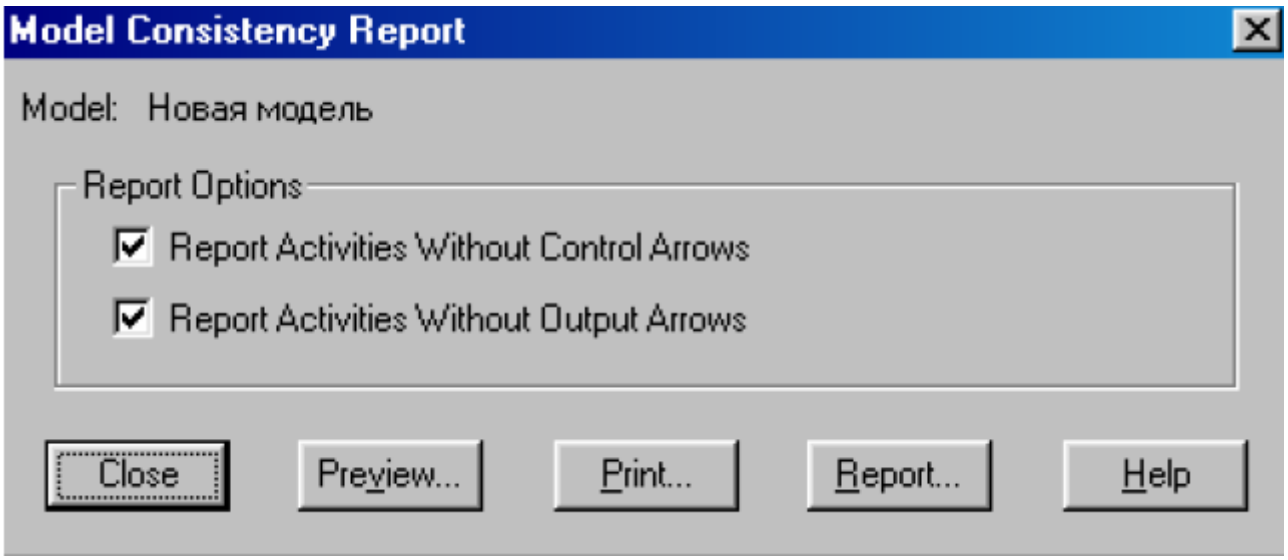


Рис.18. Отчет проверки синтаксиса модели

Нажав на кнопку Print Вы можете напечатать полученный отчет, кнопка Report позволяет сохранить сформированный отчет либо в формате txt , либо как файл Consistency Report, имеющий расширение bpc.

4. Формирование отчета Node Tree

Для наглядного представления количества уровней декомпо-зиции и отношений между родительскими и дочерними диаграм-мами следует сформировать отчет Node Tree. Для этого нужно вызвать диалог Diagram/Add Node Tree.После чего появится диалого-вое окно, где будет предложено название отчета (можно написать другое) – Node Tree Name, верхний уровень диаграммы, с которого следует начать строить отчет – Top level activity, и выбрать количество уровней который будет иметь отчет – Number of levels (Рис.19).

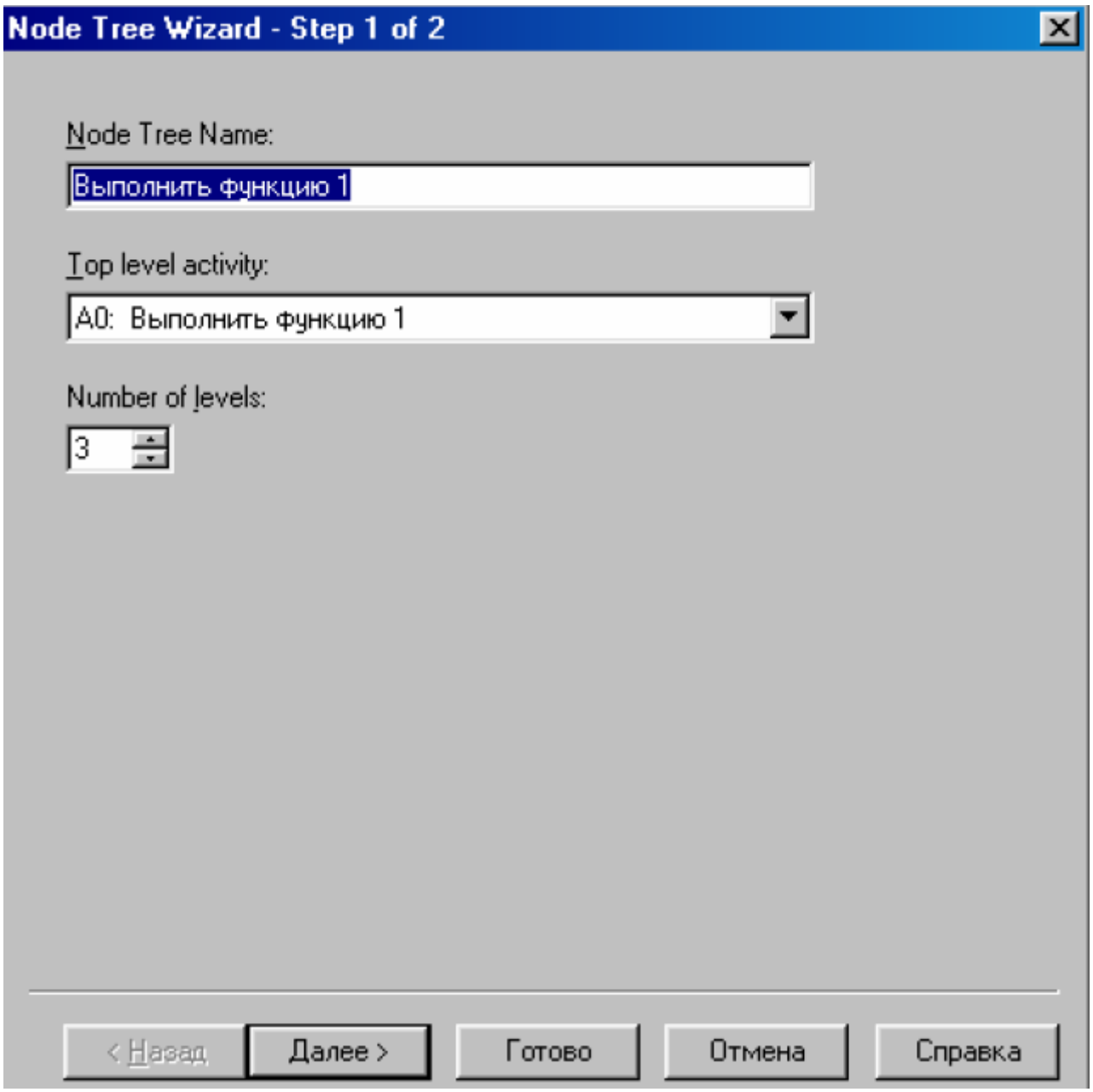


Рис.19. Окно построения отчета Node Tree

При нажатии кнопки далее можно изменить, либо оставить прежними параметры отчета. Затем следует нажать кнопку готово и появится сформированный отчет. Отчет имеет древовидную структуру (Рис.20).

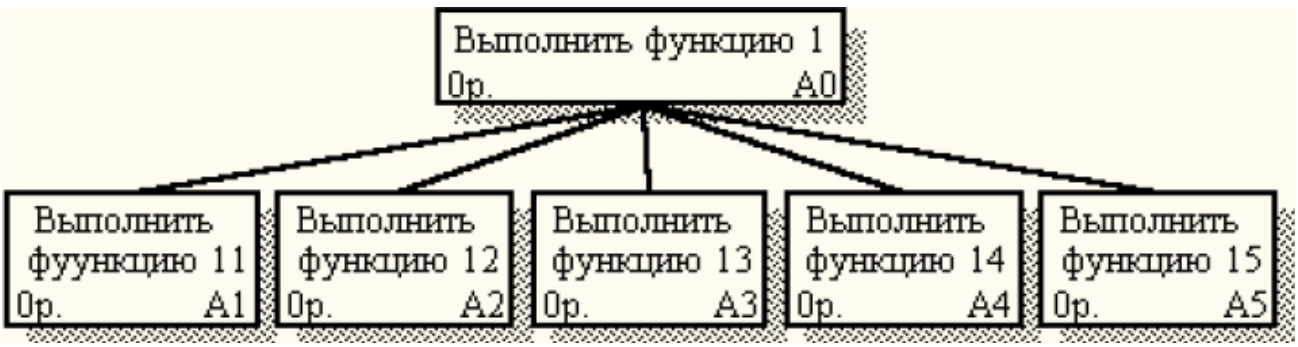


Рис.20. Пример дерева узлов Node Tree

ЗАДАНИЕ НА РАБОТУ

1. Выполнить приведенные выше примеры в пакете BPWin.

2. Ответить на контрольные вопросы.

Контрольные вопросы к практическому занятию №5

1. Каково назначение пакета BPWin?
2. Что включает в себя функциональная модель?
3. Что такое рабочая область BPWin?
4. Что содержат панели инструментов BPWin?
5. Назовите основные виды синтаксических ошибок в BPWin.
6. Что такое дерево узлов Node Tree?

Практическое занятие № 6.
Объектное моделирование ИС средством Ration Rose, количественный анализ диаграмм UML

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы);
- протокол выполнения заданий;
- заключение.

Время работы: 4 часа.

1. Общая характеристика CASE-системы Rational Rose

Rational Rose — это CASE-система для визуального моделирования объектно-ориентированных программных продуктов. Визуальное моделирование – это процесс графического описания разрабатываемого программного обеспечения.

В Rational Rose реализованы общепринятые стандарты на рабочий интерфейс программы, подобно известным средам визуального программирования. После установки системы на компьютер пользователя, ее запуск в среде MS Windows приводит к появлению на экране рабочего интерфейса (рис.1.1).

Рабочий интерфейс Rational Rose состоит из различных элементов, основными из которых являются: главное меню программы, стандартная панель инструментов, окно браузера, специальная панель инструментов, окно диаграммы, окно документации и окно журнала.

Рассмотрим кратко назначение и основные функции каждого из этих элементов.

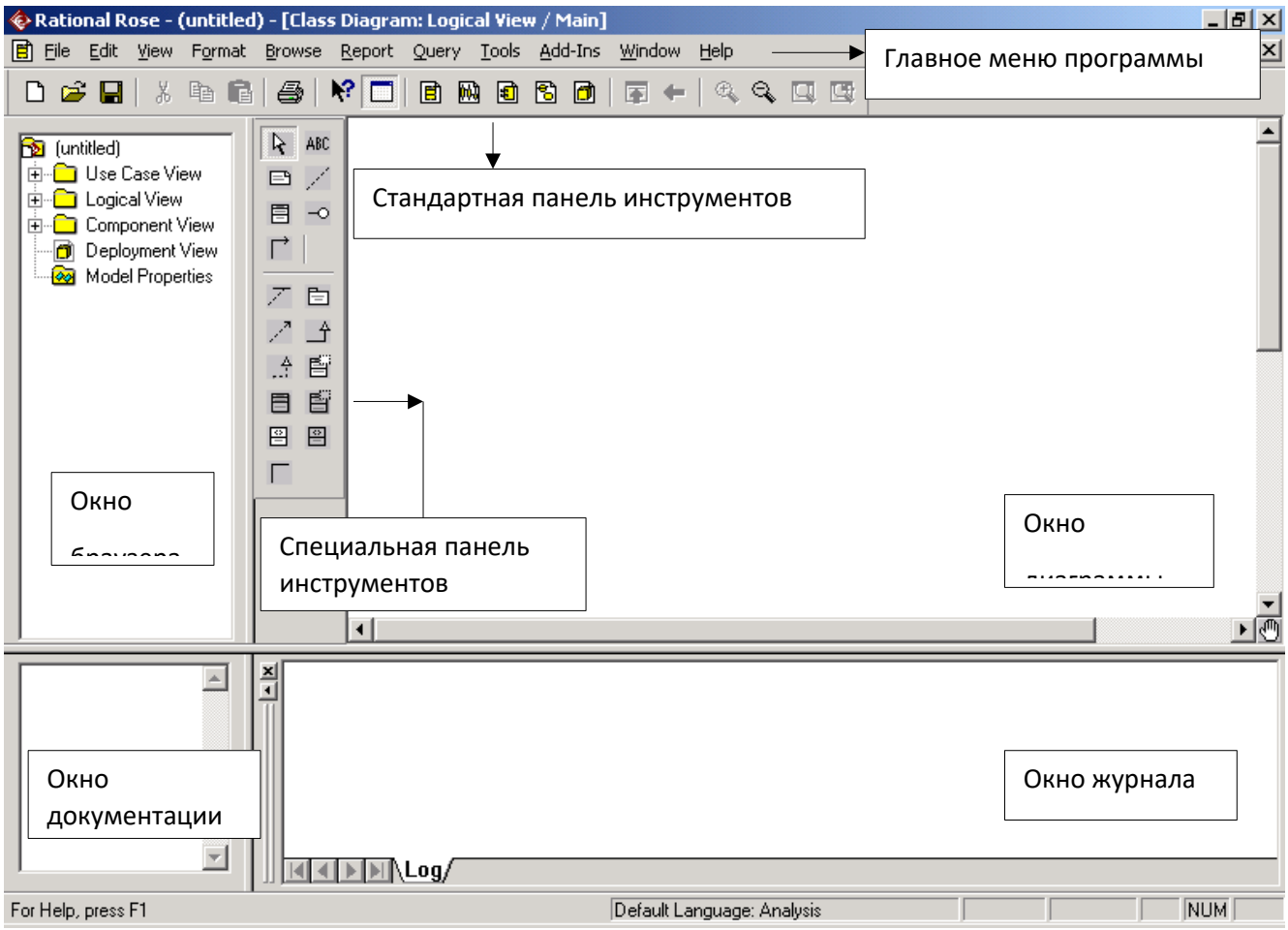


Рис.1.1. Общий вид рабочего интерфейса системы Rational Rose

Главное меню программы выполнено в общепринятом стандарте и находится под строкой заголовка среды Rational Rose. Некоторые из пунктов меню содержат хорошо знакомые функции (открытие проекта, вывод на печать диаграмм, копирование в буфер и вставка из буфера различных элементов диаграмм). Другие пункты меню специфичны и требуют дополнительного изучения (опции генерации программного кода, проверка согласованности моделей, подключение дополнительных модулей).

Стандартная панель инструментов располагается ниже главного меню программы. Она обеспечивает быстрый доступ к тем командам меню, которые выполняются разработчиками наиболее часто. Пользователь может настроить внешний вид этой панели по своему усмотрению.

Окно браузера по умолчанию располагается в левой части рабочего интерфейса под стандартной панелью инструментов. Браузер организует представления модели в виде иерархической структуры, которая упрощает навигацию и позволяет отыскать любой элемент модели в проекте. При этом любой элемент, добавляемый в модель, сразу отображается в окне браузера. Соответственно, выбрав элемент в окне браузера, его можно визуализировать в окне диаграммы или изменить его спецификацию.

Специальная панель инструментов располагается между окном браузера и окном диаграммы в средней части рабочего интерфейса. По умолчанию предлагается панель инструментов для построения диаграммы классов модели. Расположение специальной панели инструментов и состав ее кнопок можно изменять. Назначение кнопок отражено в тексте всплывающих подсказок, появляющихся после задержки указателя мыши над соответствующей кнопкой.

Окно диаграммы является основной рабочей областью ее интерфейса, в которой визуализируются различные представления модели проекта. По умолчанию окно диаграммы располагается в правой части рабочего интерфейса, однако его расположение

и размеры также можно изменить. Одновременно в окне диаграммы могут присутствовать несколько диаграмм, однако активной может быть только одна из них. При активизации отдельного вида диаграммы изменяется внешний вид специальной панели инструментов, которая настраивается под конкретный вид диаграммы.

Окно документации по умолчанию может не присутствовать на экране. Иначе оно появится ниже браузера. Окно документации предназначено для документирования элементов представления модели. В него можно записывать различную информацию, которая в последующем преобразуется в комментарии и не влияет на логику выполнения программного кода. В окне документации активизируется та информация, которая относится к отдельному выделенному элементу диаграммы.

Окно журнала (Log) предназначено для автоматической записи различной служебной информации, образующейся в ходе работы с программой. В журнале фиксируется время и характер выполняемых разработчиком действий, таких как обновление модели, настройка меню и панелей инструментов, а также сообщения об ошибках, возникающих при генерации программного кода.

2. Диаграмма вариантов использования

Диаграмма вариантов использования (use case diagram) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Диаграмма вариантов использования содержит варианты использования системы, действующих лиц и связи между ними. *Вариант использования* (use case) — это описание функциональности системы. *Действующее лицо* (actor) — это всё, что взаимодействует с системой.

Часто для одной системы создается несколько диаграмм вариантов использования. На диаграмме высокого уровня, называемой в среде Rational Rose *главной* (main), указываются только *пакеты* (группы) вариантов использования. Другие диаграммы описывают совокупности вариантов использования и действующих лиц. Количество и состав создаваемых диаграмм вариантов использования полностью зависит от разработчика.

В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи *коммуникации* (communication), *использования* (uses), *расширения* (extends) и *обобщения действующего лица* (actor generalization).

Связи коммуникации (кнопка Unidirectional Association – однонаправленная ассоциация) существуют между вариантом использования и действующим лицом. Направление стрелки показывает, кто инициирует коммуникацию. Информация при этом может двигаться в обоих направлениях. Вариант использования также может инициировать коммуникацию с действующим лицом. Каждый вариант использования должен быть инициирован действующим лицом; исключения составляют лишь варианты использования в связях использования и расширения.

Связи использования (кнопка Generalization – обобщение) позволяют одному варианту использования задействовать функциональность другого. С помощью таких связей обычно моделируют многократно применяемую функциональность, встречающуюся в двух и более вариантах использования. Связь использования предполагает, что один вариант использования всегда применяет функциональные возможности другого.

Связи расширения (кнопка Generalization – обобщение) позволяют варианту использования только при необходимости применять функциональные возможности, предоставляемые другим вариантом использования.

Абстрактный вариант использования не запускается непосредственно действующим лицом. Он обеспечивает некоторую дополнительную функциональность, которая может применяться другими вариантами использования. Таким образом, абстрактные варианты использования участвуют в связях использования или расширения.

Для документирования процесса обработки данных, реализуемого в рамках варианта использования, используют *поток событий* (flow of events). Этот документ подробно описывает, что будут делать пользователи системы и что — сама система. Поток событий также не должен зависеть от программной реализации.

На языке UML *стереотипы* (stereotypes) используются для выделения категорий элементов модели. Для действующего лица не поставляется никаких других стереотипов, кроме стереотипа Actor (Действующее лицо). Однако вы всегда можете определить свои собственные стереотипы и использовать их в ваших моделях.

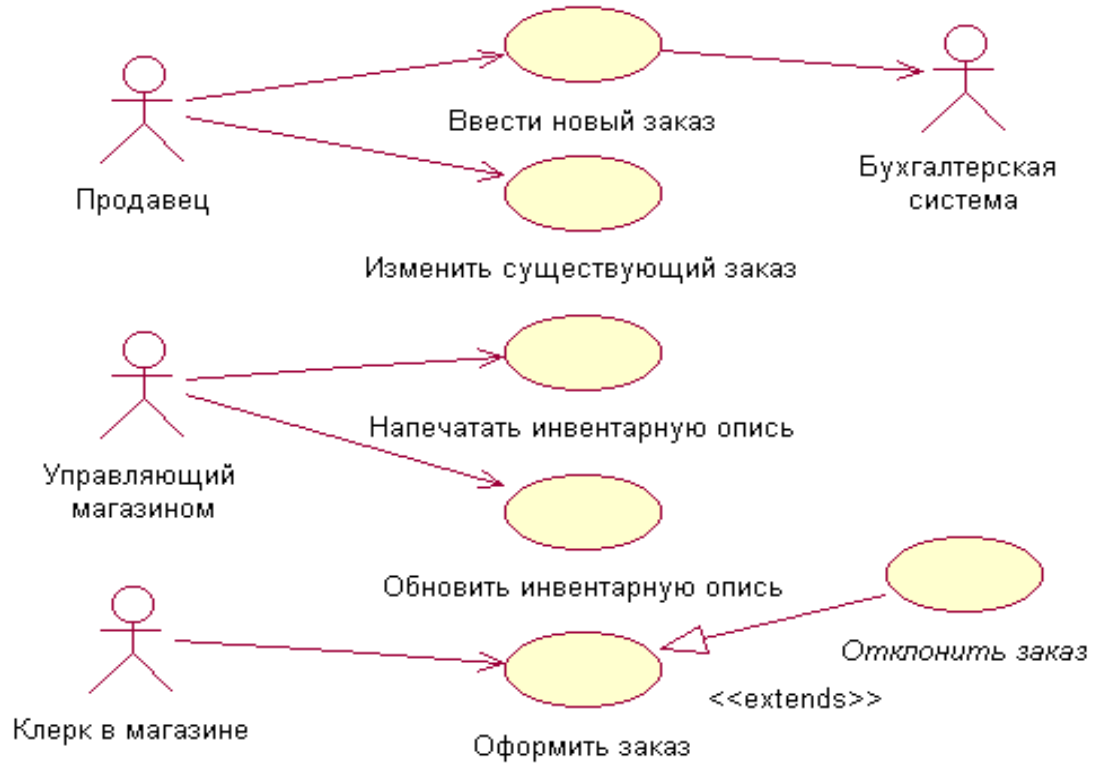
В среде Rose можно указать, сколько экземпляров конкретного действующего лица будет использоваться. Чтобы зафиксировать этот факт, можно использовать поле *Cardinality* (Множественность) окна спецификации. *Абстрактным* называется действующее лицо, не имеющее экземпляров. Иными словами, его множественность равна нулю.

С помощью *связи обобщения действующего лица* (actor generalization relationship) показывают, что у нескольких действующих лиц имеются общие черты. На языке UML связь обобщения действующего лица изображают в виде стрелки *Generalization* (Обобщение) от конкретного действующего лица к абстрактному действующему лицу. Связи этого типа создаются не всегда. В общем случае они нужны, если поведение действующего лица одного типа отличается от поведения другого настолько, что это *затрагивает систему.*

Упражнение 1

Создайте диаграмму вариантов использования для системы обработки заказов (см. рис.2.1).

Рис.2.1. Диаграмма вариантов использования для системы обработки заказов



Этапы выполнения упражнения

Создание диаграммы вариантов использования, вариантов использования и действующих лиц

1. Дважды щелкнув мышью на главной диаграмме вариантов использования (*Main*) в браузере, откройте её.

2. С помощью кнопки *Use Case* (Вариант использования) панели инструментов поместите на диаграмму новый вариант использования и назовите его *Ввести новый заказ*.

3. Повторив п.2, поместите на диаграмму остальные варианты использования: *Изменить существующий заказ*, *Напечатать инвентарную опись*, *Обновить инвентарную опись*, *Оформить заказ* и *Отклонить заказ*.

4. С помощью кнопки *Actor* (Действующее лицо) панели инструментов поместите на диаграмму новое действующее лицо и назовите его *Продавец*.

5. Повторив шаг 4, поместите на диаграмму остальных действующих лиц: *Управляющий магазином*, *Клерк в магазине* и *Бухгалтерская система*.

Создание абстрактного варианта использования

Щелкните правой кнопкой мыши на варианте использования *Отклонить заказ* на диаграмме. В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию). Установите флажок *Abstract* (Абстрактный), чтобы сделать этот вариант использования абстрактным.

Добавление ассоциаций

1. С помощью кнопки *Unidirectional Association* (Однонаправленная ассоциация) панели инструментов нарисуйте ассоциацию между действующим лицом *Продавец* и вариантом использования *Ввести новый заказ*.

2. Повторив п.1, поместите на диаграмму остальные ассоциации.

Добавление связи расширения

1. С помощью кнопки *Generalization* (Обобщение) панели инструментов нарисуйте связь между вариантом использования *Отклонить заказ* и вариантом использования *Оформить заказ*.

2. Щелкните правой кнопкой мыши на новой связи между вариантами использования *Отклонить заказ* и *Оформить заказ*. В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию).

3. В раскрывающемся списке стереотипов введите слово *extends* (расширение), затем нажмите *OK*. Надпись *«extends»* появится на линии данной связи.

Добавление описаний к вариантам использования

1. Выделите в браузере вариант использования *Ввести новый заказ*. В окне документации введите следующее описание: *«Этот вариант использования дает клиенту возможность ввести новый заказ в систему».*

2. С помощью окна документации добавьте описания ко всем остальным вариантам использования.

Добавление описаний к действующему лицу

1. Выделите в браузере действующее лицо *Продавец*. В окне документации введите следующее описание: *«Продавец — это служащий, старающийся продать товар».*

2. С помощью окна документации добавьте описания к остальным действующим лицам.

Прикрепление файла к варианту использования

1. Создайте файл *OrderFlow.doc* для основного потока событий, описывающего вариант использования *Ввести новый заказ*. Введите в него следующую информацию:

Основной поток событий для варианта использования «Ввести новый заказ»

1. *Продавец выбирает в имеющемся меню пункт «Создать новый заказ».*
2. *Система выводит форму «Детали заказа»*
3. *Продавец вводит номер заказа, заказчика и то, что заказано.*
4. *Продавец сохраняет заказ.*
5. *Система создаёт новый заказ и сохраняет его в базе данных.*

2. Щелкните правой кнопкой мыши на варианте использования *Ввести новый заказ*. В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию). Перейдите на вкладку *Files* (Файлы). Щелкните правой кнопкой мыши в белой области спецификации и в открывшемся меню выберите пункт *Insert File* (Вставить файл).

Укажите файл *OpenFlow.doc* и нажмите на кнопку *Open* (Открыть), чтобы прикрепить файл к варианту использования.

3. Диаграммы взаимодействия

Существуют два типа диаграмм взаимодействия (interaction diagrams): *диаграммы последовательности* (sequence diagrams) и *кооперативные диаграммы* (collaboration diagrams). Диаграммы взаимодействия отображают события, участвующие в процессе обработки информации варианта использования, и сообщения, которыми обмениваются объекты. События на диаграмме последовательности упорядочены во времени, а кооперативная диаграмма организована вокруг самих объектов.

Диаграммы взаимодействия визуализируют детали, описанные в потоке событий, представляя их в форме более удобной для разработчика. Главное на этих диаграммах – объекты, которые должны быть созданы для реализации функциональных возможностей, заложенных в вариант использования. На диаграммах последовательности и кооперативных диаграммах могут быть показаны объекты, классы или то и другое вместе.

Объектом называется сущность, инкапсулирующая в себе некоторые данные и поведение. Данные объекта называются *атрибутами* (attributes). Поведение объекта представляется его *операциями* (operations). Принадлежность объекта некоторому классу определяет данные и поведение, которыми должен обладать этот объект.

У каждого объекта на диаграмме последовательности имеется *линия жизни* (lifeline), изображаемая в виде вертикальной штриховой линии под объектом. Сообщения, соответствующие коммуникациям между объектами, находятся между линиями жизни объектов. Сообщения показывают, что один объект вызывает функцию другого. После определения операций классов каждое сообщение станет операцией.

На любой диаграмме последовательности или кооперативной диаграмме должен быть объект-действующее лицо. Он является внешним стимулом, дающим системе команду на выполнение определенной функции. Объекты-действующие лица на диаграмме взаимодействия соответствуют действующим лицам, которые взаимодействуют с вариантом использования на диаграмме вариантов использования.

На диаграммах последовательности и кооперативных диаграммах изображают объекты, участвующие в потоке одного конкретного варианта использования. После того как на диаграмму поместили объект-действующее лицо, нужно добавить туда и все остальные объекты. Участвующие в конкретной диаграмме последовательности или кооперативной диаграмме объекты можно выявить, изучив документацию по этому сценарию или имена существительные, встречающиеся в потоке событий. Каждый объект диаграммы последовательности или кооперативной диаграммы может быть соотнесен с классом. После добавления объектов можно приступать к работе с сообщениями между ними.

Сообщение (message) – это связь между объектами, в которой один из них (клиент) требует от другого (сервера) выполнения каких-то действий. При генерации кода сообщения транслируются в вызовы функций. На диаграмме последовательности сообщение изображают в виде стрелки, которая проводится между линиями жизни двух объектов или от линии жизни объекта к самой себе. Сообщения располагают в хронологическом порядке сверху вниз.

Как правило, на каждой диаграмме взаимодействия имеется управляющий объект, отвечающий за управление последовательностью событий сценария. Обратите внимание, что управляющий объект не реализует никаких бизнес-процессов, он лишь посылает сообщения другим объектам. Управляющий объект отвечает за координацию действий других объектов и за делегирование ответственности. По этой причине такие объекты называют еще *объектами-менеджерами* (manager objects).

Упражнение 2

Создайте диаграмму последовательности и кооперативную диаграмму, отражающую ввод нового заказа в систему обработки заказов. Готовая диаграмма последовательности показана на рис.3.3.

Этапы выполнения упражнения

Настройка

В меню модели выберите пункт *Tools > Options* (Инструменты > Параметры). Перейдите на вкладку *Diagram* (Диаграмма). Установите флажки *Sequence numbering, Collaboration numbering* и *Focus of control*. Нажмите *OK*, чтобы выйти из окна параметров.

Создание диаграммы последовательности

Щелкните правой кнопкой мыши на логическом представлении браузера (Logical view). В открывшемся меню выберите пункт *New > Sequence Diagram* (Создать > Диаграмма последовательности). Назовите новую диаграмму *Add order* (Ввод заказа). Дважды щелкнув на этой диаграмме, откройте ее.

Добавление на диаграмму действующего лица и объектов

1. Перетащите действующее лицо *Salesperson* (Продавец) из браузера на диаграмму (предварительно необходимо переименовать действующее лицо в браузере: *Продавец → Salesperson*).
2. Нажмите кнопку *Object* (Объект) панели инструментов. Щелкните мышью в верхней части диаграммы, чтобы поместить туда новый объект. Назовите объект *Order Options Form* (Выбор варианта заказа).
3. Повторив п.2, поместите на диаграмму объекты: *Order Detail Form* (Форма деталей заказа) и *Order #1234* (Заказ №1234).

Добавление сообщений на диаграмму

1. На панели инструментов нажмите кнопку *Object Message* (Сообщение объекта). Проведите мышью от линии жизни действующего лица *Salesperson* (Продавец) к линии жизни объекта *Order Options Form* (Выбор варианта заказа). Выделив сообщение, введите его имя — *Create new order* (Создать новый заказ).
2. Повторив п.1, поместите на диаграмму остальные сообщения (см. рис.3.1): *Open form* (Открыть форму), *Enter order number, customer, order items* (Ввести номер заказа, заказчика и число заказываемых предметов), *Save the order* (Сохранить заказ), *Create new, blank order* (Создать пустой заказ), *Set the order number, customer, order items* (Ввести номер заказа, заказчика и число заказываемых предметов) и *Save the order* (Сохранить заказ).

Завершен первый этап работы. Готовая диаграмма последовательности представлена на рис.3.1.

Теперь нужно позаботиться об управляющих объектах и о взаимодействии с базой данных. Как видно из диаграммы, объект *Order Detail Form* имеет множество ответственностей, с которыми лучше всего мог бы справиться управляющий объект. Кроме того, новый заказ должен сохранять себя в базе данных сам. Вероятно, эту обязанность лучше было бы переложить на другой объект.

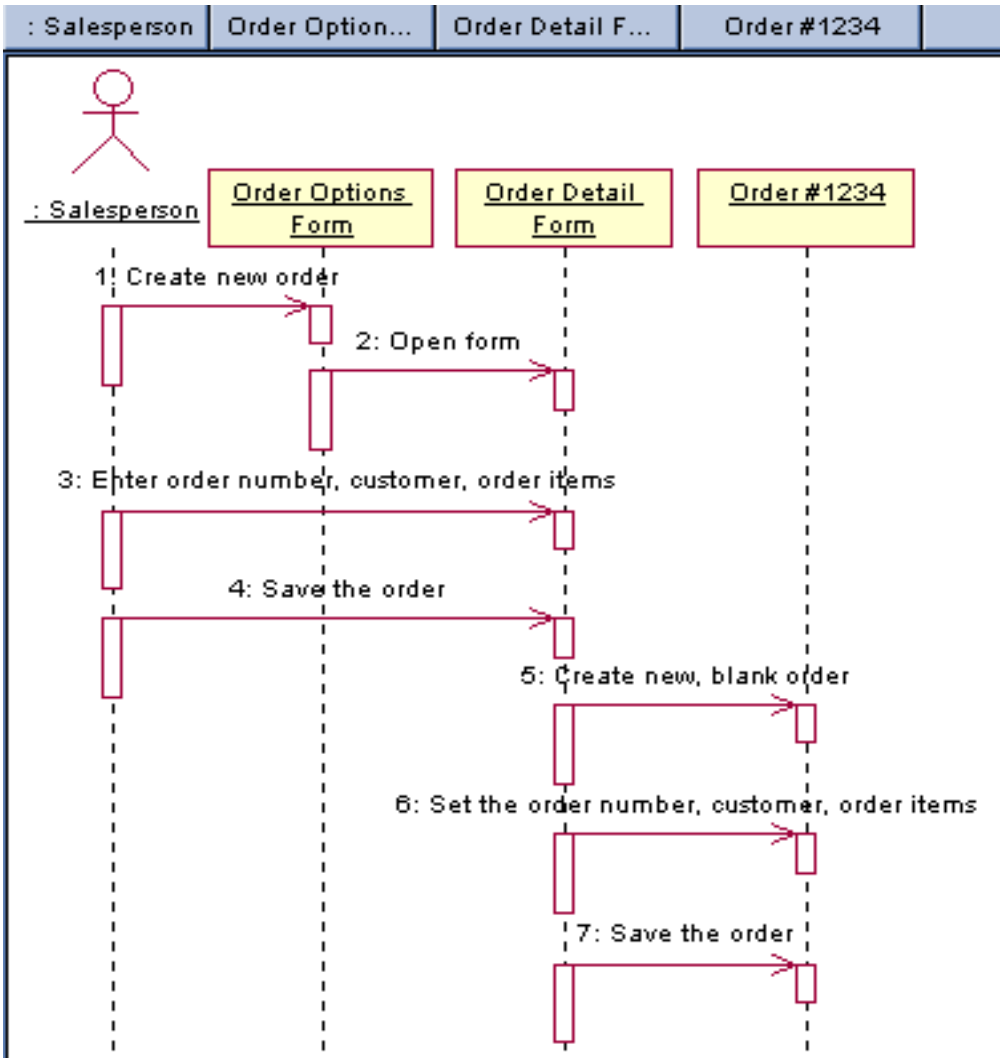


Рис.3.1. Диаграмма последовательности для ввода нового заказа после завершения первого этапа работы

Добавление на диаграмму дополнительных объектов

1. Нажмите кнопку *Object* (Объект) панели инструментов. Щелкните мышью между объектами *Order Detail Form* и *Order #1234*, чтобы поместить туда новый объект. Введите имя объекта — *Order Manager* (Управляющий заказами).
2. Нажмите кнопку *Object* (Объект) панели инструментов. Новый объект расположите справа от *Order #1234*. Введите его имя — *Transaction Manager* (Управляющий транзакциями).

Назначение ответственностей объектам

1. Выделите сообщение *5: Create new, blank order* (Создать пустой заказ). Нажав комбинацию клавиш CTRL+D, удалите это сообщение.
2. Повторите п.1 для удаления двух последних сообщений: *Set the order number, customer, order items* (Вести номер заказа, заказчика и число заказываемых предметов) и *Save the order* (Сохранить заказ).
3. Нажмите кнопку *Object Message* (Сообщение объекта) панели инструментов. Поместите на диаграмму новое сообщение, расположив его под сообщением 4 между *Order Detail Form* и *Order Manager*. Назовите его *Save the order* (Сохранить заказ).
4. Повторите п.3 для добавления сообщений с шестого по девятое и назовите их (см. рис.3.2): *Create new, blank order* (Создать новый заказ), *Set the order number, customer, order items* (Вести номер заказа, заказчика и число заказываемых предметов), *Save the order* (Сохранить заказ) и *Collect order information* (Информация о заказе).

5. На панели инструментов нажмите кнопку *Message to Self* (Сообщение себе). Щелкнув на линии жизни объекта *Transaction Manager* (Управляющий транзакциями) ниже сообщения 9, добавьте туда рефлексивное сообщение. Назовите его *Save the order information to the database* (Сохранить информацию о заказе в базе данных).

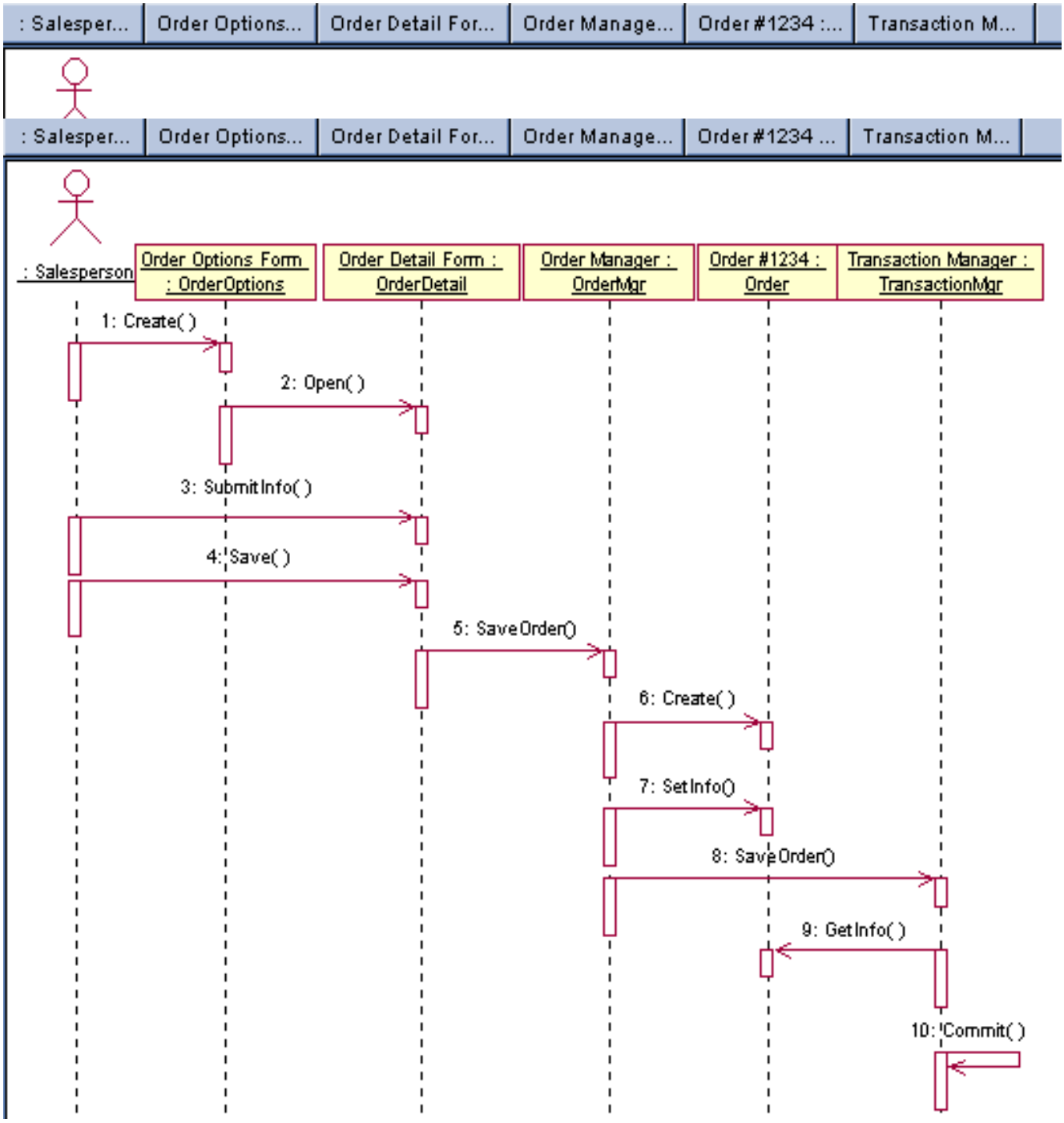
Соотнесение объектов с классами

1.Щелкните правой кнопкой мыши на объекте *Order Options Form* (Выбор варианта заказа). В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию). В раскрывающемся списке классов выберите пункт *<New>* (Создать). Появится окно спецификации класса. В поле *Name* введите *OrderOptions* (Выбор заказа). Щелкните на кнопке *OK*. Вы вернетесь в окно спецификации объекта. В списке классов выберите класс *OrderOptions*. Щелкните на кнопке *OK*, чтобы вернуться к диаграмме. Теперь объект называется *Order Options Form : OrderOptions*.

2. Для соотнесения остальных объектов с классами повторите п.1 (см. рис.3.2): класс *OrderDetail* соотнесите с объектом *Order Detail Form*, класс *OrderMgr* — с объектом *Order Manager*, класс *Order* — с объектом *Order #1234*, класс *TransactionMgr* — с объектом *Transaction Manager*.

По завершении этих действий ваша диаграмма должна иметь вид, представленный на рис. 3.2.

Рис.3.2. Диаграмма последовательности с новыми объектами и именами классов



Соотнесение сообщений с операциями

1. Щелкните правой кнопкой мыши на сообщении *1: Create new order* (Создать новый заказ). В открывшемся меню выберите пункт *<new operation>* (создать операцию). Появится окно спецификации операции. В поле *Name* введите имя операции *Create* (Создать). Нажмите на кнопку *OK*, чтобы закрыть окно спецификации операции. Еще раз щелкните правой кнопкой мыши на сообщении *Create new order*. В открывшемся меню выберите новую операцию *Create()*.
2. Повторите п.1, чтобы соотнести с операциями все остальные сообщения (см. рис.3.3)

Диаграмма последовательности должна выглядеть, как показано на рис.3.3.

Рис.3.3. Диаграмма последовательности с показанными на ней операциями

Создание Кооперативной диаграммы

Для создания кооперативной диаграммы достаточно нажать клавишу *F5*, предварительно выделив соответствующую диаграмму последовательности в браузере.

4. Диаграмма классов. Классы и пакеты

На *диаграммах классов* отображаются классы и пакеты системы. Это статические картины фрагментов системы и связей между ними.

В среде Rose диаграммы классов создаются в логическом представлении модели. Обычно для описания системы создают несколько диаграмм классов. На одних показывают некоторое подмножество классов и отношения между классами подмножества. На других отображают то же подмножество, но вместе с атрибутами и операциями классов. Третьи соответствуют только пакетам классов и отношениям между ними. Для представления полной картины системы можно разработать столько диаграмм классов, сколько требуется.

По умолчанию существует одна диаграмма классов, называемая *главной* (*Main*) и располагающаяся непосредственно под логическим представлением в браузере. На этой диаграмме показывают пакеты классов модели. Внутри каждого пакета также имеется главная диаграмма, включающая в себя все классы этого пакета.

Класс— это некоторая сущность, инкапсулирующая данные и поведение. В соответствии с традиционным подходом данные располагаются в базе данных, а поведением занимается собственно приложение.

Выявление классов можно начать с изучения потока событий сценария. Другим способом является анализ диаграмм последовательности и кооперативных диаграмм.

После создания диаграммы классов нужно добавить новые классы в модель. Доступны классы нескольких типов: регулярные, параметризованные, классы-наполнители, утилиты классов, утилиты параметризованных классов, утилиты классов-наполнителей и метаклассы.

Rose предоставляет ряд возможностей по детализации классов. Каждому классу можно дать имя, определить его стереотип, указать видимость, а также задать несколько других параметров.

На языке UML определены три основных стереотипа: *Boundary* (Граница), *Entity* (Объект), *Control* (Управление).

Пограничными классами (boundary classes) называются такие классы, которые расположены на границе системы со всем остальным миром. Они включают в себя формы, отчеты, интерфейсы с аппаратурой (такой, как принтеры или сканеры) и интерфейсы с другими системами.

Классы-сущности (entity classes) содержат информацию, хранимую постоянно. Классы-сущности можно обнаружить в потоке событий и на диаграммах взаимодействия. Они имеют наибольшее значение для пользователя, и потому в их названиях часто применяют термины из предметной области. Как правило, для каждого класса-сущности создают таблицу в базе данных.

Управляющие классы (control classes) отвечают за координацию действий других классов. Обычно у каждого варианта использования имеется один управляющий класс, контролирующий последовательность событий этого варианта использования.

Помимо упомянутых выше стереотипов, вы можете создавать и свои собственные.

Параметр *Visibility* (видимость) показывает, будет ли класс виден вне своего пакета. Вы можете указать для класса одно из следующих значений: *Public* (открытый), *Protected* (защищенный)**,** *Private* (закрытый) и *Package or Implementation* (пакет или реализация).

Поле *Cardinality* (Множественность) позволяет указать, сколько у данного класса должно быть экземпляров. Множественность управляющего класса обычно равна 1.

Абстрактным называется класс, который не наполняется конкретным содержанием (не инстанцируется). Обычно абстрактные классы применяют при работе с наследованием. В них содержатся данные и поведение, общие для нескольких других классов.

Пакеты (packages) применяются для группирования классов, обладающих некоторой общностью.

Для объединения классов существует несколько наиболее распространенных подходов. Во-первых, можно группировать классы по стереотипу. Второй подход заключается в объединении классов по их функциональности. Наконец, применяют комбинацию двух указанных подходов.

Очередным этапом разработки модели является добавление пакетов. Пакеты классов создают в логическом представлении браузера.

Упражнение 3

Необходимо сгруппировать в пакеты классы системы. Постройте диаграмму классов для отображения пакетов, диаграммы классов для представления классов в каждом пакете и диаграмму классов для представления всех классов варианта использования *Ввести новый заказ*.

Этапы выполнения упражнения

Настройка

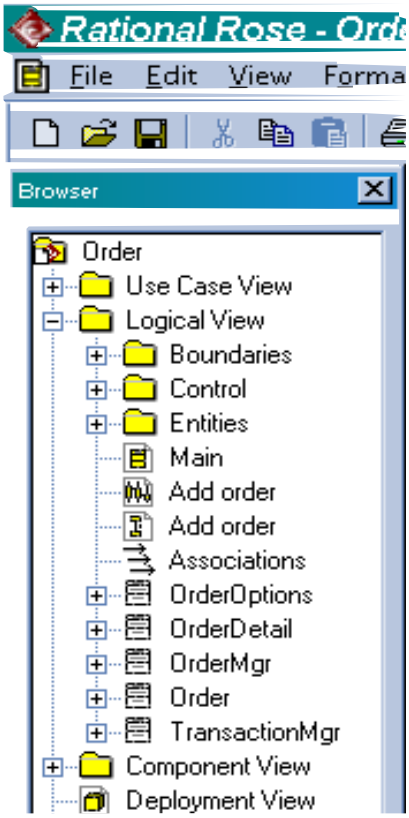
1. В меню модели выберите пункт *Tools* > *Options* (Инструменты > Параметры). Перейдите на вкладку *Diagram* (Диаграмма).
2. Убедитесь, что установлены флажки: *Show stereotypes* (Показать стереотипы), *Show All Attributes* (Показать все атрибуты) и *Show All Operations* (Показать все операции).
3. Убедитесь, что сброшены флажки *Suppress Attributes* (Подавить вывод атрибутов) и *Suppress Operations* (Подавить вывод операций).

Создание пакетов

1. Щелкните правой кнопкой мыши на *Логическом представлении* браузера (Logical view). В открывшемся меню выберите пункт *New* > *Package* (Создать > Пакет). Назовите новый пакет *Entities* (Сущности).
2. Повторив п.1, создайте пакеты *Boundaries* (Границы) и *Control* (Управление).

Браузер должен теперь иметь вид, показанный на рис. 4.1.

Рис.4.1. Пакеты системы обработки заказов



Создание главной диаграммы классов

1. Дважды щелкнув мышью на главной диаграмме классов, находящейся под логическим представлением браузера, откройте ее.

2. Перетащите пакеты *Entities*, *Boundaries* и *Control* из браузера на диаграмму.

Главная диаграмма классов должна выглядеть, как показано на рис.4.2.

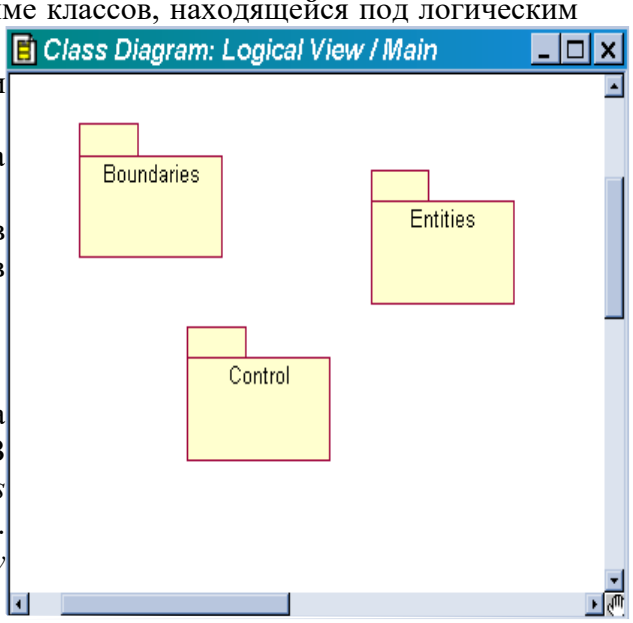


Рис.4.2. Главная диаграмма классов системы обработки заказов

Создание диаграммы классов для сценария «Ввести новый заказ» с отображением всех классов

1. Щелкните правой кнопкой мыши на логическом представлении браузера. В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмма классов). Назовите новую диаграмму классов *Add New Order* (Ввод нового заказа).

2. Дважды щелкнув мышью на этой диаграмме в браузере, откройте ее. Перетащите из браузера все классы (*OrderOptions, OrderDetail, Order, OrderMgr* и *TransactionMgr*).

Полученная диаграмма классов представлена на рис.4.3.

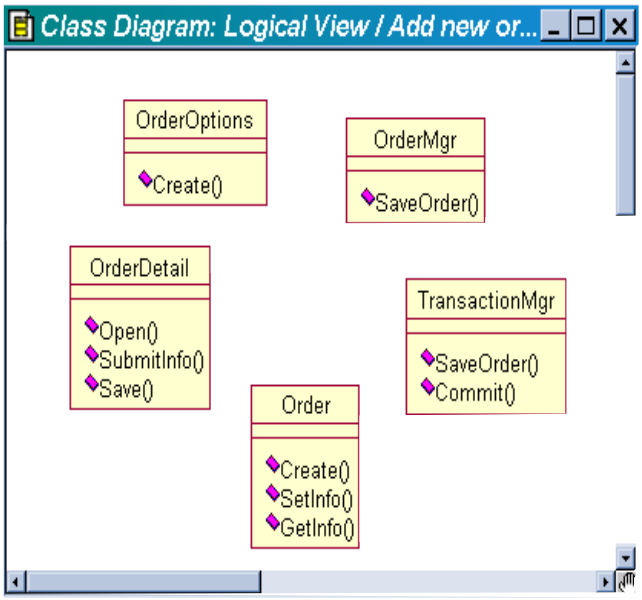


Рис.4.3. Диаграмма классов Add New Order

Добавление стереотипов к классам

1. Щелкните правой кнопкой мыши на классе *OrderOptions* диаграммы. В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию). В поле стереотипа введите слово *Boundary*. Нажмите на кнопку *OK*. Щелкните правой кнопкой мыши на классе *OrderDetail* диаграммы. В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию). В раскрывающемся списке поля стереотипов будет указан стереотип *Boundary*. Выделите его и нажмите на кнопку *OK*.
2. Повторив п.1, свяжите классы *OrderMgr* и *TransactionMgr* со стереотипом *Control*, а класс *Order* — со стереотипом *Entity*.

Объединение классов в пакеты

1. В браузере перетащите классы *OrderOptions* и *OrderDetail* в пакет *Boundaries*.
2. Перетащите классы *OrderMgr* и *TransactionMgr* в пакет *Control*.
3. Перетащите класс *Order* в пакет *Entities*.

Добавление диаграмм классов к каждому пакету

1. В браузере щелкните правой кнопкой мыши на пакете *Boundaries*. В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмма классов). Введите имя новой диаграммы — *Main* (Главная).
2. Дважды щелкнув мышью на этой диаграмме, откройте ее. Перетащите на нее из браузера классы *OrderOptions* и *OrderDetail*. Закройте диаграмму.

Главная диаграмма классов пакета *Boundaries* должна теперь выглядеть, как показано на рис.4.4.

Рис.4.4. Главная диаграмма классов пакета Boundaries.

3. В браузере щелкните правой кнопкой мыши на пакете *Entities*. В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмма Классов). Введите имя новой диаграммы — *Main* (Главная).4. Дважды щелкнув мышью на этой диаграмме, откройте ее. Перетащите на нее из браузера класс *Order*. Закройте диаграмму.

Главная диаграмма классов пакета *Entities* должна теперь иметь вид, представленный на рис.4.5.

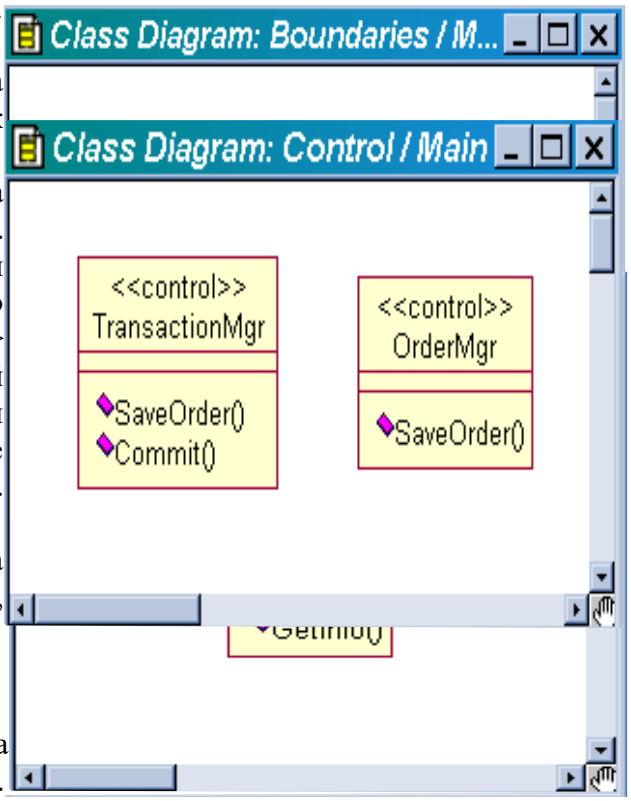


Рис. 4.5. Главная диаграмма классов пакета Entities.

5. В браузере щелкните правой кнопкой мыши на пакете Control. В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмма классов). Введите имя новой диаграммы — *Main* (Главная).
6. Дважды щелкнув мышью на этой диаграмме, откройте ее. Перетащите на нее из браузера классы *OrderMgr* и *TransactionMgr*. Закройте диаграмму.

Главная диаграмма классов пакета *Control* должна выглядеть, как показано на рис.4.6.

Рис.4.6. Главная диаграмма классов пакета Control

5. Атрибуты и операции классов

Классы инкапсулируют атрибуты (данные) и действующие на них операции (поведение). *Атрибут* – это фрагмент информации, связанный с классом. Rose дает возможность добавлять атрибуты (один или несколько) к классам модели.

Атрибуты можно выявить, изучая описание варианта использования или документацию, содержащую требования к системе. Кроме этого, если структура базы данных известна, то поля ее таблиц могут быть атрибутами. Определяя атрибуты, необходимо, чтобы каждый из них можно было соотнести с требованиями к системе. Далее внимательно соотнесите атрибуты с соответствующими классами.

В Rose можно определить подробные спецификации атрибута. Они включают в себя, помимо прочего, тип данных, значение по умолчанию, стереотип и видимость атрибута.

Одной из главных характеристик атрибута является его тип данных. Он специфичен для используемого языка. В качестве типов данных можно использовать либо встроенные типы данных языка программирования (string, integer, long и т.д.), либо определенные в модели имена классов.

Стереотип атрибута является способом его классификации. Например, некоторые атрибуты могут соответствовать полям базы данных, а другие нет. Для каждого такого типа можно определить свой стереотип. В Rose необязательно назначать стереотипы атрибутам. Стереотипы не требуются для генерации кода, но при их использовании легче читать и понимать модель.

Атрибуты могут иметь значения по умолчанию. Для генерации кода задавать начальные значения не обязательно. Тем не менее при их наличии генерируемый код будет соответствующим образом инициализировать атрибут.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим нужно указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется *видимостью атрибута* (attribute visibility).

Допустимы четыре значения этого параметра: *Public* (Общий), *Private* (Закрытый), *Protected* (Защищенный) и *Package or Implementation* (Пакетный).

В общем случае атрибуты рекомендуется делать закрытыми или защищенными. Это позволяет лучше контролировать сам атрибут и код. Задаваемые параметры видимости влияют на генерируемый код.

В среде Rose поддерживаются два набора нотаций видимости: нотация UML (+, -, #) и пиктограммы видимости Rose. На диаграмме классов разрешается применять любую из этих нотаций.

Операцией называется связанное с классом поведение. Операция состоит из трех частей: имени, параметров и типа возвращаемого значения. Параметры — это аргументы, получаемые операцией «на входе». Тип возвращаемого значения относится к результату действия операции. На диаграмме классов можно показывать как имена операций, так и их параметры и типы возвращаемого значения.

Следует рассмотреть четыре различных типа операций.

Операции реализации (implementor operations) реализуют некоторую бизнес-функциональность. Такие операции можно найти, исследуя диаграммы взаимодействия. Диаграммы этого типа фокусируются на бизнес-функциональности, и каждое сообщение диаграммы скорее всего можно соотнести с операцией реализации.

Операции управления (manager operations) управляют созданием и разрушением объектов. В эту категорию попадают конструкторы и деструкторы классов. В среде Rose не требуется вручную создавать конструкторы и деструкторы классов. При генерации кода предоставляется возможность сделать это автоматически.

Атрибуты обычно бывают закрытыми или защищенными. Тем не менее другие классы иногда должны просматривать или изменять их значения. Для этого предназначены *операции доступа* (access operations). Эти операции могут содержать любые правила и условия проверки, которые необходимо выполнить, прежде чем изменить атрибут.

Вспомогательными (helper operations) называются такие операции класса, которые необходимы ему для выполнения его ответственностей, но о которых другие классы не должны ничего знать. Это закрытые и защищенные операции класса.

Вспомогательные операции можно обнаружить на диаграммах последовательности и кооперативных диаграммах. Часто такие операции являются рефлексивными сообщениями.

Возвращаемым классом (return class) операции называется тип данных ее результата. При определении возвращаемого класса можно использовать либо встроенные типы языка программирования (такие, как string, char или integer), либо определенные в вашей модели классы.

Как и в случае других элементов модели, для классификации операций создаются их стереотипы. Существуют четыре наиболее распространенных стереотипа операций: *Implementor* (Реализация), *Manager* (Управляющая), *Access* (Доступ) и *Helper* (Вспомогательная). Назначение операциям стереотипов не требуется для генерации кода. Тем не менее они облегчают понимание модели.

Как уже упоминалось выше, видимость показывает, каким образом данные и поведение инкапсулируются в класс. При генерации кода Rose учтет установленную видимость атрибутов и операций.

Упражнение 4

Добавьте атрибуты и операции к классам диаграммы классов «Ввод нового заказа» (Add New Order). При этом учтите специфические для языка реализации особенности. Установите параметры так, чтобы показывать все атрибуты, все операции и их сигнатуры. Примените нотацию UML.

Этапы выполнения упражнения

Настройка

1. В меню модели выберите пункт *Tools* > *Options* (Инструменты > Параметры). Перейдите на вкладку *Diagram*.
2. Убедитесь, что установлены флажки: *Show visibility* (Показать видимость), *Show stereotypes* (Показать стереотипы), *Show operation signatures* (Показать сигнатуры операций), *Show all attributes* (Показать все атрибуты) и *Show all operations* (Показать все операции).
3. Убедитесь, что флажки *Suppress attributes* (Подавить атрибуты) и *Suppress operations* (Подавить операции) сброшены.
4. Перейдите на вкладку *Notation* (Нотация). Убедитесь, что флажок *Visibility as icons* (Отображать пиктограммы) сброшен.

Добавление нового класса

1. Найдите в браузере диаграмму классов *Add New Order*. Дважды щелкнув мышью на диаграмме, откройте ее.
2. Нажмите кнопку *Class* панели инструментов. Щелкните мышью внутри диаграммы, чтобы поместить туда новый класс. Назовите его *OrderItem*.
3. Назначьте этому классу стереотип *Entity*. В браузере перетащите класс в пакет *Entities.*

Добавление атрибутов

1. Щелкните правой кнопкой мыши на классе *Order*. В открывшемся меню выберите пункт *New Attribute* (Создать атрибут). Введите новый атрибут *OrderNumber : Integer*
2. Нажмите клавишу *Enter*. Введите следующий атрибут *CustomerName : String*
3. Повторив п.2, добавьте атрибуты *OrderDate : Date* и *OrderFillDate : Date*
4. Щелкните правой кнопкой мыши на классе *OrderItem*. В открывшемся меню выберите пункт *New Attribute* (Создать атрибут). Введите новый атрибут *ItemID : Integer*
5. Нажмите клавишу *Enter*. Введите следующий атрибут *ItemDescription : String*

Добавление операций к классу OrderItem

1. Щелкните правой кнопкой мыши на классе *OrderItem*. В открывшемся меню выберите пункт *New Operation* (Создать операцию). Введите новую операцию *Create*
2. Нажмите клавишу *Enter*. Введите следующую операцию *SetInfo*
3. Нажмите клавишу *Enter*. Введите операцию *GetInfo*

Подробное описание операций с помощью диаграммы классов

1. Щелкнув мышью на классе *Order*, выделите его. Щелкните на этом классе еще раз, чтобы переместить курсор внутрь.
2. Отредактируйте операцию *Create()*, чтобы она выглядела следующим образом: *Create() : Boolean*
3. Отредактируйте операцию SetInfo(): *SetInfo(OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date) : Boolean*
4. Отредактируйте операцию GetInfo(): *GetInfo() : String*

Подробное описание операций с помощью браузера

1. Найдите в браузере класс *OrderItem*. Раскройте этот класс, щелкнув на значке "+" рядом с ним. В браузере появятся атрибуты и операции класса.
2. Дважды щелкнув мышью на операции *GetInfo()*, откройте окно ее спецификации. В раскрывающемся списке *Return class* (Возвращаемый класс) укажите *String*. Щелкнув мышью на кнопке *OK*, закройте окно спецификации операции.
3. Дважды щелкните в браузере на операции *SetInfo()* класса *OrderItem*, чтобы открыть окно ее спецификации. В раскрывающемся списке *Return class* укажите *Boolean*. Перейдите на вкладку *Detail* (Подробно). Щелкните правой кнопкой мыши в области аргументов, чтобы добавить туда новый параметр. В открывшемся меню выберите пункт *Insert* (Вставить). Rose добавит аргумент под названием *argument*. Щелкнув один раз на этом слове, выделите его и измените имя аргумента на *ID*. Щелкните на колонке *Type* (Тип). В раскрывающемся списке типов выберите *Integer*. Щелкните на колонке *Default* (По умолчанию), чтобы добавить значение аргумента по умолчанию. Введите число *0*. Нажав на кнопку *OK*, закройте окно спецификации операции.
4. Дважды щелкните на операции *Create()* класса *OrderItem*, чтобы открыть окно ее спецификации. В раскрывающемся списке *Return class* укажите *Boolean*. Нажав на кнопку *OK*, закройте окно спецификации операции.

Подробное описание операций

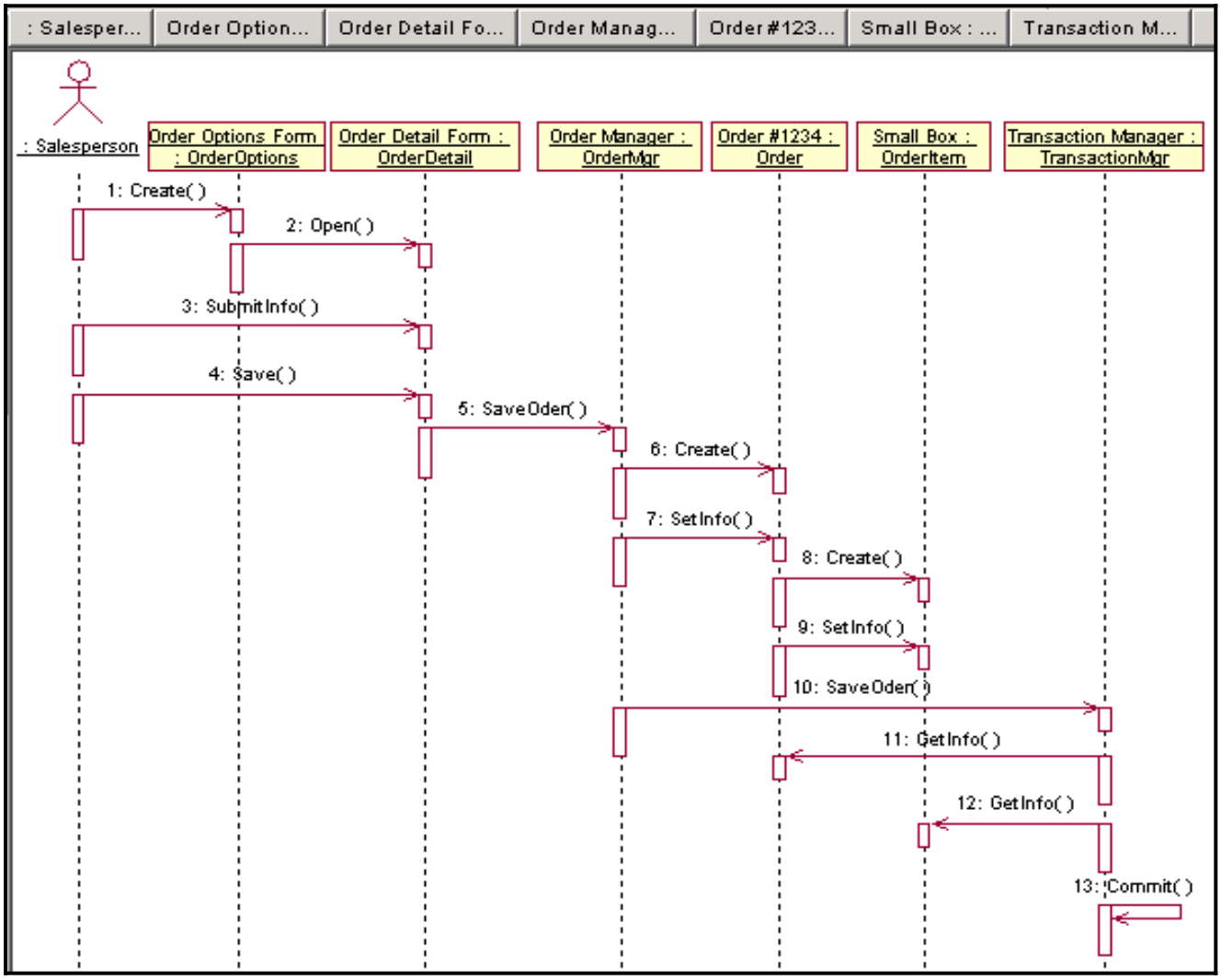
1. Используя браузер или диаграмму классов, введите следующие сигнатуры операций класса *OrderDetail*:
Open() : Boolean, *SubmitInfo() : Boolean* и *Save() : Boolean*
2. Используя браузер или диаграмму классов, введите сигнатуру операций класса *OrderOptions*: *Create() : Boolean*
3. Используя браузер или диаграмму классов, введите сигнатуру операций класса *OrderMgr*: *SaveOrder(OrderID : Integer) : Boolean*
4. Используя браузер или диаграмму классов, введите сигнатуры операций класса *TransactionMgr*: *SaveOrder(OrderID : Integer) : Boolean* и *Commit() : Integer*

Модификация диаграммы последовательности

1. Скорректируйте диаграмму последовательности *Add order* (Ввод заказа) согласно рис.5.3.

2. Получите соответствующую диаграмму кооперации *Add order* (Ввод заказа), нажав клавишу *F5*.

Рис.5.3. Модификация диаграммы последовательности



6. Связи на диаграмме классов

Связь представляет собой семантическую взаимосвязь между классами. Она позволяет классу узнавать об атрибутах, операциях и связях другого класса. Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения.

Ассоциация (association) — это семантическая связь между классами. Ее рисуют на диаграмме классов в виде обыкновенной линии. После того как классы связали ассоциацией, они могут передавать друг другу сообщения на диаграмме последовательности или кооперативной диаграмме. Ассоциации могут быть двунаправленными или однонаправленными. На языке UML двунаправленные ассоциации изображают в виде простой линии без стрелок или со стрелками с обеих сторон. На однонаправленной ассоциации ставят только одну стрелку, показывающую направление связи. После определения ассоциации Rose помещает в классы соответствующие дополнительные атрибуты.

Связь зависимости (dependency) также отражает связь между классами, но делает это несколько иначе. Зависимости всегда однонаправленные, они показывают, что один класс зависит от определений, сделанных в другом. Специальные атрибуты для классов, связанных зависимостью, не создаются. Зависимости изображают в виде стрелки, проведенной пунктирной линией.

Агрегация (aggregation) представляет собой более тесную форму ассоциации. Агрегация — это связь между целым и его частями. Агрегацию изображают в виде линии с незакрашенным ромбиком у класса, являющегося целым.

Отношение композиции является частным случаем отношения агрегации. Это отношение служит для выделения специальной формы отношения «часть-целое», при которой части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его составные части. Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который является классом-композицией или «целым».

С помощью *обобщений* (generalization) показывают связи наследования между двумя классами. На языке UML связь наследования называют обобщением и изображают в виде соответствующей стрелки от класса-потомка к классу-предку.

Зависимости можно устанавливать не только между классами, но и между пакетами. Фактически, это единственный тип связей, существующий между пакетами. Как и в случае классов, зависимость между пакетами изображают пунктирной линией. Зависимости определяют возможность повторного использования пакетов. Зависимости между пакетами можно обнаружить, исследуя связи на диаграмме классов. Если два класса из различных пакетов связаны, эти пакеты также связаны. Создавая зависимости между пакетами, старайтесь избегать циклических зависимостей.

До начала генерации кода необходимо указать также множественность связи, иначе будут заданы значения по умолчанию. *Множественность* (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью связи с одним экземпляром другого класса в данный момент времени. Ее индикаторы устанавливаются на обоих концах линии связи. Значение множественности позволяет понять, является ли данная связь обязательной.

Как и другим элементам модели, связям разрешается назначать *стереотипы*. Они применяются для классификации связей. Стереотипы пишут вдоль линии связи в двойных угловых скобках.

Упражнение 5

Необходимо определить связи между классами на диаграмме классов «Ввод нового заказа» (см. рис.6.1) и зависимости между пакетами на главной диаграмме классов системы (см. рис.6.2).

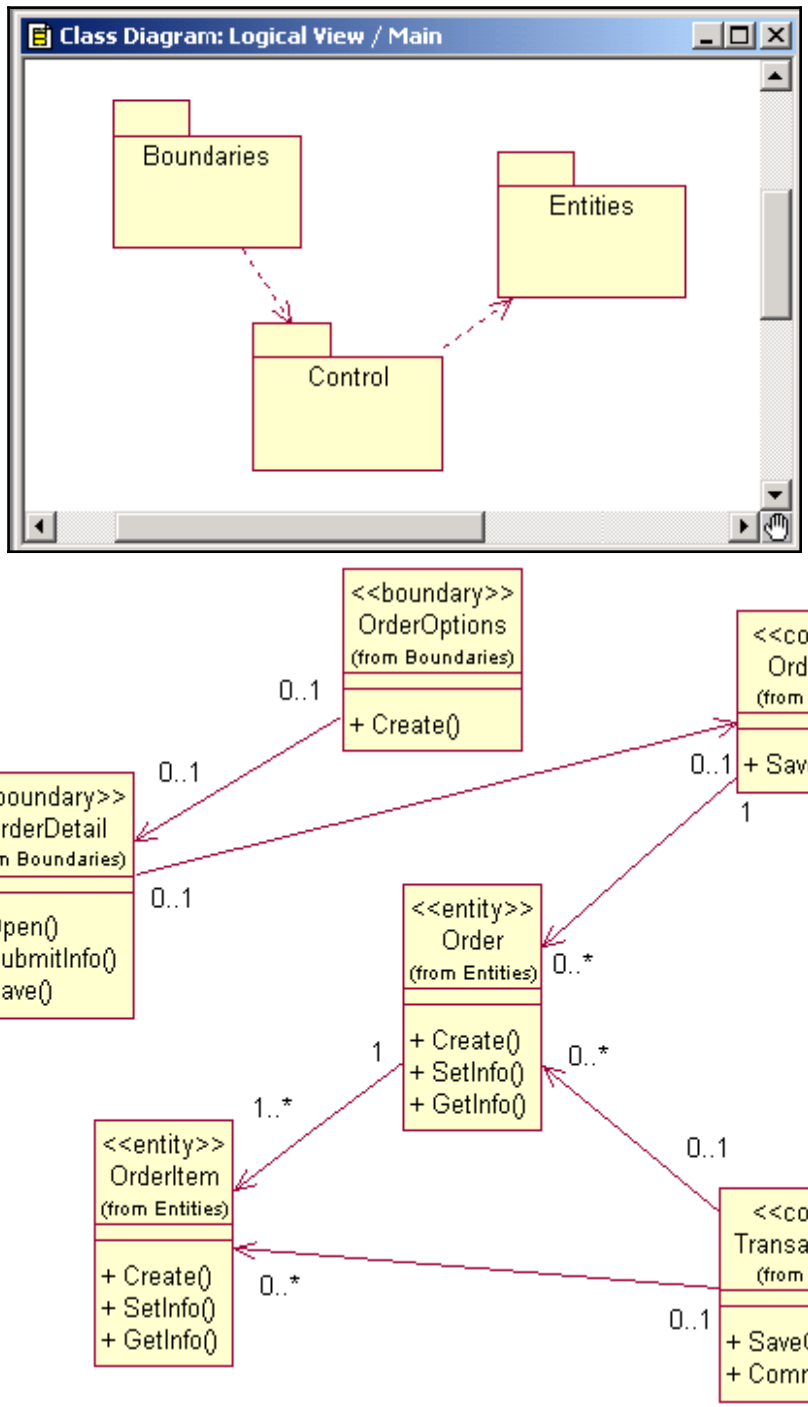
Этапы выполнения упражнения

Настройка

1. Найдите в браузере диаграмму классов *Add New Order* (Ввод нового заказа). Дважды щелкнув на диаграмме, откройте ее. Проверьте, имеется ли на панели инструментов диаграммы кнопка *Unidirectional Association* (Однонаправленная ассоциация). Если ее нет, продолжите настройку, выполняя п.2. Если есть, то приступайте к добавлению ассоциаций.
2. Щелкните правой кнопкой мыши на панели инструментов диаграммы и в открывшемся меню выберите пункт *Customize* (Настроить). Добавьте на панель кнопку *Creates a unidirectional association* (Создать однонаправленную ассоциацию).

Добавление ассоциаций

1. Нажмите кнопку *Unidirectional Association* панели инструментов. Проведите ассоциацию от класса *OrderOptions* к классу *OrderDetail*.
2. Повторив п.1, создайте остальные ассоциации (см. рис.6.1)
3. Щелкните правой кнопкой мыши на однонаправленной ассоциации между классами *OrderOptions* и *OrderDetail* со стороны класса *OrderOptions*. В открывшемся меню выберите пункт *Multiplicity > Zero or One* (Множественность **>** Нуль или один). Щелкните правой кнопкой мыши на другом конце однонаправленной ассоциации. В открывшемся меню выберите пункт *Multiplicity > Zero or One* (Множественность **>** Нуль или один).



4. Повторив п.3, добавьте на диаграмму значения множественности для остальных ассоциаций, как показано на рис. 6.1.

Рис.6.1. Ассоциации сценария «Ввести новый заказ»

Отображение зависимостей между пакетами

1. Найдите в браузере главную диаграмму классов системы (*Main*) и откройте ее.
2. Нажмите кнопку *Dependency* (Зависимость) панели инструментов. Щелкните мышью на пакете *Boundaries* главной диаграммы классов. Проведите линию зависимости к пакету *Control* (см. рис.6.2).
3. Повторив п.2, проведите зависимость от пакета *Control* к пакету *Entities*.

Рис. 6.2. Зависимости главной диаграммы классов системы

7. Диаграмма состояний

Диаграммы состояний предназначены для моделирования различных состояний, в которых может находиться объект. В то время как диаграммы классов показывают

статическую картину классов и их связей, диаграммы состояний применяются при описании динамики поведения системы.

В среде Rose можно создать одну диаграмму состояний для класса. На ней отображаются все определенные для этого класса состояния и переходы. В браузере диаграмма состояний располагается ниже класса. Диаграммы состояний не нужно создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него, вероятно, потребуется такая диаграмма. Диаграммы состояний необходимы в основном для документирования.

Состоянием (state) называется одно из возможных условий, в которых может существовать объект. Для выявления состояний объекта необходимо исследовать две области модели: значения атрибутов объекта и связи с другими объектами.

С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и историю состояния.

Деятельностью (activity) называется поведение, реализуемое объектом, когда он находится в данном состоянии.

Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. В отличие от деятельности, входное действие рассматривается как непрерываемое.

Выходное действие (exit action) подобно входному. Однако оно осуществляется как составная часть процесса выхода из данного состояния. Как и входное, выходное действие является непрерываемым.

Все описанные элементы можно добавить в модель Rose с помощью вкладки *Detail* (Подробно) окна спецификации состояния.

Переходом (transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может переходить из одного состояния в другое. Переходы могут быть *рефлексивными*: объект переходит в то же состояние, в котором он в настоящий момент находится. У перехода существует несколько спецификаций: события, аргументы, ограждающие условия, действия и посылаемые события.

Событие (event) — это то, что вызывает переход из одного состояния в другое. Событие помещают на диаграмме вдоль линии перехода. Для отображения события на диаграмме можно использовать как имя операции, так и обычную фразу. У событий могут быть *аргументы*. Rose позволяет добавлять аргументы к событиям.

Большинство переходов должно иметь события, так как именно они инициируют переход. Тем не менее бывают и автоматические переходы, не имеющие событий.

Ограждающие условия (guard conditions) определяют, когда переход может быть выполнен, а когда нет. На диаграмме ограждающие условия заключают в квадратные скобки и размещают вдоль линии перехода после имени события. Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия.

Действием (action) является непрерываемое поведение, выполняющееся как часть перехода. Входные и выходные действия показывают внутри состояния. Другие действия изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния. Действие размещают вдоль линии перехода после имени события, ему предшествует косая черта (/).

На диаграмме имеются два специальных состояния — *начальное* (start) и *конечное* (stop). Начальное состояние соответствует состоянию объекта в момент его создания. Конечное состояние соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще.

Кроме этого на диаграмме можно вкладывать состояния одно в другое. Вложенные состояния называются *подсостояниями* (substates), а те, в которые они вложены, — *суперсостояниями* (superstates). Если у нескольких состояний имеются идентичные переходы, эти состояния можно сгруппировать вместе в суперсостояние.

Упражнение 6

Постройте диаграмму состояний для класса *Order*, показанную на рис.7.1.

Этапы выполнения упражнения

Создание диаграммы состояний

Найдите в браузере класс *Order*. Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт *New* > *Statechart Diagram* (Создать диаграмму состояний).

Добавление начального и конечного состояний

1. Нажмите кнопку *Start State* (Начальное состояние) панели инструментов. Поместите это состояние на диаграмму.
2. Нажмите кнопку *End State* (Конечное состояние) панели инструментов. Также поместите это состояние на диаграмму.

Добавление суперсостояния

Нажмите кнопку *State* (Состояние) панели инструментов. Поместите это состояние на диаграмму.

Добавление оставшихся состояний

1. На панели инструментов нажмите кнопку *State* (Состояние). Поместите состояние на диаграмму. Назовите состояние *Cancelled* (Отменен).
2. Применяя описанный в п.1 метод, поместите остальные состояния на диаграмму (см. рис.7.1).

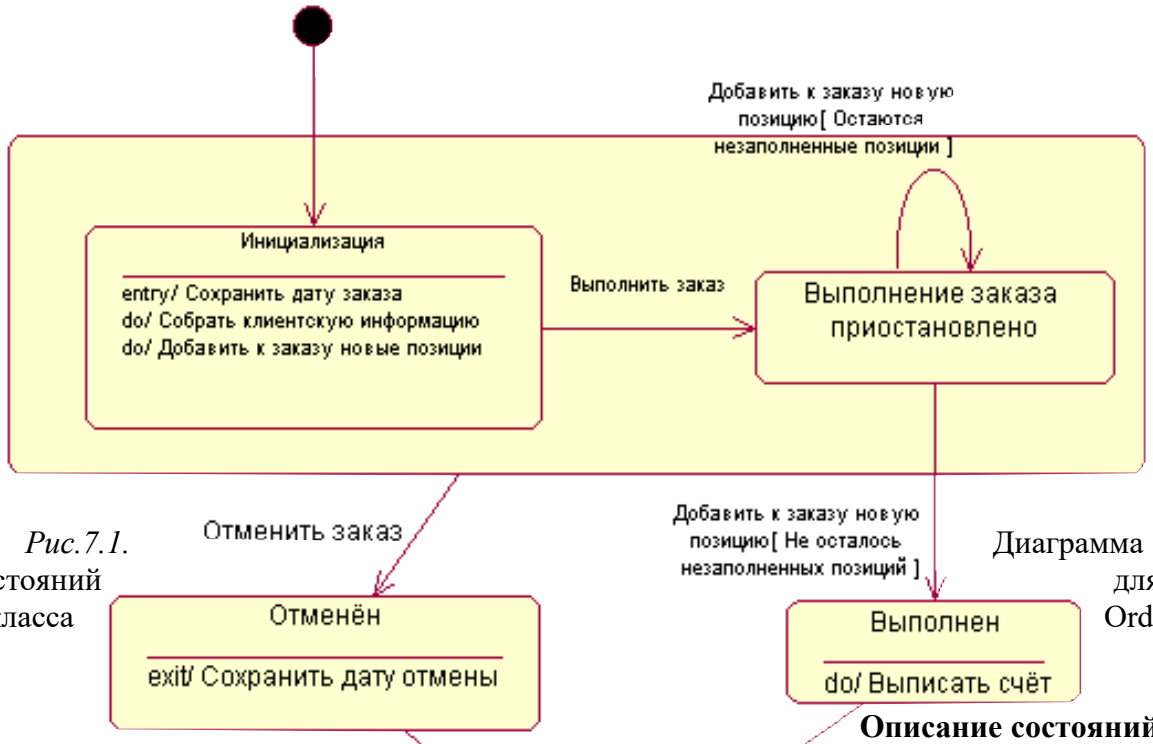


Рис.7.1. Диаграмма состояний для класса Order

Описание состояний

1. Дважды щелкните мышью на состоянии *Initialization* (Инициализация). Перейдите на вкладку *Detail* (Подробно). Щелкните правой кнопкой мыши в окне *Actions* (Действия). В открывшемся меню выберите пункт *Insert* (Вставить). Дважды щелкните мышью на новом действии. Назовите его *Store order date* (Сохранить дату заказа). Убедитесь, что в окне *When* (Когда) указан пункт *On Entry* (На входе).

2. Повторив п.1, добавьте следующие действия: *Collect customer info* (Собрать клиентскую информацию), в окне *When* укажите *Do* (Выполнить до завершения); *Add order items* (Добавить к заказу новые позиции), укажите *Do* (Выполнить до завершения). Нажмите два раза на *OK*, чтобы закрыть спецификацию.
3. Дважды щелкните мышью на состоянии *Cancelled* (Отменен). Повторив п.1, добавьте действие *Store cancellation data* (Сохранить дату отмены), укажите *On Exit* (На выходе). Нажмите два раза на *OK*, чтобы закрыть спецификацию.
4. Дважды щелкните мышью на состоянии *Filled* (Выполнен). Повторив п.1, добавьте действие *Bill customer* (Выписать счет), укажите *Do* (Выполнить до завершения). Нажмите два раза на *OK*, чтобы закрыть спецификацию.

Добавление переходов

1. Нажмите кнопку *Transition* (Переход) панели инструментов. Щелкните мышью на начальном состоянии. Проведите линию перехода к состоянию *Initialization* (Инициализация).
2. Повторив п.1, создайте остальные переходы (см. рис.7.1).
3. На панели инструментов нажмите кнопку *Transition to Self* (Переход к себе). Щелкните мышью на состоянии *Pending* (Выполнение заказа приостановлено).

Описание переходов

1. Дважды щелкнув мышью на переходе от состояния *Initialization* (Инициализация) к состоянию *Pending* (Выполнение заказа приостановлено), откройте окно спецификации перехода. В поле *Event* (Событие) введите фразу *Finalize order* (Выполнить заказ). Щелкнув на кнопке *OK*, закройте окно спецификации.
2. Повторив п.1, добавьте событие *Cancel Order* (Отменить заказ) к переходу между суперсостоянием и состоянием *Cancelled* (Отменен).
3. Дважды щелкнув мышью на переходе от состояния *Pending* (Выполнение заказа приостановлено) к состоянию *Filled* (Выполнен), откройте окно его спецификации. В поле *Event* (Событие) введите фразу *Add order item* (Добавить к заказу новую позицию). Перейдите на вкладку *Detail* (Подробно). В поле *Condition* (Условие) введите *No unfilled items remaining* (Не осталось незаполненных позиций). Щелкнув на кнопке *OK*, закройте окно спецификации.
4. Дважды щелкните мышью на рефлексивном переходе (Transition to Self) состояния *Pending* (Выполнение заказа приостановлено). В поле *Event* (Событие) введите фразу *Add order item* (Добавить к заказу новую позицию). Перейдите на вкладку *Detail* (Подробно). В поле *Condition* (Условие) введите *Unfilled items remaining* (Остаются незаполненные позиции). Щелкнув на кнопке *OK*, закройте окно спецификации.

8. Диаграмма деятельности

При моделировании поведения проектируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для отражения процесса выполнения операций в языке UML используются *диаграммы деятельности* (activity diagrams).

Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами — переходы от одного состояния действия к другому. Таким образом, диаграммы деятельности можно считать частным случаем диаграмм состояний.

Состояние действия является специальным случаем состояния с некоторым входным действием и по крайней мере одним выходящим из состояния переходом. Состояние действия используется в моделировании одного шага выполнения алгоритма (процедуры) или потока управления. Рекомендуется в качестве простого действия использовать глагол с пояснительными словами. Если же действие может быть представлено в некотором формальном виде, то целесообразно записать его на языке реализации проекта. На диаграмме деятельность (activity) изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами.

Каждая диаграмма деятельности должна иметь единственное начальное (start state) и конечное состояние (end state). Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз.

При построении диаграммы деятельности используют переходы, которые срабатывают сразу, как только закончится действие в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой (state transition). Если из состояния действия выходит единственный переход, то он может быть никак не помечен. Если же таких переходов несколько, то сработать может только один из них. Именно в этом случае для каждого из таких переходов должно быть явно записано *сторожевое условие* в квадратных скобках. При этом для всех выходящих из некоторого состояния 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 некоторого промежуточного результата. Такая ситуация получила название *ветвления.* Графически ветвление на диаграмме деятельности обозначается небольшим ромбом (decision), внутри которого нет никакого текста.

В языке UML используют специальный символ для разделения и слияния параллельных вычислений или потоков управления. Таким символом является прямая черта (horizontal synchronization) — отрезок горизонтальной линии, толщина которой несколько шире основных сплошных линий диаграммы деятельности. При этом *разделение* (concurrent fork) имеет один входящий переход и несколько выходящих. *Слияние* (concurrent join), наоборот, имеет несколько входящих переходов и один выходящий.

Диаграммы деятельности могут быть использованы при моделировании бизнес-процессов. В этом случае желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несёт ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому.

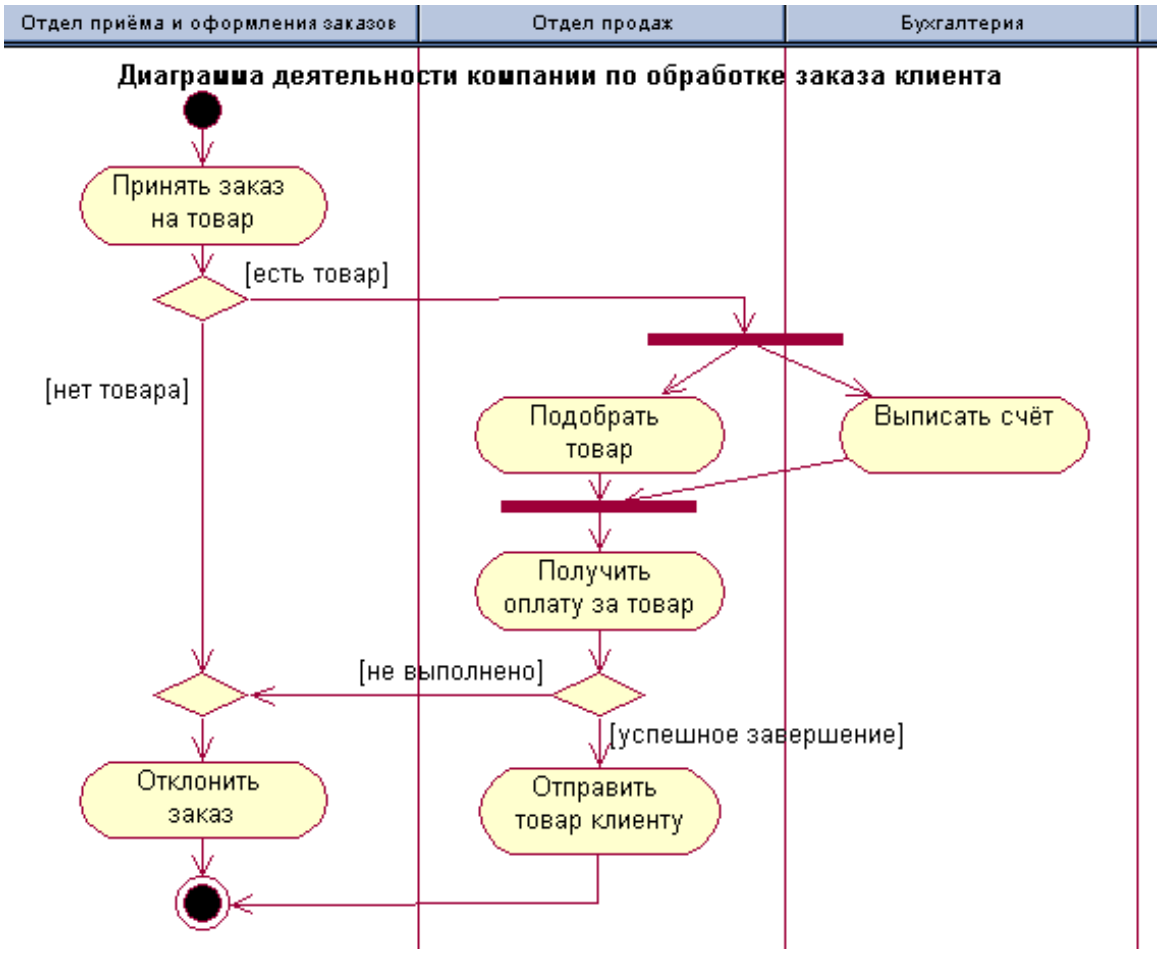
Для моделирования этих особенностей в языке UML используется специальная конструкция, получившая название *дорожки* (swimlane). При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии образуют дорожку, а группа состояний действия между этими линиями выполняется отдельным подразделением компании. Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение компании.

Любая деятельность может быть подвергнута дальнейшей декомпозиции. При построении диаграммы деятельности, представляющей собой декомпозицию деятельности более высокого уровня, на ней должна присутствовать только одна начальная точка. Однако конечных точек может быть столько, сколько выходов у деятельности более высокого уровня.

Упражнение 7

Постройте диаграмму деятельности компании по обработке заказа клиента (см. рис.8.1) и диаграмму деятельности, представляющую декомпозицию деятельности *Получить оплату за товар* (см. рис.8.2).

Рис.8.1. Диаграмма деятельности компании по обработке заказа клиента



Этапы выполнения упражнения

Создание диаграммы деятельности компании по обработке заказа

Найдите в браузере *Use Case View* (Представление вариантов использования). Щелкните на пакете представления правой кнопкой мыши и в открывшемся меню укажите пункт *New > Activity Diagram* (Создать диаграмму деятельности).

Добавление дорожек

1. Нажмите кнопку *Swimlane* (Дорожка) панели инструментов. Поместите этот элемент в верхнюю часть диаграммы. Назовите дорожку *Отдел приема и оформления заказов*.
2. Повторите п.1 и создайте следующие дорожки: *Отдел продаж* и *Бухгалтерия*.
3. Нажмите кнопку *Text Box* (Текст) панели инструментов. Поместите этот элемент под названиями дорожек на диаграмму. Введите текст названия диаграммы *Диаграмма деятельности компании по обработке заказа клиента*.

Добавление начального и конечного состояний

1. Нажмите кнопку *Start State* (Начальное состояние) панели инструментов. Поместите это состояние на дорожку *Отдел приема и оформления заказов*.
2. Нажмите кнопку *End State* (Конечное состояние) панели инструментов. Также поместите это состояние на дорожку *Отдел приема и оформления заказов*.

Добавление деятельностей

1. Нажмите кнопку *Activity* (Деятельность) панели инструментов. Поместите эту деятельность на соответствующую дорожку. Назовите деятельность *Принять заказ на товар*.
2. Повторите п.1 и создайте следующие деятельности: *Отклонить заказ*, *Подобрать товар*, *Выписать счет*, *Получить оплату за товар* и *Отправить товар клиенту*.

Добавление символов ветвления

1. Нажмите кнопку *Decision* (Символ ветвления) панели инструментов. Поместите этот элемент на соответствующую дорожку.
2. Повторите п.1 для добавления еще двух символов ветвления (см. рис.8.1).

Добавление линий синхронизации

1. Нажмите кнопку *Horizontal Synchronization* (Горизонтальная линия синхронизации) панели инструментов. Поместите этот элемент на соответствующую дорожку.
2. Повторите п.1 для добавления еще одной линии синхронизации (см. рис.8.1).

Добавление переходов

1. Нажмите кнопку *State Transition* (Переход) панели инструментов. Щелкните мышью на начальном состоянии. Проведите линию перехода к деятельности *Принять заказ на товар*.
2. Применяя описанный в п.1 метод, создайте остальные переходы на диаграмме (см. рис.8.1)

Описание переходов

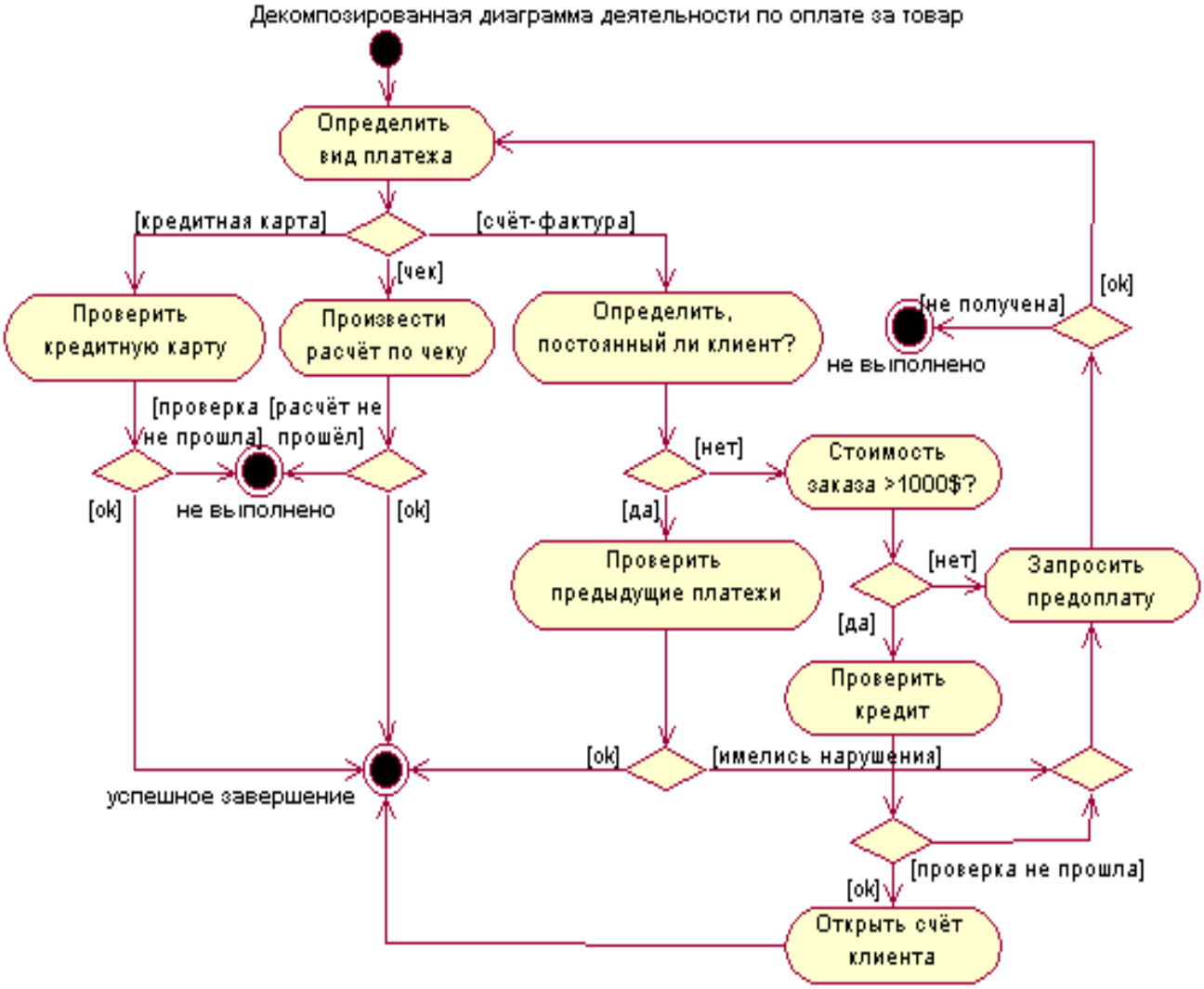
1. Дважды щелкнув мышью на переходе, откройте окно спецификации перехода.

Перейдите на вкладку *Detail* (Подробно). В поле *Guard Condition* (Сторожевое условие) введите нужную фразу, например, *есть товар*. Щелкнув на кнопке *ОК*, закройте окно спецификации.

2. Повторив п.1, занесите остальные сторожевые условия на диаграмму (см. рис.8.1.)

Создание диаграммы деятельности, представляющей декомпозицию деятельности *Получить оплату за товар*

1. Найдите в браузере деятельность *Получить оплату за товар*, которая относится к предыдущей диаграмме деятельности.
2. Щелкните на этой деятельности правой кнопкой мыши и в открывшемся меню укажите пункт *New > Activity Diagram* (Создать диаграмму деятельности).



3. Используя технологию построения предыдущей диаграммы деятельности, постройте диаграмму деятельности, представляющую декомпозицию деятельности более высокого уровня (см. рис.8.2).

Рис.8.2. Декомпозированная диаграмма деятельности по оплате за товар

9. Диаграмма компонентов

Диаграммы компонентов (Component diagrams) показывают, как выглядит модель на физическом уровне. На ней изображаются компоненты программного обеспечения вашей системы и связи между ними. При этом выделяют два типа компонентов: библиотеки исходного кода и исполняемые компоненты.

В среде Rose можно использовать различные пиктограммы для изображения компонентов разных типов, например:

- *Спецификация и тело пакета* (Package Specification and Body). Пакет в данном случае – это реализация класса. Спецификацией пакета является заголовочный файл со сведениями о прототипах функций для класса. На C++ это файл с расширением .H. Тело пакета содержит код операций класса. На C++ это файл .CPP.
- *Спецификация и тело задачи* (Task Specification and Body). Эти пиктограммы отображают пакеты, имеющие независимые потоки управления. Исполняемый файл обычно представляют как спецификацию задачи с расширением .EXE.

В среде Rose каждый класс модели преобразуется в компонент исходного кода. Созданные компоненты сразу добавляются к диаграмме компонентов. Затем указываются зависимости между отдельными компонентами, соответствующие зависимостям на этапе

компиляции или выполнения программы. Следует избегать циклических зависимостей между компонентами.

С каждым компонентом можно соотнести один или несколько классов. В результате в логическом представлении системы после имени класса появится имя соответствующего компонента, заключенное в скобки.

Как и для остальных элементов модели Rose, для компонентов можно определить подробные спецификации, в которых указываются: стереотип компонента, язык реализации, декларации (например, оператор #include языка C++), соотнесенные с компонентом классы модели.

Компоненты можно объединять в пакеты для лучшей организации. Как правило, для каждого пакета логического представления системы создается один пакет представления компонентов. Между пакетами компонентов устанавливают соответствующие зависимости.

С компонентом можно связать документацию, содержащую описание назначения компонента и его классов.

Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Диаграмма компонентов дает представление о том, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. Диаграмма показывает соответствие классов реализованным компонентам. Итак, она нужна там, где начинается генерация кода.

Упражнение 8

Необходимо создать диаграмму компонентов системы обработки заказов.

Этапы выполнения упражнения

Создание пакетов компонентов

1. Щелкните правой кнопкой мыши на представлении компонентов (Component view) в браузере. В открывшемся меню выберите пункт *New* > *Package* (Создать > Пакет). Назовите пакет *Entities* (Сущности).
2. Повторив п.1, создайте пакеты *Boundaries* (Границы) и *Control* (Управление).

Добавление пакетов на главную диаграмму компонентов

Откройте главную диаграмму компонентов, дважды щелкнув на ней мышью. Перетащите пакеты *Entities*, *Boundaries* и *Control* из браузера на главную диаграмму (см. рис.9.1).

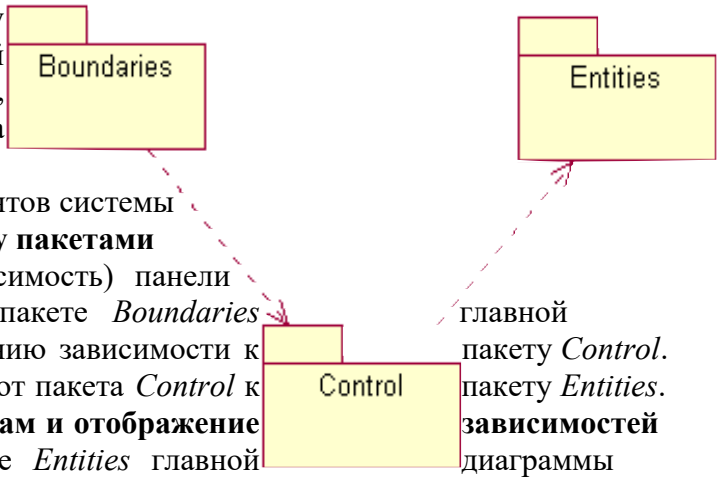


Рис. 9.1. Главная диаграмма компонентов системы

Отображение зависимостей между пакетами

1. Нажмите кнопку *Dependency* (Зависимость) панели инструментов. Щелкните мышью на пакете *Boundaries* главной диаграммы компонентов. Проведите линию зависимости к пакету *Control*.
2. Повторив п.1, проведите зависимость от пакета *Control* к пакету *Entities*.

Добавление компонентов к пакетам и отображение зависимостей

1. Дважды щелкнув мышью на пакете *Entities* главной диаграммы компонентов, откройте главную диаграмму компонентов этого пакета (см. рис.9.2).
2. Нажмите кнопку *Package Specification* (Спецификация пакета) панели инструментов. Поместите спецификацию пакета на диаграмму. Введите имя спецификации пакета — *OrderItem***.**
3. Повторив п.2, добавьте спецификацию пакета *Order***.**
4. Нажмите кнопку *Package Body* (Тело пакета) панели инструментов. Поместите его на диаграмму. Введите имя тела пакета — *OrderItem***.**

5. Повторив п.4, добавьте тело пакета *Order*.

6. Нажмите кнопку *Dependency* (Зависимость) панели инструментов. Щелкните мышью на теле пакета *OrderItem*. Проведите линию зависимости к спецификации пакета *OrderItem*.

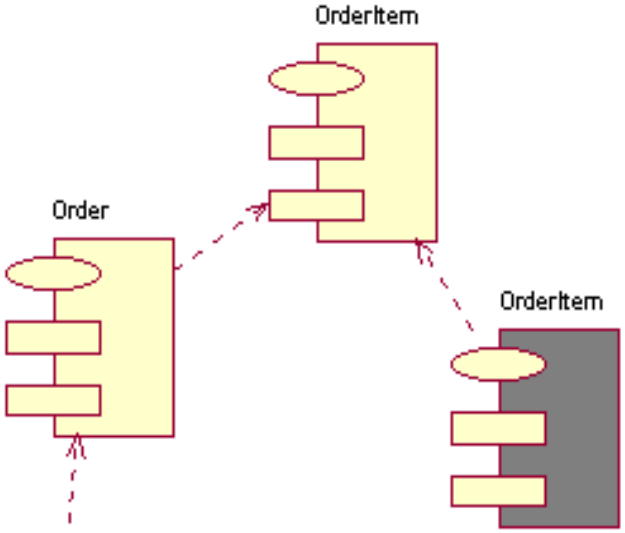


Рис.9.2. Диаграмма компонентов пакета Entities

7. Повторив шаги п. 6, добавьте линию зависимости между телом пакета *Order* и спецификацией пакета *Order*, а также линию зависимости от спецификации пакета *Order* к спецификации пакета *OrderItem*.

8. С помощью описанного метода создайте необходимые компоненты и зависимости для пакета *Control* (см. рис.9.3) и для пакета *Boundaries* (см. рис.9.4).

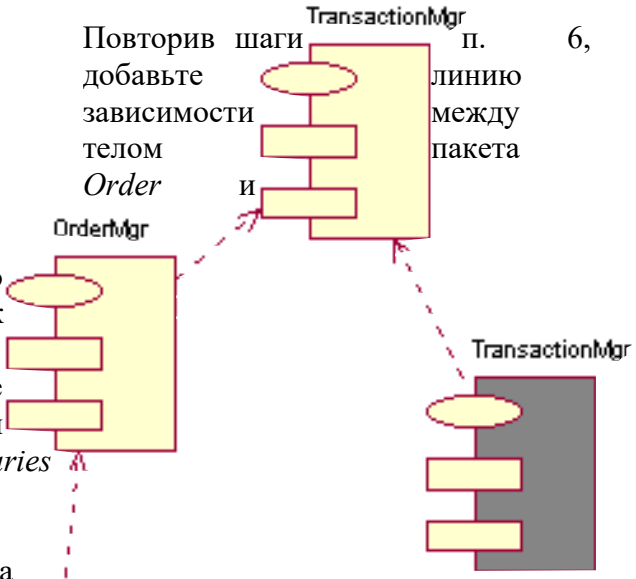
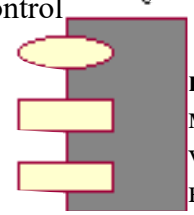
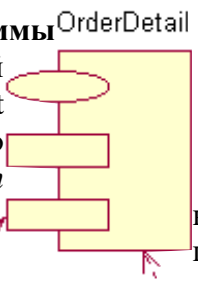


Рис.9.3. Диаграмма компонентов пакета Control

Создание диаграммы компонентов системы

Щелкните правой кнопкой мыши на представлении (Component view) в браузере. В открывшемся меню выберите пункт *New* > *Component Diagram* (Создать > Диаграмма компонентов). Назовите новую диаграмму *System*. Дважды щелкните на этой диаграмме мышью.





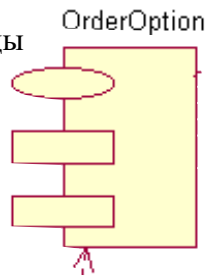
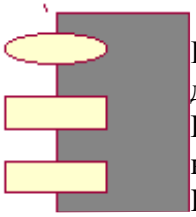


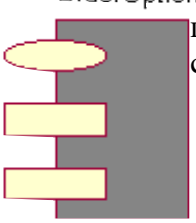
Рис.9.4. Диаграмма компонентов пакета Boundaries



Размещение компонентов на диаграмме компонентов системы

1. Разверните в браузере пакет компонентов *Entities*.

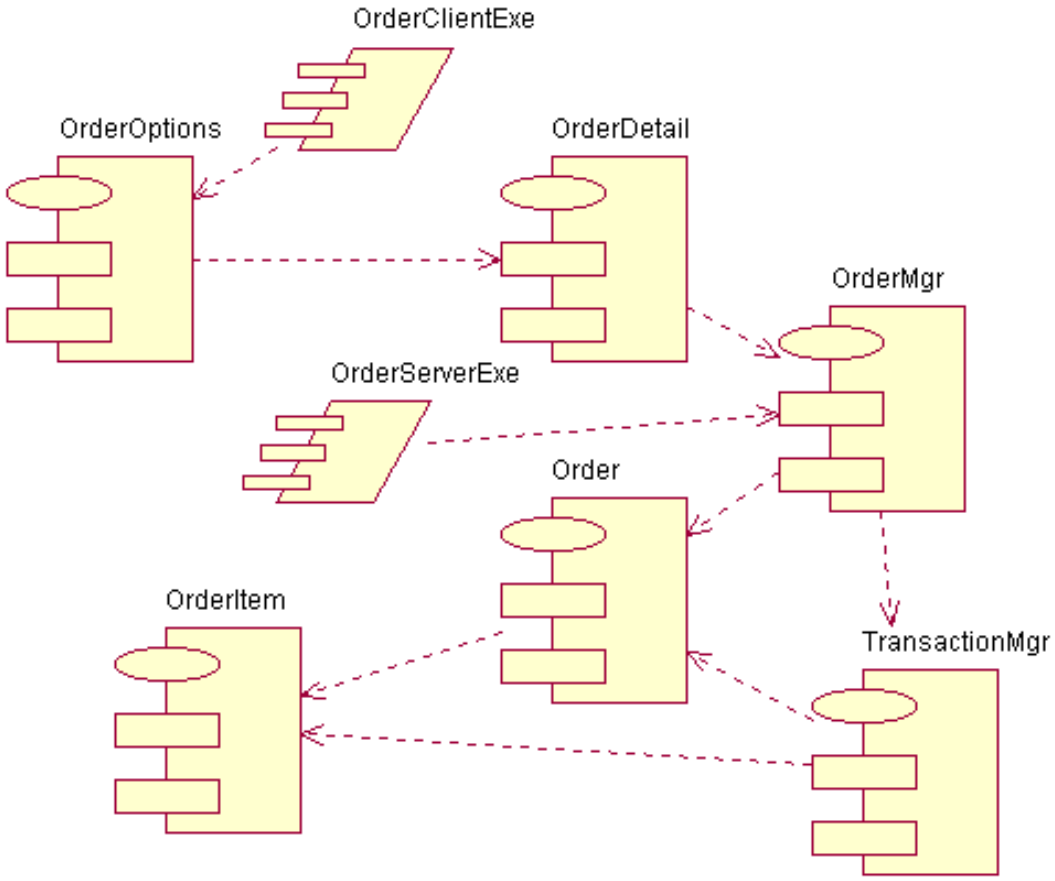
2. Щелкните мышью на спецификации пакета *Order* в пакете компонентов *Entities*. Перетащите эту спецификацию на диаграмму.



3. Повторив п.2, поместите на диаграмму спецификацию пакета *OrderItem*.
4. С помощью этого метода (см. пп.1-3) поместите на диаграмму остальные спецификации пакетов из пакета компонентов *Boundaries* и из пакета компонентов *Control* (см. рис.9.5).
5. Нажмите кнопку *Task Specification* (Спецификация задачи) панели инструментов. Поместите на диаграмму спецификацию задачи и назовите ее *OrderClientExe*.
6. Повторите п.5 для спецификации задачи *OrderServerExe*.

Добавление оставшихся зависимостей на диаграмму компонентов системы

Уже существующие зависимости будут автоматически показаны на диаграмме компонентов системы после добавления туда соответствующих компонентов. Теперь



нужно добавить остальные зависимости (см. рис.9.5).

Рис.9.5. Диаграмма компонентов системы

Соотнесение классов с компонентами

1. В логическом представлении (Logical view) браузера найдите класс *Order* пакета *Entities*. Перетащите этот класс на спецификацию пакета компонента *Order* в представлении компонентов (Component view) браузера. В результате класс *Order* будет соотнесен со спецификацией пакета компонента *Order*. Перетащите класс *Order* на тело пакета компонента *Order* в представлении компонентов браузера. В результате класс *Order* будет соотнесен с телом пакета компонента *Order.*
2. Повторив п.1, соотнесите остальные классы с соответствующими компонентами.

10. Диаграмма размещения

Представление размещения (Deployment view) отражает физическое распределение готового приложения, включая размещение и топологию сети, а также локализацию в ней компонентов системы.

Представление размещения содержит процессоры, устройства, процессы и связи между процессорами и устройствами. Все они наносятся на *диаграмму размещения* (Deployment diagram). Для системы и, следовательно, для модели Rose может быть создана только одна диаграмма размещения.

Процессором (processor) называется любая машина, имеющая вычислительную мощность, т.е. способная производить обработку данных. В эту категорию попадают серверы, рабочие станции и другие устройства, содержащие физические процессоры. В спецификациию *процессора* можно ввести информацию о его стереотипе, характеристиках и планировании.

Как и в случае других элементов модели, стереотипы применяются для классификации процессоров. Например, у вас могут быть компьютеры под управлением UNIX и другие ПК. Чтобы различать их, вы можете определить стереотипы.

Характеристики процессора — это его физическое описание. Оно может включать в себя скорость процессора и объем памяти.

Устройством (device) называется аппаратура, не обладающая вычислительной мощностью. Это, например, терминалы ввода/вывода (dumb terminals), принтеры и сканеры.

Процессоры и устройства называются также *узлами* (nodes) сети.

Как и в случае процессоров, для *устройств* можно определять различные детали, используя спецификацию устройства. Прежде всего это стереотип, применяемый для классификации устройств. Во-вторых, это характеристики, задаваемые в поле *Characteristics*. Они представляют собой физическое описание устройств.

Связью (connection) называется физическая связь между двумя процессорами, двумя устройства или процессором и устройством. Чаще всего связи отражают физическую сеть соединений между узлами вашей сети. Кроме того, это может быть ссылка Интернета, связывающая два узла.

В спецификации связи разрешается назначать стереотипы для связей и задавать характеристики связи, представляющие собой техническое описание физического соединения.

Процессом (process) называется поток обработки информации (execution), выполняющийся на процессоре. Процессом, например, считается исполняемый файл. Добавляя процессы на диаграмму, уделяйте внимание только тем из них, которые имеют отношение к проектируемой системе.

Процессы можно показывать или не показывать на диаграмме размещения. В первом случае они отображаются непосредственно под процессором (процессорами), на котором выполняются.

Процессам можно присваивать приоритеты и добавлять текстовое описание в спецификации процесса. Для добавления процесса его необходимо создать для соответствующего процессора в браузере.

При моделировании бизнес-процессов диаграмма размещения, кроме компьютеров корпоративной сети, может содержать в качестве узлов различные средства оргтехники (факсимильные устройства, многоканальные телефонные станции, множительные аппараты, экраны для презентаций и др.). При этом каждое из подобных устройств может функционировать как автономно, так и в составе корпоративной сети.

Упражнение 9

Разработайте диаграмму размещения для системы обработки заказов (см. рис.10.1).

Этапы выполнения упражнения

Добавление узлов к диаграмме размещения

1. Дважды щелкнув мышью на представлении размещения (Deployment view) в браузере, откройте диаграмму размещения.
2. Нажмите кнопку *Processor* (Процессор) панели инструментов. Щелкнув мышью на диаграмме, поместите туда процессор. Введите имя процессора *Сервер базы данных*.
3. Повторив п.2, добавьте следующие процессоры: *Сервер приложения, Клиентская рабочая станция №1* и *Клиентская рабочая станция №2*.
4. На панели инструментов нажмите кнопку *Device* (Устройство). Щелкнув мышью на диаграмме, поместите туда устройство. Назовите его *Принтер.*

Добавление связей

1. Нажмите кнопку *Connection* (Связь) панели инструментов. Щелкните мышью на процессоре *Сервер базы данных*. Проведите линию связи к процессору *Сервер приложения.*
2. Повторив п.1, добавьте остальные связи (см. рис.10.1).

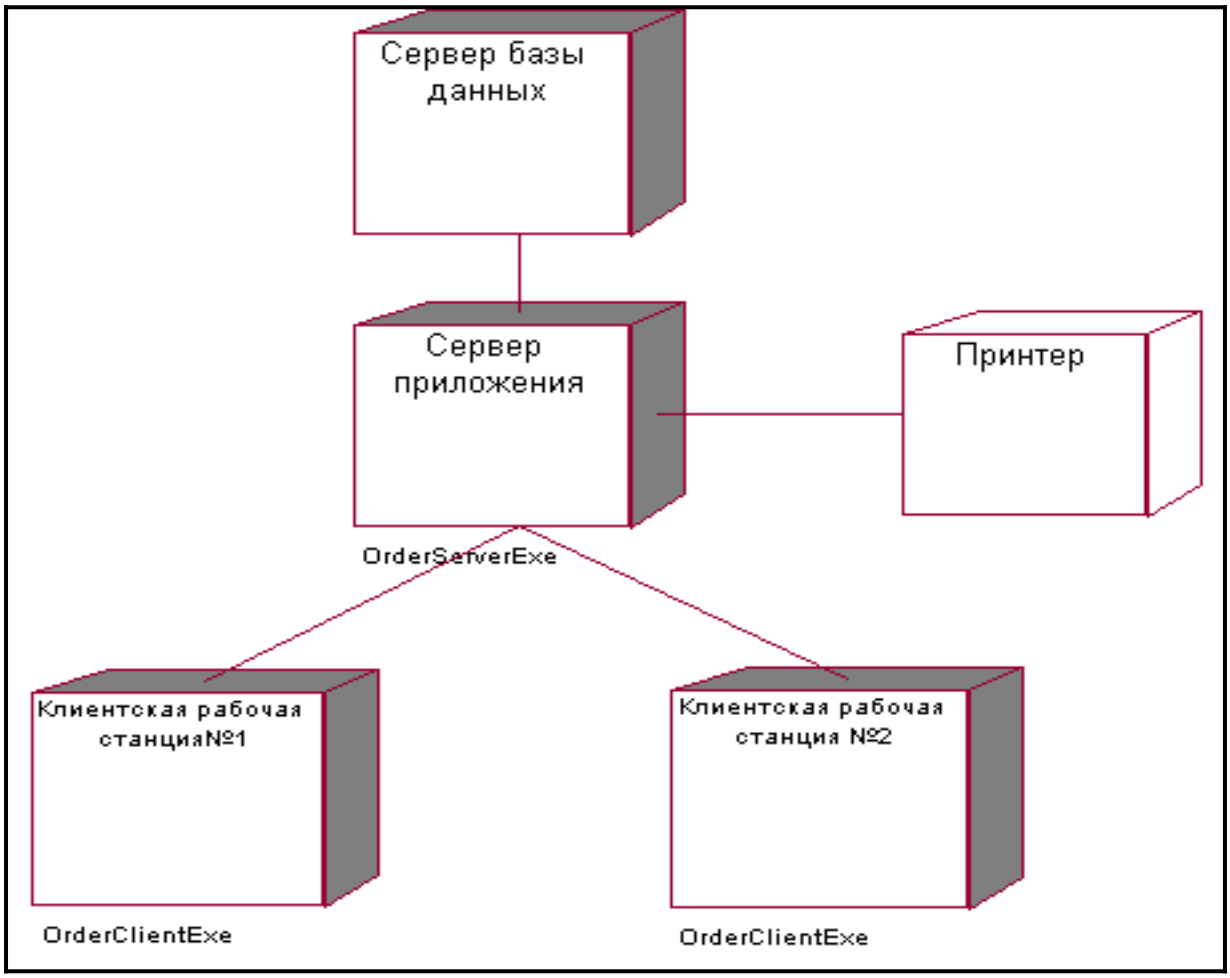


Рис.10.1. Диаграмма размещения для системы обработки заказов

Добавление процессов

1. Щелкните правой кнопкой мыши на процессоре *Сервер приложения* в браузере. В открывшемся меню выберите пункт *New* > *Process* (Создать > Процесс). Введите имя процесса — *OrderServerExe*.
2. Повторив п. 1, добавьте остальные процессы (*OrderClientExe)*.

Показ процессов на диаграмме

1. Щелкните правой кнопкой мыши на процессоре *Сервер приложения*. В открывшемся меню выберите пункт *Show Processes* (Показать процессы).
2. Повторив п.1, покажите процессы на следующих процессорах: *Клиентская рабочая станция №1* и *Клиентская рабочая станция №2*.

11. Генерация программного кода

Одно из важных свойств Rational Rose — возможность генерации программного кода, представляющего модель. Варианты генерации программ меняются в зависимости от установленной версии Rose.

Процесс генерации программного кода состоит из шести основных этапов: проверка модели; создание компонентов; отображение классов на компоненты; установка свойств генерации программного кода; выбор класса, компонента или пакета; генерация программного кода.

В разных языках реализации не все этапы обязательны. Так, программы C++ генерируются и без предварительного создания компонентов. Создавать код программ на любом языке можно, не выполняя шага проверки модели, хотя во время генерации это порой приводит к различным ошибкам.

В Rose существует не зависящее от языка реализации средство проверки моделей, применяемое для обеспечения корректности модели перед генерацией программного

кода (меню Tools > Check Model). Также можно обнаружить нарушения правил доступа, возникающие тогда, когда существует связь между двумя классами разных пакетов. При этом связи между самими пакетами нет.

Хотя не все вышеперечисленные этапы процесса генерации кода обязательны, первые пять из них рекомендуется выполнить до начала генерации программы. Проверка модели поможет выявить ее неточности и недостатки, нежелательные с точки зрения последующей генерации кода. Этапы, на которых рассматриваются компоненты, — это способ отобразить логическую схему системы на ее физическую реализацию, причем на этих этапах собирается большой объем полезной информации. Если пропустить эти этапы, то для создания компонентов Rose воспользуется пакетной структурой логического представления.

Существует возможность установки свойств генерации программного кода для классов, компонентов и других элементов модели. Этими свойствами определяется способ генерации программ. В Rose предлагаются установки по умолчанию. Вместо изменения непосредственно наборов свойств по умолчанию рекомендуется размножать (клонировать) эти наборы и изменять копии.

Во время генерации программного кода Rose выбирает информацию из логического и компонентного представлений модели. За один раз можно создать класс, компонент или целый пакет. Программный код генерируется с помощью диаграммы или браузера. В результате работы Rose получается большой объем «скелетного» (skeletal) программного кода.

По умолчанию корневой каталог, применяемый для генерации программ, ─ это каталог с системой Rose. Изменить его можно с помощью установки свойств генерации программного кода.

После генерации файлов остается сделать два шага. Сначала разработчики кодируют каждую из операций классов. Затем проектируется графический пользовательский интерфейс. Для создания экранов и форм следует пользоваться средой выбранного языка программирования.

Упражнение 10

Для модели обработки заказов сгенерируйте программный код на языке C++.

Этапы выполнения упражнения

Ввод тел пакетов на диаграмму компонентов системы

1. Откройте диаграмму компонентов системы (*System*). Выберите в браузере пакет компонентов *Entities*: тело пакета *Order*. Перетащите тело пакета *Order* на диаграмму компонентов системы.

2. Повторите п. 1 для следующих компонентов (см. рис.11.1):

Entities: тело пакета *OrderItem*, *Boundaries*: тело пакета *OrderOptions*, *Boundaries*: тело пакета *OrderDetail*, *Control*: тело пакета *TransactionMgr* и *Control*: тело пакета *OrderMgr*.

Проверка модели Rose

Выберите в меню *Tools > Check Model*. Проанализируйте и исправьте все найденные ошибки в окне журнала.

Обнаружение нарушений правил доступа

Откройте главную диаграмму классов модели (Main) в логическом представлении (Logical view) модели. Выберите в меню *Report > Show Access Violations*. Проанализируйте и исправьте все нарушения правил доступа, показанные в соответствующем диалоговом окне.

Рис.11.1. Диаграмма компонентов системы Order

Установка языка C++

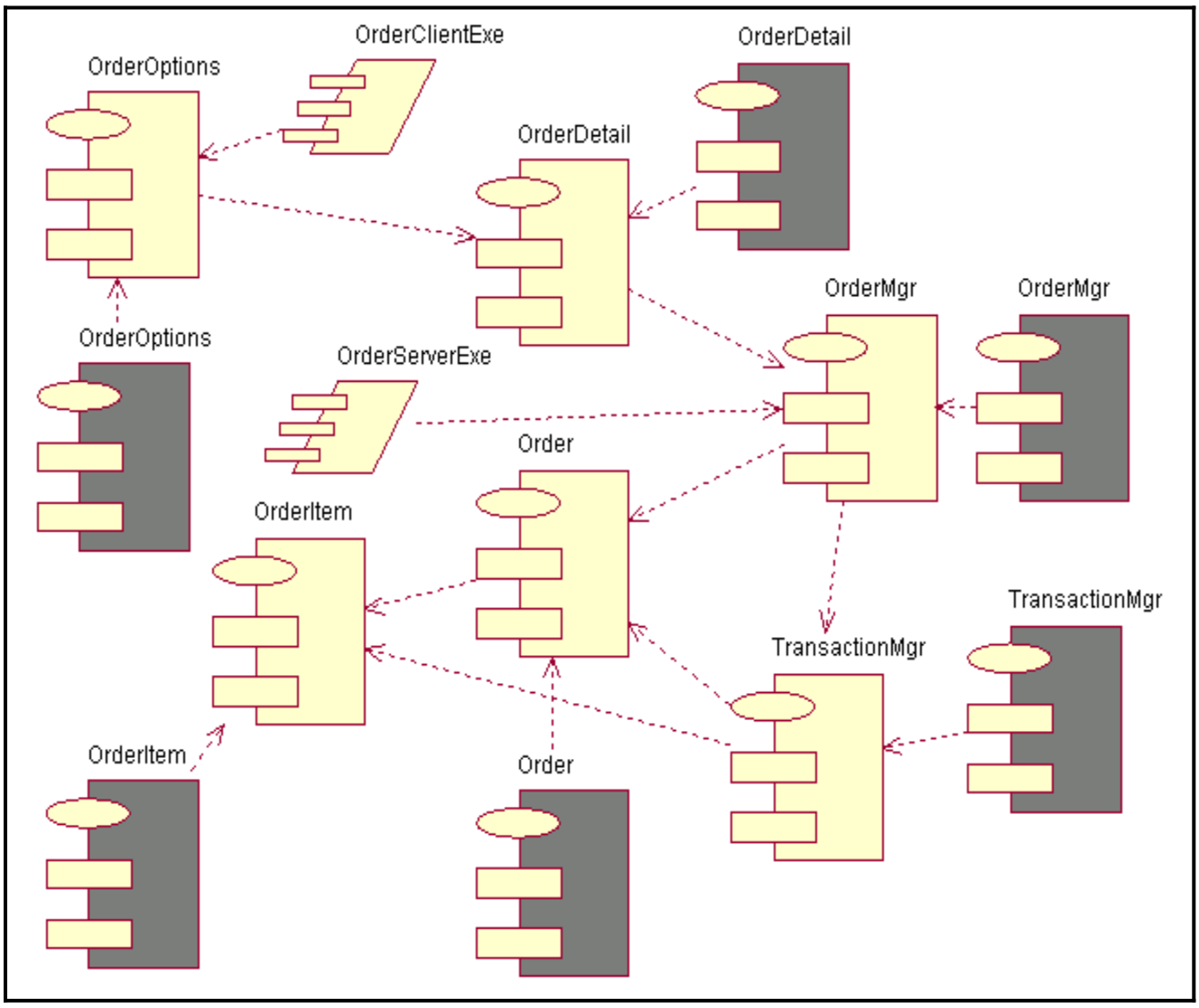
Выберите в меню *Add-Ins > Add-In Manager*. В диалоговом окне *Add-In Manager* поставьте флажок для варианта *Rose C++* и нажмите кнопку *Apply* (Применить), затем *OK*.

Назначение компонентам системы языка генерации программного кода

1. Откройте спецификацию компонента *Order* (спецификацию пакета) в пакете компонентов *Entities*. Выберите в качестве языка реализации *C++*.
2. Повторите п. 1 для остальных компонентов системы (см. рис.11.1).

Назначение свойств генерации программного кода

1. Любым способом создайте корневой каталог для генерируемого программного кода, где



будут созданы все каталоги и файлы C++ (например, *c:\order*).

2. Выберите в меню *Tools > Options* на вкладке C++ *Type: Project* для установки свойств генерации программного кода проекта.

3. Нажмите кнопку *Clone* для создания своей копии набора свойств проекта. Введите имя для нового набора свойств проекта (например, *MyPropertySet*) и выберите его в раскрывающемся списке *Set* диалогового окна *Options*.

4. Для свойства *Directory* в столбце *Value* необходимо набрать имя созданного корневого каталога для генерируемого программного кода проекта (например, *c:\order*). Нажмите

кнопку *Apply* (Применить) и *OK*.

Генерация программного кода C++

1. Откройте диаграмму компонентов системы (*System*). Выберите все компоненты на диаграмме *System* посредством меню *Edit >Select All*.
2. Выберите в меню *Tools > C++ > Code Generation*. Просмотрите созданный программный код в соответствующем каталоге для генерируемого программного кода проекта.

Практическое занятие № 7.
Выполнение пилотного проекта ИС, работа с технической документацией

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 6 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Основу отечественной нормативной базы в области документирования ПС составляет комплекс стандартов Единой системы программной документации**(ЕСПД).**

Основная и большая часть комплекса ЕСПД была разработана в 70-е и 80-егоды 20 века. Сейчас этот комплекс представляет собой систему межгосударственных стандартов стран СНГ (ГОСТ), действующих на территории Российской Федерации на основе межгосударственного соглашения по стандартизации.

Единая система программной документации - это комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации.

Стандарты ЕСПД в основном охватывают ту часть документации, которая создаётся в процессе разработки программных средств, и связаны, по большей части, с документированием функциональных характеристик программных средств.

Следует отметить, что стандарты ЕСПД (ГОСТ 19) носят рекомендательный характер. Впрочем, это относится и ко всем другим стандартам в области ПС (ГОСТ34, международному стандарту ISO/IEC и др.). Дело в том, что в соответствии с Законом РФ «О стандартизации» эти стандарты становятся обязательными на контрактной основе, т.е. при ссылке на них в договоре на разработку (поставку)программного средства.

Говоря о состоянии ЕСПД в целом, можно констатировать, что большая часть стандартов ЕСПД морально устарела. Тем не менее до пересмотра всего комплекса многие стандарты могут с пользой применяться в практике документирования программных средств.

К числу **программных** ЕСПД относит документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ.

Как известно, грамотно составленный пакет программной документации позволяет избежать при проектировании многих неприятностей. В частности, избавиться от назойливых вопросов и необоснованных претензий заказчика можно,

просто отослав пользователя к документации. Это касается прежде всего важнейшего документа — Технического задания.

Техническое задание (ТЗ) содержит совокупность требований к программному средству и может использоваться как критерий проверки и приемки разработанной программы. Поэтому достаточно полно составленное (с учетом возможности внесения дополнительных разделов) и принятое заказчиком и разработчиком ТЗ является одним из основополагающих документов проекта программного средства.

ГОСТ 19.201-78, входящий в ЕСПД, устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

ЗАДАНИЕ НА РАБОТУ

Разработать техническое задание на проектирование информационной системы, предназначенной для решения задач автоматизации деятельности организации.

Исходными данными для проектирования информационной системы являются описание предметной области и виды запросов в информационной системе (приложение 1).

Алгоритм выполнения работы

1) В соответствии с назначенным преподавателем вариантом определить наименование информационной системы (табл. 1), подлежащей проектированию входе лабораторного практикума, для удовлетворения основных требований к ней с применением системы управления базами данных Microsoft Access 2007 и/или инструментального средства Borland Turbo Delphi.

Таблица 1

№ варианта	Наименование информационной системы
1	Информационная система медицинских организаций города
2	Информационная система автопредприятия города
3	Информационная система проектной организации
4	Информационная система ГИБДД
5	Информационная система строительной организации
6	Информационная система библиотечного фонда города
7	Информационная система спортивных организаций города

8	Информационная система аэропорта
9	Информационная система гостиничного комплекса

№ варианта	Наименование информационной системы
10	Информационная система торговой организации
11	Информационная система ВУЗа
12	Информационная система железнодорожной пассажирской станции
13	Информационная система зоопарка
14	Информационная система театра
15	Информационная система фотоцентра

2) Изучить описание предметной области информационной системы (приложение 1).

3. На основании анализа описания предметной области и запросов к будущей информационной системе (приложение 1) сформулировать основные требования к ее функциям.
4. Выполнить поиск прототипа проектируемой информационной системы с применением Интернет.
5. Используя сформулированные требования к информационной системе, а также документацию пользователя на прототип найденного программного средства, разработать техническое задание в соответствии с ГОСТ 19.201-78 (приложение 2).

Контрольные вопросы к практическому занятию №7

1. Как можно охарактеризовать понятие «программная документация»?
2. Что представляет собой внешняя и внутренняя программная документация?
3. Дайте определение понятию «единая система программной документации».
4. В чем заключаются основные недостатки единой системы программной документации?
5. Дайте определение понятию «техническое задание».
6. Объясните смысл понятия «документация пользователя».
7. Какими свойствами должна обладать документация пользователя? Дайте краткую характеристику.

Практическое занятие № 8. Расчет надежности ИС

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;

- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Характеристики надежности. *Надежность ИС* – это свойство сохранять заданные характеристики при определенных условиях эксплуатации. Надежность ИС определяется безотказностью и восстанавливаемостью соответствующего программного обеспечения (ПО).

Безотказность ПО – его свойство сохранять работоспособность в процессе обработки информации, которую можно определить вероятностью работы без отказов при определенных условиях внешней среды в течение заданного периода наблюдения.

Отказ программы – это недопустимое отклонение характеристик процесса функционирования программы от требуемых. Определенные условия внешней среды – это совокупность входных данных и состояния вычислительной системы.

Заданный период наблюдения обычно соответствует необходимому числу прогонов программы для решения задачи.

Безотказность ПО можно охарактеризовать также *средним временем между двумя отказами* в процессе выполнения программы. С точки зрения надежности принципиальное отличие программных средств от аппаратных состоит в том, что программы не изнашиваются и не подвержены физическому старению в процессе работы.

Поэтому характеристики надежности ПО зависят от тщательности разработки и отладки, а также от условий хранения носителей программ. Безотказность ПО определяется его корректностью, а значит целиком зависит от наличия в нем ошибок, внесенных на

этапе создания и хранения, в то время как безотказность аппаратных средств зависит в ос-

новном от случайных отказов, связанных с физическими изменениями параметров эле-

ментов. Вид обрабатываемых данных не влияет на аппаратуру, но может привести к отка-

зам ПО.

Интенсивность отказов ПО с течением времени уменьшается, так как в процессе эксплуатации обнаруживаются и устраняются его скрытые ошибки.

Восстанавливаемость программы может быть оценена сравнительной продолжительностью устранения ошибки в программе и восстановления ее работоспособности. Восстановление после отказа может заключаться в корректировке текста программы, исправлении данных, внесении изменений в организацию вычислительного процесса. Восстанавливаемость зависит от сложности структуры комплекса программ, от

алгоритмического языка, от качества документации и т. д. Можно также говорить об

устойчивости ПО, понимая под этим способность ограничивать последствия собственных

ошибок и противостоять неблагоприятным условиям внешней среды. Устойчивость ПО

может быть повышена с помощью разных форм структурной, информационной и временной избыточности, позволяющей иметь дублирующие модули программ, альтернативные

пути для решения одной задачи, позволяющих осуществлять контроль за процессом ис-

полнения программ (зацикливание, блокировка и т. д.).

Причины отказов ПО:

□□ошибки, скрытые в самой программе;

□□искажение входной информации, подлежащей обработке;

□□неверные действия пользователя (могут быть связаны с некорректной документацией);

□□неисправность аппаратуры, на которой реализуется вычислительный процесс.

Последствия и признаки появления ошибок в ПО. В зависимости от степени серь-езности последствий ошибок в ПО отклонения выполнения программой заданных функ-

ций можно охарактеризовать следующим образом: полное прекращение выполнения

функций на длительное или неопределенное время или кратковременное прекращение хо-

да вычислительного процесса.

Симптомы проявления ошибки в программе:

□□преждевременное окончание программы;

□□увеличение времени выполнения программы (зацикливание);

□□потери или искажение накопленных данных;

□□нарушение порядка вызова отдельных программ.

Для устранения ошибок программы необходимо предусмотреть специальные сред-

ства диагностики типа кодов завершения, вводить в ПО контрольные точки, обеспечить

возможность рестарта с контрольных точек.

Аналитические модели надежности программ. Аналитические модели дают воз-

можность исследовать закономерности появления ошибок ПО, а также прогнозировать

надежность его эксплуатации ПО. Модели строятся в предположении, что появление

ошибок является случайным событием и имеет вероятностный характер.

Функцию надежности $P(t)$ можно определить как вероятность того, что ошибка по-

явится в программе не ранее чем через время t . Обратная величина $Q(t)$ – вероятность

того, что ошибка произойдет за время t .

Из этих характеристик можно вывести величину интенсивности отказов $\lambda(t)$, которая

будет определяться плотностью вероятности возникновения отказа

( )

( ) ( ).

dP t

t P t

dt

$\lambda = -$

Для ПО характерно ступенчатое изменение $\lambda(t)$. Поэтому наиболее простая модель надежности ПО – это дискретная модель

$( ) (( )) , _i \lambda\ t \square\square K\ M \square i\ t = \lambda$

где M – постоянная, характеризующая начальное число ошибок;

$i(t)$ – число отказов, устраняемое в момент времени t;

K – эмпирический коэффициент, зависящий от характеристик системы.

Эти параметры можно найти на основании последовательности наблюдений интервалов между обнаружением ошибок по методу ***максимального правдоподобия***, с помощью которого производится точечная оценка неизвестных параметров априорно известного закона распределения случайной величины. Суть этого метода – оценивание неизвестного параметра путем максимизации ***функции правдоподобия*** (на практике используется логарифмическая функция правдоподобия).

Если $i(t)$ постоянна – получаем простую линейную модель.

ЗАДАНИЕ НА РАБОТУ

Задание: определить время тестирования, необходимое для достижения указанного времени наработки на отказ. Исходные данные: время первого отказа, время второго отказа, время наработки на отказ.

1. Изучить свойства надежности ИС.
2. Изучить классификацию моделей надежности ИС.
3. Изучить особенности модели надежности программ с дискретным увеличением времени наработки на отказ.
4. Определить время тестирования, необходимое для достижения, указанного времени наработки на отказ. Исходные данные: время первого отказа, время второго отказа, время наработки на отказ.
5. Подготовить данные для отчета.

Контрольные вопросы к практическому занятию №8

1. Понятие надежности ИС.
2. Краткая характеристика аналитических моделей надежности.
3. Особенности аналитической модели с дискретным увеличением времени наработки на отказ.

Практическое занятие № 9.
Создание фрагментов проектной документации

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 6 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

TDBGrid находиться на вкладке DataControls, предназначен для отображения данных БД в нашей программе (приложении)

TDataSource находится на вкладке Data Access, предназначен для связи нашей сетки отображения данных, с самой БД

TADOConnection находится на вкладке ADO, предназначен для подключения нашей БД по определенному провайдеру

TADOQuery находится также на вкладке ADO, предназначен для получение нужных результатов из нашей БД.

DBGrid и DBEdit находится во вкладке DataControls

ЗАДАНИЕ НА РАБОТУ

Для готового программного модуля, создать руководство пользователя программного продукта.

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.

Контрольные вопросы к практическому занятию №9

1. Что такое ТЗ?
2. Что такое руководство пользователя?
3. Что такое руководство администратора?
4. Что такое руководство по техническому обслуживанию?
5. Что относиться к эксплуатационным документам?

Практическое занятие № 10.
Создание фрагментов эксплуатационной документации

Цель работы: целью работы является изучение файловой структуры диска и основных ее элементов, основных сервисных функции операционной системы MS-DOS и приобретение практических навыков их использования.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- основные теоретические положения (ответы на вопросы и форматы основных команд ОС, изученных в ходе выполнения работы);
- протокол выполнения заданий;
- заключение.

Время работы: 8 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Программная документация, включает:

1. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2. **текст программы** (запись программы с необходимыми комментариями);
3. **описание программы** (сведения о логической структуре и функционировании программы);
4. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5. **эксплуатационные документы.**

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов. Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);
- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.
- Руководство программиста

  Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:
- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);
- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

  Руководство оператора

  Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:
- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);
- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

ЗАДАНИЕ НА РАБОТУ

1. В файле приложении Microsoft Office Excel 2003 на листе с КТП назначить параметры столбцов в соответствии со строкой 27 .
2. В файле приложении Microsoft Office Excel 2003 на листе с КТП последовательно объединить ячейки в соответствии со строками 14-25.
3. Нанести соответствующие надписи ячейках.

4. Выделить группы ячеек утолщённой линией.
5. Скопировать сформированный лист в количестве 3 в данной книге и расположить их последовательно.
6. Удалить на первом листе КТП строки типа М, А, Б, Р.Преобразовать поле документа в Титульный лист и переименовать лист как Титульный (ТЛ).
7. Последующие листы переименовать как КТП1, КТП2.

Контрольные вопросы к практическому занятию №10

1. Что такое ТЗ?
2. Что такое руководство пользователя?
3. Что такое руководство администратора?
4. Что такое руководство по техническому обслуживанию?
5. Что относиться к эксплуатационным документам?

Список литературы

1. Устройство и функционирование ОС Windows. Практикум к курсу «Операционные системы» [Электронный ресурс]: учебное пособие/ Коньков К.А.— Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017.— 208 c.— Режим доступа: http://www.iprbookshop.ru/67369.html
2. Информационные системы : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. — 2-е изд. — М. : ФОРУМ : ИНФРА-М, 2018. — 448 с. : ил. — (Высшее образование). - Режим доступа: http://znanium.com/catalog/product/953245;
3. https://e.lanbook.com/reader/book/111916/#1 В.И. Петренко Защита персональных данных в информационных системах;
4. http://biblioclub.ru/index.php?page=book&id=493441; Современные информационные каналы и системы связи : учебник / В.А. Майстренко, А.А. Соловьев, М.Ю. Пляскин, А.И. Тихонов
5. Проектирование информационных систем. Курс лекций [Электронный ресурс]: учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий/ Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.— Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017.— 303 c.— Режим доступа: http://www.iprbookshop.ru/67376.html
6. Технологии командной разработки программного обеспечения информационных систем [Электронный ресурс]/ Долженко А.И.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 300 c.— Режим доступа: http://www.iprbookshop.ru/39569.html
7. Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008 [Электронный ресурс]/ Бурков А.В.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 310 c.— Режим доступа: http://www.iprbookshop.ru/52166.html
8. Технологии разработки современных информационных систем на платформе Microsoft.NET [Электронный ресурс]/ Павлова Е.А.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 128 c.— Режим доступа: http://www.iprbookshop.ru/52196.html