

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Минцаев Магомед Шавалович

Должность: Ректор

Дата подписания: 12.10.2023 15:32:09

Уникальный программный ключ:

236bcc35c296f119d6aafdc22846a2c051dbcc0111a86243e582591e4304dc

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

имени академика М.Д. Миллионщикова

Кафедра «Информационные технологии»

Бетербиева А.И.

Методические рекомендации к лабораторным работам по дисциплине

«Модели и методы проектирования информационных систем»

для студентов, обучающихся по направлению подготовки

09.03.02 «Информационные системы и технологии»

Магистр

Грозный 2023

Составители:

Бетербиева А.И., ассистент кафедры «Информационные технологии»

Рецензент:

Методические указания предназначены для бакалавров по направлению подготовки 09.03.02 Информационные системы и технологии института прикладных информационных технологий.

Методические рекомендации рассмотрены и утверждены на заседании кафедры «Информационные технологии»

Протокол № _ от _____ г

Рекомендовано к изданию редакционно-издательским советом ГГНТУ.

© Федеральное государственное бюджетное образовательное учреждение высшего образования «Грозненский государственный нефтяной технический университет имени академика М.Д. Миллионщикова», 2023

Содержание

Введение	4
Лабораторная работа №1. Выделение жизненных циклов проектирования компьютерных систем	6
Лабораторная работа №2. Разработка концептуальной модели и модели декомпозиции процесса (стандарт IDEF0)	17
Лабораторная работа №3. Создание диаграммы прецедентов.....	25
Лабораторная работа №4. Создание диаграмм взаимодействия	34
Лабораторная работа №5. Создание диаграммы классов.....	38
Лабораторная работа № 6. Создание диаграмм деятельности.....	47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	50

Введение

Разработка информационных систем является важной задачей в современном мире, где все больше компаний и организаций полагаются на технологии для улучшения процессов и повышения эффективности своей деятельности. Проектирование информационных систем включает в себя широкий спектр деятельности, от анализа требований и создания моделей до разработки, тестирования и внедрения.

Целью данного методического пособия является предоставить студентам необходимые знания, инструменты и методы для успешного проектирования информационных систем. Рекомендации охватывают основные концепции и подходы, используемые в процессе проектирования, а также предлагают практические рекомендации и примеры для лучшего понимания и применения материала. Для достижения этой цели перед студентами ставятся следующие задачи:

1. **Ознакомление с теоретическими основами:** Методические рекомендации предлагают студентам комплексное представление о различных моделях и методах, используемых в проектировании информационных систем. Это включает изучение концепций системного анализа, архитектуры информационных систем и принципов проектирования пользовательского интерфейса.

2. **Практическое применение знаний:** Рекомендации включают реальные примеры и практические задания, позволяющие студентам применять полученные знания на практике. Студенты разработают модели информационных систем, проведут анализ требований, разработают и протестируют пользовательский интерфейс, а также разработают план процесса разработки информационной системы.

3. **Развитие творческого подхода:** Методические рекомендации стимулируют творческое мышление студентов и развивают их способность принимать решения в сложных ситуациях. Студенты будут подходить к

проектированию информационных систем с инновационным взглядом, пытаясь найти оптимальные и эффективные решения.

Основная задача методических указаний – помочь студентам развить свои навыки и компетенции в проектировании информационных систем.

Лабораторная работа №1. Выделение жизненных циклов проектирования компьютерных систем

Цель работы: ознакомиться с моделями жизненного цикла информационных систем, определить достоинства и недостатки моделей, выбрать модель построения информационной системы индивидуального проектного задания и программные средства, составить план реализации индивидуального проектного задания.

Краткие теоретические сведения.

1. Жизненный цикл информационной системы

Разработка сложных информационных систем (ИС) невозможна без тщательно обдуманного методологического подхода. Понятие жизненного цикла является одним из базовых понятий методологии проектирования информационных систем.

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:

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

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

- Организационные процессы (создание инфраструктуры, управление, обучение, совершенствование).

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:

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

- 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.

- Rational Unified Process (RUP) – концепция итеративной (спиральной) разработки программного обеспечения, предложенная фирмой Rational Software (ныне – подразделение IBM). Жизненный цикл ИС представляет собой четыре фазы: начало (inception), исследование (elaboration), конструирование (construction) и внедрение (transition). Каждая фаза может содержать в себе несколько итераций. Кроме того, завершение всех четырех фаз не всегда означает завершение работы над проектом – его развитие может продолжиться новым циклом. В рамках итераций производится создание взаимосогласованных моделей, которые описываются на специально разработанном языке UML (Unified Modeling Language).

- Microsoft Solution Framework (MSF). Итерационная методология разработки приложений, аналогичная RUP. Так же включает четыре фазы: анализ, проектирование, разработка, стабилизация и предполагает использование объектно-ориентированного моделирования. По сравнению с RUP в большей степени ориентирована на разработку ИС для бизнеса.

- Extreme Programming (XP). Экстремальное программирование – это самая новая среди рассматриваемых методологий (первые идеи были сформированы в середине 1990-ых). Основные принципы: командная работа, эффективное взаимодействие между заказчиком и исполнителем в течение всего времени разработки ИС, использование последовательно дорабатываемых прототипов, достижение максимальной гибкости разработки (адаптация к изменяющимся требованиям заказчика).

2. 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.

Выбор модели жизненного цикла зависит от специфики, масштаба, сложности проекта и набора условий, в которых АИС создается и функционирует.

В соответствии с известными моделями ЖЦ программного обеспечения определяют модели ЖЦ ИС —каскадную, итерационную, спиральную.

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.

Выделяют **пять** устойчивых этапов разработки, практически не зависящих от предметной области (рис.1.1):



Рис.1.1. Этапы разработки

На первом этапе проводится исследование проблемной области, формулируются требования заказчика. Результатом данного этапа является техническое задание (задание на разработку), согласованное со всеми заинтересованными сторонами.

В ходе второго этапа, согласно требованиям технического задания, разрабатываются те или иные проектные решения. В результате появляется комплект проектной документации.

Третий этап — реализация проекта; по существу, разработка программного обеспечения (кодирование) в соответствии с проектными решениями предыдущего этапа. Методы реализации при этом принципиального значения не имеют. Результатом выполнения этапа является готовый программный продукт.

На четвертом этапе проводится проверка полученного программного обеспечения на предмет соответствия требованиям, заявленным в техническом задании. Опытная эксплуатация позволяет выявить различного рода скрытые недостатки, проявляющиеся в реальных условиях работы ИС.

Последний этап — сдача готового проекта.

На каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности. На заключительных этапах разрабатывается пользовательская документация, охватывающая все предусмотренные стандартами виды обеспечения АИС (организационное, информационное, программное, техническое и т. д.). Последовательное выполнение этапов работ позволяет планировать сроки завершения и соответствующие затраты.

При этом для каскадной модели необходимо отметить существенную задержку в получении результатов, сложность параллельного ведения работ по проекту и сложность управления проектом, и как следствие, высокий уровень риска и ненадежность вложений в ИС. Кроме того, ошибки и недоработки на любом из этапов проявляются, как правило, на последующих этапах работ, что приводит к необходимости возврата.

II. Итерационная модель заключается в серии коротких циклов (шагов) по планированию, реализации, изучению, действию (рис.1.2). 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.



Рис.1.2. Итерационная модель

Создание сложных ИС предполагает проведение согласований проектных решений, полученных при реализации отдельных задач. Подход к проектированию «снизу — вверх» обуславливает необходимость таких итераций возвратов, когда проектные решения по отдельным задачам объединяются в общие системные. При этом возникает потребность в пересмотре ранее сформировавшихся требований.

В итерационной модели межэтапные корректировки обеспечивают меньшую трудоемкость разработки по сравнению с каскадной моделью.

При этом, время жизни каждого этапа растягивается на весь период работы, вследствие большого числа итераций возникают рассогласования выполнения проектных решений и документации, возможно появление на стадии внедрения необходимости перепроектирования всей системы.

III. Спиральная модель, в отличие от каскадной, но аналогично предыдущей предполагает итерационный процесс разработки ИС (рис. 1.3). При этом возрастает значение начальных этапов, таких как анализ и проектирование, на которых проверяется и обосновывается реализуемость технических решений путем создания прототипов.

Каждая итерация представляет собой законченный цикл разработки, приводящий к выпуску внутренней или внешней версии изделия (или подмножества конечного продукта), которое совершенствуется от итерации к итерации, чтобы стать законченной системой:

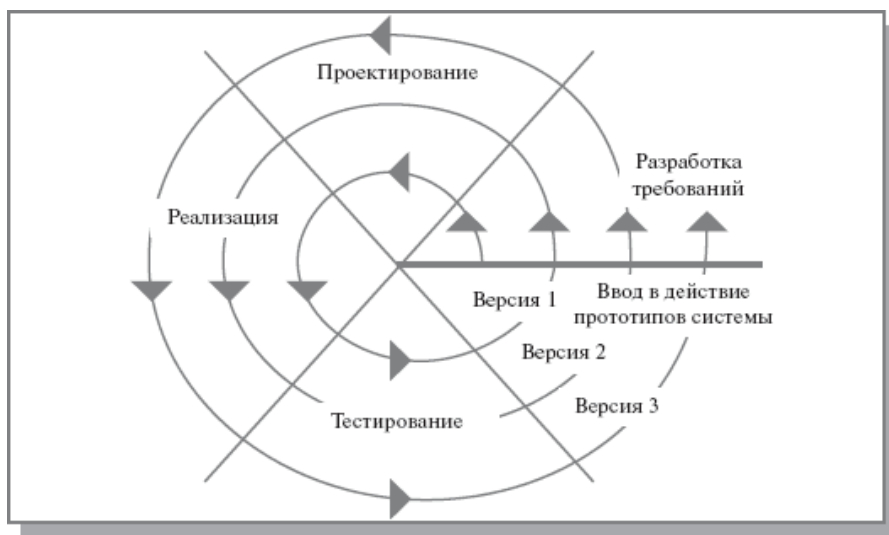


Рис.1.3. Спиральная модель

Таким образом, каждый виток спирали соответствует созданию фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы на следующем витке спирали. Каждая итерация служит для углубления и последовательной конкретизации деталей проекта, в результате этого выбирается обоснованный вариант окончательной реализации.

Использование спиральной модели позволяет осуществлять переход на следующий этап выполнения проекта, не дожидаясь полного завершения текущего, — недоделанную работу можно будет выполнить на следующей итерации. Главная задача каждой итерации — как можно быстрее создать работоспособный продукт для демонстрации пользователям. Таким образом, существенно упрощается процесс внесения уточнений и дополнений проект.

Спиральный подход к разработке программного обеспечения позволяет преодолеть большинство недостатков каскадной модели, кроме того, обеспечивает ряд дополнительных возможностей, делая процесс разработки более гибким. При использовании спиральной модели существенно упрощается внесение изменений в проект при изменении требований заказчика, происходит снижение уровня рисков (уровень рисков максимален в начале разработки проекта, по мере продвижения разработки он снижается), обеспечивается большая гибкость в управлении проектом за счет возможности внесения тактических изменений в разрабатываемое изделие, возможность совершенствовать процесс разработки — в результате анализа в конце каждой итерации проводится оценка изменений в организации разработки; на следующей итерации она улучшается, упрощается повторное использование компонентов, поскольку гораздо проще выявить (идентифицировать) общие части проекта, когда они уже частично разработаны, чем пытаться выделить их в самом начале проекта. Спиральная модель позволяет получить более надежную и устойчивую систему. Это связано с тем, что по мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации. Одновременно корректируются критические параметры эффективности, что в случае каскадной модели доступно только перед внедрением системы. Вовлечение пользователей в процесс проектирования и копирования приложения позволяет получать замечания и дополнения к требованиям непосредственно в процессе проектирования приложения,

сокращая время разработки. Представители заказчика получают возможность контролировать процесс создания системы и влиять на ее функциональное наполнение. Результатом является сдача в эксплуатацию системы, учитывающей большинство потребностей заказчиков.

Однако, организация проектирования ИС по спиральной модели обычно имеет высокую стоимость (поэтому ее имеет смысл использовать для сложных и дорогостоящих систем). Модель имеет сложную структуру, что может затруднить ее применение на практике неподготовленными специалистами и заказчиками. Спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может породить новый цикл работ. Большое количество промежуточных стадий усложняет ведение документации проекта. Возможны затруднения в определении момента перехода на следующую итерацию цикла. Обычно вводят временные ограничения на выполнение итерации и каждого из ее этапов.

В некоторых ситуациях применение спиральной модели невозможно или ограничено, поскольку невозможно использование/тестирование продукта, обладающего неполной функциональностью (например, военные разработки, атомная энергетика и т.д.). Поэтапное итерационное внедрение корпоративных информационных систем обычно сопряжено с организационными сложностями (перенос данных, интеграция систем, изменение бизнес-процессов, учетной политики, обучение пользователей). Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а неграмотное управление процессом внедрения может свести на нет все полученные результаты. По этой причине на этапе внедрения часто обходятся без итерационных моделей, внедряя систему «раз и навсегда».

3. Технология проектирования

Технология проектирования ИС — это совокупность методов и средств проектирования АИС, а также методов и средств организации проектирования (управление процессом создания и модернизации проекта ИС).

Предметом выбираемой технологии проектирования должно служить отражение взаимосвязанных процессов проектирования на всех стадиях жизненного цикла ИС.

Основные требования, предъявляемые к выбираемой технологии проектирования, следующие:

- созданный с помощью этой технологии проект должен отвечать требованиям заказчика;
- технология должна максимально отражать все этапы цикла жизни проекта;
- технология должна обеспечивать минимальные трудовые и стоимостные затраты на проектирование и сопровождение проекта;
- технология должна способствовать росту производительности труда проектировщиков;
- технология должна обеспечивать надежность процесса проектирования и эксплуатации проекта;
- технология должна способствовать простому ведению проектной документации.

Методы проектирования АИС можно классифицировать по степени использования средств автоматизации, типовых проектах решений, адаптивности к предполагаемым изменениям.

По степени автоматизации различают:

ручное проектирование, при котором проектирование компонентов АИС осуществляется без использования специальных инструментальных программных средств; программирование производится на алгоритмических языках;

компьютерное проектирование, при котором генерация или конфигурация (настройка) проектных решений производится с использованием специальных инструментальных программных средств.

По степени использования типовых проектных решений различают:

оригинальное (индивидуальное) проектирование, когда проектные решения разрабатываются «с нуля» в соответствии с требованиями к АИС;

типовое проектирование, предполагающее конфигурацию АИС из готовых типовых проектных решений (программных модулей).

По степени адаптивности проектных решений различаются следующие методы:

реконструкция — адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);

параметризация — проектные решения настраиваются в соответствии с заданными и изменяемыми параметрами;

реструктуризация модели — изменяется модель предметной области, что приводит к автоматическому переформированию проектных решений.

Сочетание различных признаков классификации методов проектирования обуславливает характер используемой технологии проектирования АИС. Выделяются два основных класса технологии проектирования: *каноническая* и *индустриальная*. Индустриальная технология проектирования в свою разбивается на два подкласса: *автоматизированное* (использование CASE-технологий) и *типовое* (параметрически-ориентированное или модельно-ориентированное) проектирование.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств

CASE-средства позволяют создавать не только продукт, практически готовый к применению, но и обеспечить “правильный” процесс его разработки. Основная цель технологии – отделить проектирование программного обеспечения от его кодирования, сборки, тестирования и максимально “скрыть” от будущих пользователей все детали разработки и функционирования ПО. При этом значительно повышается эффективность работы проектировщика: сокращается время разработки, уменьшается число программных ошибок, программные модули можно использовать при следующих разработках.

В качестве примеров популярных CASE-средств укажем программные средства компании Computer Associates, IBM-Rational Software и Oracle:

- ВРwin – моделирование бизнес-процессов;

- ERwin – моделирование баз данных и хранилищ данных;
- ERwin Examiner – проверка структуры СУБД и моделей, созданных в Erwin;
- ModelMart – среда для командной работы проектировщиков;
- Paradigm Plus – моделирование приложений и генерация объектного кода;
- Rational Rose – моделирование бизнес-процессов и компонентов приложений;
- Rational Suite AnalystStudio – пакет для аналитиков данных;
- Oracle Designer (входит в Oracle9i Developer Suite) – высокофункциональное средство проектирования программных систем и баз данных, реализующее технологию CASE и собственную методологию Oracle – CDM. Сложное CASE-средство, его имеет смысл использовать при ориентации на линейку продуктов Oracle.

4. Этапы создания ИС

Стандарт ISO/IEC 12207 определяет структуру жизненного цикла, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания информационной системы (рис.1.4):

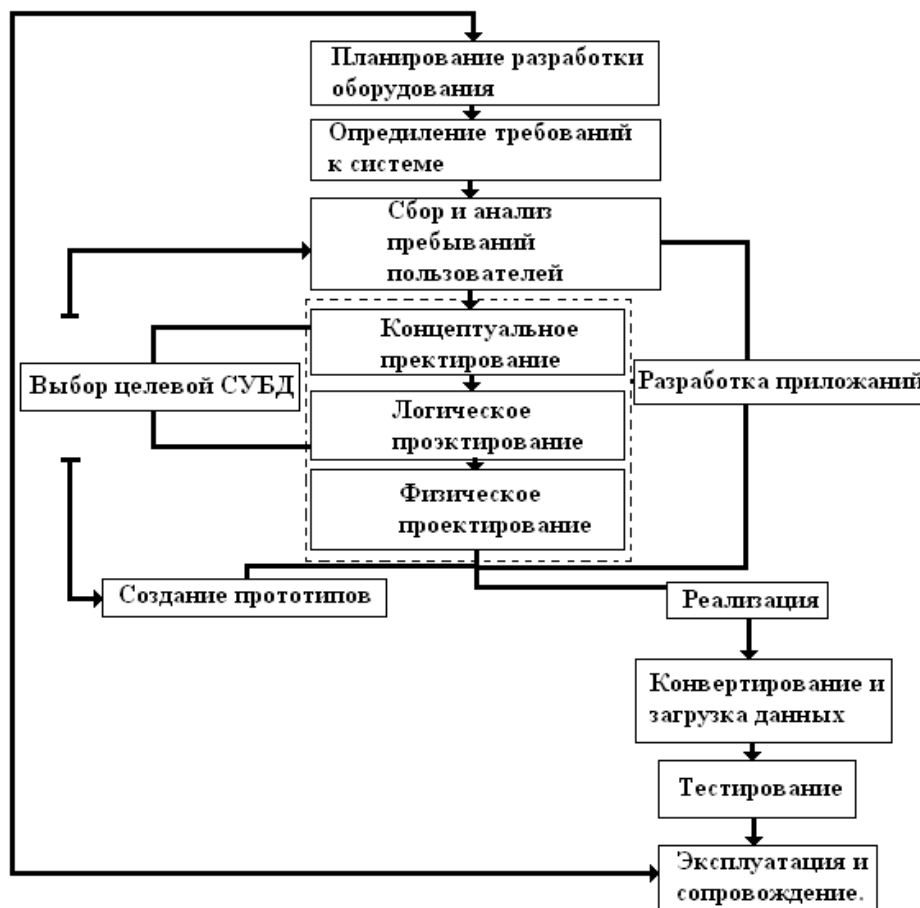


Рис.1.4. Структура жизненного цикла
Стадии создания АИС (ISO/IEC 15288)

Стадия	Описание
Формирование концепции	Анализ потребностей, выбор концепции и проектных решений
Разработка	Проектирование системы

Реализация	Изготовление системы
Эксплуатация	Ввод в эксплуатацию и использование системы
Поддержка	Обеспечение функционирования системы
Снятие с эксплуатации	Прекращение использования, демонтаж, архивирование системы

Стадии создания системы согласно требованиям ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания.»:

<i>Стадия</i>	<i>Описание</i>
Формирование требований к АИС	<ul style="list-style-type: none"> · обследование объекта и обоснование необходимости создания АИС; · формирование требований пользователей к АИС; · оформление отчета о выполненной работе и тактико-технического задания на разработку.
Разработка концепции АИС:	<ul style="list-style-type: none"> · изучение объекта автоматизации; · проведение необходимых научно-исследовательских работ; · разработка вариантов концепции АИС, удовлетворяющих требованиям пользователей; · оформление отчета и утверждение концепции.
Техническое задание:	<ul style="list-style-type: none"> · разработка и утверждение технического задания на создание АИС
Эскизный проект:	<ul style="list-style-type: none"> · разработка предварительных проектных решений по системе и ее частям; · разработка эскизной документации на АИС и ее части.
Технический проект:	<ul style="list-style-type: none"> · разработка проектных решений по системе и ее частям; · разработка документации на АИС и ее части; · разработка и оформление документации на поставку комплектующих изделий; · разработка заданий на проектирование в смежных частях проекта.
Рабочая документация:	<ul style="list-style-type: none"> · разработка рабочей документации на АИС и ее части; · разработка и адаптация программ.
Ввод в действие:	<ul style="list-style-type: none"> · подготовка объекта автоматизации; подготовка персонала; · комплектация АИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями); · строительно-монтажные работы; пусконаладочные работы; · проведение предварительных испытаний; · проведение опытной эксплуатации; · проведение приемочных испытаний.
Сопровождение АИС:	<ul style="list-style-type: none"> · выполнение работ в соответствии с гарантийными обязательствами; · послегарантийное обслуживание.

Детализация ряда работ по этапам.

<i>Наименование этапа</i>	<i>Содержание работ</i>
Планирование разработки ИС	Формулирование целей создания ИС; Поиск и обоснование оптимальных методов (способов) ее организации в условиях конкретного предприятия, Определение границ применения ИС
Определение требований к составу и распределению информации (в том числе к организации баз данных)	Определение состава пользователей и установление задач для каждого из них в процессе проектирования и эксплуатации ИС; Выявление существующих рабочих процессов и документов
Разработка единого описания характеристик объектов	Сбор и анализ требований к описанию информационных объектов от всех потенциальных пользователей ИС
Разработка и исследование моделей проекта ИС как СУБД	Концептуальное, логическое и физическое моделирования баз данных, в том числе: Определение объектов базы данных; Определение связей, их мощностей и обязательности, атрибутов; Повторный анализ сущностей: связь между сущностью и предметной областью, рабочие процессы, влияющие на сущность, взаимодействие между сущностями, атрибуты, анализ доменов, выбор типа ограничений; Нормализация данных; Выбор архитектуры базы данных системы; Таблицы; Связи; Индексы; Запросы; Защита данных: уровни защиты; Отслеживание и регистрация событий в ИС.
Проектирование пользовательского интерфейса	Выбор модели интерфейса; Архитектура пользовательского интерфейса: поддержка рабочих процессов; Уровни пользовательского интерфейса; Критерии пользовательского интерфейса; Связь между сущностями и формами; Выбор элементов управления; Поддержка целостности данных; Отчеты; Поддержка пользователя
Обоснование и выбор программной системы для разработки ИС	Оценка ожидаемых затрат на разработку и эксплуатацию ИС в условиях предприятия
Разработка проекта (прототипа) ИС	Создание прототипа (модели ИС) средствами визуального проектирования (например, Access)
Разработка и реализация ИС	Разработка серверной и клиентской частей ИС как баз данных и прикладных программ
Загрузка данных	Заполнение базы данных информацией

Тестирование	Проверка работы ИС и устранение возникающих ошибок работы приложений
Эксплуатация и сопровождение	Разработка организационных мероприятий по внедрению ИС, наблюдение за работой системы и при необходимости внесение изменений в программные приложения.

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»

<i>Модель ЖЦ ИС</i>	<i>Достоинства</i>	<i>Недостатки</i>
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:

<i>№ п/п</i>	<i>Название стадии (этапа) работ</i>	<i>Содержание работ</i>	<i>Результат работ</i>	<i>Применяемые программные средства</i>
1				

Лабораторная работа №2. Разработка концептуальной модели и модели декомпозиции процесса (стандарт IDEF0)

Цель работы: получить навыки построения диаграмм прецедентов.

Задание:

1. Создание концептуальной модели
2. Создание диаграммы декомпозиции

Теоретический материал

Методология **IDEF0** предназначена для моделирования деятельности организации. На начальном этапе разработки проекта создается модель, предназначенная для описания существующих бизнес-процессов и технологических процессов на предприятии по принципу «AS - IS» («Как есть»), причем, что немаловажно, она представляет предприятие с позиции сотрудников, которые на нем работают и досконально знают все нюансы, в том числе и неформальные. AS-IS – «что мы делаем сегодня», перед тем, как перепрыгнуть на то, «что мы будем делать завтра».

Модель деятельности или, иначе говоря, **функциональная модель**. **Функциональная модель** рассматривает систему как набор *действий*, в котором каждое действие преобразует некоторый *объект* или набор *объектов*. Технология **IDEF 0** использует принцип **функциональной декомпозиции** систем (разбиение системы на фрагменты). **Принцип декомпозиции** означает, что функциональную модель следует строить по правилу «*сверху вниз*», от *общего* вида модели к *частным* моделям. Поэтому обычно функциональная модель решения задачи представляет собой совокупность *частных функциональных моделей*.

Функциональные модели выделяют действия посредством представления в виде специального элемента – *блока*. **Блок** – основной структурный элемент функциональной модели, графическим представлением которой является **диаграмма**. Наименование действия – **отглагольное существительное** или **глагол**. В результате декомпозиции модели создается совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Методология **IDEF 0** основана на четырех основных понятиях: функциональный блок (узел), интерфейсная дуга, декомпозиция, глоссарий.

Функциональный блок

Фундаментальным понятием технологии **IDEF 0** является понятие **функционального блока**. Он предназначен для представления определенного вида *деятельности (Activity)*, которая представляет собой некоторую конкретную **функцию** в рамках рассматриваемой системы. Эта функция в свою очередь означает некоторое действие (набор действий), имеющее фиксированную цель и приводящее к некоторому конечному результату.

Функциональный блок изображается прямоугольником, стороны которого имеют следующие значения:

- Верхняя сторона – управление.
- Нижняя сторона – механизм.

- Правая сторона – выход.
- Левая сторона – вход.

Функциональный блок имеет имя, которое задается в глагольном наклонении или отглагольным существительным. Взаимодействие между действием и окружающим его миром, в том числе и с другими действиями, отображается с помощью интерфейсных дуг (стрелок).

Интерфейсная дуга

Интерфейсная дуга отображает элемент системы, который или **обрабатывается** функциональным блоком, или **оказывает иное влияние** на деятельность (функцию), отображенную данным функциональным блоком. **Интерфейсная дуга** изображается в виде стрелки, которая обозначает **носителя**, обеспечивающего перенос данных или объектов от одной деятельности к другой. Стрелки описывают взаимодействие работ с внешним миром и между собой, представляют собой некую информацию и именуются **существительными**.

Имя стрелки указывает ее **роль** (совокупность возможных ролей обозначается – **ICOM**):

Вход функционального блока – **Input**.

Управление – **Control**.

Выход функционального блока – **Output**.

Механизм – **Mechanism**.

Вход (Input) – это материал или информация, которые используются или преобразуются работой для получения результата (выхода). Стрелки входа может не быть.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Оно влияет на работу, но не преобразуется работой. Стрелка рисуется как входящая в верхнюю грань работы. Каждый функциональный блок должен иметь как минимум одну стрелку управления.

Часто сложно определить, являются ли данные входом или управлением. Если данные в работе изменяются или перерабатываются, то это, скорее всего, вход, а если нет – управление. Если определить статус стрелки сложно, рекомендуется рисовать стрелку управления.

Выход (Output) – материал или информация, которые производятся работой. Обязательна хотя бы одна стрелка выхода, исходящая из правой грани работы.

Механизм исполнения (Mechanism) – ресурсы, которые выполняют работу (например, персонал, станки, устройства и т. п.). Стрелка рисуется как входящая в нижнюю грань работы. Стрелки механизма могут не изображаться. В общем виде функциональный блок показан на рис. 2.1.



Рис. 2.1 Функциональный блок

На рис. 2.1 все интерфейсные дуги показаны в виде поименованных стрелок. По требованиям стандарта каждый функциональный блок должен иметь по крайней мере один выход и одно управление, так как каждая задача (процесс) должны иметь хотя бы один выход и хотя бы одно правило ее решения. Интерфейсная дуга «Механизм» может не изображаться.

Из нескольких функциональных блоков, соединенных интерфейсными дугами требуемым образом, строится **функциональная модель**.

Следует обратить внимание на то, что стрелки могут разветвляться, осуществляя требуемые соединения функциональных блоков.

Входом функционального блока может быть не только *выход* другого функционального блока, но и его *управление* или даже *механизм*. В результате в функциональной модели могут быть различные, достаточно сложные и необычные процессы решения задач в информационной системе.

Создание диаграмм в технологии IDEF0

При разработке модели деятельности организации следует использовать три типа диаграмм:

- ***I тип диаграммы – Контекстная диаграмма*** (может быть только одна) – вершина древовидной структуры, которая представляет собой наиболее абстрактный уровень описания системы и ее взаимодействие с внешней средой. В ней определена ***контекстная функция***;

- ***II тип диаграммы – Диаграмма декомпозиции***.

Диаграммы декомпозиции предназначены для детализации работы и содержат *родственные работы*, т. е. *дочерние работы*, имеющие общую *родительскую работу*. Работы нижнего уровня – это то же самое, что работы верхнего уровня, но в более детальном изложении. Диаграммы создаются аналитиком для того, чтобы провести сеанс экспертизы, т. е. обсудить диаграмму со специалистом предметной области.

После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы *без замечаний* можно приступить к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели.

Необходимо установить число работ не более шести (3–6), иначе диаграмма плохо читается (перенасыщена). Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание.

В диаграмме декомпозиции слева вверху располагается работа наиболее важная и выполняемая первой. Последовательно вниз идут работы менее важные или выполняемые позже.

- ***III тип диаграммы – Диаграмма дерева узлов*** показывает иерархическую зависимость работ, но не взаимосвязи между работами (этих диаграмм может быть сколько угодно, т. к. дерево может быть построено на любую глубину и не обязательно с корня).

Технологический процесс IDEF0-моделирования:

CASE-средство BPWin имеет простой и понятный пользовательский интерфейс для построения требуемых функциональных моделей и сценариев. Он зависит от используемой технологии. На рисунке 2.2 показано окно BPWin (**Computer Associates BPWin**).

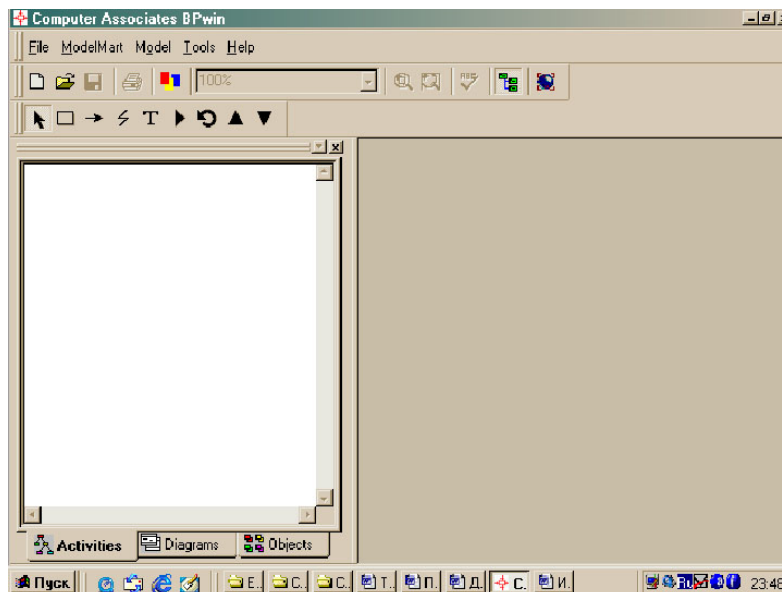















Рис. 2.2 Окно BPWin

Основная панель инструментов окна *Computer Associates BPwin* содержит следующие кнопки:

-  – создание новой модели,
-  – открытие имеющейся модели,
-  – сохранение построенной модели,
-  – печать модели,
-  – выбор масштаба,
-  – масштабирование,
-  – проверка правописания,
-  – включение/выключение навигатора модели,
-  – включение/выключение Model Mart.

Навигатор модели показывает состав модели по уровням разработки. С его помощью можно легко и быстро переходить с уровня на уровень. Работа с навигатором модели аналогична работе с Проводником системы Windows.

Панель специальных инструментов содержит следующие основные кнопки:

-  – редактирование функциональных блоков и стрелок,
-  – добавление функционального блока в модель,
-  – создание стрелок,
-  – переход на верхний уровень модели,

▼ – декомпозиция модели.

Окно модели является местом создания функциональной модели исследуемой системы. В нем строятся и редактируются функциональные блоки, рисуются и редактируются стрелки, осуществляется декомпозиция.

Подготовка модели

1. Нажать кнопку создания модели для вызова окна диалогового окна **BPWin** (рис. 2.3):

В диалоговом окне **BPWin** произвести следующие действия:

- выбрать **Business Process (IDEF0)**;
- задать имя модели и нажать кнопку **OK**;
- в окне **Properties for New Model** зафиксировать фамилию автора модели;
- нажать кнопку **OK**.

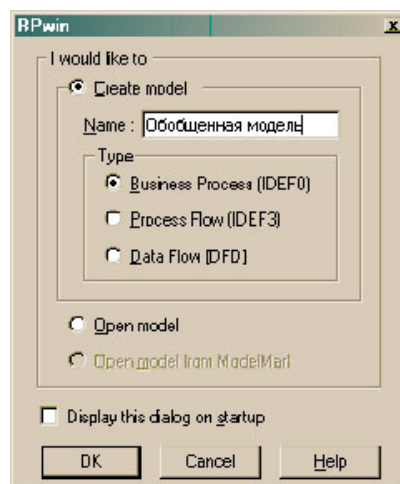


Рис. 2.3 Диалогового окна BPWin

2. Командой Model/Model Properties вызвать диалоговое окно Model Properties (рис. 2.4), в котором оформить свойства модели в соответствии с методическими рекомендациями.

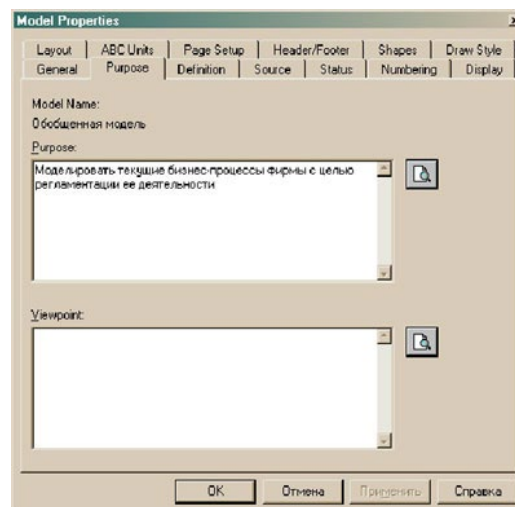


Рис. 2.4 Диалоговое окно Model Properties

Выполнение лабораторной работы

Первый уровень

1. Оформить функциональный блок в окне модели, выполнив следующие действия:

- в контекстном меню функционального блока выбрать команду **Name...**;
- в диалоговом окне **Activity Properties** (рис. 2.5) в закладке **Name** задать **имя** работы (краткое), помещаемой в данный функциональный блок, а в закладке **Definition** в поле **Definition** вписать достаточно подробное **описание** работы;
- в закладке **Font** задать шрифт **Arial Cyr** и установить флажки, позволяющие использовать этот шрифт во всех функциональных блоках диаграммы (**All activities in this diagram, All activities in this model** и **Change all occurrences of this font in the model**), после чего нажать кнопку **ОК**.

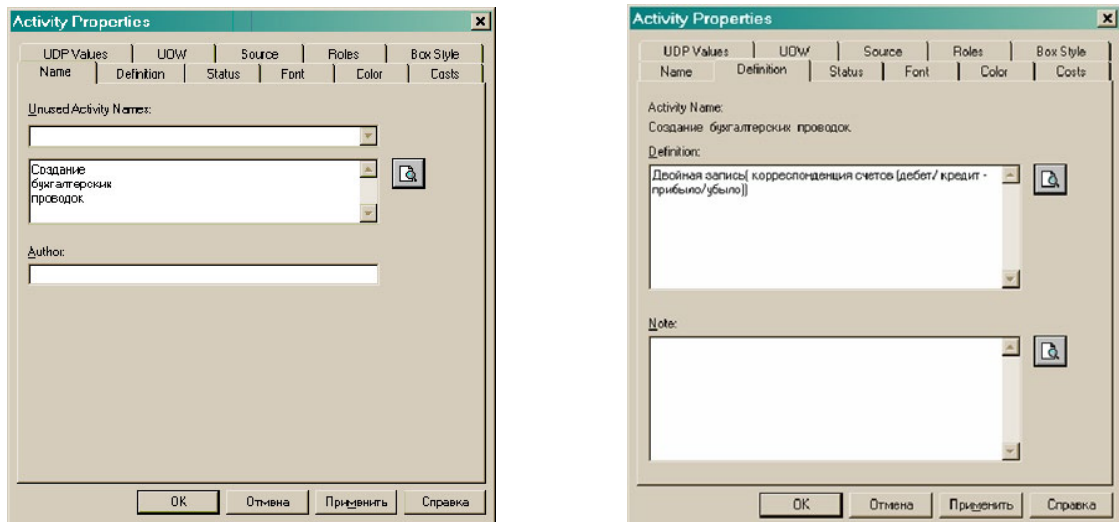


Рис. 2.5 Диалоговое окно Activity Properties

2. Оформить стрелку Вход, выполнив следующие действия:

- нажать кнопку создания стрелки (Precedence Arrow Tool – [→]);
- дважды щелкнуть на левой границе окна модели, а затем щелкнуть на левой границе функционального блока (рис. 2.6);
- нажать кнопку редактирования стрелок (Pointer Tool – [↖]);
- в контекстном меню стрелки выбрать команду **Name...**;

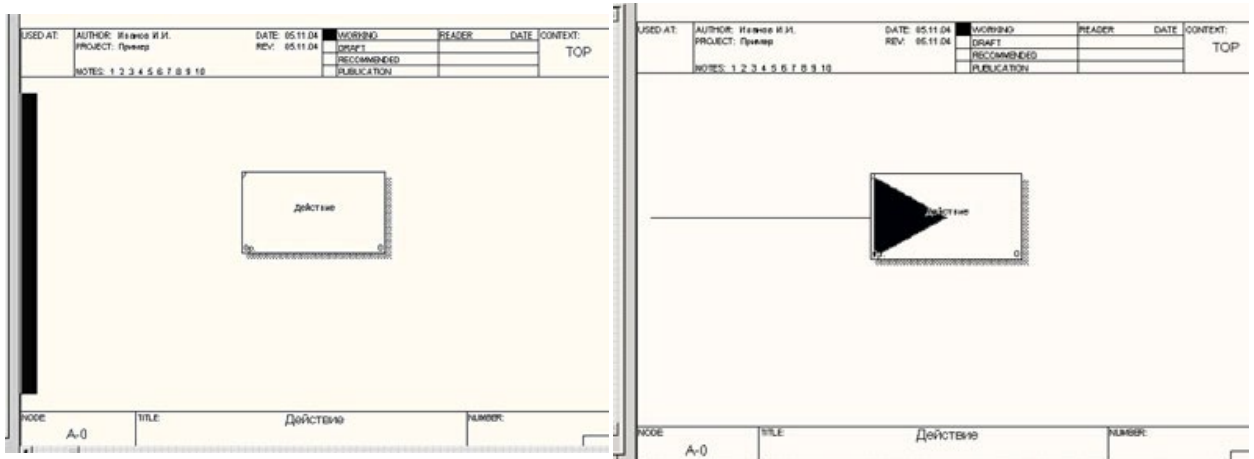


Рис. 2.6 Оформление стрелки Вход

- в диалоговом окне Arrow Properties (рис. 2.7), в закладке Name задать имя стрелки (краткое), а в закладке Definition в поле Definition вписать достаточно подробное описание назначения этой стрелки;

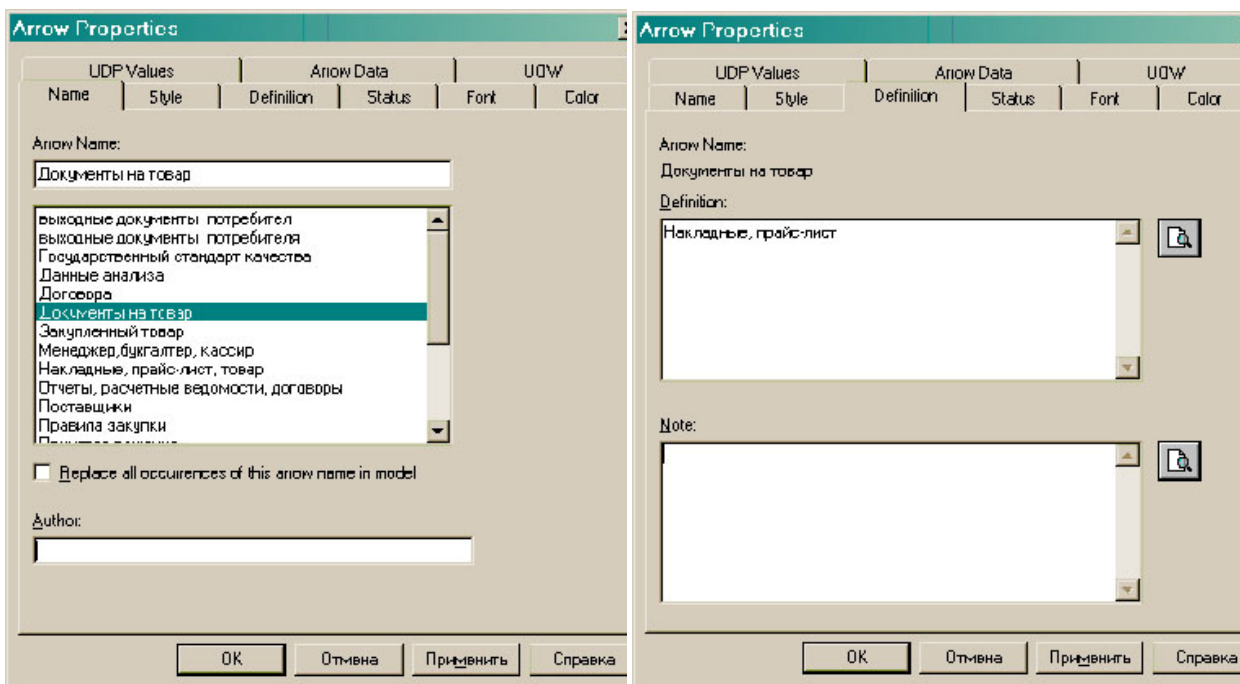


Рис. 2.7 Диалоговое окно Arrow Properties

- в контекстном меню стрелки выбрать команду Font...;
- в диалоговом окне Arrow Properties (рис. 2.8), в закладке Font задать шрифт Arial Cyr и установить флажки, позволяющие использовать этот шрифт для всех стрелок диаграммы (All Arrows in this diagram, All Arrows *in this model*, All instances of this Arrow и Change all occurrences of this font in the model);

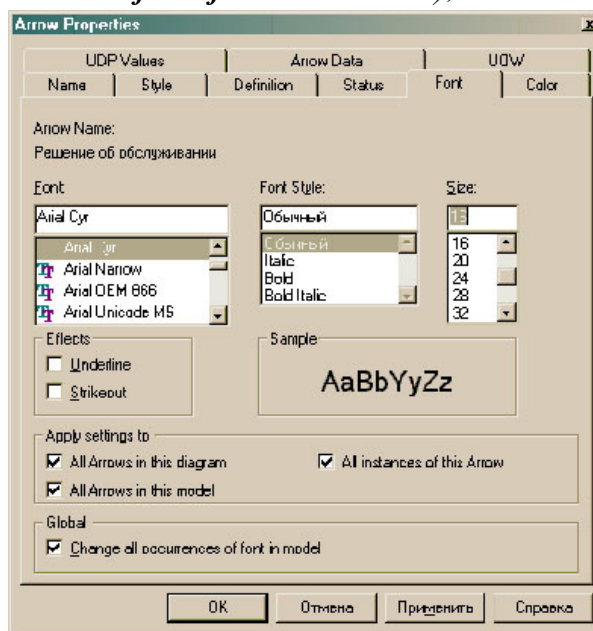


Рис. 2.8 Диалоговое окно Arrow Properties

3. Оформить стрелку **Выход**, для чего повторить п. 2, заменив *левые* границы *правыми*;
4. Оформить стрелку **Управление**, для чего повторить п. 2, заменив *левые* границы *верхними*;
5. Оформить стрелку **Механизм**, для чего повторить п. 2, заменив *левые* границы *нижними*.

Второй уровень

1. Перейти на нижний уровень моделирования кнопкой декомпозиции модели **[▼]**.

В диалоговом окне *Activity Box Count* указать:

- тип модели – **IDEF0**;
- число функциональных блоков нижнего уровня (от трех до шести, иначе диаграмма станет перегруженной и будет трудно читаться).

2. Оформить наследованные с первого уровня стрелки, выполнив следующие действия:

- нажать кнопку создания стрелки;
- щелкнуть по наконечнику стрелки **Вход (Управление, Механизм)**;
- щелкнуть по соответствующей границе требуемого функционального блока.

3. Оформить внутренние стрелки, выполнив следующие действия:

- нажать кнопку создания стрелки;
- щелкнуть по правой границе функционального блока;
- щелкнуть по соответствующей границе связанного функционального блока.

4. Создать разветвления стрелок, выполнив следующие действия:

- нажать кнопку редактирования стрелки;
- щелкнуть по фрагменту стрелки;
- щелкнуть по требуемой границе функционального блока.

5. Создать слияние стрелок, выполнив следующие действия:

- нажать кнопку редактирования стрелки;
- щелкнуть по границе функционального блока;
- щелкнуть по фрагменту стрелки;
- повторить п.п. 2–5 для всех функциональных блоков уровня.

Задание

Задание на выполнение лабораторной работы выдает преподаватель

Лабораторная работа №3. Создание диаграммы прецедентов

Цель работы: получить навыки построения диаграмм прецедентов.

Задание:

1. создать главную диаграмму прецедентов, задав на ней варианты использования и актеров;
2. добавить отношения между актерами и вариантами использования;
3. создать дополнительную диаграмму прецедентов;
4. добавить описания к актерам и вариантам использования;
5. для каждого варианта использования задать поток событий в виде отдельного файла и прикрепить его к варианту использования.

Содержание отчета:

1. созданные диаграммы прецедентов;
2. краткое описание каждого актера и прецедента;
3. описание потока событий для каждого варианта использования.

Пример выполнения работы

1. Предварительные действия - создание новой модели

Примечание: В этом и последующих примерах мы будем проектировать систему для предметной области "Предприятие по сборке и продаже компьютеров".

Работа с Rational Rose начинается с создания модели. При запуске программы автоматически появляется диалоговое окно для создания модели (рис. 3.1). Для его вызова также можно выбрать пункт меню File -> New. Во вкладке *New* выбираем необходимый каркас (в нашем случае *J2EE*). Во всех лабораторных работах мы будем создавать диаграммы для созданной модели.



Рис. 3.1. Окно создания новой модели

2. Создание главной диаграммы прецедентов

По умолчанию в представлении Вариантов Использования браузера уже существует главная диаграмма прецедентов (*Main*) (рис. 3.2). Для ее заполнения необходимо открыть ее, дважды щелкнув по ней, и разместить на ней актеров и варианты использования.

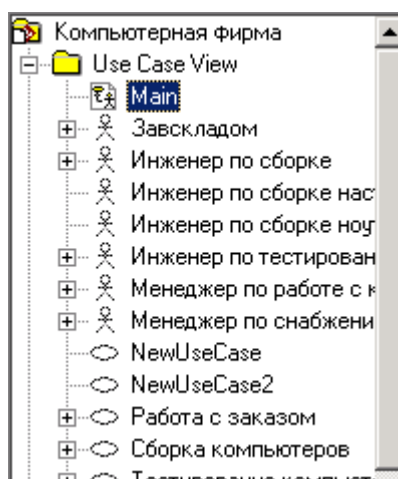


Рис. 3.2. Представление Вариантов Использования

Для нашей предметной области мы выделили следующих актеров:

Актер	Краткое описание
Менеджер по работе с клиентами	Сотрудник, который общается с заказчиком и работает с заказом
Менеджер по снабжению	Сотрудник, который занимается закупкой необходимых комплектующих
Инженер по сборке настольных компьютеров	Сотрудник, который занимается сборкой настольных компьютеров
Инженер по сборке ноутбуков	Сотрудник, который занимается сборкой ноутбуков
Инженер по тестированию	Сотрудник, который занимается тестированием собранных компьютеров
Завскладом	Сотрудник, который заведует складом комплектующих

Рассмотрим теперь, какие возможности должна предоставлять наша система:

- актер *Менеджер по работе с клиентами* использует систему для оформления, редактирования заказов и управления информацией о клиентах предприятия;
- актер *Менеджер по снабжению* использует систему для просмотра перечня необходимых для закупки комплектующих и ведения информации о снабжении;
- актер *Инженер по сборке настольных компьютеров* использует систему для просмотра нарядов на сборку персональных компьютеров, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- актер *Инженер по сборке ноутбуков* использует систему для просмотра нарядов на сборку ноутбуков, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- актер *Инженер по тестированию* использует систему для просмотра нарядов на тестирование собранной продукции и отметки о ходе выполнения работы;

- актер *Завскладом* использует систему для учета поступления и выдачи комплектующих.

На основании вышеизложенного можно выделить следующие прецеденты:

Прецедент	Краткое описание
Работа с заказом	Запускается менеджером по работе с клиентами. Позволяет вносить, изменять, удалять или просматривать заказ.
Управление информацией о клиенте	Запускается менеджером по работе с клиентами. Позволяет добавлять, изменять или удалять клиентов, а также просматривать информацию о клиентах.
Управление информацией о поставщиках	Запускается менеджером по снабжению. Позволяет добавлять, изменять или удалять поставщиков, а также просматривать информацию о поставщиках.
Управление информацией о комплектующих	Запускается менеджером по снабжению. Позволяет просматривать информацию о комплектующих, производить анализ их расходования, прогнозировать необходимое их количество и делать заказ.
Сборка компьютеров	Запускается инженером по сборке. Позволяет просматривать наряды на сборку компьютеров и делать отметки о ходе выполнения работы.
Требование необходимых комплектующих	Запускается инженером по сборке. Предназначено для затребования необходимых комплектующие со склада.
Тестирование компьютеров	Запускается инженером по тестированию. Позволяет просмотреть список компьютеров, подлежащих тестированию и сделать отметки о ходе выполнения работ.
Учет поступления и выдачи комплектующих	Запускается завскладом. Позволяет вести учет поступления и выдачи запчастей и комплектующих.

Созданная главная диаграмма прецедентов показана на рис. 3.3:

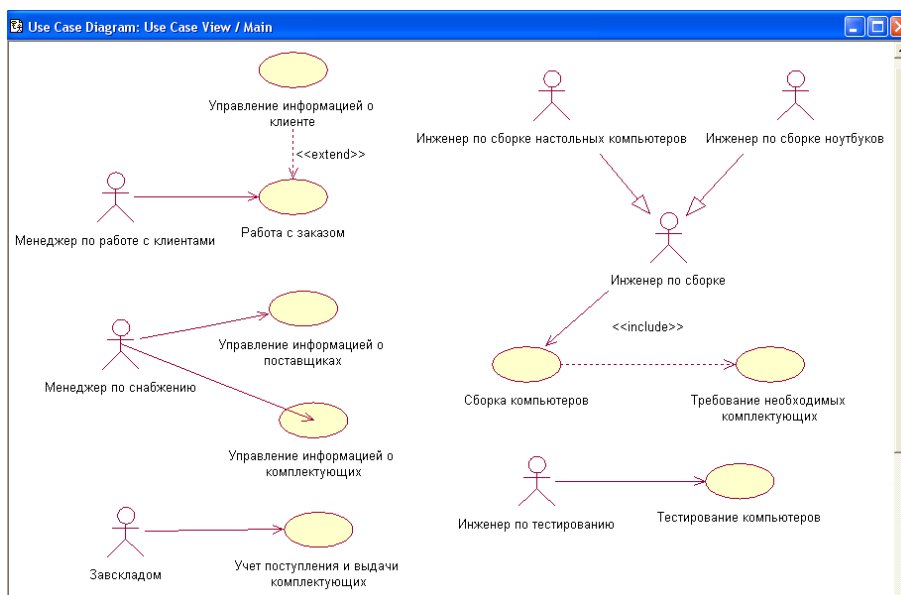


Рис. 3.3. Главная диаграмма прецедентов

Рассмотрим теперь отношения между актерами и прецедентами. В языке UML возможен только один тип отношений между актером и прецедентом - отношение коммуникации. Поэтому всех актеров мы связали с прецедентами отношением *Unidirectional Association*. Поскольку другой тип отношений здесь мы задать не можем, то стереотип *communicate* можно не указывать (он неявно подразумевается).

Для прецедента *Сборка компьютеров* не имеет значение какой именно актер будет с ним взаимодействовать - *Инженер по сборке настольных компьютеров* или *Инженер по сборке ноутбуков*. Поэтому мы ввели еще одного актера - *Инженер по сборке*, с которым связали первых двух актеров отношением обобщения (*Generalization*).

Отношение между прецедентами *Работа с заказом* и *Управление информацией о клиенте* - отношение расширения, поскольку когда актер *Менеджер по работе с клиентами* работает с заказом (оформляет, меняет и т.д.), то не всегда при этом он управляет информацией о клиентах.

Отношение между прецедентами *Сборка компьютеров* и *Требование необходимых комплектующих* - отношение включения, поскольку для сборки компьютеров обязательно нужно заказывать необходимые комплектующие со склада.

3. Поток событий для прецедентов главной диаграммы прецедентов

Потоки событий для прецедентов будем описывать по следующему шаблону:

- X.1 предусловия;
- X.2 главный поток;
- X.3 под-потоки;
- X.4 альтернативные потоки;
- X.5 постусловия.

где X - число от единицы до количества прецедентов.

4. Создание дополнительной диаграммы прецедентов

Как видно из описания потока событий для всех прецедентов каждый из них включает проверку пользователя. Проверка осуществляется единообразно для любого прецедента. Поэтому ее можно представить в виде отдельного прецедента *Аутентификация пользователя*, связанного отношением включения со всеми остальными. Результат создания диаграммы показан на рис. 3.4:

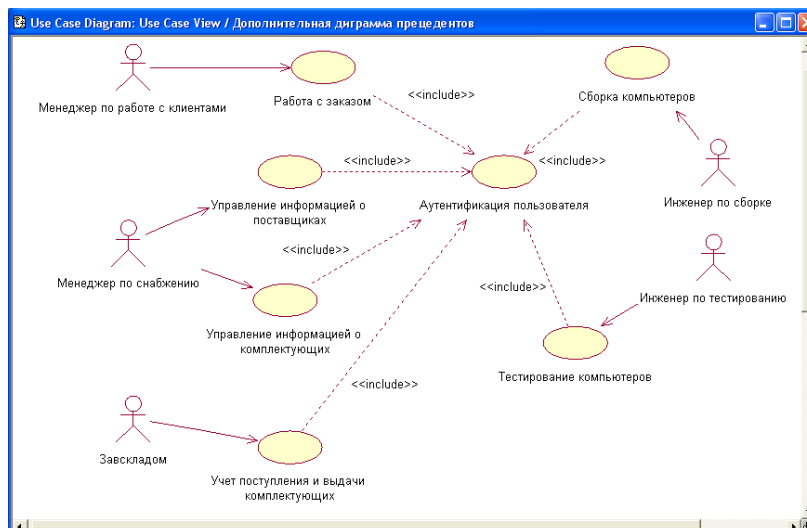


Рис. 3.4. Дополнительная диаграмма прецедентов

Примеры выполнения

1. Поток событий для прецедента «Работа с заказом»

1.1 Предусловия

Если заказ оформляется для нового клиента, то под-поток *добавить нового клиента* (Add a New Client) прецедента *Управление информацией о клиенте* должен быть выполнен перед его началом.

1.2 Главный поток

Прецедент начинает выполняться, когда менеджер подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить* (Add), *изменить* (Change), *удалить* (Delete), *просмотреть* (View) или *выйти* (Exit).

Если выбрана операция *добавить* (Add), S-1: выполняется поток *добавить новый заказ* (Add a New Order).

Если выбрана операция *изменить* (Change), S-2: выполняется поток *изменить заказ* (Change Order).

Если выбрана операция *удалить* (Delete), S-3: выполняется поток *удалить заказ* (Delete Order).

Если выбрана операция *просмотреть* (View), S-4: выполняется поток *просмотреть заказ* (View Order).

Если выбрана операция *выйти* (Exit) прецедент завершается.

1.3 Под-потоки

S-1: *добавить новый заказ* (Add a New Order)

Система отображает диалоговое окно, содержащее поле, в котором менеджер должен выбрать тип компьютера (настольный или ноутбук). Пользователь выбирает

необходимый тип. Система отображает поле для выбора клиента и список возможных комплектующих для выбранного типа компьютера, в котором менеджер отмечает выбранные клиентом комплектующие. Менеджер заполняет поля (E-2). Система запоминает введенные данные и распечатывает счет для оплаты. Затем прецедент начинается сначала.

S-2: *изменить заказ (Change Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система отображает информацию о данном заказе. Менеджер делает необходимые изменения (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-3: *удалить заказ (Delete Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система удаляет выбранный заказ (E-4). Затем прецедент начинается сначала.

S-4: *просмотреть заказ (View Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система отображает информацию о выбранном заказе. Когда менеджер просмотрит информацию, прецедент начнется сначала.

1.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: выбраны не все комплектующие, необходимые для сборки компьютера или комплектующих нет в наличии. Менеджер должен изменить состав компьютера или завершить прецедент.

E-3: введен неправильный номер заказа. Менеджер должен повторить ввод или завершить прецедент.

E-4: система не может удалить заказ. Информация сохраняется, система удалит заказ позже. Выполнение прецедента продолжается.

2. Поток событий для прецедента «Управление информацией о клиенте»

2.1 Предусловия

2.2 Главный поток

Прецедент начинает выполняться, когда менеджер подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить (Add)*, *изменить (Change)*, *удалить (Delete)*, *просмотреть (View)* или *выйти (Exit)*.

Если выбрана операция *добавить (Add)*, S-1: выполняется поток *добавить нового клиента (Add a New Client)*.

Если выбрана операция *изменить (Change)*, S-2: выполняется поток *изменить данные о клиенте (Change Client Data)*.

Если выбрана операция *удалить (Delete)*, S-3: выполняется поток *удалить клиента (Delete Client)*.

Если выбрана операция *просмотреть (View)*, S-4: выполняется поток *просмотреть данные о клиенте (View Client Data)*.

Если выбрана операция *выйти (Exit)* прецедент завершается.

2.3 Под-потоки

S-1: *добавить нового клиента (Add a New Client)*

Система отображает диалоговое окно, содержащее поля для ввода данных о новом клиенте. Пользователь заполняет поля (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-2: *изменить данные о клиенте (Change Client Data)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-3). Система отображает информацию о данном клиенте. Менеджер делает необходимые изменения (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-3: *удалить клиента (Delete Client)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-2). Система удаляет выбранного клиента (E-4). Затем прецедент начинается сначала.

S-4: *просмотреть данные о клиенте (View Client Data)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-3). Система отображает информацию о выбранном клиенте. Когда менеджер просмотрит информацию, прецедент начнется сначала.

2.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: заполнены не все поля. Менеджер должен заполнить незаполненные поля или завершить прецедент.

E-3: введен неправильный номер клиента. Менеджер должен повторить ввод или завершить прецедент.

E-4: система не может удалить клиента. Информация сохраняется, система удалит клиента позже. Выполнение прецедента продолжается.

3. Поток событий для прецедента «Учет поступления и выдачи комплектующих»

3.1 Предусловия

3.2 Главный поток

Прецедент начинает выполняться, когда завскладом подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить (Add)*, *отметить (Mark)* или *выйти (Exit)*.

Если выбрана операция *добавить (Add)*, S-1: выполняется поток *внести поступившие комплектующие (Add a New Components)*.

Если выбрана операция *отметить (Mark)*, S-2: выполняется поток *сделать отметку о выдаче комплектующих (Mark Components)*.

Если выбрана операция *выйти (Exit)* прецедент завершается.

3.3 Под-потоки

S-1: *внести поступившие комплектующие (Add a New Components)*

Система отображает диалоговое окно, содержащее поля для ввода наименования комплектующих, их количества, поставщика. Завскладом заполняет указанные поля (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-2: *сделать отметку о выдаче комплектующих (Change Order)*

Система отображает список комплектующих, находящихся на складе. Завскладом напротив нужных комплектующих вводит количество выданных (E-3). Система запоминает введенные данные. Затем прецедент начинается сначала.

3.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: заполнены не все поля. Пользователь должен заполнить пропущенные поля или завершить прецедент.

E-3: указано количество выданных комплектующих, превышающее их количество на складе. Пользователь должен повторить ввод или завершить прецедент.

4. Поток событий для прецедента «Сборка компьютеров»

4.1 Предусловия

4.2 Главный поток

Прецедент начинает выполняться, когда инженер по сборке подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *просмотреть (View)*, *отметить (Mark)* или *выйти (Exit)*.

Если выбрана операция *просмотреть (View)*, S-1: выполняется поток *Просмотреть наряд на сборку компьютера (View an Make Computer Order)*.

Если выбрана операция *отметить (Mark)*, S-2: выполняется поток *сделать отметку о статусе собираемого компьютера по наряду (Mark Computer)*.

Если выбрана операция *выйти (Exit)* прецедент завершается.

4.3 Под-потоки

S-1: *Просмотреть наряд на сборку компьютера (View an Make Computer Order)*

Система отображает диалоговое окно, содержащее список нарядов и поле для ввода номера наряда. Инженер выбирает необходимый наряд из списка или вводит его номер в поле (E-2). Система отображает информацию о выбранном наряде. Когда инженер просмотрит информацию, прецедент начнется сначала.

S-2: *сделать отметку о статусе собираемого компьютера (Mark Computer)*

Система отображает диалоговое окно, содержащее список нарядов. Возле необходимого наряда инженер делает отметку о статусе компьютера по данному наряду. Инженер сохраняет изменения. Затем прецедент начинается сначала.

4.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: заполнены не все поля. Пользователь должен заполнить пропущенные поля или завершить прецедент.

E-3: введен неправильный номер наряда. Инженер должен повторить ввод или завершить прецедент.

5. Поток событий для прецедента «Требование необходимых комплектующих»

5.1 Предусловия

5.2 Главный поток

Прецедент начинает выполняться, когда инженер по сборке подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *просмотреть (View)*, *затребовать (Order)* или *выйти (Exit)*.

Если выбрана операция *просмотреть (View)*, S-1: выполняется поток *просмотреть затребованные комплектующие на складе (View Ordered Components on Warehouse)*.

Если выбрана операция *затребовать (Order)*, S-2: выполняется поток *затребовать необходимые комплектующие на складе (Order Required Components on Warehouse)*.

Если выбрана операция *выйти (Exit)* прецедент завершается.

5.3 Под-потоки

S-1: *Просмотреть затребованные комплектующие на складе (View Ordered Components on Warehouse)*

Система отображает следующую информацию обо всех сделанных заказах данным инженером по сборке: дата затребования, наименование комплектующих, их количество, заказ выполнен или нет. Когда инженер по сборке просмотрел список, он уведомляет систему. Прецедент начинается сначала.

S-2: *затребовать необходимые комплектующие на складе (Order Required Components on Warehouse)*

Система отображает диалоговое окно, содержащее поля для ввода списка необходимых комплектующих и их количества. Инженер по сборке заполняет его. Система запоминает введенные данные. Затем прецедент начинается сначала.

5.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

Описание потоков событий для прецедентов *Управление информацией о поставщиках* и *Управление информацией о комплектующих* аналогично описанию для прецедента *Управление информацией о клиенте*; для прецедента *Тестирование компьютеров* - прецеденту *Сборка компьютеров*.

Лабораторная работа №4. Создание диаграмм взаимодействия

Цель работы: получить навыки построения диаграмм последовательности и кооперации.

Задание: создать диаграмму последовательности и кооперации для одного из сценариев любого прецедента, созданного в лабораторной работе № 2.

Пример выполнения работы.

Создавать диаграммы взаимодействия будем для сценария "Добавить новый заказ" прецедента "Работа с заказом". В этом сценарии кроме основного потока существуют еще и альтернативные потоки. Хотя стандарт языка UML допускает ветвления на диаграммах последовательности и кооперации, мы, чтобы не загромождать наши диаграммы, ограничимся рассмотрением только случая, когда пользователь правильно вводит свой пароль, правильно заполняет необходимые поля и введенные данные без ошибок сохраняются в базе данных. В случае необходимости альтернативные потоки можно показать на дополнительных диаграммах последовательности и кооперации.

Диаграммы взаимодействия будем создавать в Логическом представлении браузера. Для того, чтобы отделить эти диаграммы от других (которые мы уже создали или создадим в дальнейшем), создадим вначале новый пакет в Логическом представлении - *Диаграммы взаимодействия*, в котором будут располагаться созданные далее диаграммы.

Построение любой диаграммы взаимодействия начинается с определения перечня объектов, которые будут участвовать во взаимодействии. Для выбранного сценария в лабораторной работе № 3 была разработана диаграмма классов. Экземпляры классов этой диаграммы и будут участниками диаграмм взаимодействия.

Примечание: Rational Rose позволяет, имея одну из двух типов диаграмм взаимодействия, создать вторую. Для этого необходимо открыть имеющуюся диаграмму взаимодействия и выбрать пункт меню Browse > Create Sequence (Collaboration) Diagram. Автоматически будет создана диаграмма второго типа с таким же именем, в том же пакете и с таким же содержимым, что и первая. Единственный недостаток этого приема - в созданной диаграмме элементы не будут автоматически выравниваться. Поэтому если исходная диаграмма достаточно большая, то в созданной диаграмме сложно будет разобраться, т.к. элементы могут налезать друг на друга.

В данной работе мы оба типа диаграмм взаимодействия будем строить с нуля.

Создание диаграммы последовательности для сценария "Добавить новый заказ" прецедента "Работа с заказом"

Для создания диаграммы последовательности необходимо щелкнуть правой кнопкой мыши по пакету Диаграммы взаимодействия и в появившемся меню выбрать пункт New > Sequence Diagram, ввести ее имя, после чего дважды щелкнуть по ней в браузере, чтобы открыть ее (рис. 4.1).

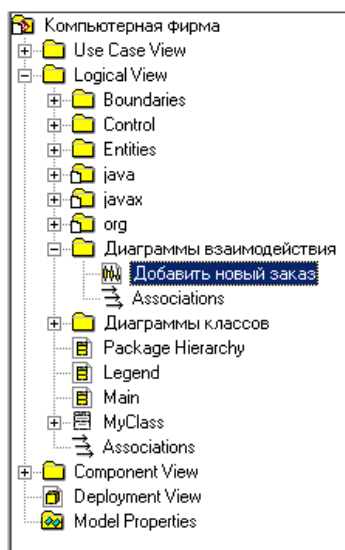


Рис. 4.1. Создание диаграммы последовательности

Построение диаграммы последовательности начинается с размещения на ней объектов, которые будут обмениваться сообщениями. Сначала необходимо разместить объекты, которые посылают сообщения, а потом объекты, получающие их. Инициатором взаимодействия выступает актер *Менеджер по работе с клиентами*. Поэтому на диаграмме он будет находиться в левом углу. Далее размещаем (рис. 4.2):

- объект класса *OrderOptions* (*Параметры работы с заказом*), отвечающий за выбор возможного действия с заказом в рассматриваемом прецеденте;
- объект класса *AddNewOrder* (*Добавление нового заказа*), отвечающий за добавление заказа;
- объект класса *OrderManager* (*Менеджер по работе с заказами*), отвечающий за обработку потока событий рассматриваемого прецедента;
- объект класса *Order* (*Заказ*);
- объект класса *Client* (*Клиент*);
- объект класса *ComponentPart* (*Комплектующее изделие*).

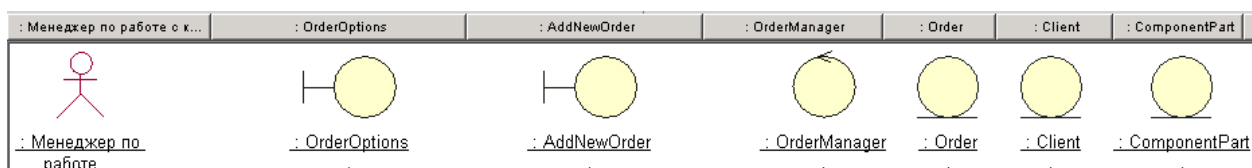


Рис. 4.2. Расположение объектов на диаграмме последовательности

Теперь на диаграмме следует разместить сообщения, которыми будут обмениваться объекты (таблица 4.1, рис.4.3):

Таблица 4.1

Номер сообщения	Объект - отправитель сообщения	Объект - получатель сообщения	Название

1	Менеджер по работе с клиентами	OrderOptions	ввод пароля
2	OrderOptions	OrderOptions	проверка пароля
3	Менеджер по работе с клиентами	OrderOptions	выбор операции "добавить"
4	OrderOptions	AddNewOrder	отображение полей ввода
5	Менеджер по работе с клиентами	AddNewOrder	выбор типа компьютера
6	AddNewOrder	OrderManager	получение списка клиентов
7	OrderManager	Client	получение списка клиентов
8	Client	AddNewOrder	список клиентов
9	AddNewOrder	AddNewOrder	отображение списка клиентов
10	Менеджер по работе с клиентами	AddNewOrder	выбор клиента
11	AddNewOrder	OrderManager	получение списка комплектующих
12	OrderManager	ComponentPart	получение списка комплектующих
13	ComponentPart	AddNewOrder	список комплектующих
14	AddNewOrder	AddNewOrder	отображение списка комплектующих
15	Менеджер по работе с клиентами	AddNewOrder	* выбор необходимых комплектующих
16	Менеджер по работе с клиентами	AddNewOrder	сохранить заказ
17	AddNewOrder	OrderManager	передача управления
18	OrderManager	Order	сохранить

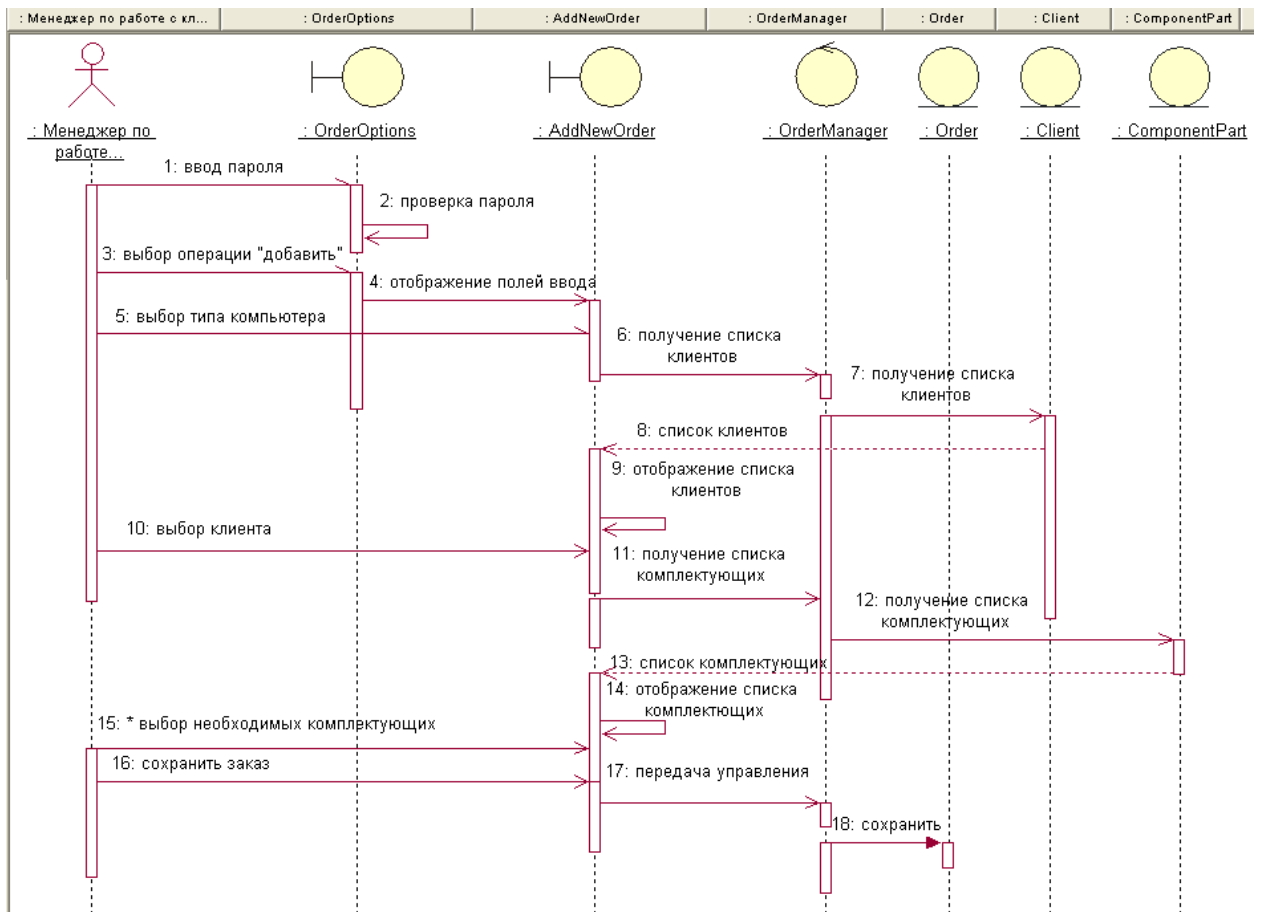


Рис. 4.3. Итоговая диаграмма последовательности

Для отображения номера сообщения в Rational Rose необходимо в меню Tools > Options > вкладка Diagram поставить галочку возле надписи Sequence numbering.

Лабораторная работа №5. Создание диаграммы классов

Цель работы: получить навыки построения диаграмм классов, создания пакетов и группировки классов в пакеты.

Задание:

1. создать диаграмму классов* для одного из сценариев диаграммы прецедентов, созданной в предыдущей лабораторной работе. Для каждого класса необходимо задать атрибуты и операции. Каждый класс должен быть подробно задокументирован - необходимо задать текстовое описание самого класса, описания его атрибутов и операций;
2. создать пакеты для группировки классов, созданных в пункте 1;
3. сгруппировать классы из пункта 1 в пакеты;
4. для каждого пакета создать свою диаграмму классов.
5. разработать главную диаграмму классов.

**Примечание: рассматривать диаграмму классов рекомендуется с концептуальной точки зрения, которая используется на начальных этапах моделирования и разработки.*

Содержание отчета:

1. созданные диаграммы классов (для диаграммы классов из пункта 2 задания должен быть указан сценарий, для которого данная диаграмма построена);
2. краткое описание каждого созданного класса и отношений между классами.

Диаграммы классов (class diagram) используются для моделирования статического вида системы с точки зрения проектирования. *Диаграмма классов* - диаграмма, на которой показано множество классов, интерфейсов, коопераций и отношений между ними. Используется в следующих целях:

- для моделирования словаря системы: предполагает принятие решения о том, какие абстракции являются частью системы, а какие - нет. С помощью диаграмм классов можно определить эти абстракции и их обязанности;
- для моделирования простых коопераций. Кооперация - это сообщество классов, интерфейсов и других элементов, работающих совместно для обеспечения некоторого кооперативного поведения;
- для моделирования логической схемы базы данных.

Согласно Мартину Фаулеру существуют три различные точки зрения на построение диаграмм классов или любой другой модели:

- концептуальная точка зрения - диаграммы классов служат для представления понятий изучаемой предметной области. Эти понятия будут соответствовать реализующим их классам, но прямое соответствие может отсутствовать. Концептуальная модель может иметь слабое отношение или вообще не иметь никакого отношения к реализующему ее программному обеспечению, поэтому ее можно рассматривать без привязки к какому-то языку программирования;
- точка зрения спецификации - рассматривается программная система, при этом рассматривается только ее интерфейсы, но не реализация;
- точка зрения реализации - классы диаграммы соответствуют реальным классам программной системы.

Пример выполнения работы

1. Создание диаграммы классов для сценария "Добавить новый заказ" прецедента "Работа с заказом"

Диаграммы классов будем рассматривать с концептуальной точки зрения. Для упрощения задачи и чтобы не загромождать диаграммы несущественными деталями методы setX, getX для каждого атрибута X классов задавать не будем. Создадим в Логическом представлении браузера новую диаграмму классов и назовем ее "Add New Order". В поле документации запишем для нее следующий текст: "Диаграмма классов для сценария "Добавить новый заказ" прецедента "Работа с заказом"

Заполнение диаграммы начнем с определения классов-сущностей. Рассматриваемый сценарий состоит из:

- самого заказа;
- клиента, который делает заказ;
- комплектующих изделий, которые входят в заказ.

Создадим классы-сущности Order (Заказ), Client (Клиент) и ComponentPart (Комплектующее изделие). Поскольку в один заказ может входить много разных комплектующих изделий, и одно комплектующее изделие может входить во много заказов, то введем еще один класс-сущность OrderItem (Состав заказа). Опишем каждый класс.

Класс *Client*:

Параметр	Значение
Комментарий	Класс, представляющий собой клиента фирмы
Атрибуты	name : String - наименование клиента address : String - адрес клиента phone : String - телефон клиента Все атрибуты имеют модификатор доступа - private
Операции	AddClient() - добавление нового клиента RemoveClient() - удаление существующего клиента GetInfo() - получить информацию о клиенте Все операции имеют модификатор доступа - public

Класс *Order*:

Параметр	Значение
Комментарий	Класс, представляющий собой заказ, который делает клиент
Атрибуты	orderNumber : Integer - номер заказа orderDate : Date - дата оформления заказа orderComplete : Date - дата выполнения заказа Все атрибуты имеют модификатор доступа - private
Операции	Create() - создание нового заказа SetInfo() - занести информацию о заказе

	GetInfo() - получить информацию о заказе Все операции имеют модификатор доступа - public
--	---

Класс *OrderItem*:

Параметр	Значение
Комментарий	Класс, представляющий собой пункт заказа, который делает клиент
Атрибуты	itemNumber : Integer - номер пункта заказа quantity : Integer - количество комплектующих изделий price : Double - цена за единицу Все атрибуты имеют модификатор доступа - private
Операции	Create() - создание новой строки заказа SetInfo() - занести информацию о строке заказа GetInfo() - получить информацию о строке заказа Все операции имеют модификатор доступа - public

Класс *ComponentPart*:

Параметр	Значение
Комментарий	Класс, представляющий собой комплектующие изделия
Атрибуты	name : String - наименование manufacturer : String - производитель price : Double - цена за единицу description - описание Все атрибуты имеют модификатор доступа - private
Операции	AddComponent() - добавление нового комплектующего изделия RemoveComponent() - удаление комплектующего изделия GetInfo() - получить информацию о комплектующем изделии Все операции имеют модификатор доступа - public

Результат создания классов-сущностей показан на рис. 5.1:

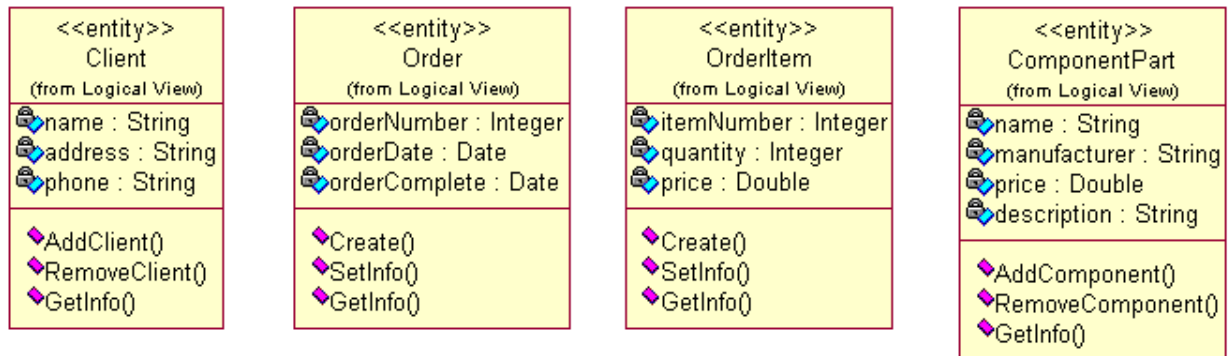


Рис. 5.1. Созданные классы-сущности

Добавим отношения между классами (рис. 5.2):

- класс Client и Order - отношение ассоциации, поскольку данные два класса просто связаны друг с другом и никакие другие типы связей здесь применить нельзя. Один клиент может сделать несколько заказов, каждый заказ поступает только от одного клиента, поэтому кратность связи со стороны класса Client - 1, со стороны Order - 1..n;
- класс Order и OrderItem - отношение композиции, поскольку строка заказа является частью заказа, и без него существовать не может. В один заказ может входить несколько строк заказа, строка заказа относится только к одному заказу, поэтому кратность связи со стороны Order - 1, со стороны OrderItem - 1..n;
- класс OrderItem и ComponentPart - отношение агрегации, поскольку комплектующие изделия являются частями строки заказа, но и те, и другие, являются самостоятельными классами. Одно комплектующее изделие может входить во много строк заказа, в одну строку заказа входит только одно комплектующее изделие, поэтому кратность связи со стороны ComponentPart - 1, со стороны OrderItem - 1..n.

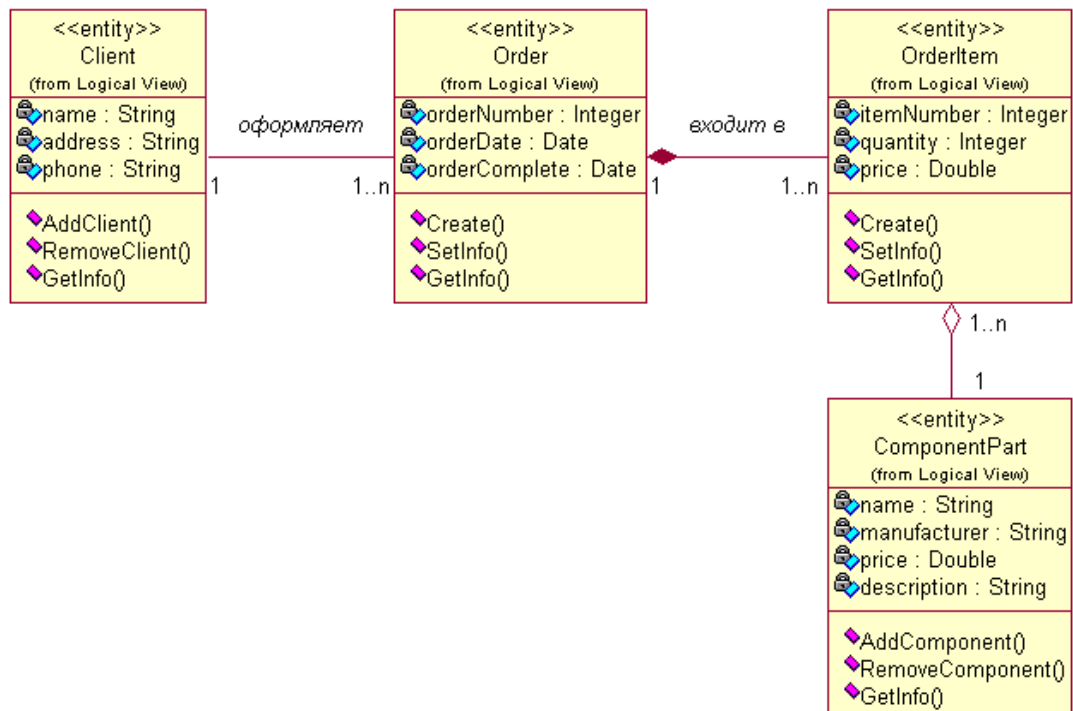


Рис. 5.2. Классы-сущности и отношения между ними

Добавим теперь на диаграмму граничные и управляющие классы (рис. 5.3). Рассматриваемый сценарий - это только одно из действий, которые обеспечивает прецедент "Работа с заказом".

Прецедент также позволяет просмотреть, отредактировать или удалить заказ. Это означает, что необходимо предусмотреть механизм, который позволяет выбирать необходимое действие.

Создадим для этого граничный класс `OrderOptions` (Параметры работы с заказом) с комментарием "Класс, обеспечивающий механизм работы с заказами".

Также создадим граничный класс `AddNewOrder` (Добавление нового заказа), который будет служить для добавления новых заказов (комментарий - "Класс служит для добавления новых заказов").

Отношение между этими классами - агрегация, поскольку в данном случае класс `AddNewOrder` рассматривается как часть класса `OrderOptions`, частями которого также будут классы для просмотра, редактирования и удаления заказов. Кратность связи 1 к 1, поскольку в состав класса `OrderOptions` входит только один класс `AddNewOrder`.

Перейдем теперь к управляющим классам. Добавим управляющий класс `OrderManager` (Менеджер по работе с заказами) с комментарием "Управляющий класс для обработки потока событий прецедента "Работа с заказами"", который будет обеспечивать обработку потока событий для рассматриваемого прецедента.

Данный класс будет связан с классами `AddNewOrder` и `Order`. Отношение между классами `AddNewOrder` и `OrderManager` - однонаправленная ассоциация с кратностью связи 1 к 1, поскольку один экземпляр класса `AddNewOrder` взаимодействует только с одним экземпляром класса `OrderManager`. Отношение между классами `OrderManager` и `Order` - однонаправленная ассоциация с кратностью связи 1 к 1..n, поскольку один класс `OrderManager` может взаимодействовать с несколькими классами `Order`.

Окончательный вариант диаграммы классов показан на рис. 5.3:

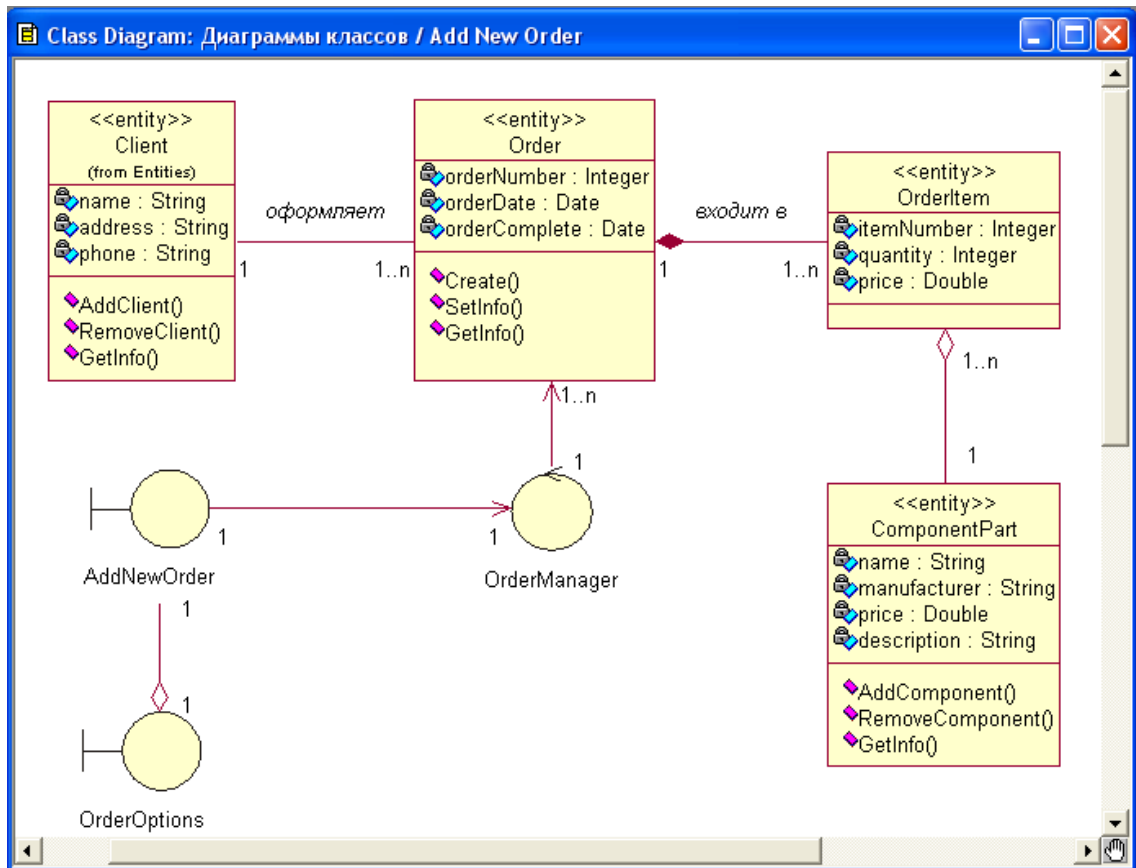


Рис. 5.3. Итоговая диаграмма классов

2. Создание пакетов

Пакеты предназначены для группировки элементов в группы по определенным критериям. В простейшем случае классы можно группировать по их стереотипам. Создадим три пакета: Entities (классы-сущности), Boundaries (граничные классы) и Control (управляющие классы). Для этого необходимо щелкнуть правой кнопкой мыши на Логическом представлении браузера (Logical View), в появившемся контекстном меню выбрать пункт New > Package (Создать > Пакет), и ввести имя пакета. Результат создания пакетов показан на рис.5.4:

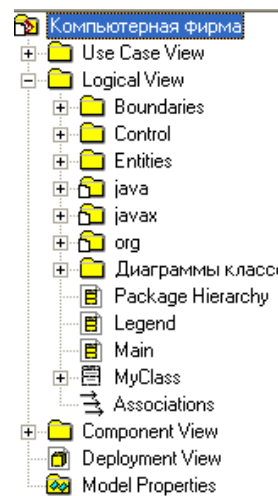


Рис. 5.4. Созданные пакеты

В поле документации (Documentation) для каждого пакета зададим комментарий:

- для пакета *Entities* комментарий: пакет содержит классы-сущности;
- для пакета *Boundaries* комментарий: пакет содержит граничные классы;
- для пакета *Control* комментарий: пакет содержит управляющие классы.

3. Группировка классов в пакеты

Группировка классов в пакеты осуществляется путем перетаскивания в Логическом представлении браузера соответствующего класса в соответствующий пакет. Группировать созданные классы будем следующим образом:

- классы *Client*, *Order*, *OrderItem* и *ComponentPart* перенесем в пакет *Entities*;
- классы *OrderOptions* и *AddNewOrder* перенесем в пакет *Boundary*;
- класс *OrderManager* перенесем в пакет *Control*.

Итоговой результат приведен на рис. 5.5.

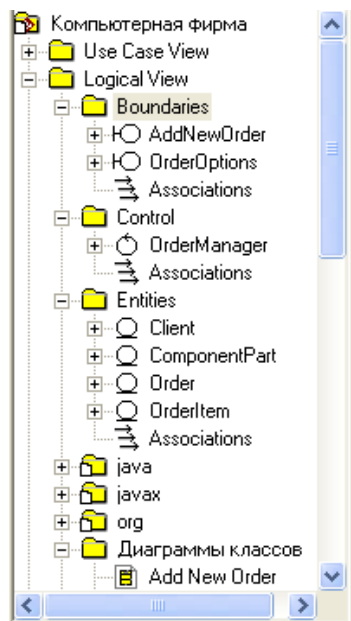


Рис. 5.5. Классы и пакеты для сценария "Добавление нового заказа"

4. Добавление диаграммы классов для каждого пакета

Для добавления диаграммы к пакету следует щелкнуть правой кнопкой мыши по пакету, в появившемся контекстном меню выбрать пункт *New > Class Diagram* (Создать > Диаграмма Классов), ввести имя класса *Main* (Главная), далее открыть диаграмму, дважды щелкнув по ней, и перенести на нее нужные классы. Отношения между классами, принадлежащие одному пакету, будут перенесены автоматически. Результат создания диаграммы классов для пакета *Boundaries* показан на рис.5.6, для пакета *Control* - на рис.5.7, для пакета *Entities* - на рис. 5.8.

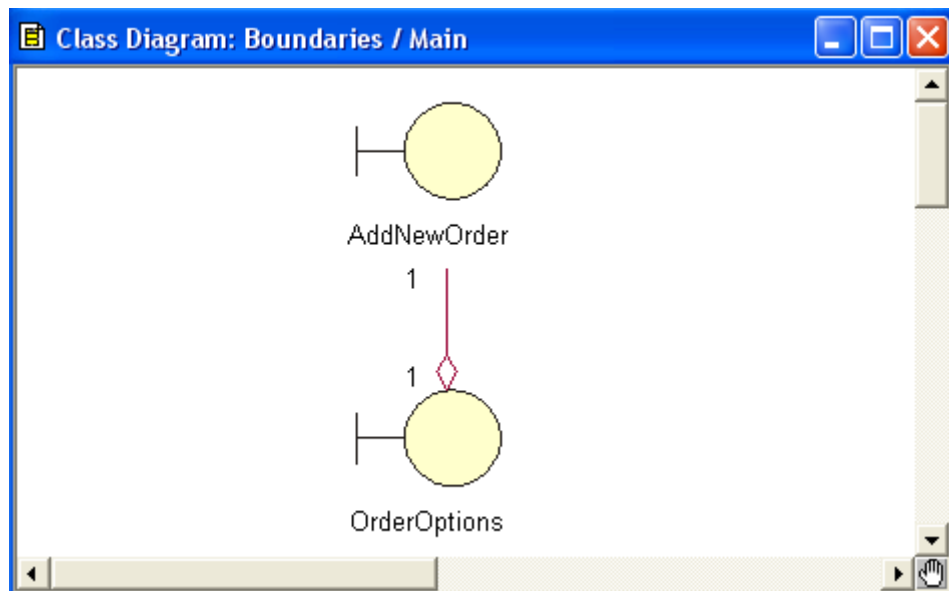


Рис. 5.6. Диаграмма классов пакета Boundaries

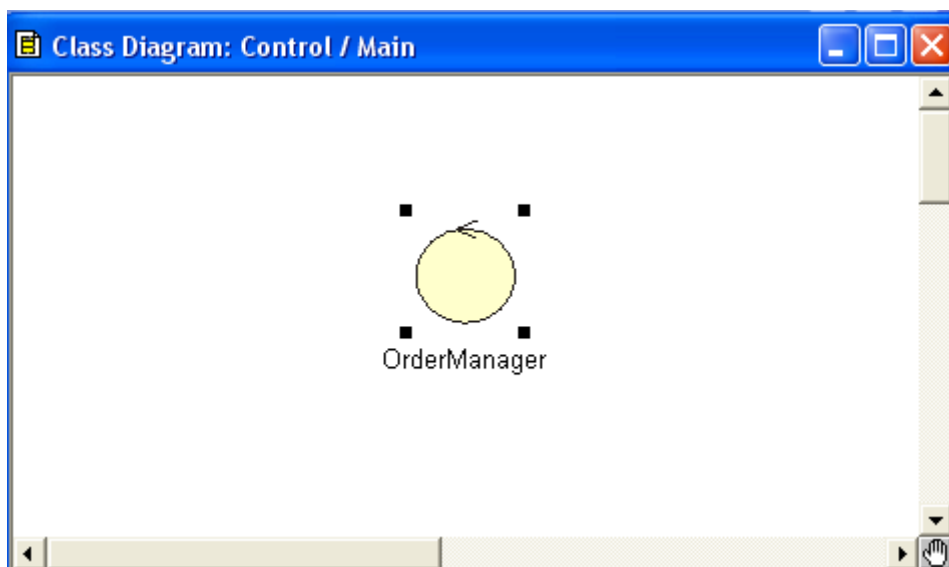


Рис. 5.7. Диаграмма классов пакета Control

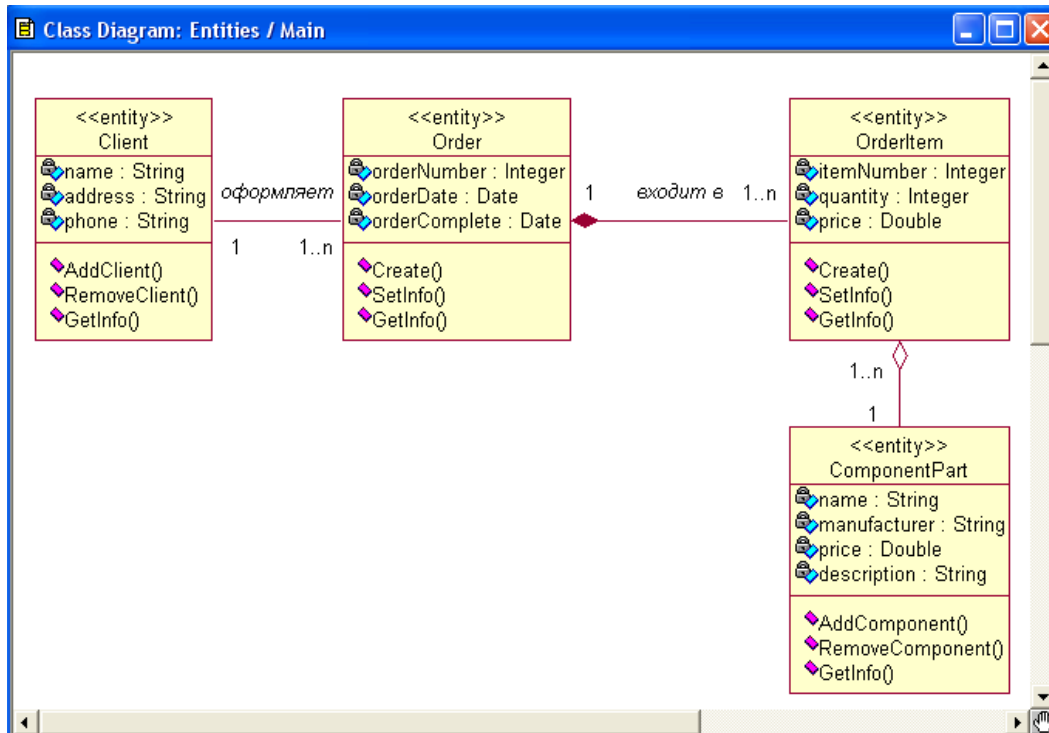


Рис. 5.8. Диаграмма классов пакета Entities

5. Создание главной диаграммы классов

Главная диаграмма в логическом представлении модели обычно отображает пакеты системы. По умолчанию в Логическом представлении браузера уже существует главная диаграмма классов (*Main*). Для ее заполнения необходимо открыть ее, дважды щелкнув по ней в Логическом представлении браузера, и перетащить на нее три созданные нами пакеты (рис.5.9):

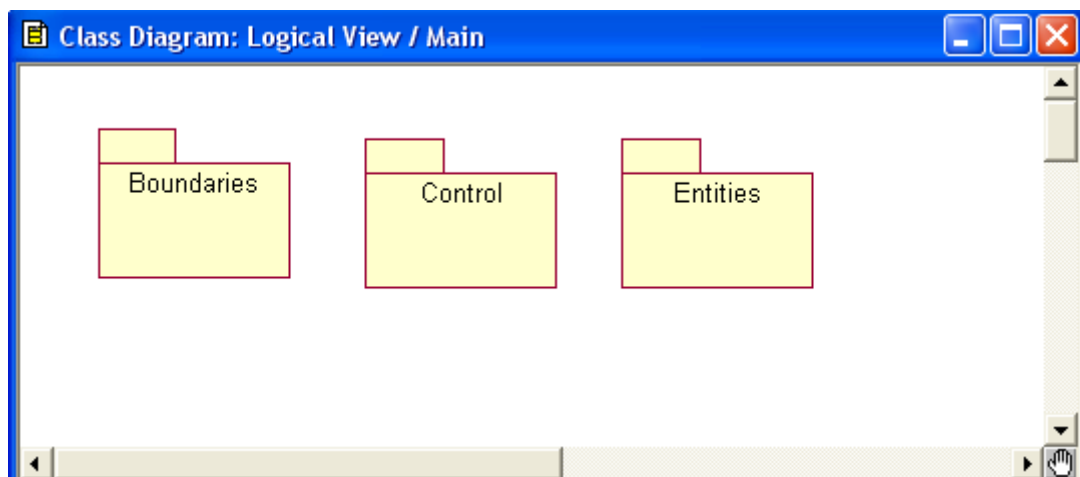


Рис. 5.9. Главная диаграмма классов

Лабораторная работа № 6. Создание диаграмм деятельности

Цель работы: получить навыки построения диаграмм деятельности.

Задание:

1. создать диаграмму деятельности, описывающую один из бизнес-процессов выбранной предметной области;
2. создать диаграмму деятельности, описывающую поток событий одного из вариантов использования, созданного в лабораторной работе № 2.

Содержание отчета: созданные диаграммы деятельности с указанием того, какой бизнес-процесс и поток событий какого варианта использования они описывают.

Пример выполнения работы

1. Создание диаграммы деятельности для бизнес-процесса предприятия по сборке компьютеров

Рассмотрим целом, что происходит на предприятии от момента оформления заказа на сборку компьютера до выдачи готового компьютера. После оформления заказа менеджер по работе с клиентами передает его менеджеру по сборке, который прежде чем начать сборку заказывает необходимые комплектующие со склада. На складе заведующий подбирает необходимые комплектующие (в случае их отсутствия заказывает их у менеджера по снабжению) и передает их инженеру по сборке. После получения комплектующих менеджер по сборке осуществляет сборку компьютера и передает его инженеру по тестированию. Если компьютер не прошел тестирование, он возвращается для повторной сборки. При успешном завершении тестирования компьютер передается на склад на хранение. Со склада компьютер по требованию передается инженеру по работе с клиентами, который оформляет на него документы и выдает клиенту.

Для создания диаграммы действий необходимо щелкнуть правой кнопкой мыши по Представлению Вариантов Исползования и в появившемся меню выбрать пункт New > Activity Diagram, ввести ее имя, после чего дважды щелкнуть по ней в браузере, чтобы открыть ее.

Результат построения диаграммы показан на рис. 6.1:

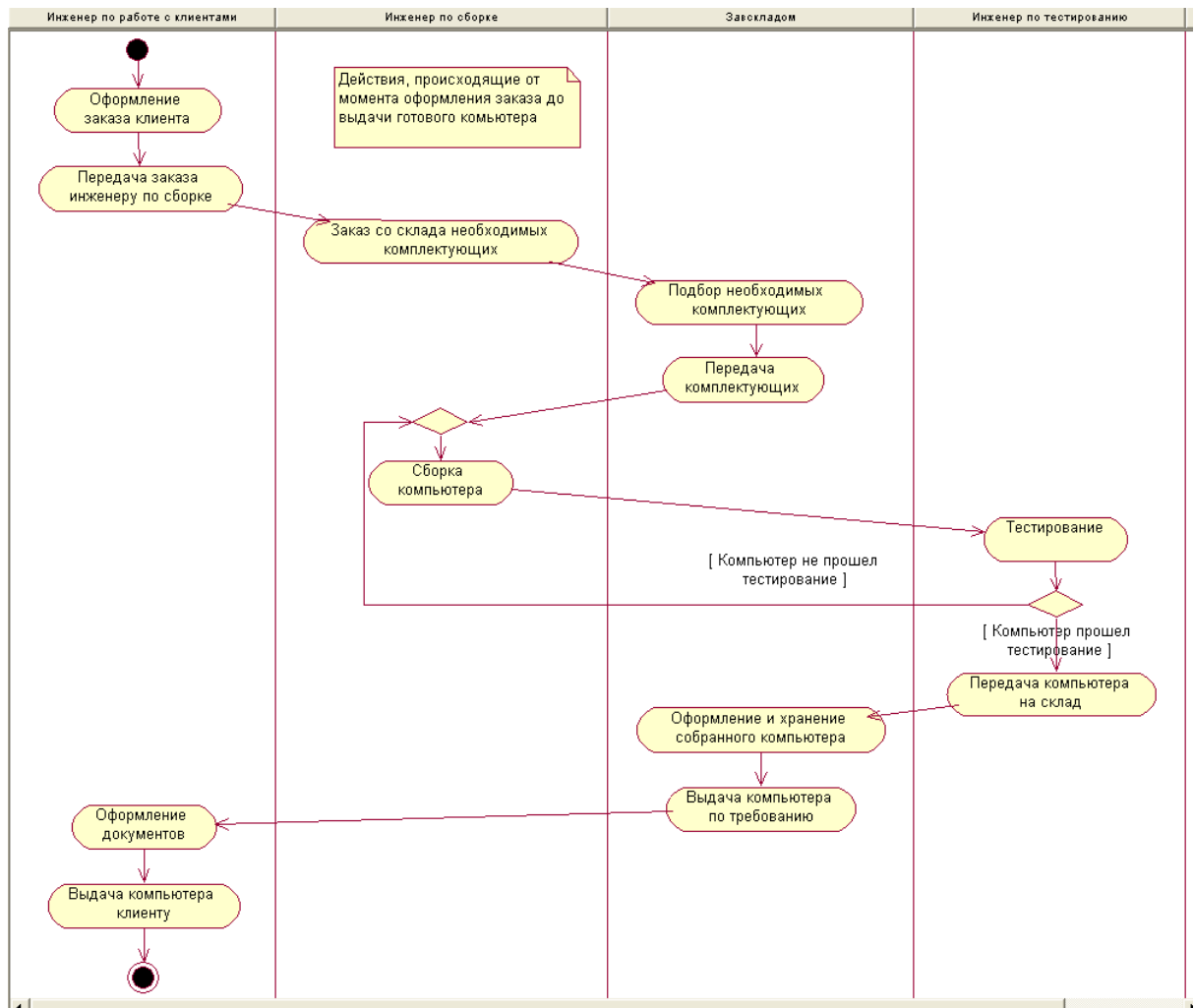


Рис. 6.1. Диаграмма деятельности бизнес-процесса

2. Создание диаграммы деятельности потока события варианта использования "Работа с заказом"

Поток событий варианта использования "Работа с заказом" состоит из главного потока, под-потоков и альтернативных потоков. Чтобы не загромождать диаграмму покажем поток событий на нескольких диаграммах деятельности. На первой из них (условно назовем ее главной) покажем действия для основного потока и связанный с ним альтернативный поток (рис. 6.2). Под-потоки можно будет показать путем декомпозиции соответствующего действия главной диаграммы.

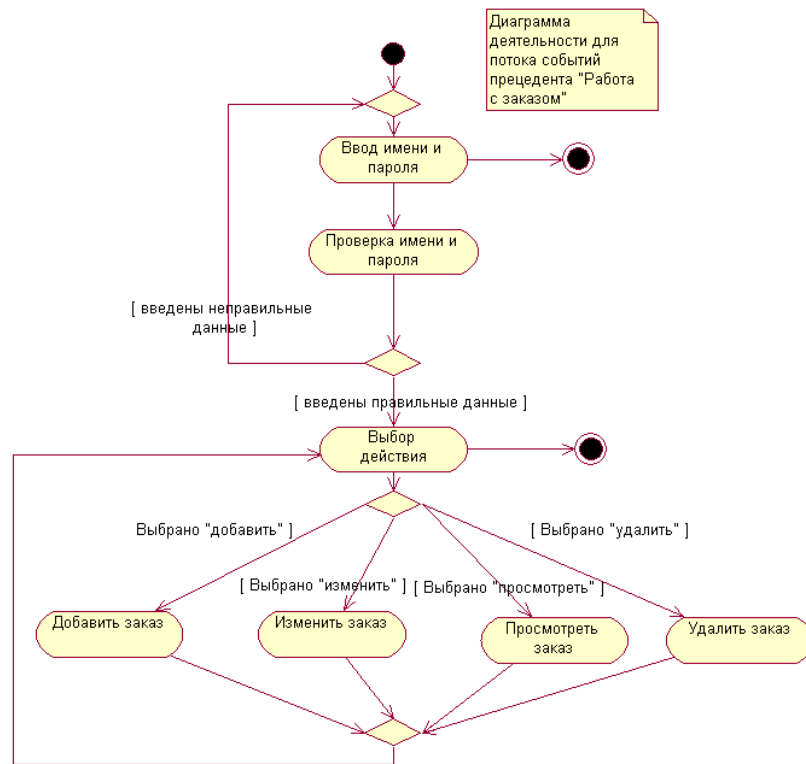


Рис. 6.2. Диаграмма деятельности для потока событий прецедента "Работа с заказом"

Для декомпозиции действия диаграммы деятельности следует щелкнуть по ней правой кнопкой мыши и в появившемся меню выбрать пункт Sub Diagrams > New Activity Diagram.

Пример декомпозиции действия На рис. 6.3 показана диаграмма деятельности для под-потока "Добавить заказ", которая является декомпозицией действия "Добавить заказ" главной диаграммы деятельности.

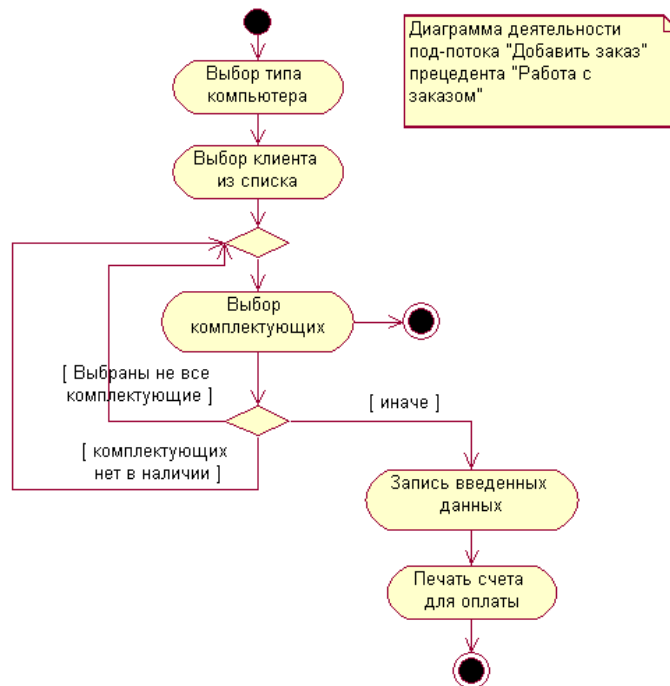


Рис. 6.3. Диаграмма деятельности для действия "Добавить заказ"

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Куклина, И. Г. Методы и средства проектирования информационных систем : учебное пособие / И. Г. Куклина, К. А. Сафонов. — Нижний Новгород : Нижегородский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. — 84 с.
2. Иванова, О. Г. Методы и средства проектирования информационных систем и технологий. Основы UML : учебное пособие / О. Г. Иванова, Ю. Ю. Громов. — Тамбов : Тамбовский государственный технический университет, ЭБС АСВ, 2020. — 80 с.
3. Цехановский, В. В. Проектирование информационных систем: архитектуры и платформы : учебное пособие / В. В. Цехановский, А. И. Водяхо. — Москва : Ай Пи Ар Медиа, 2023. — 240 с.
4. Грекул, В. И. Проектирование информационных систем : учебное пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 299 с.