РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Д.В. Глазунов, О.В. Игнатьева, С.Л. Никитченко,
З.В. Лященко, А.М. Лященко, А.А. Феденко

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ И САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

МДК.01.01 «Проектирование информационных ресурсов»

для специальности
09.02.09 Веб-разработка

Ростов-на-Дону
2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Методические указания к лабораторным работам и по выполнению самостоятельной работы студентов составлены в соответствии с ФГОС СПО и рабочей программой профессионального модуля МДК.01.01 «Проектирование информационных ресурсов», которые являются частью программы подготовки специалистов среднего звена специальности 09.02.09 Веб-разработка.

Рабочей программой дисциплины МДК.01.01 «Проектирование информационных ресурсов» предусмотрено на выполнение лабораторных работ – 40 часов и самостоятельной работы студентов – 26 часов. При выполнении лабораторных работ и при организации самостоятельной работы студентов используются активные и интерактивные формы обучения - просмотр и обсуждение учебных видеофильмов, групповая дискуссия, лекция - консультация, моделирование производственных процессов и ситуаций, обсуждение в группах, тренинг, кейс-метод, защита практических и лабораторных работ и другие.

Цель методических рекомендаций - оказание методической помощи студентам в выполнении лабораторных работ и в организации их самостоятельной работы по изучению учебного материала, для расширения, углубления и закрепления знаний и умений, а также формирования профессиональных (ПК) компетенций.

Код и содержание компетенции	Умения	Знания
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам	Уметь: применять математические модели, методы и средства проектирования информационных и автоматизированных систем	Знать: методологию и основные методы математического моделирования, классификацию и условия применения моделей, основные методы и средства проектирования информационных и автоматизированных систем, инструментальные средства моделирования и проектирования информационных и автоматизированных систем
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности	Уметь: применять на практике информационные технологии при проектировании информационных систем; применять унифицированный язык моделирования UML при проектировании информационных систем	Знать: порядок их применения и программное обеспечение в профессиональной деятельности в том числе с использованием цифровых средств
ПК 1.1 Проектировать информационные ресурсы.	Уметь: осуществлять проектирование программного обеспечения с использованием инструментальных средств проектирования и моделирования; читать схемы, чертежи, технологическую документацию	Знать: основные этапы, методологию, технологию и средства проектирования информационных ресурсов

1 ПЛАН РАСПРЕДЕЛЕНИЯ УЧЕБНОЙ НАГРУЗКИ

Объем дисциплины в академических часах с указанием количества академических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся

Вид учебной работы	**Объем часов**
Объем образовательной программы учебной дисциплины	108
в том числе:	
Лекции (теоретическое обучение)	40
Лабораторные работы	40
Самостоятельная работа	26
Промежуточная аттестация (в форме зачета)	2

Содержание дисциплины, структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

Содержание дисциплины

№	**Раздел дисциплины**	**Изучаемые компетенции**
1	Введение в проектирование информационных ресурсов	ОК 01, ОК 02, ПК 1.1
2	Объектно-ориентированная методология анализа и проектирования систем.	ОК 01, ОК 02, ПК 1.1
3	Диаграммы прецедентов	ОК 01, ОК 02, ПК 1.1
4	Диаграммы классов	ОК 01, ОК 02, ПК 1.1
5	Диаграммы последовательности	ОК 01, ОК 02, ПК 1.1
6	Диаграммы кооперации	ОК 01, ОК 02, ПК 1.1
7	Диаграммы поведения	ОК 01, ОК 02, ПК 1.1
8	Диаграммы деятельности и состояний	ОК 01, ОК 02, ПК 1.1
9	Диаграммы компонентов	ОК 01, ОК 02, ПК 1.1
10	Диаграммы развертывания	ОК 01, ОК 02, ПК 1.1

Отведенное количество часов по видам учебных занятий и работы

Вид обучения: 2 года 10 месяцев очное

Номер раздела данной дисциплины	**Трудоемкость в часах по видам занятий**		
	Лекции	**Лабораторные работы**	**Самоподготовка**
1	6	4	8
2	2	4	2
3	4	4	2
4	4	4	2
5	4	4	2
6	4	4	2
7	4	4	2
8	4	4	2
9	4	4	2
10	4	4	2
Итого	40	40	26

Лекционные занятия

Вид обучения: 2 года 10 месяцев очное

Семестр № 4

Наименование лекционных занятий	Трудоемкость аудиторной работы, часы
Раздел № 1 Введение в проектирование информационных ресурсов	
1) История развития и предпосылки создания методологии проектирования ИС 2) Цели и задачи методологии проектирования. 3) Классификация информационных систем. Модели жизненного цикла разработки программного обеспечения ИС. 4) Каноническое проектирование ИС 5) Этапы проектирования ИС. 6) Процессы жизненного цикла программных средств. 7) Инструментальные средства проектирования и моделирования ИС. 8) Стандартизация и критерии качества проекта ИС	6
Раздел № 2 Объектно-ориентированная методология анализа и проектирования систем.	
Основные концепции ООАиП. Унифицированный язык моделирования UML	2
Раздел № 3 Диаграммы прецедентов	
Определение и описание структуры диаграммы прецедентов	4
Раздел № 4 Диаграммы классов	
Определение и описание структуры диаграммы классов	4
Раздел № 5 Диаграммы последовательности	
Определение и описание структуры диаграммы последовательности	4
Раздел № 6 Диаграммы кооперации	
Определение и описание структуры диаграммы кооперации	4
Раздел №7 Диаграммы поведения	
Определение и описание структуры диаграммы поведения	4
Раздел №8 Диаграммы деятельности и состояний	
Определение и описание структуры диаграмм деятельности и состояний	4
Раздел №9 Диаграммы компонентов	
Определение и описание структуры диаграммы компонентов	4
Раздел №10 Диаграммы развертывания	
Определение и описание структуры диаграммы развертывания	4

Лабораторный практикум

Вид обучения: 2 года 10 месяцев очное

Семестр № 4

Наименование (тематика) практических работ, семинаров	Трудоемкость аудиторной работы, часы
Раздел № 1	
Описание и анализ предметной области	4
Раздел № 2	
Изучение CASE-средства ArgoUML	4

Наименование (тематика) практических работ, семинаров	Трудоемкость аудиторной работы, часы
Раздел № 3	
Построение диаграммы прецедентов.	4
Раздел № 4	
Построение диаграммы классов	4
Раздел № 5	
Построение диаграммы последовательности	4
Раздел № 6	
Построение диаграммы кооперации	4
Раздел № 7	
Построение диаграммы поведения	4
Раздел № 8	
Построение диаграмм деятельности и состояний	4
Раздел № 9	
Построение диаграммы компонентов	4
Раздел № 10	
Построение диаграммы развертывания	4

Самостоятельное изучение учебного материала (самоподготовка)

Вид обучения: 2 года 10 месяцев очное

Номер раздела данной дисциплины	Наименование тем, вопросов, вынесенных для самостоятельного изучения	Трудоемкость внеаудиторной работы, часы
1	Введение в проектирование информационных ресурсов. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	8
2	Объектно-ориентированная методология анализа и проектирования систем Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
3	Диаграммы прецедентов. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
4	Диаграммы классов. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
5	Диаграммы последовательности. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
6	Диаграммы кооперации. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2

Номер раздела данной дисциплины	Наименование тем, вопросов, вынесенных для самостоятельного изучения	Трудоемкость внеаудиторной работы, часы
7	Диаграммы поведения. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
8	Диаграммы деятельности и состояний Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
9	Диаграммы компонентов. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2
10	Диаграммы развертывания. Обработка результатов лабораторных работ. Подготовка к текущей и промежуточной аттестации.	2

ЛАБОРАТОРНАЯ РАБОТА №1

Изучение CASE-средства ArgoUML

Цель работы

Изучить возможности CASE-средства ArgoUML и способы построения диаграмм языка UML.

При разработке современного программного обеспечения разработчики сталкиваются с большими сложностями, которые вызываются четырьмя основными причинами [1]:

- сложность реальной предметной области, из которой исходит заказ на разработку;
- трудность управления процессом разработки;
- необходимость обеспечить достаточную гибкость программы;
- отсутствие удобных способов описания поведения больших систем.

Дополнительные сложности возникают, кроме того в результате изменения требований к информационной системе уже в процессе разработки, которые могут меняться, что может привести к бесконечному развитию и изменению процесса разработки самой системы.

Таким образом, все эти факторы, а также неумение правильно создавать сложные информационные системы приводят к тому, что разработка проекта выходит за рамки установленных сроков и бюджетов.

Следование определенным принципам проектирования информационных систем позволяет значительно снизить вероятность неблагоприятного исхода. К таким принципам относится использование метода объектно-ориентированного проектирования, унифицированного языка моделирования UML (*Unified Modeling Language*), который позволяет сократить разрыв между *видением* системы и ее воплощением, а также применение при проектировании CASE-систем.

Объектно-ориентированный подход позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы. Этим объясняется интерес программистов к объектно-ориентированному подходу и объектно-ориентированным языкам программирования.

UML помогает отобразить видение системы и дает возможность обсуждать его со всеми заинтересованными лицами. Это достигается общедоступным набором обозначений и диаграмм.

В современных условиях создание сложных программных приложений проблематично без использования систем автоматизированного конструирования программного обеспечения (CASE-систем). CASE-системы существенно сокращают сроки и затраты разработки, оказывая помощь инженеру в проведении рутинных операций, облегчая его работу на самых разных этапах жизненного цикла разработки. На сегодняшний день существует множество программных продуктов, один из которых и рассмотрен в настоящем пособии.

В данном учебно-методическом пособии последовательно рассматривается построение основных канонических диаграмм языка UML, используемых при проектировании информационных систем: диаграмма вариантов использования или прецедентов (use case dia- gram), диаграмма классов (class diagram),

диаграммы поведения (behavior diagrams), диаграмма состояний (statechart diagram), диаграмма деятельности (activity diagram), диаграммы взаимодействия (interaction diagrams), диаграмма последовательности (sequence diagram), диаграмма кооперации (collaboration diagram), диаграммы реализации (implementation diagrams), диаграмма компонентов (component diagram), диаграмма развертывания (deployment diagram).

Перечень этих диаграмм и их названия являются каноническими в том смысле, что представляют собой неотъемлемую часть графической нотации UML. Каждая из этих диаграмм детализирует и конкретизирует различные представления о модели сложной системы в терминах языка UML:

- *диаграмма вариантов* использования (прецедентов) представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.

- *диаграмма классов* является, по своей сути, логической моделью, отражающей статические аспекты структурного построения сложной системы.

- *диаграммы поведения* также являются разновидностями логической модели, которые отражают динамические аспекты функционирования сложной системы;

- *диаграммы реализации* служат для представления физических компонентов сложной системы и поэтому относятся к ее физической модели.

Таким образом, модель информационной системы в нотации UML представляется в виде *совокупности* указанных выше диаграмм.

Для выполнения всех лабораторных работ необходимо:

1. Ознакомиться с объектно-ориентированной CASE-системой (приложение 4. ArgoUML).
2. Ознакомиться с постановкой задачи и исходными данными (приложение 1).
3. Разработать предлагаемую в работе диаграмму.
4. Реализовать разработанную диаграмму на одном из программных продуктов (в данном случае ArgoUML).
5. Сохранить файл модели.
6. Составить отчет по проделанной работе.

Отчет оформляется по каждой лабораторной работе и состоит из следующих разделов:

1. Тема лабораторной работы.
2. Цель работы.
3. Индивидуальное задание.
4. Описание необходимых абстракций (элементов диаграмм).
5. Разработанная диаграмма.

Примечание

- Варианты индивидуальных заданий приведены в приложении 1.
- Краткий список графических обозначений для каждого типа диаграммы приведен в приложении 2.
- Пример разработки ИС с применением языка UML рассмотрен в приложении 3. Краткие сведения о работе в объектно-ориентированной CASE-системе ArgoUML изложены в приложении 4.

Ход работы и задания изложены в приложении 4 данного пособия.

ЛАБОРАТОРНАЯ РАБОТА № 2

Описание и анализ предметной области

Цель работы

Согласно варианту, выполнить описание предметной области проектируемой программной системы (см. пример из приложения 3). Провести объектный анализ полученного описания и построить модель среды с помощью диаграммы потоков данных (анализ поведения системы) и диаграммы «сущность- связь» (анализ данных). Определить назначение проектируемой ИС.

1. **Формулировка задачи. Выделение/выявление границ задачи и ИС.**

Сфера деятельности объекта управления - обслуживание клиентов в отеле.

Целью работы является создание объектно-ориентированной модели автоматизированной информационной системы приёма и обслуживания клиентов в отеле.

Выделим объект и предмет: объект – отель, а предмет – спроектированная автоматизированная информационная система приёма и обслуживания клиентов в отеле.

2. **Исследование и создание организационной структуры менеджмента и функциональности данной области. Выявления взаимосвязей между функциональными структурами системы. Описание пакета документов характерного для данной предметной области.**

В большинстве гостиниц структура управления линейно-функциональная. Во главе всей гостиницы стоит управляющий. Он занимается координацией работы менеджеров различных подразделений. Осуществляет контроль за работой всех подразделений питания. Также он представляет гостиницу на различных конференциях и семинарах.

При организации структуры управления отеля необходим творческий подход к обслуживанию клиентов. В общем случае гостиница занимается размещением клиентов и предоставляет им ряд дополнительных услуг. Главной задачей персонала отеля является удовлетворение запросов и потребностей клиентов. Проанализировав основные функции, выполняемые персоналом отеля, можно составить его организационную структуру.

Организационная структура:

- Администрация
- Бухгалтерия
- Отдел продаж и маркетинга
- Информационный отдел
- Отдел домоводства
- Кухня (напитки и пищевые продукты)
- Транспортный отдел
- Игорные заведения, центры досуга
- Отдел безопасности

Основные документы:

- Журнал регистрации заказов

- Справочник сервисных услуг
- Квитанция об оплате счета
- Справочник штрафов
- Справочник номеров
- Справочник сотрудников
- История клиента

3. Описание основных процессов работы исследуемой области. Выявление информационных потоков, определение задач интеграции с другими системами.

Административная служба.

Это первая служба с которой сталкивается гость. Основными её задачами являются:

- регистрация гостей и распределение номеров;
- ведение реестра состояния номеров;
- хранение ключей;
- оформление выездов;
- ведение счета гостя;
- координация работы горничных;
- предоставление гостям различной информации, в частности по работе гостиницы. Неотъемлемой частью административной службы является отдел резервирования номеров. Резервирование может быть осуществлено как по телефону, так и по факсу.

Менеджер административной службы должен, во-первых, обладать всей возможной информацией о гостинице. Во-вторых, он должен уметь чётко планировать работу администрации. В-третьих, должен осуществлять постоянный контроль за работой своего отдела. Все вопросы, проблемы и недоразумения гостей должны разрешаться также с его помощью.

Бухгалтерия.

Бухгалтерия документально оформляет совершаемые хозяйственные операции на предприятии: снабжение, приобретение товаров, сырья, материалов, расчётные операции с поставщиками, транспортными организациями, бюджетом, составляет калькуляцию произведённой продукции, проводит инвентаризацию, начисляет заработную плату и налоги, ведёт отчётность предприятия.

Отдел продаж и маркетинга.

Этот отдел является наиболее теневым подразделением гостиницы. Обязанности работников, этого отдела можно подразделить на 4 группы:

1. Основная цель отдела маркетинга, заключается в продаже продукции и услуг отеля.
2. Продажа, услуги по организации конференций и бизнес семинаров.
3. Реклама.
4. Связь с общественностью.

Отдел маркетинга представляет собой некий аналитический центр, который аккумулирует различного рода информацию и на её основе строит стратегию продаж.

Информационный отдел.

Состоит из систем разделённых на несколько модулей:

1. Модуль менеджера. Можно назвать системой поддержки управления гостиницей. Он может генерировать отчёты в автоматическом режиме. Например, можно задать системе в в определённое время распечатывать отчёты о деятельности различных подразделений или же находить в процессе дня некоторые пиковые значения заранее определённых параметров.

2. Модуль административной службы. Это центральная система, которая аккумулирует данные о техническом состоянии номеров, текущем статусе номеров, текущих расценках за номера, текущей занятости отеля. Также с её помощью осуществляется резервирование и регистрация гостей, ведётся учёт их истории для выявления постоянных клиентов.

3. Модуль отдела резервирования. Предназначен для автоматизации работы отдела резервирования номеров.

4. Модуль подразделения питания. Информация от поставщиков попадает на кухню шеф-повару. Если она касается пополнения склада, то она проходит через бухгалтерию, где поступивший товар оформляется документально. Из отдела обслуживания в номерах на центральный кухонный терминал попадают заказы от гостей отеля. Также заказы поступают и из торгового зала.

5. Модуль отдела горничных. Используется для поддержания информации о состоянии номеров для управленческих нужд и для отдела резервирования. Каждому номеру присваивается статус либо «свободно» либо «занято» в общем случае.

Информационные системы отдела маркетинга не включены в основную сеть гостиничного комплекса, так как этот отдел находится как бы снаружи отеля. Работники в нем, представляют собой неких «поставщиков» гостей отеля и поэтому им не требуется прямая связь с внутренними подразделениями гостиницы. Она лишь требуется в случае изменения финансовых условий проживания и увеличения набора предлагаемых услуг.

Отдел домоводства.

Включает в себя все службы по поддержанию чистоты и порядка в отеле. Уборке номерного фонда, коридоров, холлов, мест отдыха и т.д. Также он следит за своевременной заменой постельного белья и принадлежностей. В его состав могут входить прачечные, химчистки и другие службы оказывающие необходимые услуги гостям.

Кухня.

К этой службе относятся:

1. Рестораны.
2. Бары.
3. Кафе.
4. Обслуживание в номерах.

Транспортный отдел.

Предназначен для доставки клиентов в отель, предоставление им машин в пользование, а также для своевременной доставки продуктов питания и напитков.

Игорные заведения, центры досуга.

Предлагают гостям гостиницы различные услуги игорного бизнеса (казино, игровые автоматы, все возможные виды игр и развлечений). Центры досуга предоставляют разнообразные кружки по интересам и увлечениям. Организуют все возможные экскурсии, встречи, концерты и т.д.

Отдел безопасности.

Службы безопасности должны внимательно следить за тем, чтобы постояльцы и работники не пали жертвами домогательств пьяных субъектов и имели возможность покинуть игорное заведение с крупным выигрышем в кармане. Офицеры службы безопасности играют очень важную роль в обеспечении спокойной, лишённой каких- либо неприятностей жизни своих гостей.

4. Определение и описание процессов предприятия, которые будут автоматизированы (функции будущей АИС). Описание существующих взаимосвязей между функциями будущей АИС.

Для данной предметной области необходимо автоматизировать процесс доступа к данным разными категориями служащих отеля.

5. Составление предложений по технологической структуре АИС: на каком уровне будет создана система(национальном, региональном, местном...),описание информационной сети, топология(если система основана в сети),если система внедрена на одном компьютере,состав и основа технических средств (операционная система, СУБД, серверы, средства необходимые для создания клиентской части...)

Система предполагается быть созданной на местном уровне.

Так как пользователями системы будут являться несколько структурных подразделений гостиницы, необходимым условием будет наличие локальной сети .

Программное обеспечение предполагается быть разработанным под операционную систему Windows (не раньше версии XP). Для её функционирования требуется СУБД Microsoft SQL Server.

6. Техническое задание -формулировка требований:

- **функциональных (что должна выполнять АИС)**
- **не функциональных (описание характеристик системы:качество, соответствие требований предприятия стандартам, скорость, требования к железу).**

Разрабатываемая информационная система предназначена для работы в гостиницах и отелях. Исходя из собственного представления системы составим ряд приблизительных требований к ней.

Сформулируем требования к системе:

1. Система предназначена для ввода, хранения и обработки информации о клиентах, информации о заказах, о счетах за проживание и дополнительные услуги, о совершенных нарушениях и уплаченных за них суммах;
2. Журнал регистрации заказов содержит дату въезда, дату выезда, № комнаты, паспортные данные клиента (ФИО клиента, место регистрации, № паспорта), ФИО сотрудника, принявшего заказ.
3. Справочник сервисных услуг должен содержать тип услуг, стоимость.
4. Квитанция об оплате счета должна содержать дату въезда, дату выезда, ФИО клиента, общую стоимость (проживания в номере, штрафы, стоимость дополнительных услуг).
5. Справочник штрафов должен содержать тип правонарушения, величину штрафа.
6. Справочник номеров должен содержать № комнаты, стоимость проживания в сутки, категорию номера.

7. Справочник сотрудников должен содержать ФИО сотрудника, занимаемую должность.

8. История клиента должна содержать ФИО клиента, тип нарушений, величину штрафа, наименование дополнительной услуги, стоимость дополнительной услуги.

9. Система должна обеспечить выполнение действий, связанных с сервисными услугами:

1. Ввод новой сервисной услуги в справочник
2. Внесение стоимости новой сервисной услуги в справочник
3. Удаление сервисной услуги.
4. Удаление стоимости сервисной услуги из справочника.
5. Корректирование сервисной услуги
6. Корректирование сервисной услуги в справочнике.

10. Система должна поддерживать следующие варианты сервисных услуг:

1. Бассейн
2. Тенистый корт
3. Фитнесс центр
4. Массажный салон
5. Интернет

11. Система должна обеспечить выполнение действий, связанных с штрафными санкциями:

1. Ввод новой штрафной санкции в справочник
2. Внесение стоимости новой штрафной санкции в справочник
3. Удаление штрафной санкции из справочника.
4. Удаление стоимости штрафной санкции из справочника.
5. Корректирование штрафной санкции
6. Корректирование стоимости штрафной санкции из справочника

12. Система должна поддерживать следующие варианты нарушений:

1. Разбитое стекло
2. Сломанная мебель
3. Потерянный ключ
4. Нарушение порядка
5. Разбитая посуда

13. Система должна поддерживать следующие категории номеров:

1. Бизнес класс
2. Эконом
3. Полу люкс
4. Люкс

ЛАБОРАТОРНАЯ РАБОТА № 3

Построение диаграммы прецедентов

Цель работы

Изучить правила оформления диаграммы вариантов использования (диаграммы прецедентов) и диаграммы анализа. Научится выделять особенности функционального поведения проектируемой системы.

Теоретические сведения

Построение диаграммы вариантов использования (прецедентов) (Use Case Diagram)

Визуальное моделирование системы в UML (Unified Modeling Language) можно представить как некоторый процесс поуровневого спуска от наиболее общей и абстрактной модели исходной системы к логической, а затем и к физической модели. Для достижения этих целей вначале строится модель в форме диаграммы вариантов использования (*use case diagram*), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Разработка диаграммы вариантов использования преследует цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом актером (*actor*) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство или программа, которые могут служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (*use case*) служит для описания сервисов, которые система предоставляет актеру.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия ее внутренней структуры. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением.

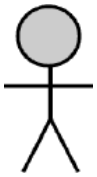
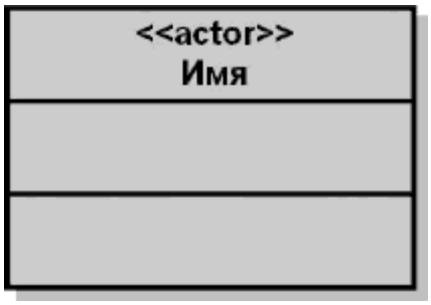
Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того как система за-

кончит обработку запроса пользователя, она должна возвратиться в исходное состояние, в котором готова к выполнению следующих запросов.

Графическое обозначение для диаграммы прецедентов дано в табл. 3.1.

Таблица 3.1

Графическое обозначение элементов

№	Название	Обозначение	Примечание
1	Эктор (actor) или актер	 или 	Обозначает набор ролей пользователя (понимается в широком смысле: человек, внешняя сущность, класс, другая система), взаимодействующего с некоторой сущностью (системой, подсистемой, классом).
2	*Интерфейс* (interface)		Интерфейсы определяют операций, которые обеспечивают необходимый набор сервисов или функциональности для актеров.
3	Прецедент (use-case)		Обозначает выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым актёрами результатам.
4	*Примечания* (notes)		Произвольная текстовая информация, имеющая непосредственное отношение к контексту разрабатываемого проекта

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов.Один актер может взаимодействовать с несколькими вариантами исполь- зования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодейство- вать с несколькими актерами, предоставляя для всех них свой сервис.

Следует заметить, что ***два варианта использования***, определенные для одной и той же сущности, ***не могут взаимодействовать друг с другом***, поскольку каждый из них самостоятельно описывает законченный вариант использования этой сущности.

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

- Отношение *ассоциации* (association relationship)
- Отношение *расширения* (extend relationship)
- Отношение *обобщения* (generalization relationship)
- Отношение *включения* (include relationship)

Общие свойства вариантов использования могут быть представлены тремя различными способами, а именно с помощью отношений расширения, обобщения и включения.

Отношение ассоциации

Применительно к диаграммам вариантов использования ассоциация – это отношение, которое устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Пример этого типа отношения приведен на рис. 3.1. Эта линия может иметь условные обозначения, такие как имя и кратность.

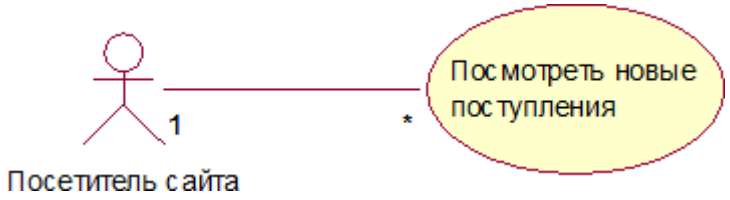


Рис. 3.1. Пример графического представления отношения ассоциации между актером и вариантом использования

Кратность (multiplicity) ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации, и характеризует количество экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и символа звездочка.

Для диаграмм вариантов использования наиболее распространенными являются четыре основные формы записи кратности отношения ассоциации:

- целое неотрицательное число (включая 0). Предназначено для указания кратности, которая является строго фиксированной для элемента соответствующей ассоциации. В этом случае количество экземпляров актеров или вариантов использования, которые могут выступать в качестве элементов отношения ассоциации, в точности равно указанному числу;
- два целых неотрицательных числа, разделенные двумя точками. Данная запись соответствует нотации для множества или интервала целых чисел, кото-

рая применяется в некоторых языках программирования для обозначения границ массива элементов. Эту запись следует понимать как множество целых неотрицательных чисел, следующих в последовательно возрастающем порядке;

- два символа, разделенные двумя точками. При этом первый из них является целым неотрицательным числом или 0, а второй – специальным символом «*», который обозначает произвольное конечное целое неотрицательное число, значение которого неизвестно на момент задания соответствующего отношения ассоциации;
- единственный символ «*», который является сокращением записи интервала «0..*».

Если кратность отношения ассоциации не указана, то, по умолчанию, принимается значение равное 1.

Отношение расширения

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров. Отношение расширения является направленным и указывает, что применительно к отдельным примерам некоторого варианта использования должны быть выполнены конкретные условия, определенные для расширения данного варианта использования.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой, направленной от того варианта использования, который является расширением для исходного варианта использования (рис. 3.2).

Рис. 3.2. Пример графического изображения отношения расширения между вариантами использования.

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования. Данное отношение включает в себя некоторое условие и ссылки на точки расширения вбазовом варианте использования. Чтобы расширение имело место, должно быть выполнено определенное условие данного отношения.

Семантика отношения расширения определяется следующим образом. Если экземпляр варианта использования выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляр другого варианта использования, которая является первой из всех точек расширения у исходного варианта, то проверяется условие данного отношения. Если условие выполняется, исходная последовательность действий расширяется посредством включения действий экземпляра другого

варианта использования.

Отношение обобщения

Отношение обобщения служит для указания того факта, что некоторый вариант использования *A* может быть обобщен до варианта использования *B*. В этом случае вариант *A* будет являться специализацией варианта *B*. При этом *B* называется предком или родителем по отношению *A*, а вариант *A* – потомком по отношению к варианту использования *B*. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное отношение обозначается сплошной линией со стрелкой в форме не закрашенного треугольника, которая указывает на родительский вариант использования (рис. 3.3).

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других.

Рис. 3.3. Пример графического изображения отношения обобщения между вариантами использования

Отношение включения

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

Семантика этого отношения определяется следующим образом. Когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, экземпляр первого варианта использования выполняет последовательность действий, определяющую поведение экземпляра второго варианта использования, после чего продолжает выполнение действий своего поведения. При этом предполагается, что даже если экземпляр первого варианта использования может иметь несколько включаемых в себя экземпляров других вариантов, выполняемые ими действия должны закончиться к некоторому моменту, после чего должно быть продолжено выполнение прерванных действий экземпляра первого варианта использования в соответствии с заданным

для него поведением. Пример графического обозначения данного типа отношения приведен на рис. 3.4.

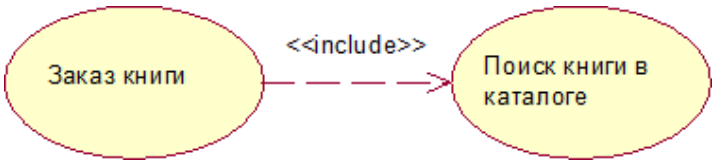


Рис. 3.4. Пример графического изображения отношения включения между вариантами использования.

Построение диаграммы анализа

Диаграмма анализа предназначена для описания происходящих в системе бизнес-процессов, модели поведения системы и ее элементов. Данная диаграмма является оптимальным средством для описания бизнес-процессов и их особенностей.

Диаграмма анализа представляет собой упрощенную версию диаграммы процессов Эриссона-Пенкера, предназначенной для моделирования бизнес-процессов в сложных корпоративных системах.

Основными компонентами диаграммы анализа являются:

- бизнес-процесс,
- ресурс и информация,
- событие,
- выход,
- цель.

Бизнес-процесс

Бизнес-процесс представляет собой набор действий, направленных на получение конкретного результата для конкретного актера. Для каждого бизнес-процесса четко заранее определяются входы и выходы. Обозначение бизнес-процесса представлено на рисунке 3.5.

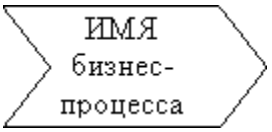


Рис. 3.5. Изображение бизнес-процесса на диаграмме анализа

Ресурс и информация

В процессе своей реализации бизнес-процесс использует ресурсы. В качестве ресурсов может выступать информация от внешних или внутренних источников, от других актеров или же от других бизнес-процессов.

В отличие от ресурсов, информация не используется бизнес-процессом, она несет информативный характер, объясняет или уточняет определенный нюанс. Обозначение ресурса и информации представлено на рисунке 3.6.

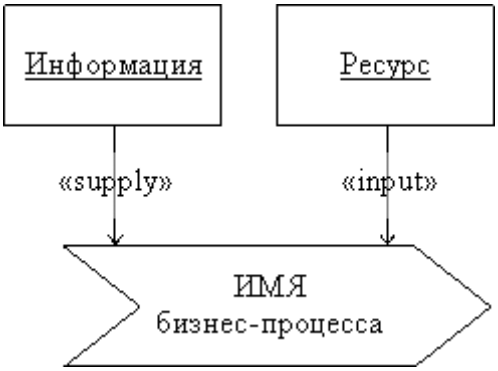


Рис. 3.6. Изображение ресурса и информации на диаграмме анализа

Событие

Событием является какой-либо объект, момент времени, дата, уведомление или какой-либо другой триггер, после срабатывания которого начинается выполнение бизнес-процесса. Обозначение события представлено на рис. 3.7.

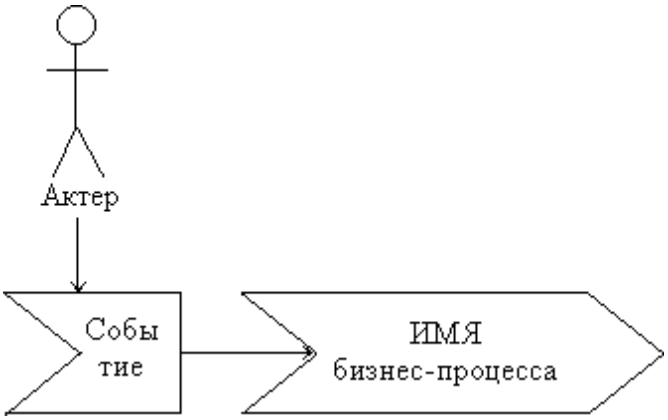


Рис. 3.7. Изображение события на диаграмме анализа

Выход

В результате выполнения бизнес-процесса, как правило, формируется один или несколько выходных результатов. Выход может представлять собой физический объект (например, отчет), преобразование имеющихся ресурсов к новым условиям (ежедневное расписание или список) или общий результат деятельности (например, заказ).

Обозначение выхода представлено на рисунке 3.8.

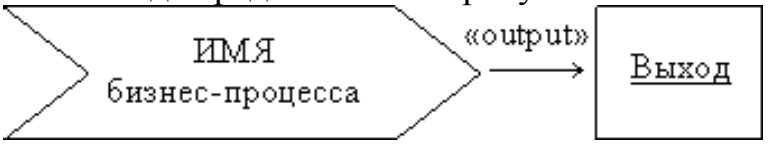


Рис. 3.8. Изображение выхода на диаграмме анализа

Цель

Любой бизнес-процесс имеет одну или несколько четко определенных целей. Именно цель является причиной организации и выполнения бизнес-процесса. Обозначение цели представлено на рисунке 3.9.

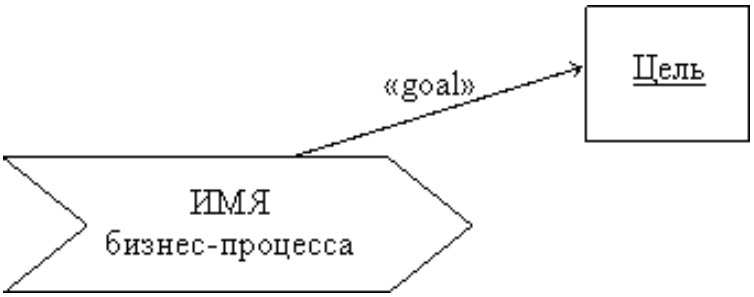


Рис. 3.9. Изображение цели на диаграмме анализа

2.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы вариантов использования.
3. Спецификация поведения элементов Use Case диаграммы вариантов использования.
4. Спецификация других элементов диаграммы.
5. Разработанные диаграммы анализа.
6. Выводы.

2.4 Контрольные вопросы

1. Назначение диаграммы вариантов использования.
2. Цели разработки диаграммы вариантов использования.
3. Элементы диаграммы вариантов использования. Актеры.
4. Элементы диаграммы вариантов использования. Отношения.
5. Элементы диаграммы вариантов использования. Use Case.
6. Цели разработки анализа
7. Элементы диаграммы анализа Бизнес-процессы.
8. Элементы диаграммы анализа Ресурсы и информация.
9. Элементы диаграммы анализа События.
10. Элементы диаграммы анализа Выходы.
11. Элементы диаграммы анализа Цели.

ЛАБОРАТОРНАЯ РАБОТА № 4

Построение диаграммы классов

Цель работы

Изучить правила оформления диаграммы классов. Научится разрабатывать статическую структуру модели системы в терминологии классов объектно-ориентированного программирования.

Теоретические сведения

Диаграмма классов

Диаграмма классов (*class diagram*) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. Эти знания интерпретируются в базовых понятиях языка UML, таких как классы, интерфейсы и отношения между ними и их составляющими компонентами. При этом отдельные компоненты этой диаграммы могут образовывать пакеты для представления более общей модели системы. Если диаграмма классов является частью некоторого пакета, то ее компоненты должны соответствовать элементам этого пакета, включая возможные ссылки на элементы из других пакетов.

В общем случае пакет структурной статической модели может быть представлен в виде одной или нескольких диаграмм классов. Декомпозиция некоторого представления на отдельные диаграммы выполняется с целью удобства и графической визуализации структурных взаимосвязей предметной области. При этом компоненты диаграммы соответствуют элементам статической семантической модели. Модель системы, в свою очередь, должна быть согласована с внутренней структурой классов, которая описывается на языке UML.

Класс

Класс (*class*) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 4.1). В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса. По мере проработки отдельных компонентов диаграммы, описания классов дополняются атрибутами и операциями.Предполагается, что окончательный вариант диаграммы содержит наибо- лее полное описание классов, которые

состоят из трех разделов или секций.

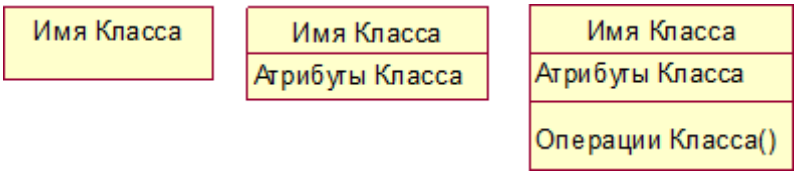


Рис. 4.1. Графическое изображение класса на диаграмме классов

Отношения между классами

Кроме внутреннего устройства или структуры классов на соответствующей диаграмме указываются различные отношения между классами. При этом совокупность типов таких отношений зафиксирована в языке UML и предопределена семантикой этих типов отношений. Базовыми отношениями или связями в языке UML являются:

- отношение зависимости (dependency relationship);
- отношение ассоциации (association relationship);
- отношение обобщения (generalization relationship);
- отношение реализации (realization relationship).

Каждое из этих отношений имеет собственное графическое представление на диаграмме, которое отражает взаимосвязи между объектами соответствующих классов.

Отношение зависимости

Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, которое не является отношением ассоциации, обобщения или реализации. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели (рис. 4.2).

Рис. 4.2. Графическое изображение отношения зависимости на диаграмме классов

Отношение ассоциации

Отношение ассоциации соответствует наличию некоторого отношения между классами.

Наиболее простой случай данного отношения – бинарная ассоциация. Она связывает в точности два класса и, как исключение, может связывать класс с самим собой (рис. 4.3).

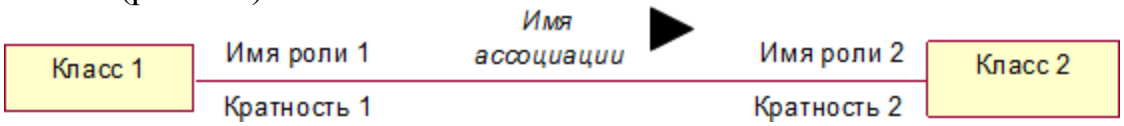


Рис. 4.3. Графическое изображение отношения бинарной ассоциации между классами

Отношение обобщения

Отношение обобщения (рис. 4.4) является отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML.

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

Рис. 4.4. Графическое изображение отношения обобщения

Отношение реализации

Отношение реализации имеет место между двумя элементами модели в том случае, если один элемент (клиент) реализует поведение, заданное другим (поставщиком). Отношение реализации подразделяется на:

- отношение агрегации (*aggregation relationship*);
- отношение композиции (*composition relationship*).

Отношение агрегации

Отношение агрегации (рис. 4.5) имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

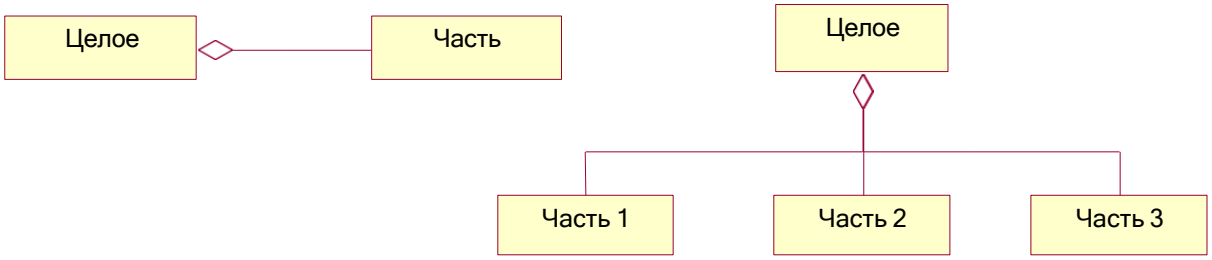


Рис. 4.5. Графическое изображение отношения агрегации в языке UML

Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа «часть-целое». Раскрывая внутреннюю структуру системы, отношение агрегации показывает, из каких компонентов состоит система и как они связаны между собой.

Отношение композиции

Отношение композиции (рис. 4.6) служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого.

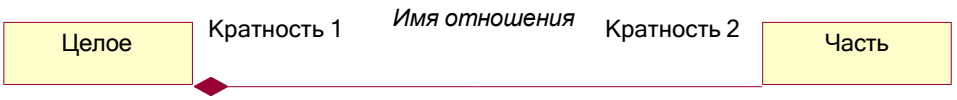


Рис. 4.6. Графическое изображение отношения композиции в языке UML

Дополнительные элементы

На диаграмме классов, кроме того указывают:

- *Интерфейсы* (рис. 4.7)

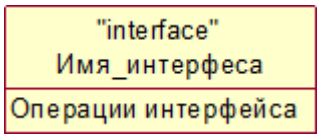


Рис. 4.7. Графическое изображение интерфейса на диаграмме классов

- *Объект (object)*. Является отдельным экземпляром класса, который создается на этапе выполнения программы (рис. 4.8)

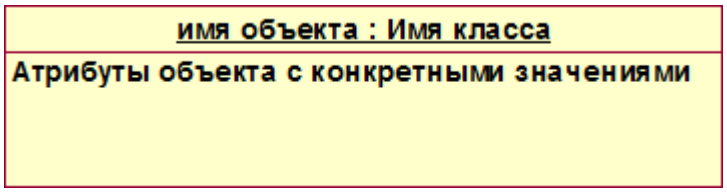


Рис. 4.8. Графическое изображение объекта на диаграмме классов

- *Шаблон (template)* или параметризованный класс (*parametrized class*) предназначен для обозначения такого класса, который имеет один (или более) нефиксированный формальный параметр (рис. 4.9).

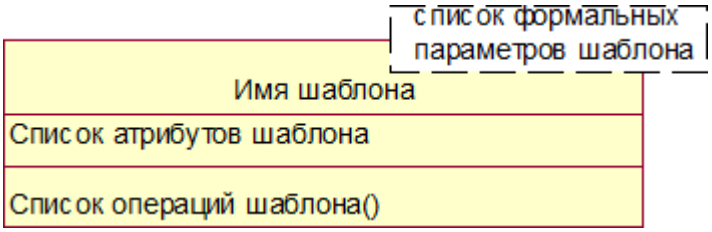


Рис. 4.9. Графическое изображение шаблона (параметризованного класса)

Рекомендации по построению диаграммы классов

Процесс разработки диаграммы классов занимает центральное место в ООАП сложных систем. От умения правильно выбрать классы и установить между ними взаимосвязи часто зависит не только успех процесса проектирования, но и производительность выполнения программы.

После разработки диаграммы классов процесс ООАП может быть продолжен в двух направлениях. С одной стороны, если поведение системы тривиально, то можно приступить к разработке диаграмм кооперации и компонентов.

Однако для сложных динамических систем поведение представляет важнейший аспект их функционирования. Детализация поведения осуществляется последовательно при разработке диаграмм состояний, последовательности и деятельности.

Содержание отчета

1. Наименование и цель работы, номер варианта
2. Разработанные диаграммы классов.
3. Описание элементов диаграммы классов (включая отношения).
4. Выводы.

Контрольные вопросы

1. Назначение диаграммы классов.
2. Цели разработки диаграммы классов.
3. Элементы диаграммы классов. Классы.
4. Элементы диаграммы классов. Отношения.
5. Элементы диаграммы классов. Объекты.

ЛАБОРАТОРНАЯ РАБОТА № 5

Построение диаграммы последовательности

Цель работы

Изучить правила оформления диаграммы последовательности. Изучить особенности взаимодействия объектов проектируемой системы.

Теоретические сведения

Функциональность элементов диаграммы вариантов использования отображается графически на диаграммах взаимодействия. Эти диаграммы содержат объекты и сообщения между объектами, которые показывают реализацию поведения.

Диаграмма последовательности (Sequence Diagram)

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения. Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия. Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют некоторый порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Второе измерение диаграммы последовательности – вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и также образуют, порядок по времени своего возникновения. Пример диаграммы последовательности дан на рис. 5.1.

Линия жизни объекта

Линия жизни объекта (*object lifeline*) изображается (см. рис. 5.1) пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности от самой верхней ее части до самой нижней.

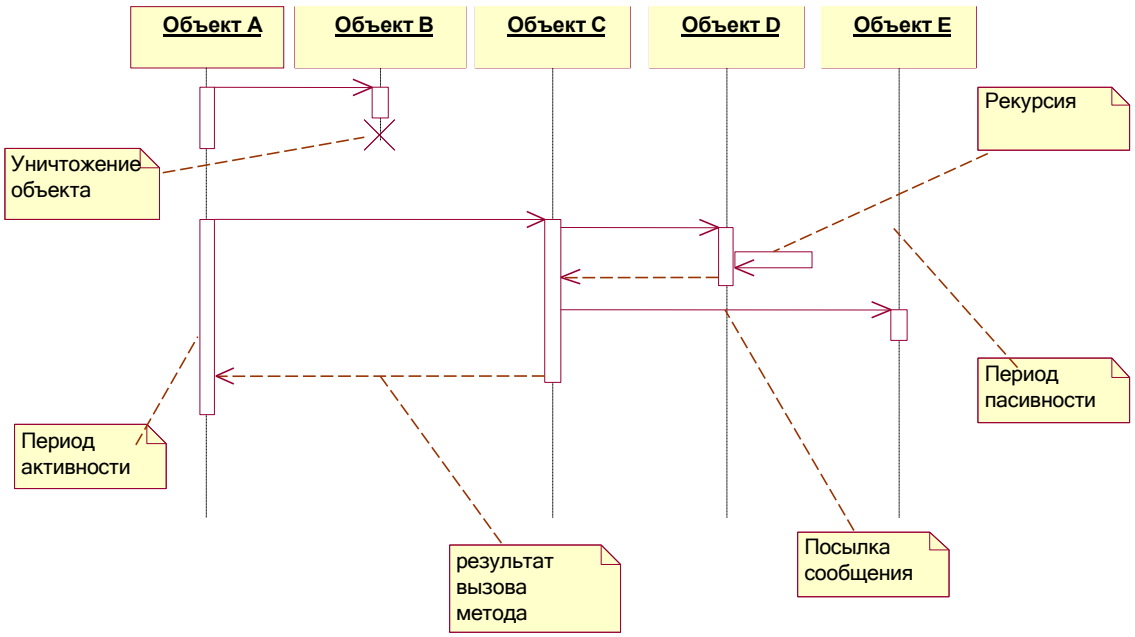


Рис. 5.1. Пример диаграммы последовательности

Отдельные объекты, исполнив свою роль в системе, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML используется специальный символ в форме латинской буквы «X».

Фокус управления

В процессе функционирования объектно-ориентированных систем одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия или в состоянии пассивного ожидания сообщений от других объектов. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название фокуса управления (focus of control). Фокус управления изображается в форме вытянутого узкого прямоугольника, верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а ее нижняя сторона – окончание фокуса управления (окончание активности).

В отдельных случаях инициатором взаимодействия в системе может быть актер или внешний пользователь. В этом случае актер изображается на диаграмме последовательности самым первым объектом слева со своим фокусом управления. Чаще всего актер и его фокус управления будут существовать в системе постоянно, отмечая характерную для пользователя активность в инициировании взаимодействий с системой. При этом сам актер может иметь собственное имя либо оставаться анонимным.

Сообщения

Как было отмечено выше, цель взаимодействия в контексте языка UML заключается в том, чтобы специфицировать коммуникацию между множеством взаимодействующих объектов. Каждое взаимодействие описывается совокуп-

ностью сообщений, которыми участвующие в нем объекты обмениваются между собой. В этом смысле сообщение (*message*) представляет собой законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено см. (рис. 5.1).

Таким образом, сообщения не только передают некоторую информацию, но и требуют или предполагают от принимающего объекта выполнения ожидаемых действий. Сообщения могут инициировать выполнение операций объектом соответствующего класса, а параметры этих операций передаются вместе с сообщением. На диаграмме последовательности все сообщения упорядочены по времени своего возникновения в моделируемой системе. В таком контексте каждое сообщение имеет направление от объекта, который инициирует и отправляет сообщение, к объекту, который его получает.

Обычно сообщения изображаются горизонтальными стрелками, соединяющими линии жизни или фокусы управления двух объектов на диаграмме последовательности.

В языке UML могут встречаться несколько разновидностей сообщений, каждое из которых имеет свое графическое изображение (рис. 5.2).

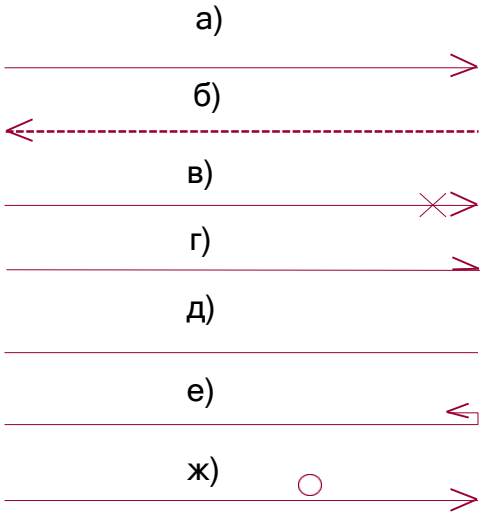


Рис. 5.2. Графическое изображение различных видов сообщений между объектами на диаграмме последовательности

Виды сообщений:

- ***simple***. Простое сообщение (рис. 5.2, а);
- ***return***. Сообщение возврата из процедуры (рис. 5.2, б);
- ***synchronous***. Клиент выполняет действия до момента посылки сообщения. Клиент ждет, пока сервер примет и обработает сообщение, не выполняя ни каких действия. Как только сервер вернет управление из сообщения, клиент продолжает свою работу (рис. 5.2, в);
- ***asynchronous***. Клиент посылает сообщение серверу и, не ожидая окончания обработки сообщения сервером, продолжает свое вы-

полнение (рис. 5.2, г);

- ***procedure call***. Вложенная последовательность действий завершается, прежде чем выполнение продолжится на более высоком уровне. Данный тип сообщения применим как для вызовов процедур, так и для взаимодействия объектов при котором клиент дожидается завершения работы всей последовательности действия сервера (рис. 5.2, д);
- ***balking***. Клиент посылает сообщение, только если сервер готов принять сообщение. Иначе сообщение не посылается (рис. 5.2, е);
- ***timeout.*** Клиент "отзывает" сообщение, если сервер не может обработать сообщение в течение определенного времени (рис. 5.2, ж).

Содержание отчета

1. Наименование и цель работы, номер варианта
2. Разработанные диаграммы последовательности.
3. Спецификация диаграмм последовательности.
4. Разработанные диаграммы кооперации уровня примеров.
5. Выводы.

Контрольные вопросы

1. Назначение диаграммы последовательности.
2. Особенности диаграммы последовательности.
3. Элементы диаграммы последовательности. Объекты.
4. Элементы диаграммы последовательности. Сообщения.
5. Диаграмма кооперации уровня спецификации.
6. Диаграмма кооперации уровня примеров.

ЛАБОРАТОРНАЯ РАБОТА № 6

Построение диаграммы кооперации

Цель работы

Изучить правила оформления диаграммы кооперации. Изучить особенности взаимодействия объектов проектируемой системы.

Теоретические сведения

Диаграмма кооперации (Collaboration Diagram)

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность диаграммы кооперации заключается в возможности графически *представить не только последовательность взаимодействия, но и все структурные отношения между объектами,* участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. Дополнительно могут быть изображены динамические связи – потоки сообщений.

Таким образом, с помощью диаграммы кооперации можно описать полный контекст взаимодействий как своеобразный временной «срез» совокупности объектов, взаимодействующих между собой для выполнения определенной задачи или цели программной системы (рис. 6.1).

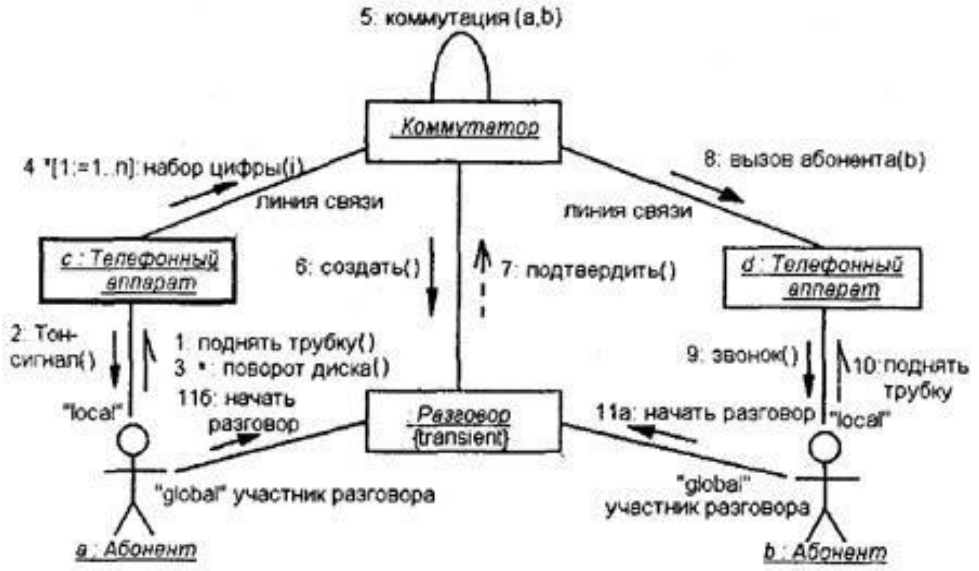


Рис. 6.1. Пример диаграммы кооперации

Кооперация

Понятие кооперации (*cooperation*) является одним из фундаментальных понятий в языке UML. Оно служит для обозначения множества взаимодействующих с определенной целью объектов в общем контексте моделируемой системы. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе. Кооперация определяет структуру поведения системы в терминах взаимодействия

участников этой кооперации.

Кооперация может быть представлена на двух уровнях:

- на уровне спецификации – показывает роли классификаторов и роли ассоциаций в рассматриваемом взаимодействии;
- на уровне примеров – указывает экземпляры и связи, образующие отдельные роли в кооперации.

Диаграмма кооперации уровня спецификации показывает роли, которые играют участвующие во взаимодействии элементы. Элементами кооперации на этом уровне являются классы и ассоциации, которые обозначают отдельные роли классификаторов и ассоциации между участниками кооперации.

Диаграмма кооперации уровня примеров представляется совокупностью объектов (экземпляры классов) и связей (экземпляры ассоциаций). При этом связи дополняются стрелками сообщений. На данном уровне показываются только объекты, имеющие непосредственное отношение к реализации операции или классификатора

Содержание отчета

6. Наименование и цель работы, номер варианта
7. Разработанные диаграммы последовательности.
8. Спецификация диаграмм последовательности.
9. Разработанные диаграммы кооперации уровня примеров.
10. Выводы.

Контрольные вопросы

7. Назначение диаграммы последовательности.
8. Особенности диаграммы последовательности.
9. Элементы диаграммы последовательности. Объекты.
10. Элементы диаграммы последовательности. Сообщения.
11. Диаграмма кооперации уровня спецификации.
12. Диаграмма кооперации уровня примеров.

ЛАБОРАТОРНАЯ РАБОТА № 7

Построение диаграммы поведения

Цель работы

Изучить правила оформления диаграмм состояний и деятельности. Научится выделять в поведении элемента системы отдельные состояния. С помощью диаграммы деятельности научится отображать особенности алгоритмов, реализующих основные функции системы.

Теоретические сведения

Диаграмма состояний

Для моделирования поведения на логическом уровне в языке UML могут использоваться сразу несколько канонических диаграмм: состояний, деятельности, последовательности и кооперации, каждая из которых фиксирует внимание на отдельном аспекте функционирования системы. В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только отдельного элемента модели (от отдельного экземпляра класса до всей системы в целом). При этом изменение состояния элемента системы может быть вызвано внешними воздействиями со стороны других подсистем или извне. Именно для описания реакции элемента модели на подобные внешние воздействия и используются диаграммы состояний.

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели.

Автомат в языке UML представляет собой некоторый формализм для моделирования поведения элементов модели и системы в целом. В метамодели UML автомат является пакетом, в котором определено множество понятий, необходимых для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом состояний и переходов. С другой стороны, автомат описывает поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла,

начиная от создания объекта и заканчивая его уничтожением. Каждая диаграмма состояний представляет некоторый автомат.

Основными понятиями, входящими в формализм автомата, являются состояние и переход.

Состояние

Понятие состояния (*state*) является фундаментальным не только в метамодели языка UML, но и в прикладном системном анализе.

В языке UML под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта (рис. 7.1).

Рис. 7.1. Графическое изображение начального и конечного состояний

Состояние на диаграмме изображается прямоугольником со скругленными вершинами (рис. 7.2). Этот прямоугольник, в свою очередь, может быть разделен на две секции горизонтальной линией. Если указана лишь одна секция, то в ней записывается только имя состояния (рис. 7.2, а). В противном случае в первой из них записывается имя состояния, а во второй – список некоторых внутренних действий или переходов в данном состоянии (рис. 7.2, б).

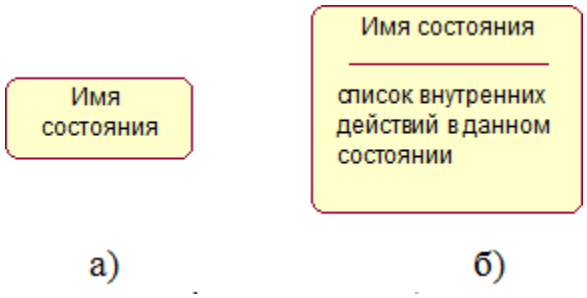


Рис. 7.2. Графическое изображение состояний

Переход

Простой переход (*simple transition*) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим (рис. 7.3). Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий. В этом случае говорят, что переход срабатывает. До срабатывания перехода объект находится в предыдущем от него состоянии, называемым исходным состоянием, или в ис-

точнике, а после его срабатывания объект находится в последующем от него состоянии (целевом состоянии).

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности (*do activity*), получении объектом сообщения или приемом сигнала. На переходе указывается имя события. Кроме того, на переходе могут указываться действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое. Срабатывание перехода может зависеть не только от наступления некоторого события, но и от выполнения определенного условия, называемого сторожевым условием. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение «истина».

Рис. 7.3. Графическое изображение перехода между состояниями

Событие

Формально, *событие (event)* представляет собой спецификацию некоторого факта, имеющего место в пространстве и во времени. Про события говорят, что они «происходят», при этом отдельные события должны быть упорядочены во времени. После наступления некоторого события нельзя уже вернуться к предыдущим событиям, если такая возможность не предусмотрена явно в модели.

В языке UML события играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание зафиксированных промежутков времени или моменты окончания выполнения определенных действий.

Имя события идентифицирует каждый отдельный переход на диаграмме состояний и может содержать строку текста, начинающуюся со строчной буквы. В этом случае принято считать переход триггерным, т. е. таким, который специфицирует событие-триггер.

Сторожевое условие

Сторожевое условие (guard condition), если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское выражение.

Если сторожевое условие принимает значение «истина», то соответствующий переход может сработать, в результате чего объект перейдет в целевое состояние. Если же сторожевое условие принимает значение «ложь», то переход не может сработать, и при отсутствии других переходов объект не может перейти в целевое состояние по этому переходу. Однако вычисление истинности сторожевого условия происходит только после возникновения ассоциированного с ним события-триггера, инициирующего соответствующий переход.

Составное состояние и подсостояние

Составное состояние (composite state) – такое сложное состояние, которое состоит из других вложенных в него состояний. Последние будут высту- пать по отношению к первому как *подсостояния (substate)*. Хотя между ними имеет место отношение композиции, графически все вершины диаграммы, ко- торые соответствуют вложенным состояниям, изображаются внутри символа составного состояния (рис. 7.4).

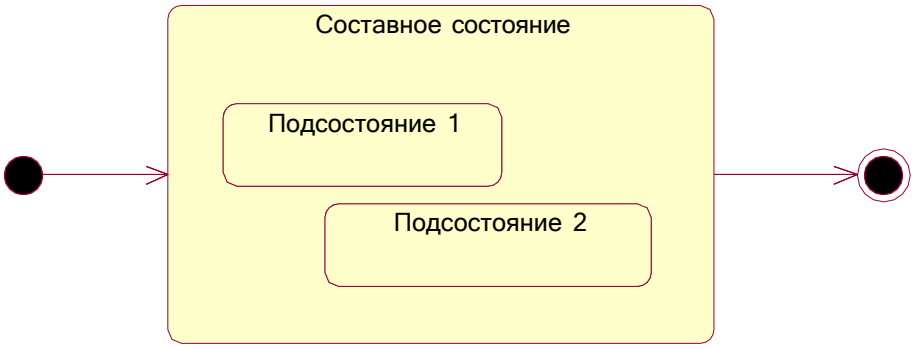


Рис. 7.4. Графическое представление составного состояния

Последовательные подсостояния

Последовательные подсостояния (sequential substates) используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостояний (рис. 7.5).

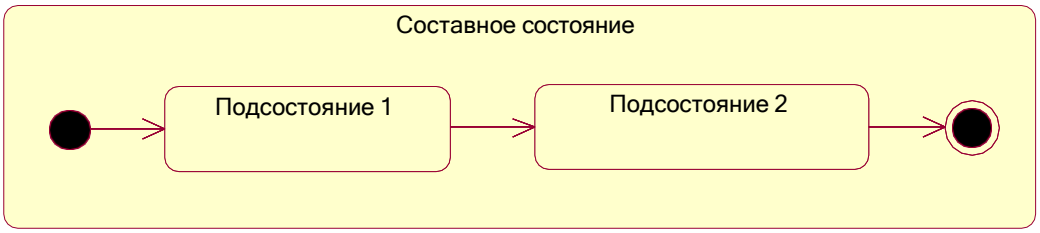


Рис. 7.5. Графическое представление последовательных подсостояний

Составное состояние может содержать в качестве вложенных подсостояний начальное и конечное состояния. При этом начальное подсостояние является исходным, если происходит переход объекта в данное составное состояние. Для последовательных подсостояний начальное и конечное состояния должны быть единственными в каждом составном состоянии.

Параллельные подсостояния

Параллельные подсостояния (concurrent substates) позволяют специфицировать два и более подавтомата (рис. 7.6), которые могут выполняться параллельно внутри составного события. Каждый из подавтоматов занимает некоторую область (регион) внутри составного состояния, которая отделяется от остальных горизонтальной пунктирной линией. Если на диаграмме состояний имеется составное состояние с вложенными параллельными подсостояниями, то объект может одновременно находиться в каждом из этих подсостояний.

Если какой-либо из подавтоматов пришел в свое конечное состояние раньше других, то он должен ожидать, пока и другие подавтоматы не придут в свои конечные состояния.

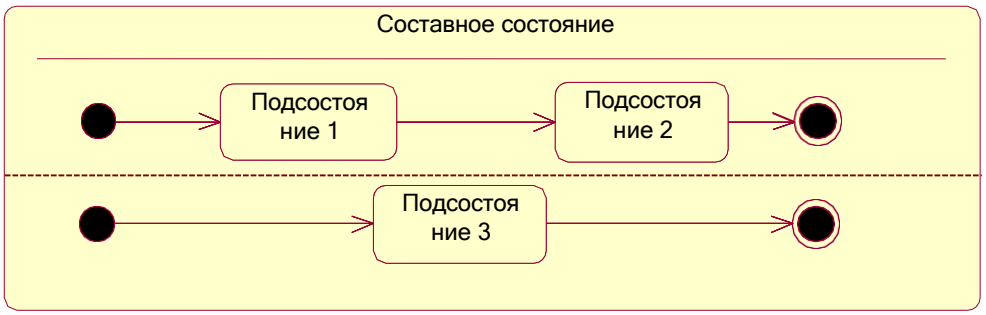


Рис. 7.6. Графическое представление параллельных подсостояний

Историческое состояние

Историческое состояние (history state) применяется в контексте составного состояния. Оно используется для запоминания того из последовательных подсостояний, которое было текущим в момент выхода из составного состояния. При этом существует две разновидности исторического состояния: недавнее и давнее (рис. 7.7).

а) б)

Рис. 7.7. Графическое изображение недавнего и давнего состояния.

Недавнее историческое состояние (shallow history state): при первом переходе в него оно заменяет собой начальное состояние подавтомата и запоминает историю только того подавтомата, к которому он относится и одного с ним уровня вложенности.

Давнее историческое состояние (deep history state) служит для запоминания всех подсостояний любого уровня вложенности для текущего подавтомата.

Составные переходы между состояниями

Современные программные системы могут реализовывать очень сложную логику поведения отдельных своих компонентов, для чего обычного перехода недостаточно. С этой целью в языке UML специфицированы дополнительные обозначения и свойства, которыми могут обладать отдельные переходы на диаграмме состояний.

Составные переходы предназначены для отражения операций распараллеливания и синхронизации. Составной переход может иметь несколько входных и несколько выходных состояний.

Составной переход может выполниться только тогда, когда все его входные состояния активны. После выполнения составного перехода все его выходные состояния становятся активными, а все входные перестают быть активными. Таким образом, автомат может одновременно находиться в нескольких со-

стояниях. Составной переход изображается как прямоугольник (или жирная линия), входные и выходные линии определяют входные и выходные состояния (рис. 7.8).

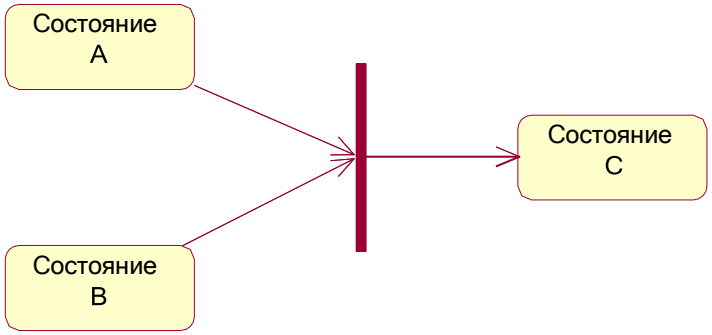


Рис. 7.8. Графическое изображение синхронизирующего перехода

ЛАБОРАТОРНАЯ РАБОТА № 8

Построение диаграмм деятельности и состояний

Цель работы: Изучить правила оформления диаграмм состояний и деятельности. Научится выделять в поведении элемента системы отдельные состояния. С по- мощью диаграммы деятельности научится отображать особенности алгорит- мов, реализующих основные функции системы.

Диаграмма деятельности

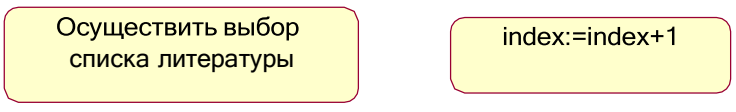
При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Традиционно для этой цели использовались блок-схемы или структурные схемы алгоритмов.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии.

Диаграмма деятельности есть не что иное, как специальный случай диаграммы состояний, в которой все или большинство состояний являются действиями или состояниями под деятельности. А все или большинство переходов являются *не триггерными переходами*, которые *срабатывают* по завершении действий или поддеятельностей в состояниях-источниках. Ниже перечислены основные элементы данной диаграммы.

Состояние действия

Состояние действия является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения алгоритма (процедуры) или потока управления. Графически состояние действия изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами (рис. 8.1). Внутри этой фигуры записывается *выражение действия* (*action- expression*), которое должно быть уникальным в пределах одной диаграммы деятельности.



а) б)

Рис. 8.1. Графическое изображение состояния действия

Переходы

При построении диаграммы деятельности используются только *не триггерные переходы*, т. е. такие, которые *срабатывают* сразу после завершения деятельности или выполнения соответствующего действия. Этот переход переводит деятельность в последующее состояние сразу, как только закончится действие в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой.

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 прямых 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, а для ее обозначения применяется специальный символ.

Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста (рис. 5.10). В этот ромб может входить только одна стрелка от того состояния действия, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей. Принято входящую стрелку присоединять к верхней или левой вершине символа ветвления. Выходящих стрелок может быть две или более, но для каждой из них явно указывается соответствующее сторожевое условие в форме булевского выражения.

В качестве примера рассмотрим фрагмент входа пользователя в систему (рис. 8.2). При входе в систему пользователю предлагается ввести свое имя.

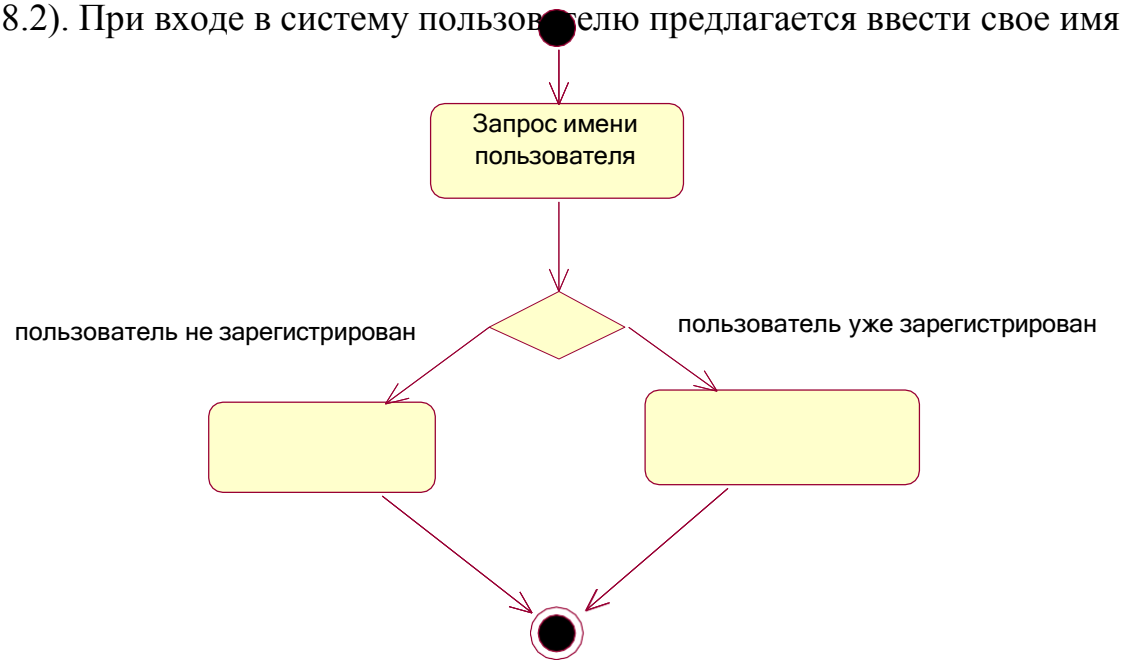


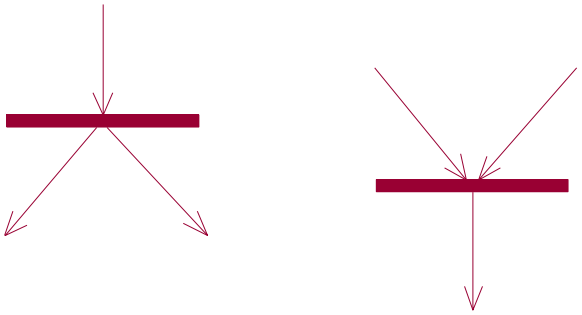
Рис. 8.2. Фрагмент диаграммы деятельности, использующей ветвление

Далее система проверяет, есть ли в системе зарегистрированный пользователь с таким именем. Если да, то происходит проверка его пароля. Если же пользователь еще не зарегистрирован, то ему предлагается регистрация.

Параллельные процессы

Для представления параллельных процессов в языке UML используется специальный символ для разделения и слияния параллельных вычислений или потоков управления (рис. 8.3). Таким символом является прямая черточка, аналогично обозначению перехода в формализме сетей Петри.

Как правило, такая черточка изображается отрезком горизонтальной линии, толщина которой несколько шире основных сплошных линий диаграммы деятельности. При этом *разделение (concurrent fork)* имеет один входящий переход и несколько выходящих (рис. 8.3, а). *Слияние (concurrent join)*, наоборот, имеет несколько входящих переходов и один выходящий (рис. 8.3, б).



а) б)

Рис. 8.3. Графическое изображение разделения (а) и слияния (б)

Дорожки

Важная область применения диаграмм деятельности связана с моделированием бизнес-процессов. Деятельность любой компании есть совокупность отдельных действий, направленных на достижение требуемого результата. Применительно к бизнес-процессам выполнение каждого действия желательно связывать с конкретным подразделением компании, которое несет 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, получившее название *дорожки (swimlanes)*. Имеется в ви-

ду визуальная аналогия с плавательными дорожками в бассейне, если смотреть на соответствующую диаграмму. При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением (отделом, группой, отделением, филиалом) компании (рис. 8.4).

Названия подразделений явно указываются в верхней части дорожки. Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение компании. Порядок следования дорожек не несет какой-либо семантической информации и определяется соображениями удобства.

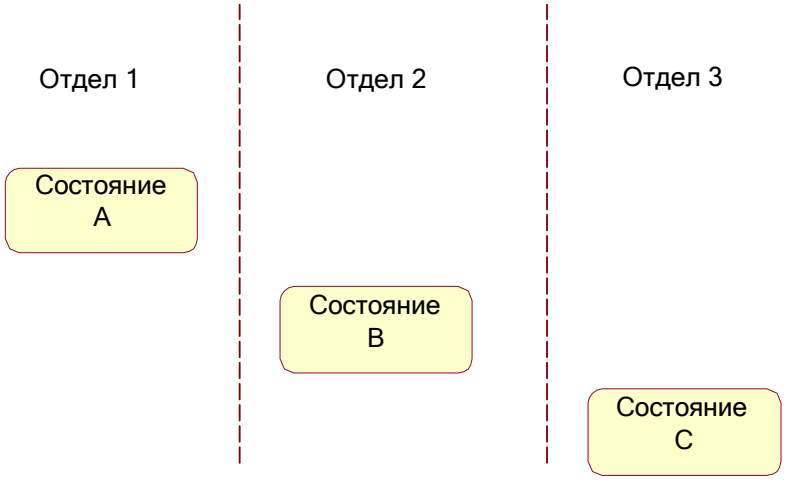


Рис. 8.4. Вариант диаграммы деятельности с дорожками

В качестве примера ниже (рис. 8.5) рассмотрен фрагмент диаграммы деятельности internet-магазина, обслуживающего клиента. В данном случае подразделениями являются веб-интерфейс магазина, менеджер по продажам и склад.

Этим подразделениям будут соответствовать три дорожки на диаграмме деятельности, каждая из которых специфицирует зону ответственности подразделения. В данном случае диаграмма деятельности заключает в себе не только информацию о последовательности выполнения рабочих действий, но и о том, какое из подразделений торговой компании должно выполнять то или иное действие (см. рис. 8.5).

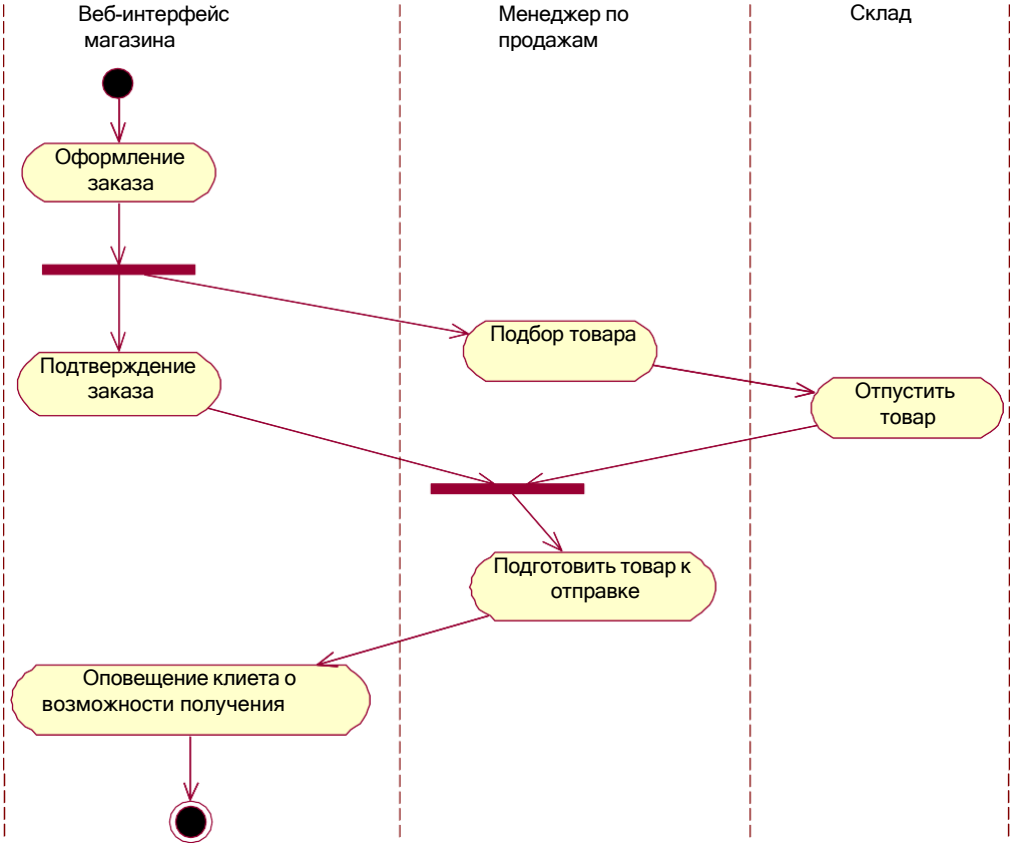


Рис. 8.5. Фрагмент диаграммы деятельности для internet-магазина

Объекты

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому, поэтому иногда возникает необходимость явно указать их на диаграмме деятельности.

Для графического представления объектов используются прямоугольник класса, с тем отличием, что имя объекта подчеркивается. Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой. Соответствующая зависимость определяет состояние конкретного объекта после выполнения предшествующего действия.

На диаграмме деятельности с дорожками расположение объекта может иметь некоторый дополнительный смысл. Например, если объект расположен на границе двух дорожек, то это может означать, что переход к следующему состоянию действия в соседней дорожке ассоциирован с готовностью некоторого документа (объект в некотором состоянии). Если же объект целиком расположен внутри дорожки, то и состояние этого объекта целиком определяется действиями данной дорожки.

Возвращаясь к предыдущему примеру с internet-магазином, можно заметить, что центральным объектом процесса продажи является заказ или вернее состояние его выполнения. Вначале, до получения, заказ как объект отсутствует

и возникает лишь после его оформления на веб-сайте. Однако этот заказ еще не оформлен до конца, поскольку менеджер по продажам должен еще подобрать конкретный товар. После его подготовки он передается на склад, где вместе с отпуском товара заказ окончательно оформляется. Наконец, после получения подтверждения эта информация заносится в заказ, и он считается выполненным и закрытым. Данная информация может быть представлена графически в виде модифицированного варианта диаграммы деятельности этого же internet-магазина (рис. 8.6).

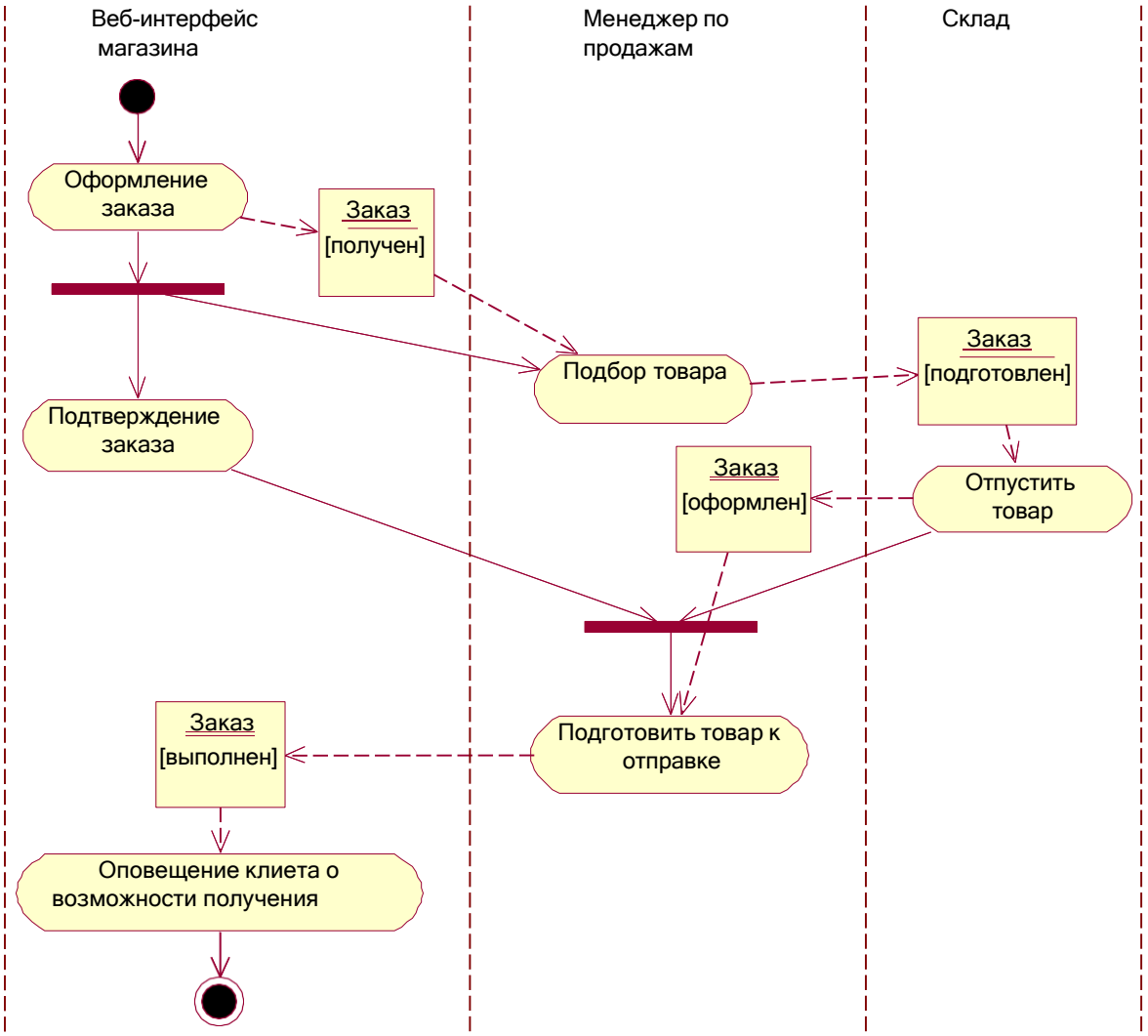


Рис. 8.6. Фрагмент диаграммы деятельности для internet-магазина

Рекомендации по построению диаграмм поведения Диаграмма состояний

По своему назначению диаграмма состояний не является обязательным представлением в модели и как бы «присоединяется» к тому элементу, который

имеет нетривиальное поведение в течение своего жизненного цикла. Наличие у системы нескольких нетривиальных состояний, отличающихся от «исправен – неисправен», служит признаком необходимости построения диаграммы состояний.

В качестве начального варианта диаграммы состояний, если нет очевидных соображений по поводу состояний объекта, можно воспользоваться этими суперсостояниями, рассматривая их как составные и детализируя их внутреннюю структуру по мере рассмотрения логики поведения объекта. При разработке диаграммы состояний нужно постоянно следить, чтобы объект в каждый момент мог находиться только в единственном состоянии. Если это не так, то данное обстоятельство может быть как следствием ошибки, так и неявным признаком наличия параллельности у поведения моделируемого объекта.

Диаграмма деятельности

Диаграммы деятельности играют важную роль в понимании процессов реализующих алгоритмы выполнения операций классов и потоков управления в моделируемой системе.

Содержание диаграммы деятельности во многом напоминает диаграмму состояний, хотя и не тождественно ей. В частности, эта диаграмма строится для отдельного класса, варианта использования, отдельной операции класса или целой подсистемы.

В случае типового проекта большинство деталей реализации действий могут быть известны заранее на основе анализа существующих систем или предшествующего опыта разработки систем-прототипов. Использование типовых решений может существенно сократить время разработки и избежать возможных ошибок при реализации проекта.

При разработке проекта новой системы, процесс функционирования которой основан на новых технологических решениях, ситуация более сложная. А именно, до начала работы над проектом могут быть неизвестны не только детали реализации отдельных деятельностей, но и само содержание этих деятельностей становится предметом разработки.

5.1 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы состояний.
3. Спецификация диаграмм состояний.
4. Разработанные диаграммы деятельности.
5. Выводы.

5.2 Контрольные вопросы

1. Назначение диаграммы состояний.
2. Особенности диаграммы состояний.
3. Элементы диаграммы состояний. Состояния.
4. Элементы диаграммы состояний. Переходы.
5. Диаграмма деятельности.

ЛАБОРАТОРНАЯ РАБОТА № 9

Построение диаграммы компонентов

Цель работы

Изучить правила оформления диаграммы компонентов. Научится разрабатывать конкретную реализацию проекта в форме программного кода

Теоретические сведения

Представление компонентов

Все рассмотренные ранее диаграммы относились к логическому уровню представления. Особенность логического представления заключается в том, что различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически, они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Полный проект программной системы представляет собой совокупность моделей логического и физического представлений, которые должны быть согласованы между собой. В языке UML для физического представления моделей систем используются так называемые диаграммы реализации (*implementation diagrams*), которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания.

Диаграмма компонентов *(Component Diagram)* описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма компонентов разрабатывается для следующих целей:

1. Визуализации общей структуры исходного кода программной системы.
2. Спецификации исполнимого варианта программной системы.
3. Обеспечения многократного использования отдельных фрагментов программного кода.
4. Представления концептуальной и физической схем баз данных.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов. Пример диаграммы компонентов дан на рис. 6.1.

Основные элементы диаграммы перечислены ниже.

Компоненты

Для представления физических сущностей в языке UML применяется специальный термин – компонент (*component*). Компонент реализует некото-

рый набор интерфейсов и служит для общего обозначения элементов физического представления модели.

Компонент предоставляет организацию в рамках физического пакета ассоциированным с ним элементам модели. Как классификатор, компонент может иметь также свои собственные свойства, такие как атрибуты и операции.

Зависимости

Зависимость *(Dependency)* не является ассоциацией, а служит для представления только факта наличия такой связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели. Отношение зависимости на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу).

Зависимости могут отражать связи модулей программы на этапе компиляции и генерации объектного кода. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

Также на диаграмме могут быть представлены отношения зависимости между компонентами и реализованными в них классами. Эта информация имеет значение для обеспечения согласования логического и физического представлений модели системы.

Кроме того диаграмма компонентов (рис. 9.1) может содержать:

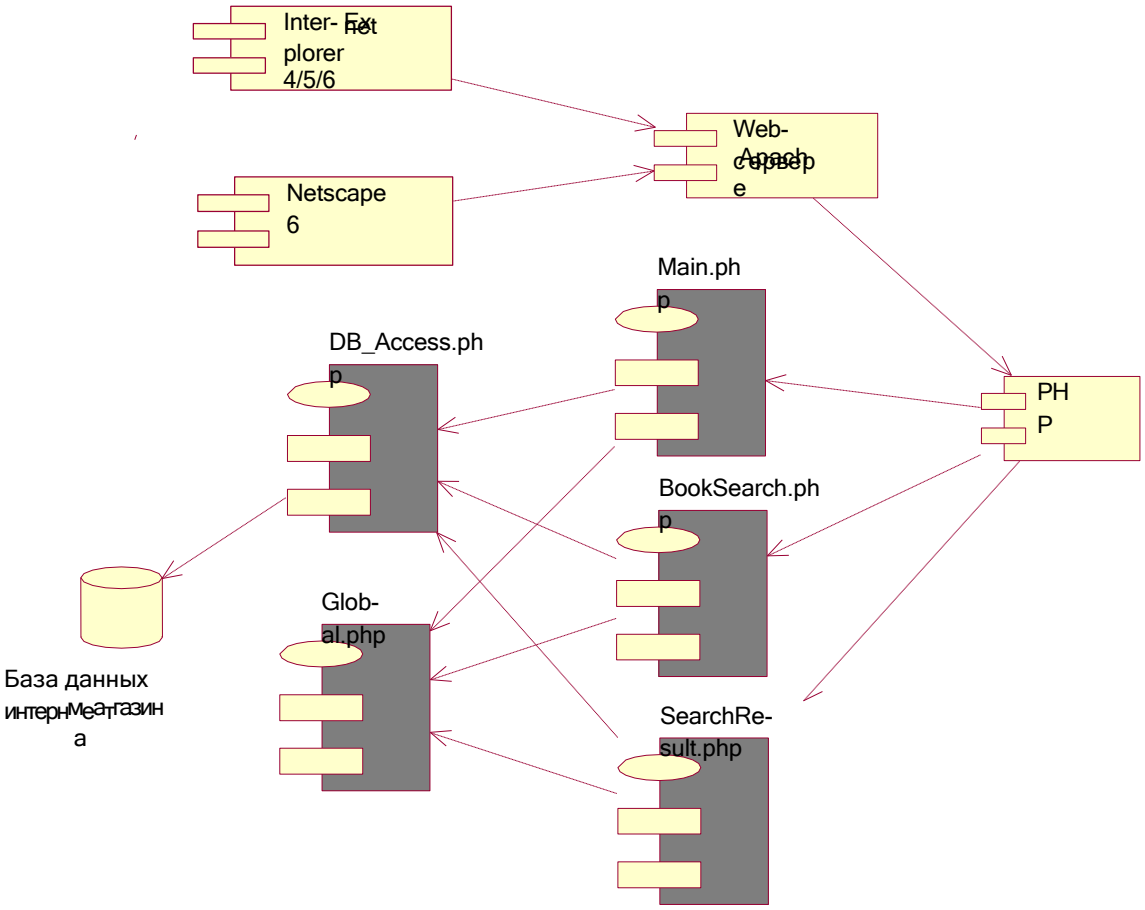


Рис. 9.1. Пример диаграммы компонентов

1. *Контейнер (Package)* позволяет отобразить контейнер, который объединяет группу компонентов в модели.
2. *Главная программа (Main program)* позволяет добавить в модель компонент, обозначающий главную программу.
3. *Тело подпрограммы (Subprogram body)* позволяет добавить в модель компонент, обозначающий тело подпрограммы, и используется также для необъектно-ориентированных компонентов.
4. *Определение/тело контейнера (Package specification/body)* отобразить определения контейнера (Package specification) и описание контейнера (Package body), которые обычно связаны между собой. Для языка C++ Package specification – это заголовочный файл с расширением .h, а Package body – это файл с расширением .cpp.
5. *Определение/тело задачи (Task specification/body)* позволяют отобразить независимые потоки в многопотоковой системе.

6.1.1 Рекомендации по построению диаграммы компонентов

Разработка диаграммы компонентов предполагает использование информации, как о логическом представлении модели системы, так и об особенностях ее физической реализации. До начала разработки необходимо определиться с выбором языковой платформы и операционной системы.

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счет рациональной декомпозиции компонентов, но и создание объектов только при их необходимости.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных. При разработке интерфейсов следует обращать внимание на согласование различных частей программной системы. Включение в модель схемы базы данных предполагает спецификацию отдельных таблиц и установление информационных связей между таблицами.

6.2 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанная диаграмма компонентов.
3. Спецификация диаграммы компонентов.
4. Выводы.

6.3 Контрольные вопросы

1. Физическая модель программной системы.
2. Назначение диаграммы компонентов.
3. Цели разработки диаграммы компонентов.
4. Элементы диаграммы компонентов. Компоненты.
5. Элементы диаграммы компонентов. Зависимости.

ЛАБОРАТОРНАЯ РАБОТА № 10

Диаграмма развертывания

Цель работы

Изучить правила оформления диаграммы вариантов использования. Научится выделять особенности функционального поведения проектируемой системы.

Теоретические сведения

Диаграмма развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Этому есть несколько причин:

- сложные программные системы могут реализовываться в сетевом варианте на различных вычислительных платформах и технологиях доступа к распределенным базам данных;
- интеграция программной системы с Интернетом определяет необходимость решения дополнительных вопросов при проектировании системы, таких как обеспечение безопасности, устойчивости доступа к информации и т. д.
- технологии доступа и обработки данных в схеме «клиент-сервер» требует размещения больших баз данных в различных сегментах корпоративной сети, их резервного копирования для обеспечения необходимой производительности системы в целом.

Диаграмма развертывания (синоним – диаграмма размещения) применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры програм- мы, являющиеся исполнимыми файлами или динамическими библиотеками.

Итак, перечислим цели, преследуемые при разработке диаграммы развертывания:

- определить распределение компонентов системы по ее физическим узлам;
- показать физические связи между всеми узлами реализации системы на этапе ее исполнения;
- выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности. Основные элементы диаграммы перечислены ниже.

Узел

Узел (*node*) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом. В качестве вычислительного ресурса узла может рассматриваться наличие хотя бы некоторого объема электронной памяти и/или процессора. В последней версии языка UML понятие узла расширено и может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства.

Разрешено показывать на диаграмме развертывания узлы с вложенными изображениями компонентов. Важно помнить, что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты.

Соединения

Кроме собственно изображений узлов на диаграмме развертывания указываются отношения между ними. В качестве отношений выступают физические соединения между узлами и зависимости между узлами и компонентами, изображения которых тоже могут присутствовать на диаграммах развертывания.

Соединения являются разновидностью ассоциации и изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими узлами. Характер соединения может быть дополнительно специфицирован примечанием, помеченным значением или ограничением.

Диаграммы развертывания могут иметь более сложную структуру, включающую вложенные компоненты, интерфейсы и другие аппаратные устройства.

Диаграмма включает в себя следующие ***элементы***:

- *Процессор (Processor)* – это устройство, способное выполнять программы. Процессор обязательно должен иметь свое имя, которое, однако, никак не связано с другими диаграммами модели по причине того, что процессор обозначает не программное обеспечение, а аппаратуру.
- *Устройство (Device).* Данный инструмент позволяет создавать на диаграмме объект устройства, неспособного выполнять программы. Каждое такое устройство также относится к аппаратному обеспечению и должно иметь общее для данного вида имя, такое как «модем» или «терминал».
- *Соединение (Connection).* Данный инструмент позволяет связать между собой устройства и процессоры. Соединение представляет собой некоторый тип кабельного или другого соединения, например, соединение при помощи сетевых карт, последовательных или параллельных портов или даже связь «Земля – спутник». В отличие от реального соединения, на диаграмме не может быть показано направление перемещения информации посредством соединения, и считается, что соединение всегда двунаправленно.

Пример диаграммы приведен на рис. 7.1.

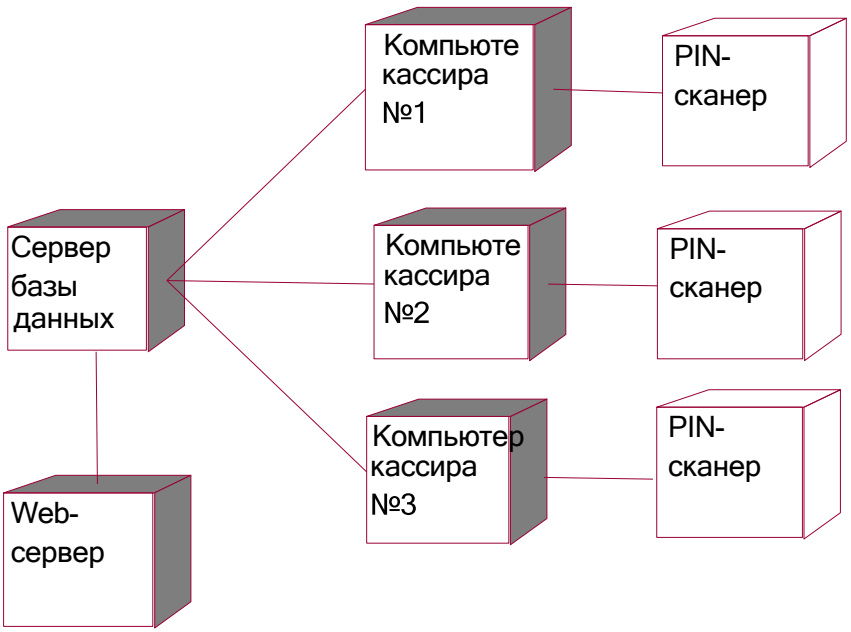


Рис. 10.1. Пример диаграммы развертывания

Рекомендации по построению диаграммы развертывания

Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций. В первую очередь специфицируются вычислительные узлы системы, обладающие памятью и/или процессором.

Дальнейшее построение диаграммы развертывания связано с размещением всех исполняемых компонентов диаграммы по узлам системы. Если отдельные исполняемые компоненты оказались не размещенными, то подобная ситуация должна быть исключена введением в модель дополнительных узлов, содержащих процессор и память.

При разработке простых программ, которые исполняются локально на одном компьютере, так же как и в случае диаграммы компонентов, необходимость в диаграмме развертывания может вообще отсутствовать. В более сложных ситуациях диаграмма развертывания строится для таких приложений, как:

- моделирование программных систем, реализующих технологию доступа к данным «клиент-сервер»;
- моделирование неоднородных распределенных архитектур (корпоративные сети);
- моделирование системы со встроенными микропроцессорами, которые могут функционировать автономно.

Как правило, разработка диаграммы развертывания осуществляется на завершающем этапе ООАП, что характеризует окончание фазы проектирования физического представления. С другой стороны, диаграмма развертывания может строиться для анализа существующей системы с целью ее последующего анализа и модификации. При этом анализ предполагает разработку этой диаграммы на его начальных этапах, что характеризует общее направление анализа от физического представления к логическому.

7.2 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанная диаграмма развертывания.
3. Спецификация диаграммы развертывания.
4. Выводы.

7.3 Контрольные вопросы

1. Назначение диаграммы развертывания.
2. Цели разработки диаграммы развертывания.
3. Элементы диаграммы развертывания. Узел.
4. Элементы диаграммы развертывания. Соединения.

2 ОБЩАЯ ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Самостоятельная работа - целенаправленная, планируемая в рамках учебного плана деятельность студентов, которая осуществляется по заданию, при методическом руководстве и контроле преподавателя, но без его непосредственного участия. Самостоятельная работа студентов является одной из важнейших составляющих образовательного процесса.

В учебном процессе учебного заведения выделяют два вида самостоятельной работы: аудиторная и внеаудиторная.

Аудиторная самостоятельная работа выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию.

Внеаудиторная — планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

Целью самостоятельной работы студентов является:

- систематизация и закрепление полученных теоретических знаний и практических умений;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов, творческой инициативы, самостоятельности, ответственности, организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, совершенствованию и самоорганизации;
- формирование общих и профессиональных компетенций.

Самостоятельная работа студентов должна быть хорошо спланирована и организована. При планировании такой работы необходимо учитывать условия, обеспечивающие её успешное выполнение:

- чёткое определение преподавателем объёма и содержания самостоятельной работы;
- определение видов консультативной помощи;
- постановка цели самостоятельной работы и критерии её оценки;
- виды и формы контроля её выполнения.

Выполняя самостоятельную работу под контролем преподавателя, студент должен:

- освоить минимум знаний;
- планировать свою самостоятельную работу в соответствии разработанным графиком;
- выполнять самостоятельную работу и отчитываться по ее результатам в соответствии с графиком представления результатов, видами и сроками отчетности по самостоятельной работе студентов.

В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, саморефлексии и становится активным самостоятельным субъектом учебной деятельности.

Таким образом, самостоятельная работа студентов оказывает важное влияние на формирование личности будущего специалиста.

Самостоятельная работа студентов является обязательной для каждого студента, объем ее определяется учебным планом в соответствии с требованиями Государственных образовательных стандартов.

При изучении тем дисциплины студенты выполняют следующие виды самостоятельной работы:

- проработка конспектов занятий, учебных изданий и специальной технической литературы;
- составление конспекта, тематических схем, таблиц;
- подготовка к лабораторным работам и практическим занятиям с использованием методических рекомендаций преподавателя;
- оформление отчетов по лабораторным работам и практическим занятиям, подготовка к их защите;
- моделирование и решение производственных процессов и ситуационных задач;
- подготовка презентаций;
- работа с электронными ресурсами в сети Интернет;
- подготовка к семинару;
- подготовка к зачетам, экзаменам.

Технология организации самостоятельной работы студентов включает использование информационных и материально-технических ресурсов образовательного учреждения. Материально-техническое и информационно - техническое обеспечение самостоятельной работы студентов включает в себя:

- библиотеку с читальным залом, укомплектованную в соответствии с существующими нормами;
- учебно-методическую базу учебных кабинетов, лабораторий и методического центра;
- компьютерные классы с возможностью работы в Интернет;
- базы практики в соответствии с заключенными договорами;
- аудитории для консультационной деятельности;
- учебную и учебно-методическую литературу, разработанную с учетом увеличения доли самостоятельной работы студентов, и иные методические материалы.

Перед выполнением внеаудиторной самостоятельной работы преподаватель проводит инструктаж по выполнению задания, в котором указывает цель задания, его содержание, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. Во время выполнения студентами внеаудиторной самостоятельной работы и при необходимости преподаватель может проводить консультации. Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений обучающихся.

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Общие методические рекомендации студенту при изучении тем дисциплины.

Большая часть самостоятельной работы выполняется студентом вне учебных занятий при подготовке домашних заданий. Общие требования к выполнению этого вида самостоятельной работы заключаются в следующем:

- активно работать на уроке, усваиваю основную часть нового материала;
- если что-то непонятно, не стесняться задавать вопросы преподавателю;
- большое задание необходимо разбивать на части и работать над каждой из них в отдельности;
- выполняя домашнее задание, надо не просто думать, что надо сделать, а еще и решать, с помощью каких средств и приемов этого можно добиться;
- в процессе приготовления домашнего задания необходимо делать перерывы;
- готовиться к докладам, рефератам, защите курсовых работ и проектов, практических и лабораторных занятий надо заранее, равномерно распределяя нагрузку, а не оставлять такую ответственную работу на последний день;
- изучая заданный материал, сначала надо его понять, а уже потом запомнить;
- научиться находить интересующую нужную информацию с помощью компьютера;
- не стесняться обращаться за помощью к взрослым и однокурсникам;
- надо составлять план устного ответа и проверять себя;
- на письменном столе должно лежать только то, что необходимо для выполнения одного задания. После его завершения со стола убираются уже использованные материалы, и кладутся те учебные принадлежности, которые необходимы для выполнения следующего задания;
- нужно решить, в какой последовательности лучше выполнять задания и сколько времени понадобится на каждое из них;
- трудный материал урока лучше повторить в тот же день, чтобы сразу закрепить его и запомнить;
- читая учебник, надо задавать самому себе вопросы по тексту.

Подготовка тематических сообщений, докладов, рефератов

Реферат доклад, сообщение (от латинского refero - передаю, сообщаю) - краткое письменное изложение материала по определенной теме с целью привития студентам навыков самостоятельного поиска и анализа информации, формирования умения подбора и изучения литературных источников, используя при этом дополнительную научную, методическую и периодическую литературу.

Тема реферата выбирается по желанию студента из списка, предлагаемого преподавателем. Тема может быть сформулирована студентом самостоятельно.

Выбранная тема согласовывается с преподавателем.

После выбора темы требуется:

- составить план реферата;
- подобрать необходимую информацию;

- изучить подобранную информацию;
- составить текст реферата.

План реферата должен включать в себя введение, основной текст и заключение. Во введении аргументируется актуальность выбранной темы, указываются цели и задачи исследования. В нем также отражается методика исследования и структура работы. Основная часть работы предполагает освещение материала в соответствии с планом. В заключении излагаются основные выводы и рекомендации по теме исследования.

Реферат оформляется согласно требованиям, установленным в учебном заведении. Он должен содержать: титульный лист, оглавление и список использованной литературы. На титульном листе указываются: название учебного заведения, название профессионального модуля, междисциплинарного курса, тема работы, курс, группа, фамилии, имена, отчества студента и руководителя работы, название города, в котором находится учебное заведение, год написания данной работы. Реферат может содержать приложения в форме схем, образцов документов и другие изображения в соответствии с темой исследования. Все страницы работы, включая оглавление и список литературы, нумеруются по порядку с титульного листа (на нем цифра не ставится) до последней страницы без пропусков и повторений. Введение, заключение, новые главы, список использованных источников и литературы должны начинаться с нового листа. Подбор литературы производится студентом из предложенного преподавателем списка литературы. Текст реферата необходимо набирать на компьютере на одной стороне листа. Размер левого поля 30 мм, правого - 15 мм, верхнего - 20 мм, нижнего - 20 мм. Шрифт - Times New Roman, размер - 14, межстрочный интервал - 1,5. Фразы, начинающиеся с новой строки, печатаются с абзацным отступом от начала строки (1,25 см). Реферат, выполненный небрежно, неразборчиво, без соблюдения требований по оформлению, возвращается студенту без проверки с указанием причин возврата на титульном листе.

Критерии оценки:

- знание и понимание проблемы;
- умение систематизировать и анализировать материал, четко и обоснованно формулировать выводы;
- «трудозатратность» (объем изученной литературы, добросовестное отношение к анализу проблемы);
- самостоятельность, способность к определению собственной позиции по проблеме и к практической адаптации материала, недопустимость плагиата;
- выполнение необходимых формальностей (точность в цитировании и указании источника текстового фрагмента, аккуратность оформления).

Проработка занятый, учебных изданий и специальной технической литературы

Работа с конспектом лекций по темам междисциплинарных курсов заключается в том, что студент после рассмотрения темы на учебных занятиях в период между очередными лекциями изучает материал конспекта. При этом непонятные положения конспекта необходимо выяснять у преподавателя на консультациях или при чтении основной и дополнительной литературы.

При работе с книгой необходимо научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги. Правильный подбор учебников рекомендуется преподавателем. Необходимая литература может быть также указана в методических разработках. Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и определения (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода). Полезно составлять опорные конспекты. При изучении материала по учебнику, полезно в тетради (на специально отведенных полях) дополнять конспект лекций, написанный на учебных занятиях. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем. Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при пропитывании записей лучше запоминались. Различают два вида чтения; первичное и вторичное. Первичное - эти внимательное, неторопливое чтение, при котором можно остановиться на трудных местах. После него не должно остаться ни одного непонятного слова. Содержание не всегда может быть понятно после первичного чтения. Задача вторичного чтения - полное усвоение смысла целого (по счету это чтение может быть и не вторым, а третьим или четвертым).

Чтение научного текста является частью познавательной деятельности. Ее цель - извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия. Выделяют четыре основные установки в чтении научного текста:

- информационно-поисковая, задача которой - найти, выделить искомую информацию;
- усваивающая, при которой усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения, излагаемые автором, так и всю логику его рассуждений;
- аналитико-критическая - читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему;
- творческая, создающая у читателя готовность в том или ином виде использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке.

Самостоятельная работа при чтении учебной литературы начинается с изучения конспекта материала, полученного при слушании лекций преподавателя. Полученную информацию необходимо осмыслить. При необходимости, в конспект лекций могут быть внесены схемы, эскизы рисунков, другая дополнительная информация.

Составление конспекта, тематических схем, таблиц

При изучении нового материала, как правило, составляется конспект. Конспект - изложение текста, которому присущи краткость, связность и

последовательность. При этом максимально точно записываются формулы, определения, схемы, трудные для запоминания места.

При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре текста. Для уточнения и дополнения необходимо оставлять поля. Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

Классификация конспектов:

- плановый конспект, для чего сначала нужно написать план текста, а затем на пункты плана делаются комментарии: свободно изложенный текст либо цитаты;
- обзорный конспект - краткое изложение данной темы с использованием нескольких источников;
- текстуальный конспект состоит из цитат одного текста;
- свободный конспект предполагает цитаты текста и собственные формулировки прочитанного текста;
- сложный - конспект, в котором отражается определенная тема или вопрос;
- хронологический конспект отражает последовательность событий;
- опорный конспект, в котором излагается информация в виде опорных знаков, слов, сигналов.

Методические рекомендации по составлению конспекта:

- определить цель написания конспекта;
- внимательно прочитать текст, уточнить в справочной литературе непонятные слова;
- выделить основные смысловые части текста;
- определить главное, составить план;
- кратко сформулировать основные положения текста, отметить аргументацию автора;
- составить текст конспекта, изложив информацию кратко и своими словами, четко следуя пунктам плана, записи следует вести четко, ясно;
- грамотно записывать цитаты, учитывая лаконичность, значимость мысли;
- в тексте конспекта желательно приводить не только тезисные положения, но и их доказательства.

При составлении тематических схем, таблиц необходимо внимательно прочитать текст соответствующий параграф учебника. Продумать «конструкцию» таблицы или схемы, расположение порядковых номеров, терминов, примеров и пояснений (и прочего). Начертить схему или таблицу и заполнить ее графы необходимым содержимым.

Подготовка к лабораторным работам и практическим занятиям, оформление отчетов по лабораторным работам и практическим занятиям, подготовка к их защите

Программы профессиональных модулей предусматривают выполнение практических и лабораторных занятий.

Лабораторное занятие - форма учебного занятия, ведущей дидактической целью которого является экспериментальное подтверждение и проверка существующих теоретических положений (законов, зависимостей), формирование учебных и профессиональных практических умений и навыков.

Практическое занятие - это одна из форм учебной работы, которая ориентирована на закрепление изученного теоретического материала, его более глубокое усвоение и формирование умения применять теоретические знания в практических целях. Особое внимание на практических занятиях уделяется выработке учебных или профессиональных навыков. Такие навыки формируются в процессе выполнения конкретных заданий - упражнений, задач - под руководством и контролем преподавателя.

Подготовка к практическим и лабораторным занятиям заключается в работе с конспектом лекций по данной теме, в изучении соответствующего раздела учебника или учебного пособия, в просмотре дополнительной литературы. Этапы подготовки к практическому или лабораторному занятию заключаются в следующем: освежить в памяти теоретические сведения, полученные на лекциях и в процессе самостоятельной работы, подобрать необходимую учебную и справочную литературу. Отобрать те материалы, которые позволят в полной мере реализовать цели и задачи предстоящей работы. Еще раз проверить соответствие отобранного материала. Студент должен прийти на лабораторное или практическое занятие подготовленным по данной теме.

При выполнении заданий практического или лабораторного занятия студент должен быть ознакомлен преподавателем с целью и ходом выполнения задания и, по необходимости, с правилами техники безопасности. Если у студентов во время выполнения заданий возникают вопросы, то преподаватель консультирует студентов. Порядок выполнения того или иного задания излагается в инструкционных картах или рабочих тетрадях.

После проведения занятия студент представляет письменный отчет, который оформляется в соответствие с принятыми в образовательном учреждении правилами. Отчеты оформляются на листах писчей бумаги формата А4 или в специальных рабочих тетрадях, разработанных преподавателем. Содержание отчета указано в инструкционных картах или рабочих тетрадях.

При подготовке к защите практических и лабораторных занятий студент должен ответить на контрольные вопросы, указанные также в инструкционных картах или рабочих тетрадях, проштудировав при этом конспект лекций, учебную литературу.

Моделирование и решение производственных процессов и ситуационных задач

При изучении дисциплины очень часто студенту приходится сталкиваться с профессиональными задачами и ситуациями, которые необходимо решить самостоятельно, как во время аудиторной работы, так и во время внеаудиторной. При решении таких задач необходимо:

- провести анализ ситуации для определения проблемы в целом; представить ситуацию и себя в качестве действующего в ней лица; проанализировать ошибочные или правильные действия всех участников ситуации;
- определить проблемные узлы - возможные причины и прогнозируемые последствия развития данной ситуации;
- рассмотреть условное прогнозирование развития ситуации: определить окончательную гипотезу, представить обоснованный и доказательный прогноз вероятностного развития ситуации; предложить варианты действий, обоснованные теоретически и, по возможности, подкрепленные практическим личным опытом, опираясь на принципы профессиональной этики; определить способы и методы воздействия на предлагаемую ситуацию;
- сформулировать итоговые выводы, используя профессиональные термины, доказательства правильности своего решения.

Подготовка презентаций

Подготовка презентации позволит студенту логически выстроить изучаемый материал, систематизировать его, сформировать коммуникативные компетенции. Материал презентации представляется в виде текста, схем, диаграмм, таблиц, которые призваны дополнить текстовую информацию или передать ее в более наглядном виде. Желательно избегать в презентации изображений, не несущих смысловой нагрузки, если они не являются частью стилевого оформления. Цвет графических изображений не должен резко контрастировать с общим стилевым оформлением слайдов, иллюстрации рекомендуется сопровождать пояснительным текстом.

Анимационные эффекты используются для привлечения внимания слушателей или для демонстрации динамики развития какого - либо процесса. В этих случаях использование анимации оправдано, но не стоит чрезмерно насыщать презентацию такими эффектами, иначе это вызовет негативную реакцию аудитории.

Звуковое сопровождение должно отражать суть или подчеркивать особенность темы слайда, презентации. Фоновая музыка не должна отвлекать внимание слушателей и заглушать слова докладчика.

Оптимальное количество слайдов, как правило, десять - пятнадцать. Для оформления слайдов презентации рекомендуется использовать несложные шаблоны, соблюдать единый стиль. Не рекомендуется на одном слайде использовать более трех цветов. Смену слайдов для управления презентацией докладчиком желательно устанавливать по щелчку без времени. Шрифт, выбираемый для презентации, должен обеспечивать читаемость информации на экране и соответствовать выбранному шаблону оформления. Не желательно использовать разные шрифты в одной презентации.

Алгоритм выстраивания презентации должен соответствовать логической структуре работы и отражать последовательность ее этапов. Независимо от алгоритма выстраивания презентации на первом слайде рекомендуется выносить следующие данные: полное наименование образовательной организации; тема презентации; фамилия, имя, отчество студента; специальность обучения; фамилия, имя, отчество руководителя. Последний слайд должен содержать фразу «Спасибо за внимание».

Работа с электронными ресурсами в сети Интернет

Для повышения эффективности самостоятельной работы студент должен учиться работать в поисковой системе сети Интернет, в электронно-библиотечной системе и использовать найденную информацию при подготовке к занятиям.

Интернет сегодня - правомерный источник научных статей, статистической и аналитической информации, и использование его наряду с книгами давно уже стало нормой. Однако, несмотря на то, что ресурсы Интернета позволяют достаточно быстро и эффективно осуществлять поиск необходимой информации, следует помнить о том, что эта информация может быть неточной или вовсе не соответствовать действительности. В связи с этим при поиске материала по заданной тематике следует обращать внимание на научные труды признанных авторов, которые посоветовали вам преподаватели.

Поиск информации можно вести по автору, заглавию, виду издания, году издания или издательству. Также в сети Интернет доступна услуга по скачиванию методических указаний и учебных пособий, подбору необходимой учебной и научно - технической литературы.

Подготовка к семинару

Семинар — это особая форма учебно-теоретических занятий, которая, как правило, служит дополнением к лекционному курсу. Семинар обычно посвящен детальному изучению отдельной темы.

Этапы подготовки к семинару:

- проанализировать тему семинара, подумать о цели и основных проблемах, вынесенных на обсуждение;
- внимательно прочитать материал, данный преподавателем по этой теме на лекции;
- изучить рекомендованную литературу, делая при этом конспекты прочитанного или выписки, которые понадобятся при обсуждении на семинаре;
- постараться сформулировать свое мнение по каждому вопросу и аргументированно его обосновать;
- записать возникшие во время самостоятельной работы с учебниками и научной литературой вопросы, чтобы затем на семинаре получить на них ответы.

При подготовке к семинарским занятиям следует руководствоваться указаниями и рекомендациями преподавателя, использовать основную и дополнительную литературу из представленного им списка.

При подготовке доклада на семинарское занятие желательно заранее обсудить с преподавателем перечень используемой литературы, за день до семинарского занятия предупредить его о необходимых для представления материала технических средствах. Напечатанный текст доклада представить преподавателю на рецензию.

Подготовка к зачетам, экзаменам

Изучение выше перечисленных тем дисциплины завершается зачетами или экзаменами.

Подготовка к зачету или экзамену способствует закреплению, углублению и обобщению знаний, получаемых в процессе обучения, а также применению их к

решению практических задач. Готовясь к зачету или экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На зачете или экзамене студент демонстрирует то, что он приобрел в процессе обучения конкретным темам междисциплинарных курсов или модулям в целом.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами, согласно графику их проведения, дается интервал времени в несколько дней. Не следует думать, что их достаточно для успешной подготовки к экзаменам. В эти дни нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов познакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки студента к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня: сон не менее 8 часов в сутки, занятия должны заканчиваться не позднее, чем за 2-3 часа до сна.

Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неутомительные занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая- либо лекция, необходимо во время ее восстановить, обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к зачету или экзамену у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных конспектов. Вначале следует просмотреть весь материал по сдаваемой теме, отметить для себя трудные вопросы, обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения. Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Правила подготовки к экзамену:

- сориентироваться во всем материале и обязательно расположить его согласно экзаменационным вопросам или вопросам, обсуждаемым на семинарах, учебных занятиях. Эта работа может занять много времени, но все остальное - уже технические детали, главное - это ориентировка в материале;
- постараться максимально запомнить материал, переосмыслить его, рассмотреть альтернативные идеи;
- подготовить «шпаргалки», главный смысл которых систематизация и оптимизация знаний, однако пользоваться таким подспорьем не рекомендуется. Это очень сложная и важная для студента работа, более сложная и важная, чем простое поглощение массы учебной информации. Если студент самостоятельно подготовил такие «шпаргалки», то, скорее всего, он и экзамены сдавать будет более уверенно, так как у него уже сформирована общая ориентировка в сложном материале. Как это ни парадоксально, но использование «шпаргалок» часто позволяет отвечающему студенту лучше демонстрировать свои познания, точнее - ориентировку в знаниях, что намного важнее знания «запомненного» и «тут же забытого» после сдачи экзамена.

При ответе на экзамене студент сначала должен продемонстрировать преподавателю усвоенный по программе обучения материал, и лишь после этого высказать иную, желательно аргументированную точку зрения.

4 МЕТОДИКА ВЫПОЛНЕНИЯ ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Получить у преподавателя задание и необходимую литературу.
2. Найти предложенную литературу на образовательном портале или в библиотеке.
3. Изучить имеющуюся литературу в электронном или печатном виде, прочитать материалы лекций, практических и (или) семинарских занятий по теме.
4. Изучить методические рекомендации.
5. Оформить работу в тетради или на компьютере в соответствии с требованиями преподавателя.
6. Сдать самостоятельную работу преподавателю, предварительно ответив на вопросы для самоконтроля.

5 МЕТОДЫ КОНТРОЛЯ И ОЦЕНКА ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Контроль результатов самостоятельной работы проводится преподавателем одновременно с текущим и промежуточным контролем знаний обучающихся. Для контроля самостоятельной работы обучающегося используются разнообразные формы и методы: фронтальный, индивидуальный, выборочный, самоконтроль, защита презентации, участие в семинарском занятии, ответы на контрольные вопросы и т. д. При контроле результатов самостоятельной работы используются следующие критерии:

- уровень освоения обучающимся учебного материала;
- умение обучающегося использовать теоретические знания при выполнении заданий;
- обоснованность и чёткость изложения ответа;
- оформления материала в соответствии с требованиями.

Критерии оценки выполненной обучающимися работы:

оценка «5» - работа выполнена без ошибок; чисто, без исправлений; тема раскрыта полностью;

оценка «4» - работа выполнена с незначительными ошибками; тема раскрыта не полностью;

оценка «3» - работа выполнена со значительными ошибками; тема практически не раскрыта;

оценка «2» - работа не выполнена.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. **Буч, Г.** Язык UML. Руководство пользователя : пер. с англ. / Г. Буч, Дж. Рамбо, А. Джекобсон. – М. : ДМК, 2010. – 432 с.
2. **Леоненков, А**. Самоучитель UML / А. Леоненков. – СПб. : БХВ-Петербург, 2013. – 304 с.
3. **Мюллер, Р. Дж**. Базы данных и UML. Проектирование / Р.Дж. Мюллер. – Лори, 2009. – 420 с.
4. **Пайлон, Д.** UML 2 для программистов / Д. Пайлон, Н. Питмен. – Питер – Москва, 2012. – 240 с.
5. **Демидов, Д.Е.** Проектирование информационных систем. Объективно-ориентированный подход к проектированию информационных систем с применением унифицированного языка моделирования UML : учеб. пособие / Д.Е. Демидов, Д.А. Ломаш ; РГУПС. – Ростов н/Д : [б. и.], 2004. – 129 с.
6. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер.с англ. – М. : Изд. дом «Вильямс», 2002. – 432 с.
7. Принципы работы с требованиями к программному обеспечению. Унифицированный подход : пер. с англ. – М. : Изд. дом «Вильямс», 2002. – 448 с.
8. UML. Основы : пер. с англ. – СПб. : Символ-Плюс, 2002. – 192 с.
9. ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы. – М. , 1990. URL: http://www.nist.ru/hr/doc/gost/34-602-89.htm.
10. **Грекул, В.И.** Проектирование информационных систем / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. – М. : Бином. 2008. – 300 с.
11. **Гусятников, В.Н.** Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков. – М. : Финансы и статистика. 2010. – 288 с.
12. **Карпович, Е.Е.** Автоматизированное проектирование информационных систем на основе современных CASE-технологий. Часть 1 / Е.Е. Карпович, Н.В. Федоров. – М. : МГГУ, 2007. – 157 с.
13. **Карпович, Е.Е.** Автоматизированное проектирование информационных систем на рснове современных CASE-технологий. Часть 2 / Е.Е. Карпович, Н.В. Федоров. – М. : МГГУ, 2007. – 143 с
14. ArgoUML User Manual : A tutorial and reference description by Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola, Jeremy Bennett, Linus Tolke, and Michiel van der Wulp, 2009 Michiel van der Wulp,379 P
15. **Гвоздева, Т.В**. Проектирование информационных систем : учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. – Ростов н/Д : Феникс,2009. – 508 с.

Приложение 1

Список индивидуальных вариантов заданий

1. Разработать модель ИС библиотеки (актеры – руководитель, библиотекарь, читатель)
2. Разработать модель ИС рекламной фирмы (актеры – руководитель, сотрудник по работе с клиентами, клиент)
3. Разработать модель ИС видеосалона (актеры – руководитель, сотрудник, клиент)
4. Разработать модель ИС магазина парфюмерии (актеры – руководитель, сотрудник, клиент)
5. Разработать модель ИС ресторана (актеры – руководитель, официант, клиент)
6. Разработать модель ИС организации по работе с абитуриентами (актеры – руководитель, сотрудник организации, абитуриент)
7. Разработать модель ИС средней школы (актеры – директор, преподаватель, ученик)
8. Разработать модель ИС провайдера Интернет (актеры – руководитель, сотрудник по работе с клиентами, клиент)
9. Разработать модель ИС работы военкомата (актеры – руководитель, сотрудник по работе с призывниками, призывник)
10. Разработать модель ИС работы центра занятости (актеры – руководитель, сотрудник по работе с безработными, безработный)
11. Разработать модель ИС системы охраны предприятия (актеры – руководитель предприятия, сотрудник предприятия, охранник)
12. Разработать модель ИС работы университета (актеры – ректор, декан, студент)
13. Разработать модель ИС супермаркета (актеры – директор, кассир, покупатель)
14. Разработать модель ИС чемпионата по хоккею (актеры – сотрудник по работе с хоккеистами, хоккеист, глава федерации хоккея)
15. Разработать модель ИС олимпийских игр (актеры – болельщик, сотрудник по работе с болельщиками, руководитель олимпийского комитета)
16. Разработать модель ИС зоопарка (актеры – посетитель, директор, кассир)
17. Разработать модель ИС театра (актеры – актер, директор, администратор)
18. Разработать модель ИС страхового агентства (актеры – клиент, директор, юрист)
19. Разработать модель ИС свадебного салона (актеры – директор, сотрудник по работе с клиентами, клиент)
20. Разработать модель ИС туристической фирмы (актеры – директор, сотрудник по работе с клиентами, клиент)
21. Разработать модель ИС парикмахерской (актеры – руководитель, парикмахер, клиент)
22. Разработать модель ИС пиццерии (актеры – руководитель, официант, клиент)

23. Разработать модель ИС аукционного дома (актеры – руководитель, сотрудник, аукционист)
24. Разработать модель ИС автопарк (актеры – директор, клиент, водитель)
25. Разработать модель ИС салона по продаже мобильных телефонов (актеры – директор, продавец-консультант, клиент)
26. Разработать модель ИС кинологического клуба (актеры – руководитель, кинолог, владелец животного)
27. Разработать модель ИС детского сада (актеры – директор, воспитатель, родитель)
28. Разработать модель ИС управления программными проектами (актеры – руководитель фирмы, разработчик, менеджер проектов)
29. Разработать модель ИС командной разработки курсовых проектов (актеры – ректор, преподаватель, заведующий кафедрой)
30. Разработать модель ИС Министерства образования (актеры – министр образования, сотрудник министерства, президент)
31. Разработать модель ИС вокзала (актеры – начальник, машинист, пассажир)
32. Разработать модель ИС благотворительного фонда (актеры – руководитель, меценат, сотрудник по работе с клиентами)
33. Разработать модель ИС учета ГАИ (актеры – адвокат, гаишник, свидетель правонарушения)
34. Разработать модель ИС работы коммунального предприятия (актеры – директор, диспетчер, житель)
35. Разработать модель ИС ремонтной мастерской (актеры – директор, мастер, клиент)
36. Разработать модель ИС троллейбусного управления (актеры – начальник, водитель, технический работник)
37. Разработать модель ИС жилищно-коммунального предприятия (актеры – начальник, диспетчер, сантехник)
38. Разработать модель ИС курьерского агентства (актеры – начальник, заказчик курьерских услуг, сотрудник по работе с клиентами)
39. Разработать модель ИС агентства по уходу за людьми преклонного возраста (актеры – руководитель, человек преклонного возраста, психолог)
40. Разработать модель ИС аптечного фонда (актеры – начальник, клиент, фармацевт)
41. Разработать модель ИС предоставления услуг мобильного оператора (актеры – начальник, потребитель, технический работник)
42. Разработать ИС кинотеатра (актеры – директор, киноман, кассир)
43. Разработать модель ИС планетария (актеры – руководитель, экскурсовод, посетитель)
44. Разработать модель ИС парка развлечений (актеры – директор, посетитель, кассир)
45. Разработать модель ИС магазина бытовой техники (актеры – директор, клиент, продавец)
46. Разработать модель ИС банка (актеры – глава, кассир, клиент)
47. Разработать модель ИС кафе (актеры – начальник, официант, посетитель)

48. Разработать модель ИС городских электрических сетей (актеры – начальник, электрик, диспетчер)
49. Разработать модель ИС ЗАГСА (актеры – руководитель, сотрудник отдела приема заявлений, желающий женится)
50. Разработать модель ИС фитнес-клуба (актеры – руководитель, тренер, клиент)
51. Разработать модель ИС работы отдела кредитования банка (актеры – глава, клиент, сотрудник, оформляющий кредиты)
52. Разработать модель ИС видеопроката (актеры – директор, сотрудник, клиент)
53. Разработать модель ИС зала игровых автоматов (актеры – директор, техник-настройщик, игрок)
54. Разработать модель ИС работы отдела кадров электромеханического завода (актеры – начальник отдела кадров, сотрудник отдела кадров, представитель биржи труда)
55. Разработать модель ИС фабрики по производству музыкальных инструментов (актеры – руководитель, сотрудник, клиент)
56. Разработать модель ИС маркетинговой фирмы (актеры – маркетолог, сотрудник по работе с клиентами, клиент)
57. Разработать модель ИС лыжной базы (актеры – директор, тренер, технический работник)
58. Разработать модель ИС авиакомпании (актеры – директор, пилот, пассажир)
59. Разработать модель ИС агентства недвижимости (актеры – руководитель, риелтор, юрист)
60. Разработать модель ИС магазина спортивных товаров (актеры – руководитель, продавец-консультант, клиент)
61. Разработать модель ИС миграционной службы (актеры – начальник, юрист, мигрант)
62. Разработать модель ИС учета пациентов поликлиники (актеры – больной, работник регистратуры, доктор)
63. Разработать модель ИС склада торговой фирмы (актеры – грузчик, кладовщик, водитель транспорта по перевозке продукции со/на склад торговой фирмы)
64. Разработать модель ИС мебельного магазина (актеры – директор, продавец-консультант, клиент)
65. Разработать модель ИС предприятия по созданию мебели (актеры – руководитель, клиент, менеджер отдела продаж)
66. Разработать модель ИС ателье по пошиву пальто (актеры – директор, клиент, швея)
67. Разработать модель ИС прачечной (актеры – клиент, прачка, сотрудник службы доставки)
68. Разработать модель ИС химчистки (актеры – клиент, сотрудник, работающий с чистящим аппаратом, сотрудник службы доставки)

69. Разработать модель ИС автосервиса (актеры – руководитель, автомеханик, клиент)
70. Разработать модель ИС лизинга автомобилей (актеры – руководитель, клиент, сотрудник по работе с клиентами)
71. Разработать модель ИС магазина товаров для туризма и отдыха (актеры – директор, продавец-консультант, клиент)
72. Разработать модель ИС деревообрабатывающего предприятия (актеры – заказчик, столяр, сотрудник по работе с клиентами)
73. Разработать модель ИС книжного магазина (актеры – директор, продавец-консультант, клиент)
74. Разработать модель ИС букмекерской конторы (актеры – начальник, сотрудник, отвечающий за прием ставок, желающий сделать ставку)
75. Разработать модель ИС травматологического пункта (актеры – медицинская сестра, больной, врач)
76. Разработать модель ИС центра тестирования выпускников (актеры – родитель выпускника, сотрудник центра, выпускник)
77. Разработать модель ИС клиники хирургия пластической хирургии (актеры – заведующий, пластический хирург, желающий сделать пластическую операцию)
78. Разработать модель ИС центра помощи ветеранам (актеры – руководитель, сотрудник центра, ветеран)
79. Разработать модель ИС центра помощи многодетным семьям (актеры – руководитель, сотрудник центра, представитель многодетной семьи)
80. Разработать модель ИС детского садика (актеры – руководитель, повар, родитель)
81. Разработать модель ИС исторического музея (актеры – руководитель, сотрудник, посетитель)
82. Разработать модель ИС работы паспортного стола (актеры – начальник, клиент, сотрудник по приему заявок)
83. Разработать модель ИС работы приемной городского главы (актеры – руководитель, сотрудник приемной, гражданин)
84. Разработать модель ИС работы магазина компьютерной техники (актеры – директор, продавец-консультант, клиент)
85. Разработать модель ИС организации защиты животных (актеры – руководитель, ветеринар, владелец животного)
86. Разработать модель ИС работы пенсионного фонда (актеры – начальник, сотрудник по работке с клиентами, пенсионер)
87. Разработать модель ИС агентства знакомств (актеры – руководитель, сотрудник по работе с клиентами, клиент)
88. Разработать модель ИС работы Ледовой арены (актеры – руководитель, охранник, посетитель)
89. Разработать модель ИС работы парка развлечений (актеры – руководитель, кассир, посетитель)
90. Разработать модель ИС работы речного вокзала (актеры – пассажир, капитан, кассир)

Приложение 2

Система обозначений языка UML

Пакеты в языке UML

Пакет – основной способ организации элементов модели в языке UML. Каждый пакет владеет всеми своими элементами, т. е. теми элементами, которые включены в него. Про соответствующие элементы пакета говорят, что они принадлежат пакету или входят в него. При этом каждый элемент может принадлежать только одному пакету.

Для графического изображения пакетов на диаграммах применяется специальный графический символ – большой прямоугольник с небольшим прямоугольником, присоединенным к левой части верхней стороны первого (рис. П2.1 а, б).

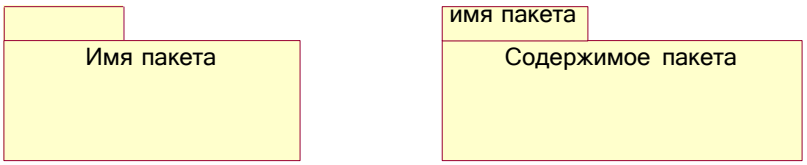


Рисунок П2.1. Графическое изображение пакета в языке UML

Внутри большого прямоугольника может записываться информация, относящаяся к данному пакету. Если такой информации нет, то внутри большого прямоугольника записывается имя пакета, которое должно быть уникальным в пределах рассматриваемой модели (рис. П2.1, а). Если же такая информация имеется, то имя пакета записывается в верхнем маленьком прямоугольнике (рис. П2.1, б).

а) б)

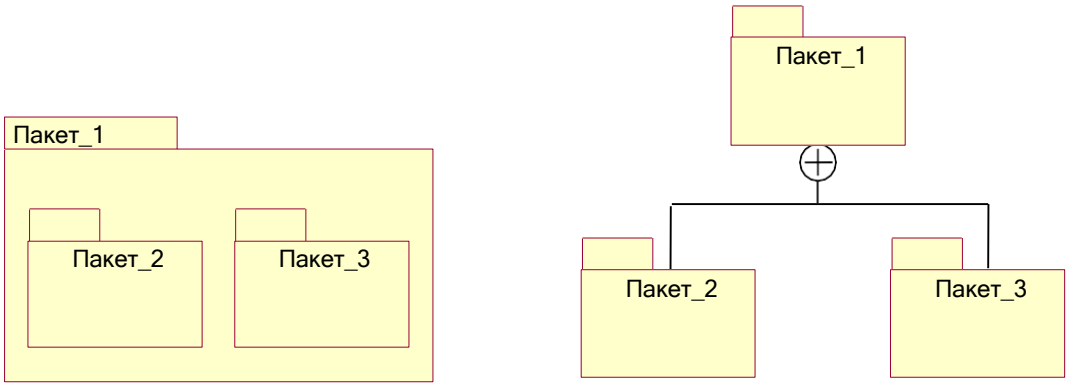


Рисунок П2.2. Графическое изображение вложенности пакетов друг в друга

Одни пакеты могут быть вложены в другие пакеты.

Таким образом, для элементов модели задается отношение вложенности пакетов, которое представляет собой иерархию. С одной стороны, в языке UML это отношение может быть изображено простым размещением одного пакета-прямоугольника внутри другого пакета-прямоугольника (рис. П2.2, а). Так, в данном случае пакет с именем Пакет_1 содержит в себе два подпакета: Пакет_2

и Пакет_3. С другой стороны, это же отношение может быть изображено с помощью отрезков линий аналогично графическому представлению дерева. В данном варианте используется символ ⊕. Этот символ означает, что под пакеты являются «собственностью» или частью контейнера, и, кроме этих под пакетов, контейнер не содержит никаких других подпакетов (рис. П2.2, б).

Использование пакетов позволяет упростить понимание модели, и поэтому они являются жизненно необходимым средством для крупных проектов. Используя механизм пакетов, разработчики могут разбить сложную систему на составные части с минимумом взаимных связей.

Диаграмма прецедентов (Use Case Diagram)

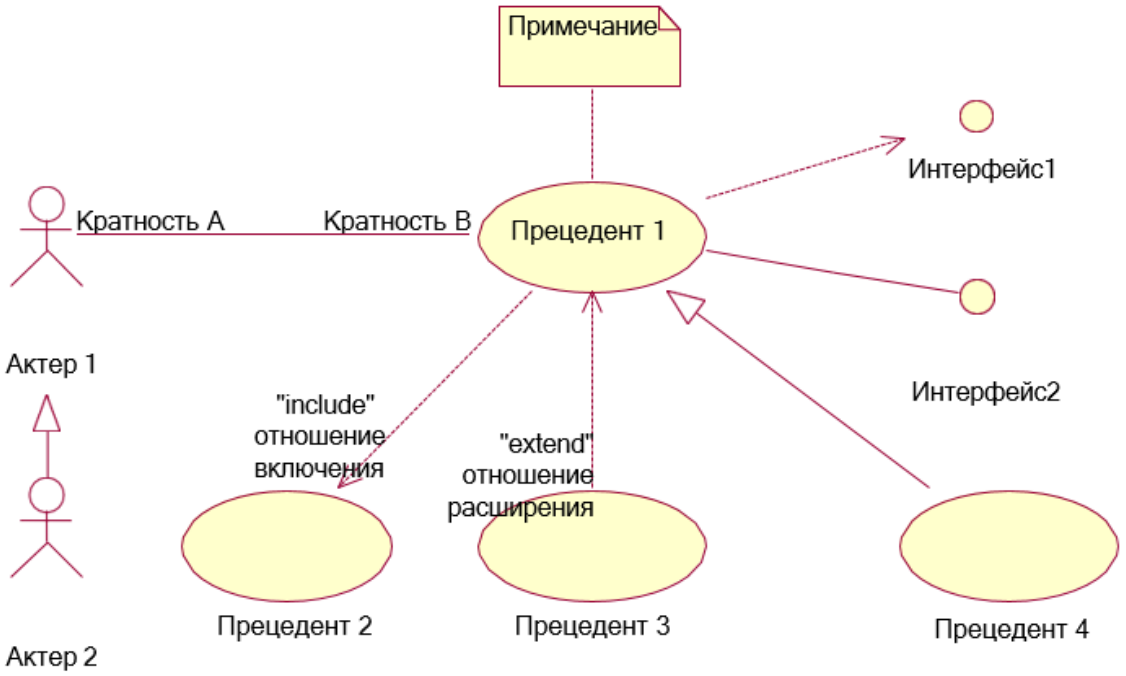


Рис. П2.3. Элементы диаграммы прецедентов

Диаграмма классов (Class Diagram)

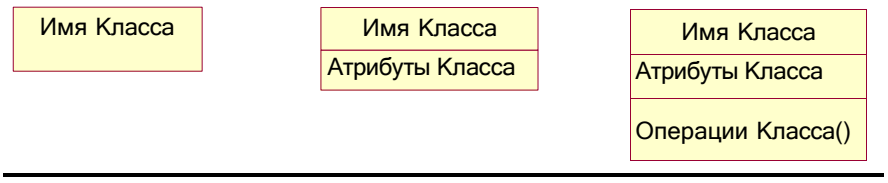


Рис. П2.4. Варианты графического изображения классов на диаграмме

Имя класса:

<Имя_пакета>::<Имя_класса>

Атрибут класса:

<квантор видимости><имя атрибута>[кратность]:

<тип атрибута>=<исходное значение>{строка-свойство}

Операция класса:

<квантор видимости><имя операции>(список параметров):

<тип возвращаемого значения>{строка-свойство}***Отношения между классами***

Рис. П2.5. Графическое изображение отношения зависимости

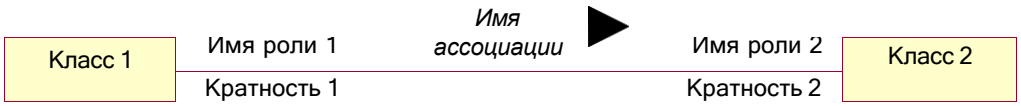


Рис. П2.6. Графическое изображение отношения бинарной ассоциации между классами

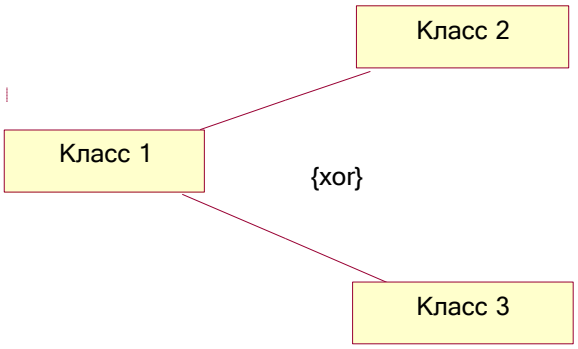


Рис. П2.7. Графическое изображение исключающей ассоциации между тремя классами

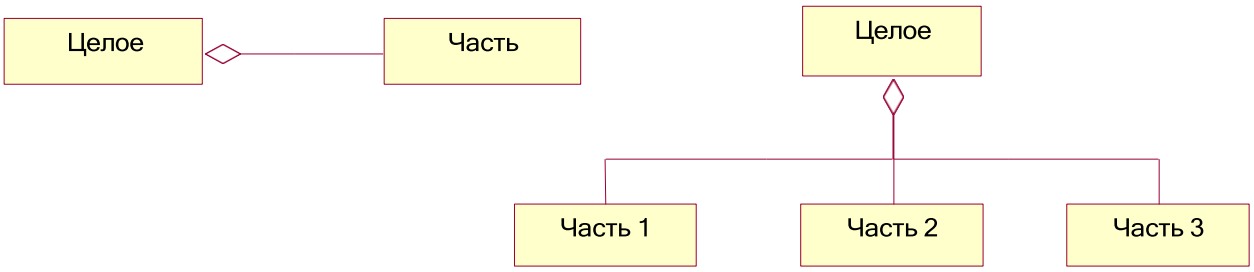


Рис. П2.8. Графическое изображение отношения агрегации в языке UML

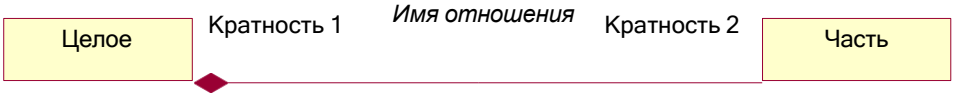


Рис. П2.9. Графическое изображение отношения композиции в языке UML

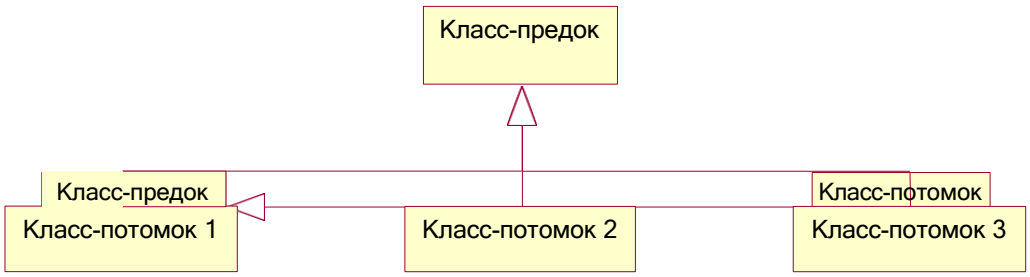


Рис. П2.10. Графическое изображение отношения обобщения

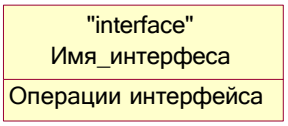


Рис. П2.11. Графическое изображение интерфейса на диаграмме классов

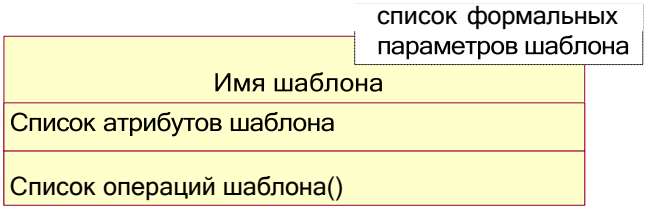


Рис. П2.12. Графическое изображение шаблона (параметризованного класса)

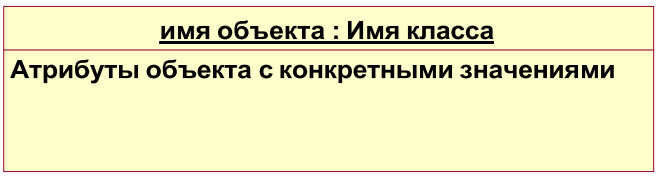


Рис. П2.13. Графическое изображение объекта

Диаграмма состояний (Statechart Diagram)

Рис. П2.14. Графическое изображение начального и конечного состояний на диаграмме состояний

Рис. П2.15. Графическое изображение недавнего и давнего состояния на диаграмме состояний

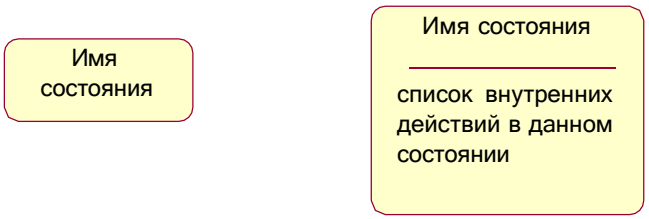


Рис. П2.16. Графическое изображение состояний на диаграмме состояний

Рис. П2.17. Графическое изображение перехода между состояниями

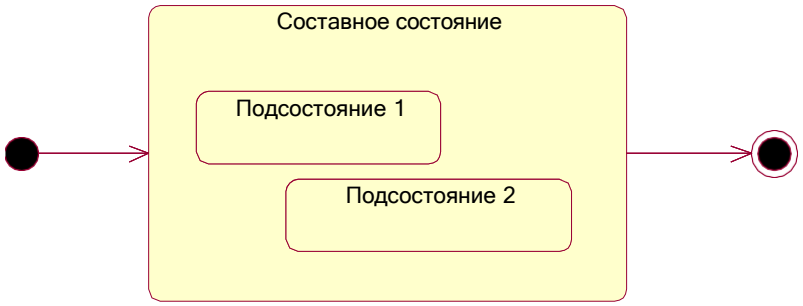


Рис. П2.18. Графическое представление составного состояния

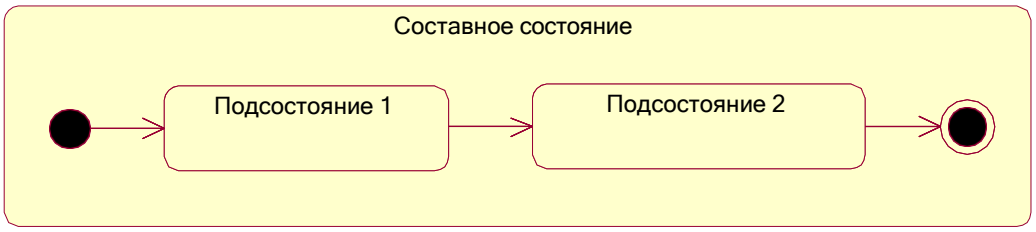


Рис. П2.19. Графическое представление последовательных подсостояний

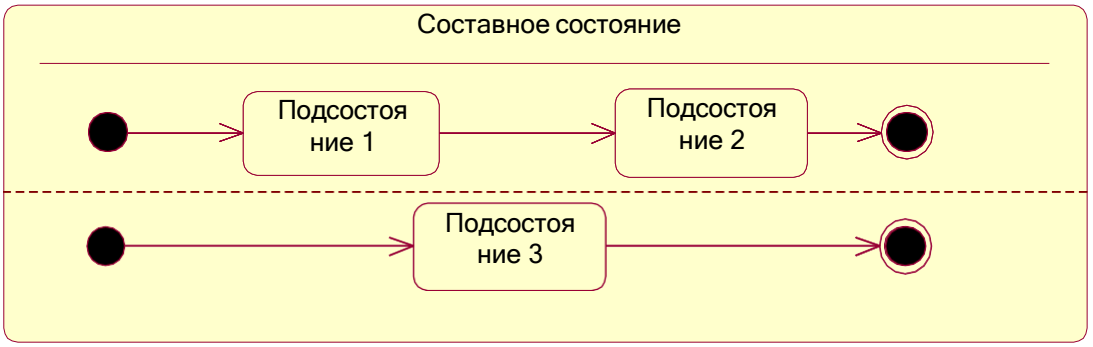


Рис. П2.20. Графическое представление параллельных подсостояний

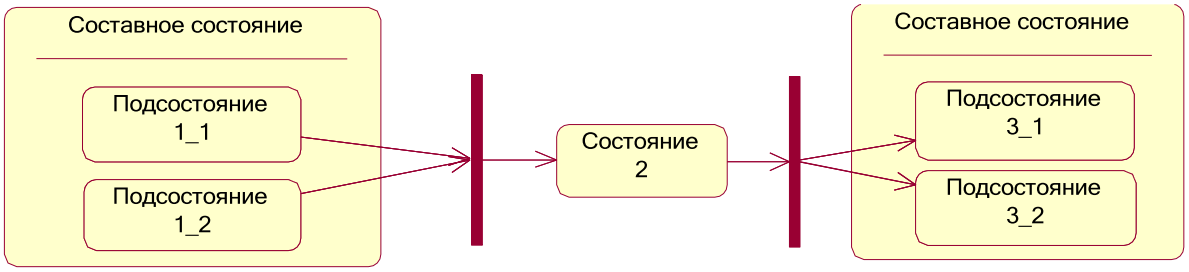


Рис. П2.21. Графическое представление перехода-соединения
и перехода-ветвления

Диаграмма деятельности (Activity Diagram)

Рис. П2.22. Графическое изображение начального и конечного состояния на диаграмме деятельности

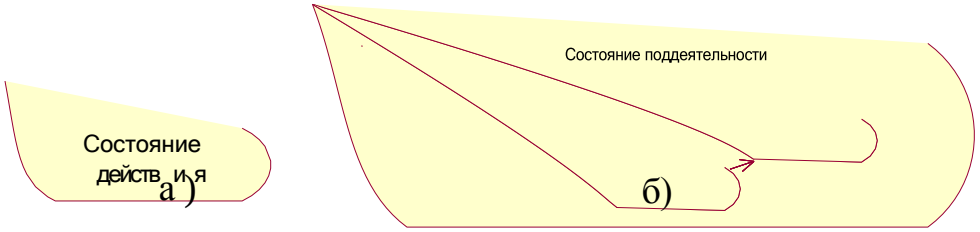


Рис. П2.23. Графическое изображение состояния действия (а)
и состояния под деятельности (б)

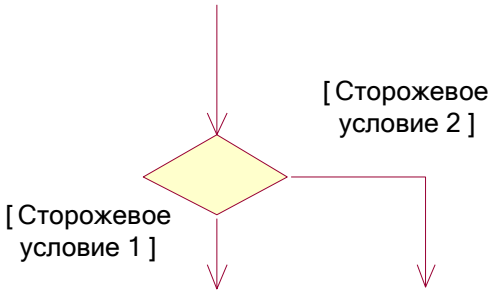


Рис. П2.24. Графическое изображение ветвления потока управления

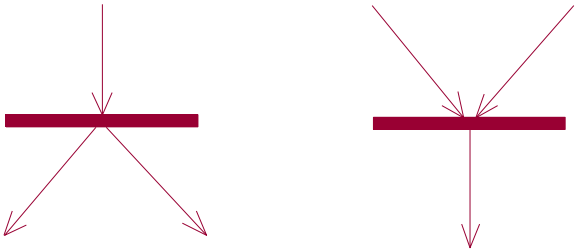


Рис. П2.25. Графическое изображение разделения и слияния
параллельных потоков управления

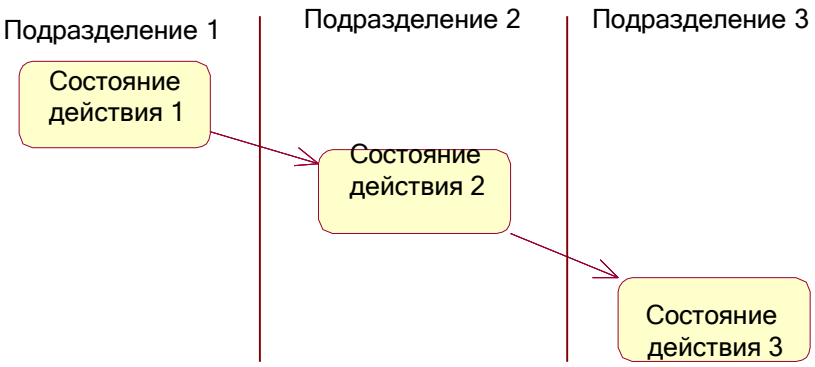


Рис. П2.26. Графическое изображение дорожек на диаграмме деятельности

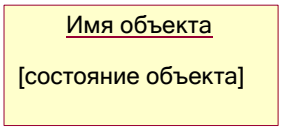


Рис. П2.27. Графическое изображение объекта на диаграмме деятельности

Диаграмма последовательности (Sequence Diagram)

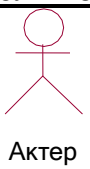


Рис. П2.28. Графическое изображение актера на диаграмме последовательности.

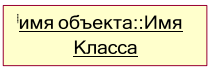


Рис. П2.29. Графическое изображение объекта и его линии жизни

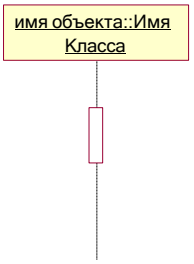


Рис. П2.30. Графическое изображение фокуса управления

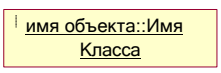


Рис. П2.31. Графическое изображение символа уничтожения объекта

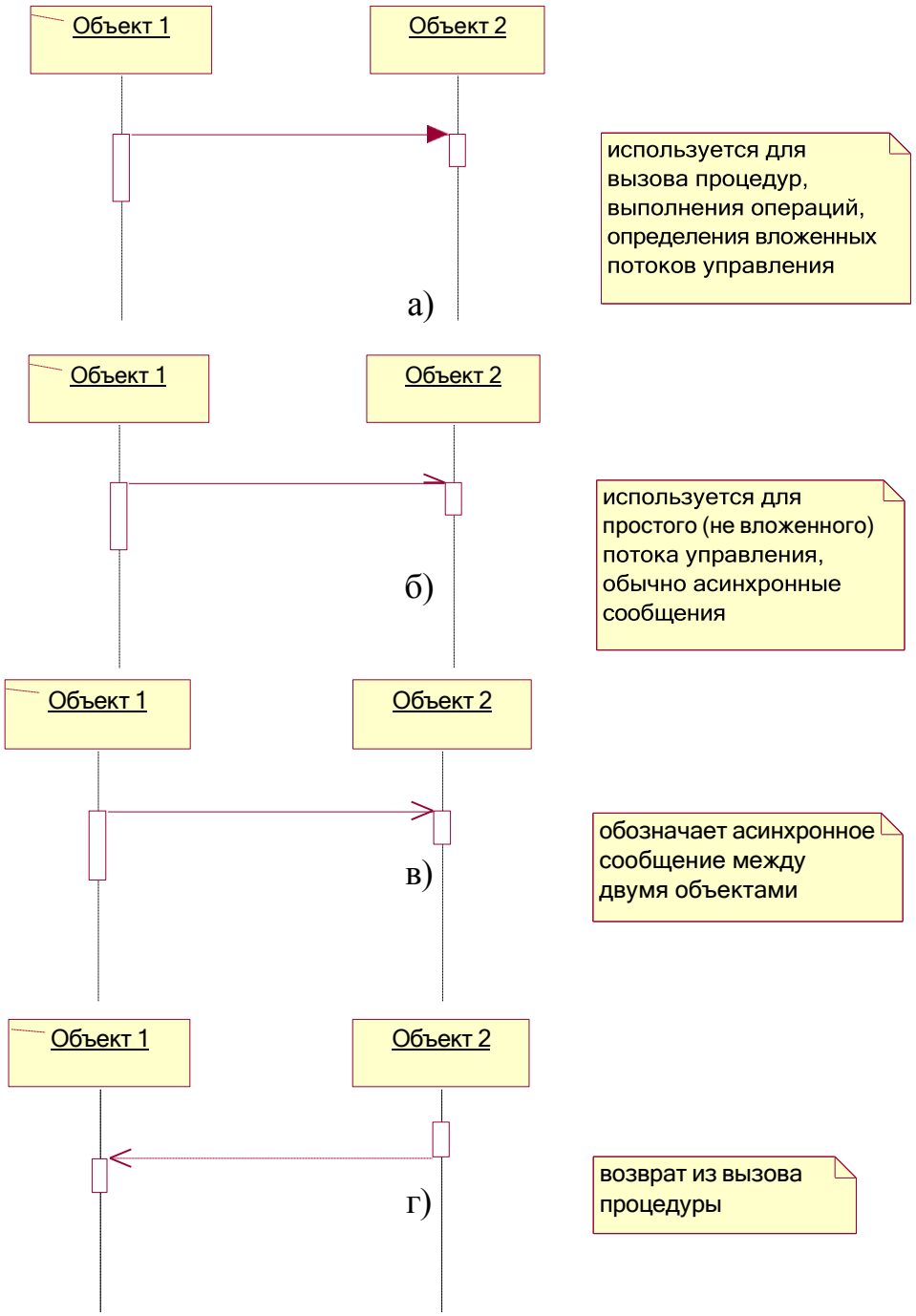


Рис. П2.32. Варианты графических изображений сообщений

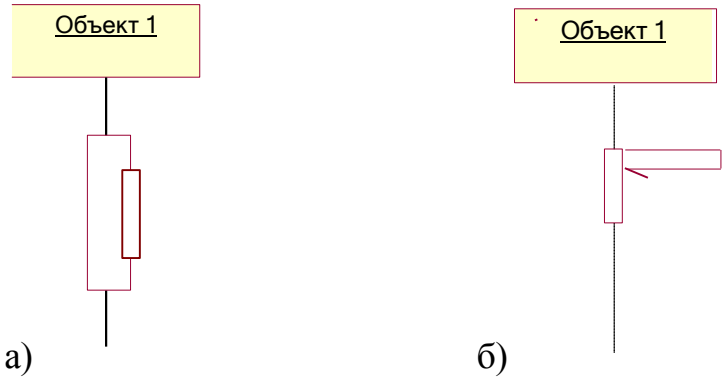


Рис. П2.33. Графическое изображение рекурсии (а) и рефлексивного сообщения (б)

Диаграмма кооперaций (Collaboration Diagram)

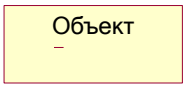
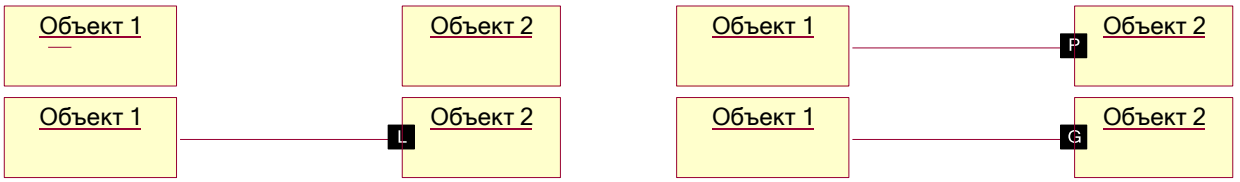


Рис. П2.34. Графическое изображение объекта на диаграмме кооперaций



P - "parameter" – параметр метода. Соответствующий объект может быть только параметром некоторого метода.
L - "local" – локальная переменная метода. Ее область видимости ограничена только соседним объектом.
G -"global" – глобальная переменная. Ее область видимости распространяется на всю диаграмму кооперации.

Рис. П2.35. Примеры графических изображений связей с различными стереотипами

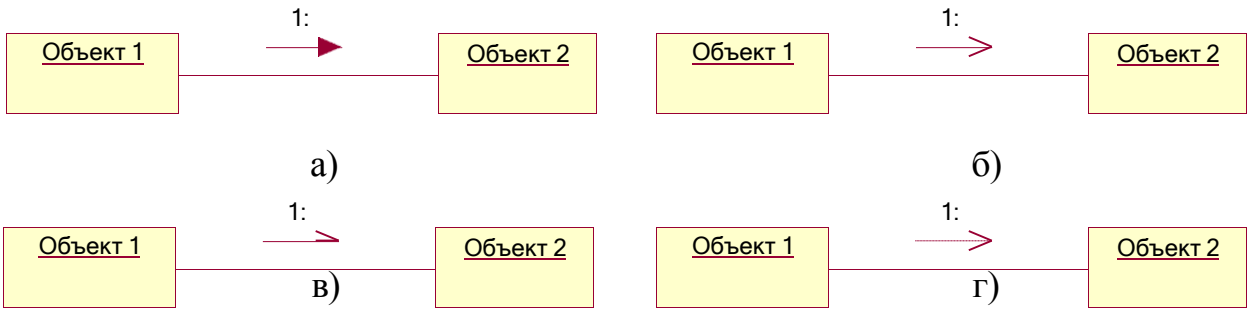


Рис. П2.36. Графическое изображение различных типов сообщений: вызов процедуры (а), простой поток управления (б), асинхронный поток управления (в), возврат из вызова процедуры (г)

Диаграмма компонентов (Component Diagram)

Рис. П2.37. Графическое изображение компонента в языке UML

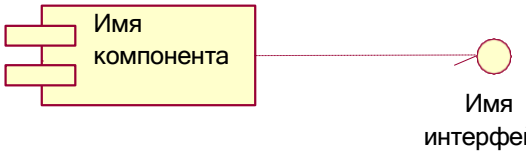


Рис. П2.38. Графическое изображение реализации интерфейса в языке UML

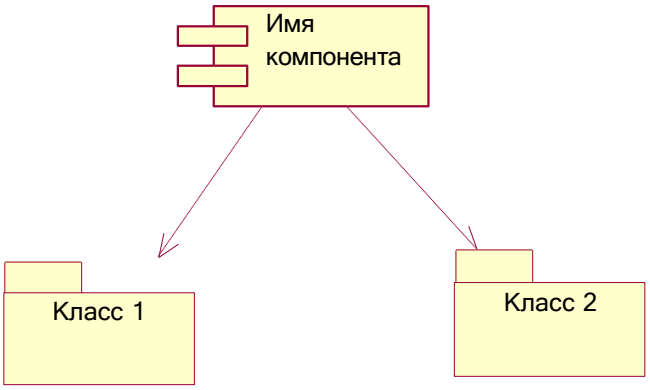


Рис. П2.39. Графическое изображение зависимости
между компонентом и классами

Диаграмма развертывания (Deployment Diagram)

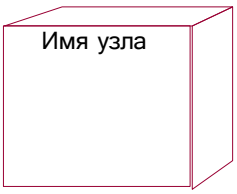


Рис. П2.40. Графическое изображение узла на диаграмме развертывания

Рис. П2.41. Графическое изображение соединения между узлами

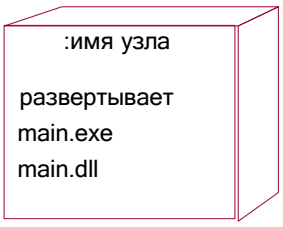


Рис. П2.42. Графическое изображение узлов-экземпляров
с размещенными в них компонентами

Приложение 3

Разработка ИС с использованием UML на примере Internet-магазина

В качестве примера для разработки информационной системы рассмотрим создание книжного *Internet-магазина*

3.1. Постановка задачи на разработку

На основании пожеланий заказчика сформулируем требования, предъявляемые к Internet-магазину:

- книжный магазин должен принимать заказы через Internet;
- книжный магазин должен поддерживать до миллиона счетов клиентов;
- книжный магазин должен предоставлять средство поиска в каталоге;
- должны быть реализованы различные методы поиска, в том числе по автору, по названию и по ключевым словам;
- книжный магазин должен предоставить различные варианты оплаты заказа, в том числе почтовым переводом, оплата через курьера, безналичным расчетом;
- разрабатываемая система должна автоматизировать процесс заказа книг со склада.

Более детально требования к разрабатываемой системе будут выявлены в процессе анализа предметной области.

3.2. Моделирование предметной области

Моделирование предметной области является основной статической частью модели UML. Построение модели предметной области начинается с выявления абстракции, существующих в реальном мире, т.е. основных концептуальных объектов, которые встречаются в системе.

Первое, что нужно сделать при построении статической модели системы, - отыскать классы, которые адекватно отражают абстракции предметной области.

Выявление классов предметной области связано с необходимостью излагать прецеденты в контексте объектной модели. Это позволит связать статические и динамические части модели. Это означает, что начинать необходимо с ключевых объектов в системе, а затем, двигаясь наружу, изучать с какими еще объектами они взаимодействуют. Таким образом, при выявлении прецедентов или динамической части системы осуществляется движение снаружи внутрь, а при создании статической модели – изнутри наружу.

Диаграммы классов позволяют создавать статическую модель предметной области, в то время как диаграммы прецедентов, активности, состояний, последовательностей и кооперации отражают динамическую составляющую предметной области.

Определение классов предметной области

Лучшими источниками классов является *высокоуровневое* описание задачи, *низкоуровневые* требования и *экспертная оценка* задачи. Одним из методов, предложенный в [1], является метод выявления ключевых существительных и именных групп из таких описаний. Таким образом, наиболее вероятно найти почти все важные доменные объекты (классы).

По мере уточнения этого перечня должно происходить следующее:

- имена существительные и именные группы становятся объектами и атрибутами;
- глаголы и глагольные группы становятся операциями и ассоциациями;
- родительный падеж показывает, что имя существительное должно быть атрибутом, а не объектом.

При построении диаграмм классов необходимо также принять предварительные решения об обобщениях (отношения вида «является»). Также необходимо принять решение об агрегировании (отношения вида «имеет») классов. В итоге модель предметной области, дополненная ассоциациями (статическими отношениями между парами классов), должна адекватно описывать аспекты задачи, независящие от времени, т.е. статические.

В качестве ключевых абстракций предметной области выявим следующие классы:

- заказ;
- заказчик;
- вариант оплаты;
- каталог книг;
- раздел каталога;
- печатное издание.

В соответствии с постановкой задачи необходимо учесть различные варианты оплаты. Это удобно осуществить посредством механизма *наследования* от класса «Вариант оплаты». В результате получим следующие подклассы:

- почтовым переводом;
- курьером;
- безналичным расчетом.

Аналогично необходимо поступить с классом «Печатное издание». Полученные подклассы будут называться:

- журнал;
- книга;
- газета.

Полученная диаграмма приведена на рис. П3.1.

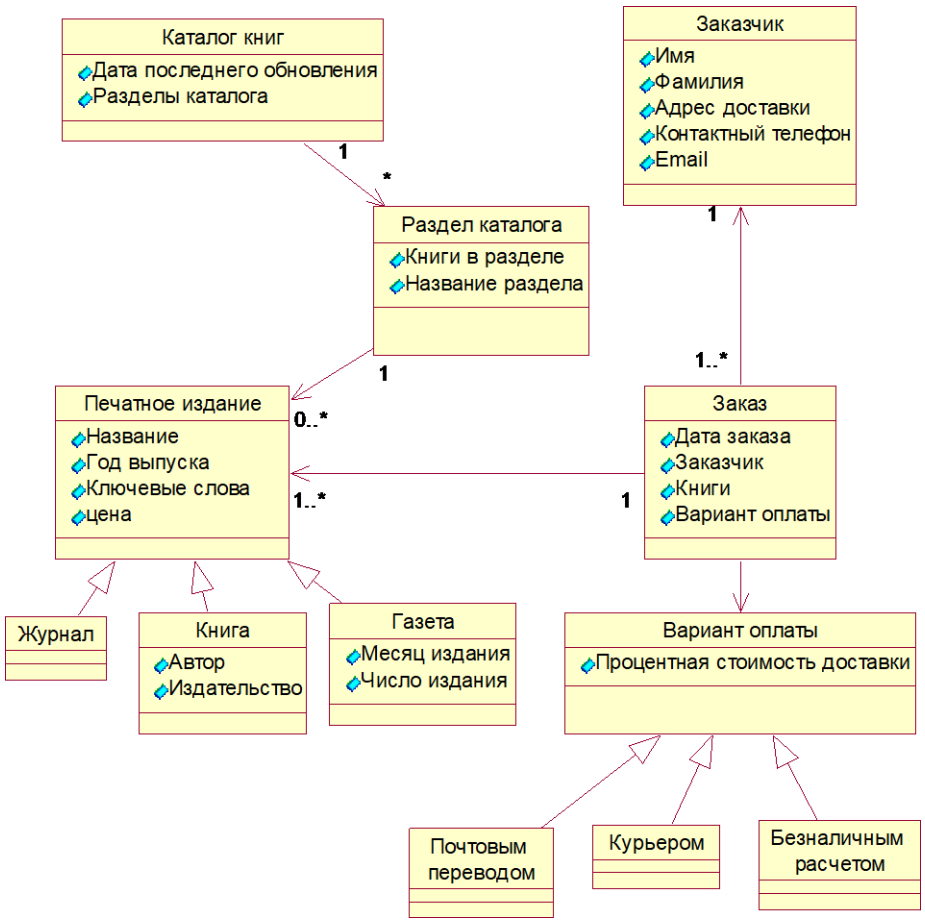


Рис. П3.1. Диаграмма классов предметной области Internet-магазина

Полученная диаграмма отражает кратность связи. Например, в одном заказе может содержаться несколько (одно или более) печатное издание; раздел каталога может содержать любое количество книг (в том числе и ни одной).

В результате анализа предметной области также выявлены основные атрибуты некоторых классов. Например, класс «Заказчик» характеризуется следующими атрибутами:

- имя;
- фамилия;
- адрес доставки;
- контактный телефон;
- Email.

Следующим шагом в моделировании предметной области является анализ прецедентов.

Определение прецедентов системы

Для решения задачи построения прецедентов для новой системы, необходимо с самого начала идентифицировать как можно больше прецедентов, а затем составить и постоянно уточнять их словесное описание.

Рекомендуется объединять прецеденты в пакеты (рис. П3.2), т. к. прецеденты определяют логические участки работы, которые можно распределить между группами разработчиков. Переходить к дальнейшим этапам разработки следует после того, как достигнуты следующие цели моделирования прецедентов:

- созданные прецеденты описывают всю требуемую функциональность системы;
- для каждого прецедента четка и кратко описана главная последовательность действий, а также все альтернативные последовательности;
- выделены сценарии, общие для нескольких прецедентов.

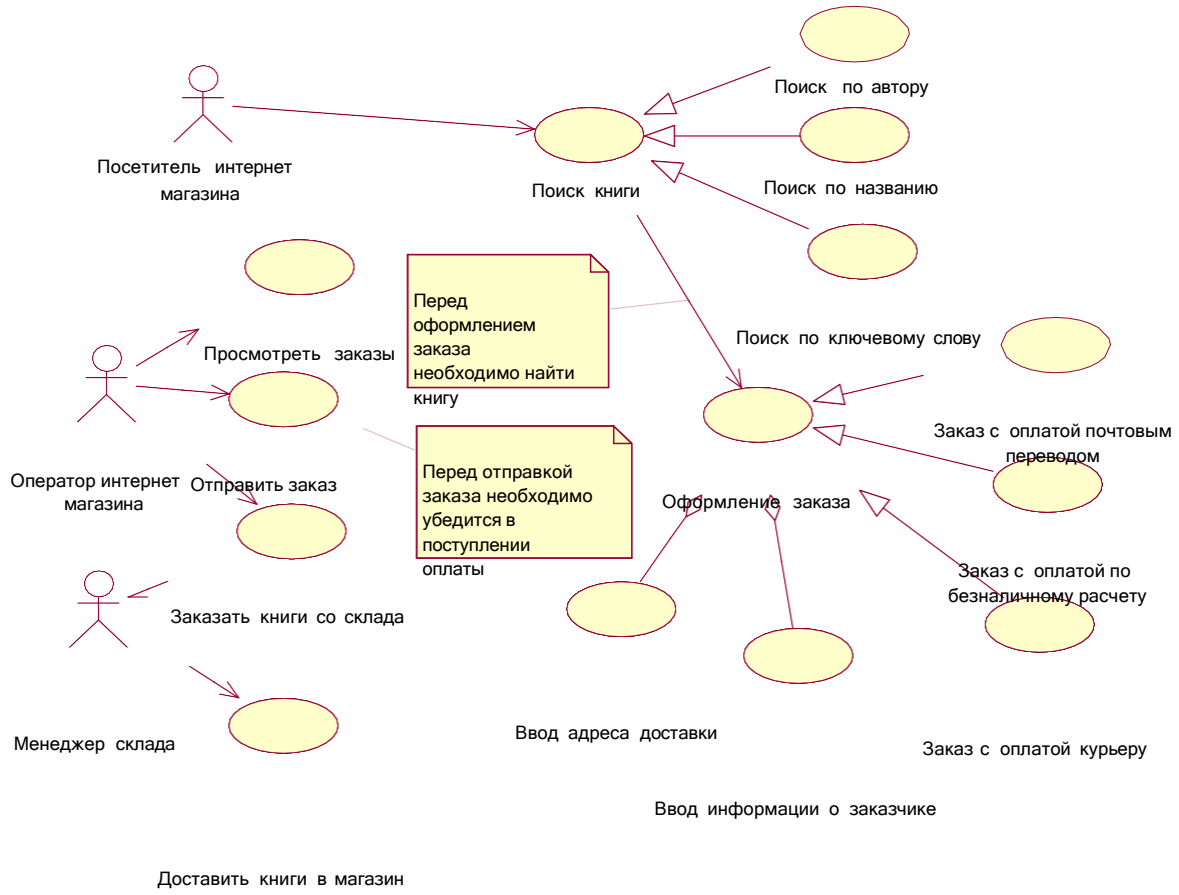


Рис. П3.2. Диаграмма прецедентов предметной области Internet-магазина

Определение взаимодействий между объектами

Взаимодействие объектов в системе определяет динамику системы. Для описания динамической модели проектируемой системы в UML используются диаграммы: состояний, активности, последовательностей и кооперации.

Динамическая модель предметной области позволяет дополнить модель классов, за счет введения вспомогательных классов информационной системы.

В качестве примеров таких диаграмм ниже рассмотрены диаграмма активности и последовательностей.

Диаграмма активности, поясняющая прецедент «Поиск», приведена на рис. П3.3.

Прецедент «Поиск по автору» поясняется диаграммой последовательностей, приведенной на рис. П3.4.

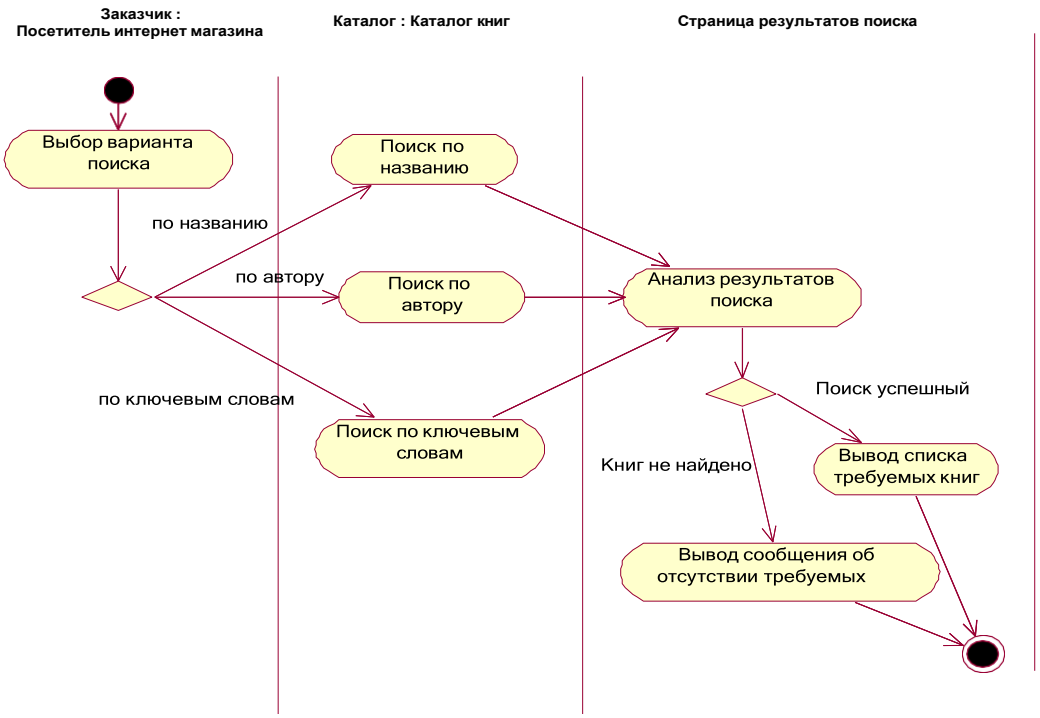


Рис. П3.3. Диаграмма активности, отражающая процедуру поиска

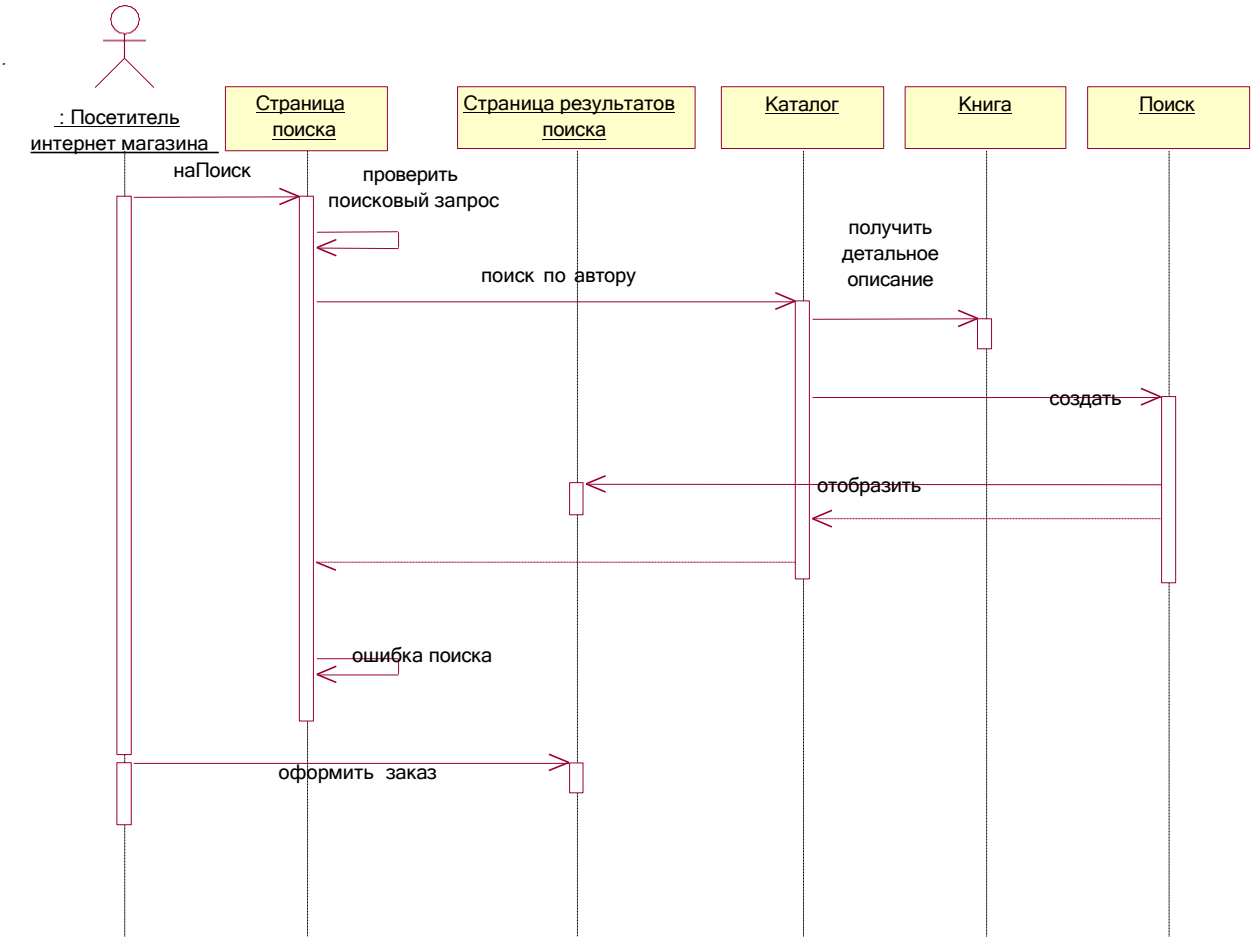


Рис. П3.4. Диаграмма последовательностей прецедента «Поиск по автору»

Главная последовательность данного прецедента:

- *посетитель* internet-магазина вводит имя автора на *странице поиска* и нажимает кнопку искать;
- система проверяет, введен ли допустимый запрос, после чего ищет в *каталоге* все *книги* указанного автора;
- для каждой найденной *книги* система извлекает существенные детали;
- затем система выводит на *странице результатов поиска* список *книг*;
- на основании результатов поиска пользователь может оформить заказ.

Альтернативная последовательность данного прецедента:

- если посетитель нажал кнопку «искать» не введя данных запроса, то система выводит сообщение об ошибке и предлагает задать другой критерий поиска;
- если в результате поиска не найдено ни одной книги, то выводится сообщение об отсутствии требуемых книг и предлагается осуществить повторный поиск.

3.3. Проектирование информационной системы
Разработка диаграммы классов

На основании диаграммы классов предметной области и диаграмм, отражающих динамику поведения системы, необходимо разработать полную диаграмму классов, дополнив и исправив список методов и атрибутов. На данном этапе могут быть введены дополнительные классы, не отражающие сущностей предметной области, но полезные при проектировании. Так, например, на рис. П3.5. показан класс OrderController, отвечающий за работу с заказами. Необходимо как можно тщательнее продумать данную диаграмму, так как она является исходной для создания кода будущей информационной системы.

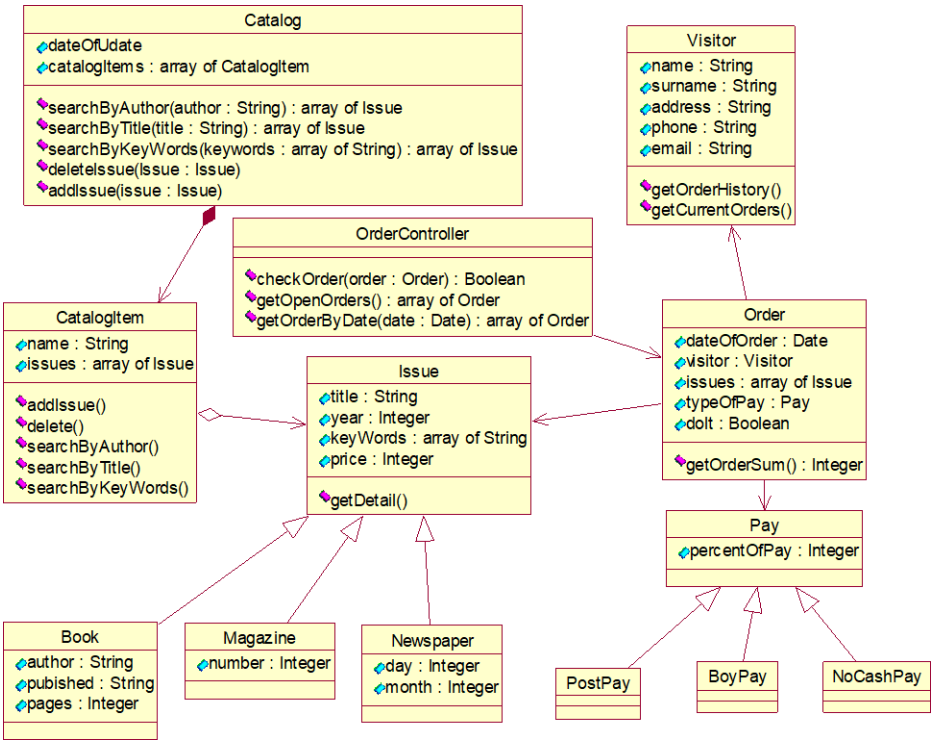


Рис. П3.5. Диаграмма классов Internet-магазина

Разработка диаграмм динамической модели

Динамическая модель системы отражает наиболее важные взаимодействия, поясняя модель системной архитектуры. На данном этапе составляются (дополняются) диаграммы динамической модели с учетом полной диаграммы классов. При этом описывается взаимодействие экземпляром отдельных классов в терминах методов и атрибутов классов. Такие модели служат мостом между требованиями к системе и её реализацией. Следовательно, с одной сто- роны модель должны быть абстрактна настолько, чтобы лишние данные не скрывали отношения между классами, а с другой стороны модель должна быть достаточно детальна для возможности принятия решения на этапе программи- рования.

При моделировании больших систем применяют диаграммы разных уровней детализации.

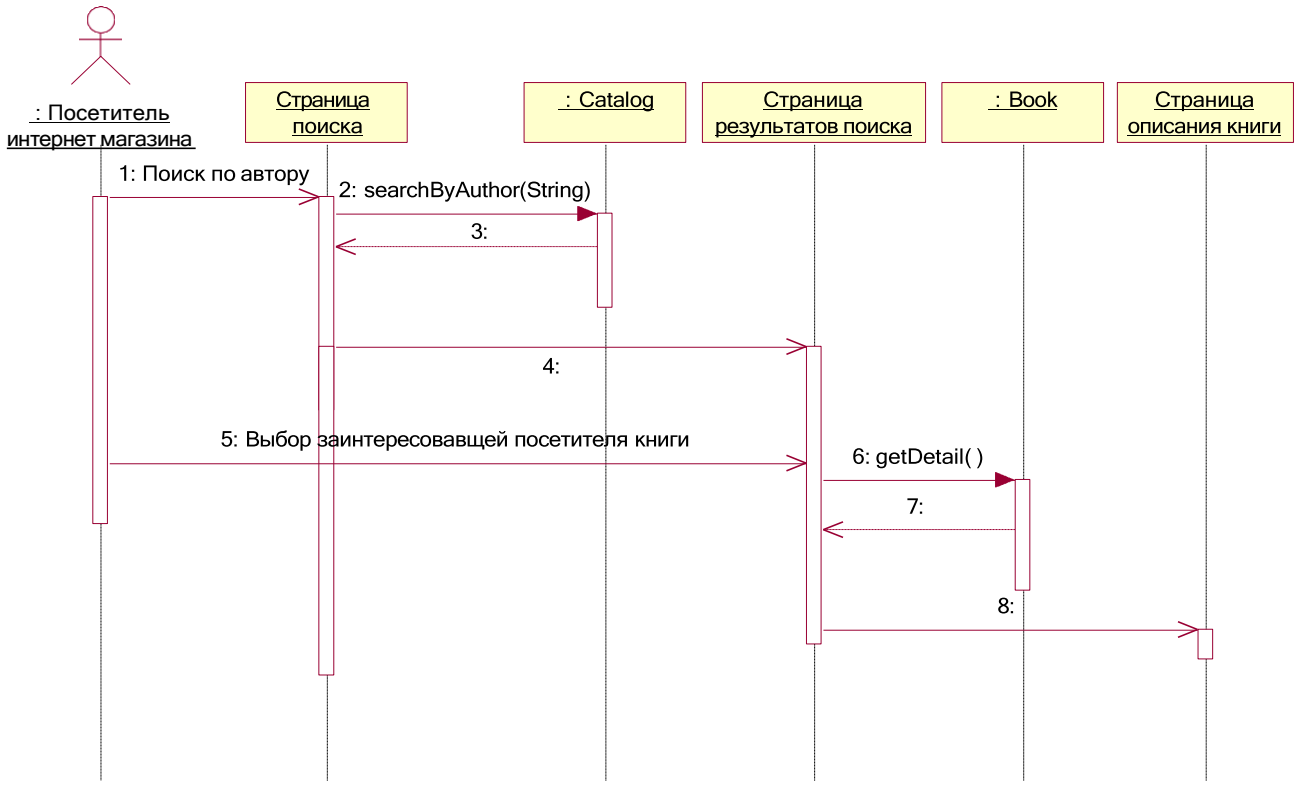


Рис. П3.6. Диаграмма последовательностей поиска по автору

Не следует пояснять все взаимодействия объектов в информационной системе, так как это затруднит понимание модели. Необходимо, однако, достаточно подробно рассмотреть взаимодействия объектов *в ключевых прецедентах*. Для наиболее полного пояснения взаимодействия объектов наиболее целесообразно применять различные виды диаграмм. Так, например, диаграмма состояний наиболее точно отражает состояния системы и переходы между ними, в то время как диаграмма активности акцентирует внимание на активности (деятельности) информационной системы. На пример, прецедент «Заказ» удобно пояснить диаграммой активности (рис. П3.7.).

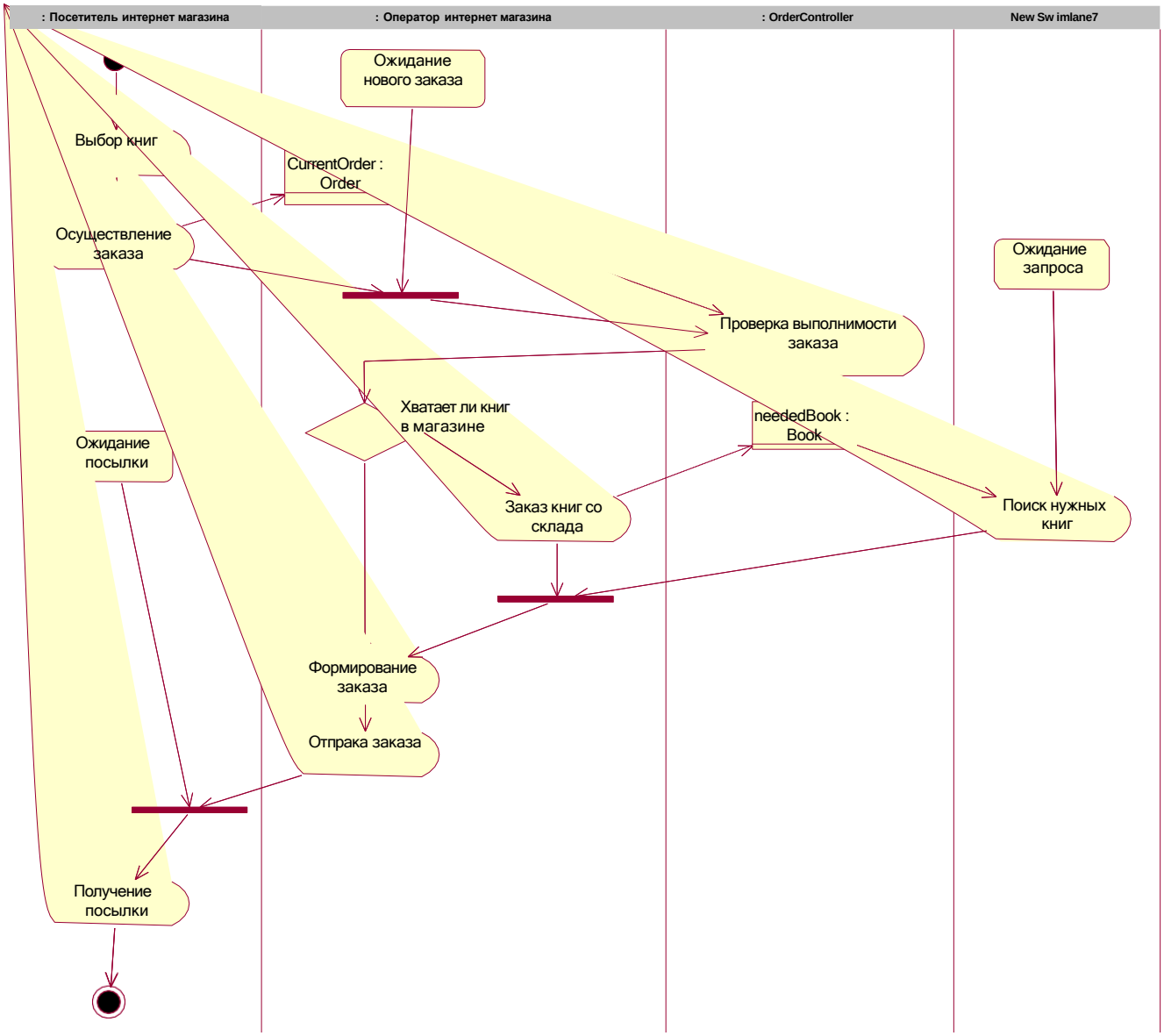


Рис. П3.7. Диаграмма активности, отражающая прецедент «Заказ»

Диаграмма компонентов

Рассмотрим компоненты разрабатываемой системы. Будем рассчитывать на применение посетителями сайта только браузеров Netscape 6 и браузера Internet Explorer выше 4 версии. В качестве Web-сервера будет применяться Apache 2.0.44. Для примера информационная система будет состоять из 5 web- страниц:

- BookCatalog (каталог книг);
- SearchPage (страница поиска);
- BookDetail (страница с подробным описанием книги);
- SearchResult (страница результатов поиска);
- OrderPage (страница заказа книг).

Помимо этого система так же включает модуль DB_Access – модуль, в котором хранятся классы, отвечающие за доступ к реляционной базе данных. С целью наибольшей гибкости информационной системы выделен модуль Con-

stant, описывающий константы, такие как: пароль для доступа к серверу баз данных, логин для доступа к серверу баз данных, путь к базе данных.

Компоненты информационной системы совместно со связями приведены на рис. П3.8.

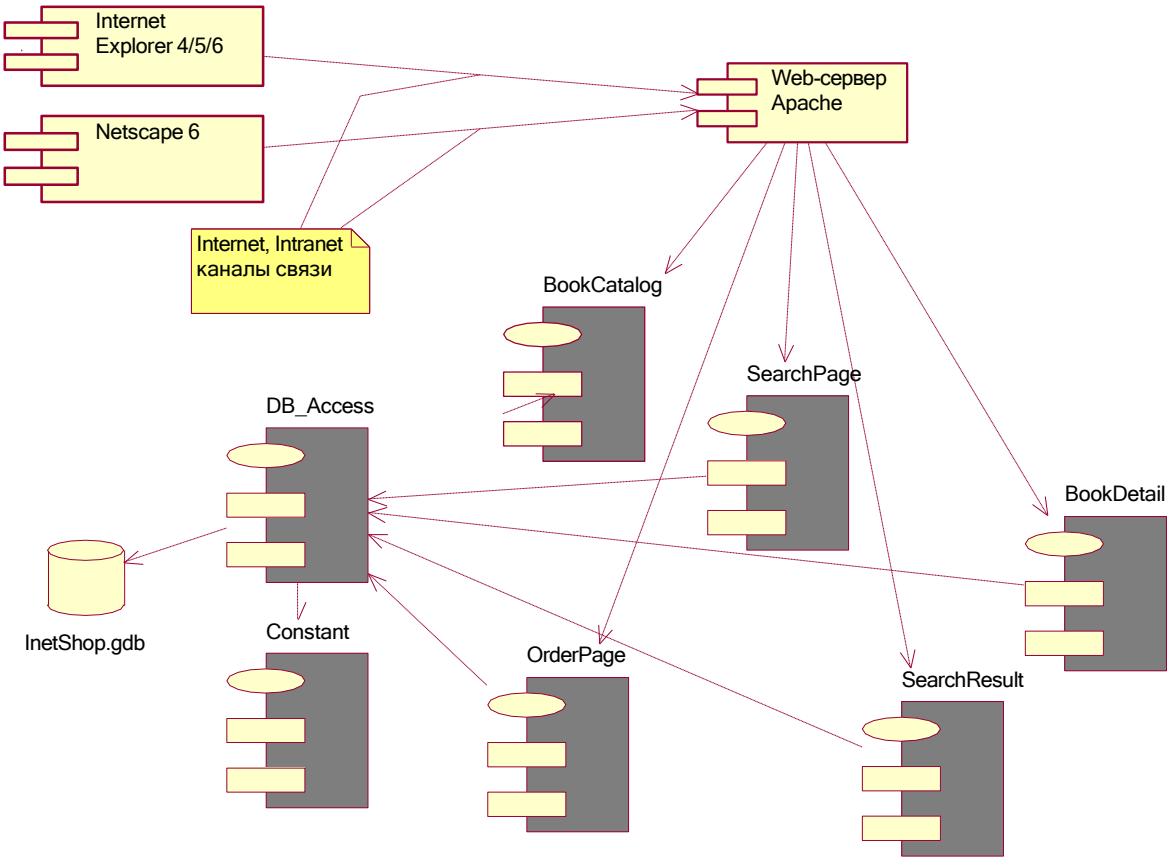


Рис. П3.8. Диаграмма компонентов Internet-магазина

Рецензирование проекта информационной системы

На стадии рецензирования проекта необходимо проверить, отвечает ли проект внутренним стандартам проектирования, принятым в организации-разработчике. Иногда целесообразно использование шаблонов проектирования. Например, может быть решение об использование фабрик для создания экземпляров классов. Или могут наличествовать стандартные механизмы доступа к реляционным базам данных. Диаграммы последовательностей и сопутствующие им детальные диаграммы классов должны отражать видение программы старшими проектировщиками.

Один из важнейших аспектов рецензирования – это тщательный анализ соответствия между каждым предложением в тексте прецедента и сообщениями напротив этого предложения на диаграммах последовательностей. Кроме того, *на моделях пригодности должна быть продемонстрирована реализуемость проекта в контексте объектной модели.* Иными словами, должна быть уверенность в том, что найдены объекты, совместная работа которых обеспечивает нужное поведение системы.

Должно быть очевидно, какие сообщения или группы сообщений в правой части диаграммы последовательностей соответствуют каждому предложению в тексте прецедента как из главной, так и изо всех альтернативных последовательностей. Это очень важно, поскольку дает возможность проследить связь между проектными решениями и функциональными требованиями.

Следующее, на что нужно обратить внимание, - это непрерывность сообщений. На диаграммах последовательностей *очень важно* правильно проставлять направление стрелок-сообщений, тем самым показывая поток управления. В любой момент должно быть ясно, какой объект владеет управлением.

Решения о распределении поведения, принимаемые проектировщиками, влияют на качество классов. Халберт (Halbert) и О'Брайен (O'Brein) [7] сформулировали четыре критерия «хорошего» класса:

- *возможность повторного использования.* Чем более общими являются классы и объекты, тем выше вероятность повторного использования их в других проектах. Необходимо отслеживать, как скажется добав- ление нового метода в класс на повторном использовании;
- *применимость.* Концепция применимости в контексте моделирования взаимодействий означает по сути дела то же самое, что и при моделировании предметной области или прецедентов. Распределяя методы по объектам на диаграммах последовательностей необходимо определять, уместен ли данный метод в данном объекте, относится ли задача, решаемая этим методом, именно к данному объекту;
- *сложность.* Определяет насколько серьезно проектировщик подходит к реализации. Вопрос ставится так: будет ли проще поместить данный метод в этот или какой-то другой объект;
- *задание реализации.* Этот критерий связан с таким вопросом: зависит ли реализация поведения от внутренних деталей соответствующего метода.

Возможно, самым важным критерием является применимость.

Далее необходимо выяснить отвечают ли классы следующим *критериям качества*:

- *связанность (coupling)* характеризует обилие связей между двумя классами. Модульность системы повысится, если классы будут слабо связанными, т.е. мало зависящими друг от друга;
- *плотность (cohesion)* характеризует, насколько сильно зависят друг от друга атрибуты и операции одного класса. Нужно стремиться к *высокой функциональной плотности*. Это означает, что для обеспечения четко определенного поведения нужна слаженная совместная работа всех элементов класса;
- *достаточность (sufficiency)* означает, что класс инкапсулирует достаточное число абстракций, присутствующих в модели, т.е. представляет собой нечто, с чем можно осмысленно и эффективно взаимодействовать. *Главный вопрос* – охватывает ли класс все взаимосвязанные прецеденты;

- *полнота (completeness)* характеризует то, в какой мере интерфейс класса покрывает все абстракции, для которых он разработан. Теоретически полный класс может повторно использоваться в любых контекстах;
- *примитивность (primitiveness)* говорит о том, что операция может быть эффективно реализована при наличии доступа к другим уже построенным элементам. Идея заключается в том, чтобы одни операции создавались как строительные блоки для других.

Еще один критерий качества диаграммы последовательностей – это достаточное количество деталей. Поскольку, диаграммы последовательностей – последняя остановка перед кодированием, то на них должен быть отражен истинный проект во всех деталях. Данный этап нельзя считать завершенным, пока *все* методы, показанные на диаграммах последовательностей, не будут включены в тот или иной класс из статической модели.

В ходе рецензирования окончательного проекта необходимо еще раз проверить соответствие детального проекта выбранной технической архитектуре.

Реализация информационной системы
Конструирование кода информационной системы

Имея в своём распоряжении диаграммы классов, диаграммы объектов, диаграммы активности и диаграммы компонентов, программисты реализуют систему в коде на конкретном языке программирования. В качестве языка программирования используем Delphi 6. Ниже приведено описание классов на языке Object Pascal, реализация методов не приводится.

```
unit Main_Unit;
interface
type

TVisitor=class;

Tissue=class
         // атрибуты
    title     :string;
    year :integer;
    keyWords  :TStrings;
    price     :integer;
         // методы
    function getDetail():TStrings;
end;

TBook=class(TIssue)
    author         :string;
    published :string;
    pages          :integer;
end;

TMagazine=class(TIssue)
```

```
        number    :byte;
    eEnd;

    TNewsPaper=class(TIssue)
        day       :byte;
        month          :byte;
    end;

    Tissues=array of TIssue;

    TCatalogItem=class
             // атрибуты
        name  :string;
        issues    : Tissues;
             // методы
        procedure addIssue(issue:TIssue);
        procedure deleteIssue(issue:TIssue);
    function searchByAu-
thor(author:string):Tissues;
    function searchByTitle(title:string):Tissues;
    function searchByKey-
Words(author:string):Tissues;
    end;

    TCatalog=class
             // атрибуты
    dateOfUpdate  :TDateTime;
    catalofItems  :array of TCatalogItem;
             // методы
    function                              searchByAu-
thor(author:string):TIssues;
    function searchByTitle(title:string):TIssues;
    function                             searchByKey-
Words(author:string):TIssues;
    procedure deleteIssue(issue:TIssue);
    procedure addIssue(issue:TIssue);
    end;

    TPay=class
        PercentOfPay   :integer;
    end;

    TPostPay=class(TPay)
    end;

    TBoyPay=class(TPay)
    end;

    TNoCashtPay=class(TPay)
    end;


    TOrder=class
```

```
            //атрибуты
        dateOfOrder         :TDate;
        visitor             :TVisitor;
        issues              :TIssues;
        typeOfPay      :TPay;
        doIt           :boolean;
            //методы
        function getOrderSum():integer;
    end;

    TOrders=array of TOrder;

    TVisitor=class
            //атрибуты
        name      :string;
        surname   :string;
        address   :string;
        phone          :string;
        email          :string;
            //методы
        function  getOrderHistory(): TOrders;
        function  getCurrentOrder(): TOrders;
    end;

    TOrderController=class
        //методы
        function                    checkOrder(order
:TOrder):boolean;
        function getOpenOrders():TOrders;
        function                getOrderByDate(date
:TDate):TOrders;
         end;
```

Тестирование информационной системы

Тестирование начинается с тестирования отдельных программных модулей, например процедур и объектов. Затем модули компонуются в подсистемы и потом в систему, при этом проводится тестирование взаимодействий между модулями. Наконец, после сборки системы, заказчик может провести серию приемочных тестов, во время которых проверяется соответствие системы её спецификации.

Для критичных систем процесс тестирования должен быть формальным. Такая формализация предполагает, что за все этапы тестирования отвечают независимые испытатели, все тесты разрабатываются отдельно и во время тестирования ведутся подробные записи.

При разработке некритических, «обычных» систем, подробные спецификации для каждого системного компонента, как правило, не создаются. При этом тестирование, как правило, сводится к пониманию программистами функциональности компонент и тестированию их работы в процессе отладки.

Обычно программисты сами тестируют написанный ими программный код для обнаружения ошибок и программных дефектов. Этот процесс называ- ется *отладкой* программы. В принципе тестирование и отладка являются раз- ными процессами. При тестировании устанавливается наличие программных ошибок. В ходе отладки устанавливается местоположение ошибок, затем они устраняются. Отладка может быть как частью процесса разработки, так и частью процесса тестирования ПО.

Целью тестирования дефектов является выявление в программной системе скрытых дефектов до того, как она будет сдана. Тестирование дефектов противоположно аттестации, в ходе которой проверяется соответствие системы своей спецификации. Во время аттестации система должна корректно работать со всеми заданными тестовыми данными. При тестировании дефектов запускается такой тест, который вызывает *некорректную* работу программы и, следовательно, выявляет дефект.

Автоматическая генерация сценариев тестирования невозможна, поскольку результаты поведения теста не всегда можно предсказать заранее.

Полное тестирование, когда проверяются все возможные последовательности выполнения программы, нереально. Поэтому тестирование, должно базироваться на некотором подмножестве всевозможных тестовых сценариев.

Из опыта тестирования больших программных систем следует, что необычные комбинации функций чаще вызывают ошибки, чем часто используемые функции.

Необходимо особо тщательно планировать тесты с некорректными входными данными, так как такие тесты наиболее часто выявляют дефекты программных продуктов.

Приложение 4

Изучение CASE-средства ArgoUML

Тема. Разработка и создание UML диаграмм.

Цель. Создание в среде ArgoUML диаграмм UML.

Ход работы

1. Ознакомиться с теоретической частью.
2. Выполнить практическое задание.
3. Ответить на контрольные вопросы.
4. Оформить отчет.

Теоретическая часть

После запуска ArgoUML появится окно. Окно ArgoUML состоит из следующих компонентов: меню, панели инструментов и четырех панелей. Вверху слева находится панель ***Explorer***, затем идет панель ***Редактирования***, затем внизу панель ***Деталей*** и ***To-Do*** панель.

В панели ***Explorer*** содержится список всех классов, интерфейсов и типов данных модели в виде дерева.

На панели ***Редактирования*** можно редактировать диаграммы.

В панели ***To-Do*** отображаются элементы моделей.

Описание пунктов меню:

Файл (File) – позволяет создавать новые проекты, сохранять и открывать проекты, импортировать, импортировать источники из другого расположения, скачать и сохранить модель из/в базу данных, печать модель, сохранить модель, сохранять конфигурацию модели и выход из программы.

Редактировать (Edit) – позволяет выбрать один или несколько UML элементов в диаграмме, а также повторить действия, перемещать элементы из диаграммы и так далее.

Вид (View) – позволяет переключаться между диаграммами, искать артефакты в модели, масштабировать диаграмму, выбирать особенное представление диаграмм и т.п.

Создать диаграмму (**Create Diagram**) -позволяет создавать любую диаграмму из семи UML диаграмм, поддерживаемых ArgoUML (class, use case, state, activity, collaboration, deployment и sequence).

Расставить (Arrange) – позволяет выровнять, распределить, упорядочить артефакты в диаграмме.

Генерация кода (Generation) -позволяет сгенерировать Java код для отдельных или всех классов.

Критика (Critique) –переключать авто рецензирование, устанавливать уровень важности проблемы дизайна и дизайн целей и увидеть критических изменения. Инструменты (***Tools***), Помощь (***Help***)

Практическая часть

Задание 1. Для создания диаграммы вариантов использования, выполните следующие действия:

1. Дважды щелкните на значке ***untitledModel.***

2. Нажмите на значок ***Диаграмма вариантов*** использования в проводнике, чтобы открыть диаграмму ***Вариантов использования***.

3. С помощью кнопки ***Use Case*** (вариант использования) на панели инструментов поместите на диаграмму новый вариант использования.

4. Назовите этот новый вариант «***Ввести новый заказ***», для этого вам нужно дважды нажать на варианте использования и ввести имя или активировать вкладку ***Свойства To-Do*** панели в поле ***Имя***: введите имя.

5. Повторите шаги 3 и 4, чтобы разместить на диаграмме остальные варианты.

6. С помощью кнопки ***Actor*** (Действующее лицо) инструментов поместите на диаграмму новое действующее лицо

7. Назовите его «***Продавец***», действия аналогичные вариантам использования.

8. Повторите шаги 6 и 7, поставив на диаграмме остальных актеров.

Задание 2. Для указания абстрактного варианта использования, выпол- ните следующие действия:

1. ***ЛКМ*** нажмите на вариант использования «***Отклонить заказ***» на диаграмме.
2. В панели ***Свойств*** в области ***modifiers*** выберите контрольный переключатель ***isAbstract (Абстрактный)***, чтобы сделать этот вариант использования абстрактным.

Задание 3. Для добавления ассоциации выполните следующие действия:

1. Нажав на кнопку ***Association (Ассоциация)***, выберите ***UniAssociation (Однонаправленная Ассоциация***), изобразите ассоциацию между актером Продавец и вариантом использования «Ввести новый заказ».
2. Повторите этот шаг, чтобы разместить на диаграмме остальные ассоциации.

Задание 4. Для добавления связи расширения выполните следующие действия:

Нажав на кнопку **Extend** на панели инструментов, нарисуйте связь между вариантом использования «***Отклонить заказы***» и вариантом использования «***Оформить заказ***». Стрелка должна быть протянута от первого варианта использования ко второму. Связь расширения означает, что вариант использования «***Отклонить заказ***», при необходимости, дополняет функциональные возможности варианта использования «***Оформить заказ***». Диаграмма должна иметь вид, как на рисунке П4.1.

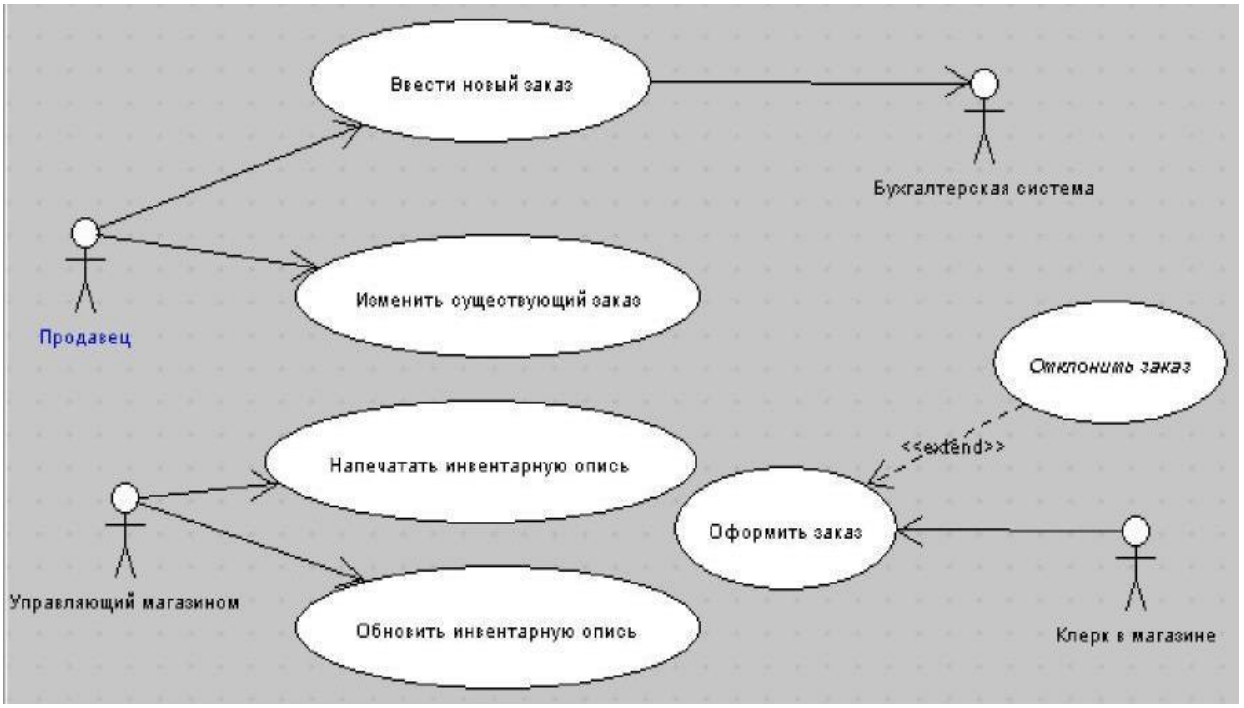


Рис. П4.1.– Диаграмма вариантов использования
для системы обработки заказов

Задание 5. Создать главную диаграмму классов.

1. Дважды щелкните на значке ***UntitledModel***, отобразится содержимое пакета. Выберите Диаграмма классов. С помощью кнопки ***Новый пакет*** и панели инструментов разместите на диаграмме новый па- кет. Во вкладке ***Свойства***, укажите имя пакета ***Entities (Сущность).***
2. Создайте пакеты ***Boundaries (Границы)*** и ***Control (Управление).***

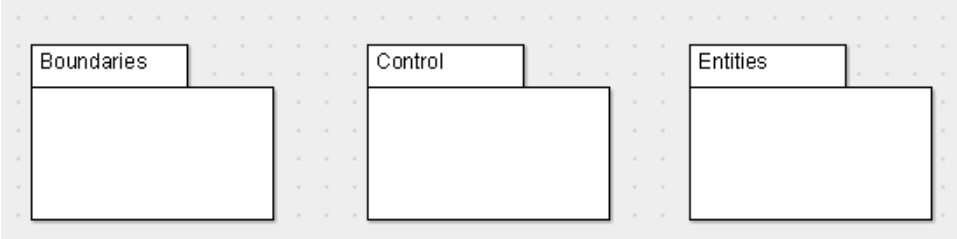


Рис. П4.2. Главная диаграмма классов системы обработки заказов

Задание 6. Создание диаграммы классов для сценария «Ввести новый заказ» со всеми классами:

1. В меню ***Создать*** диаграмму, выберите ***Диаграмма классов***.
2. Во вкладке ***Свойства*** введите имя для новой диаграммы класса ***Add New Order (Ввод нового заказа)***.
3. Щелкните в браузере на этой диаграмме, чтобы открыть ее.
4. Создайте в окне диаграммы классов классы ***OrderOptions*** (Выбор заказа), ***OrderDetail*** (Детали заказа), ***Order*** (Заказ), ***OrderMgr*** (Менеджер заказов) и ***TransactionMgr*** (Менеджер транзакций). Имя класса вводится во вкладке ***Свойства***.
5. Добавьте операции, указанные на диаграмме классов (рис.П4.3). Чтобы сделать это, выберите соответствующий класс, нажмите на кнопку ***До-***

бавить операцию, здесь вы можете ввести видимость операции. Диаграмма классов должен выглядеть как на рисунке П4.3.

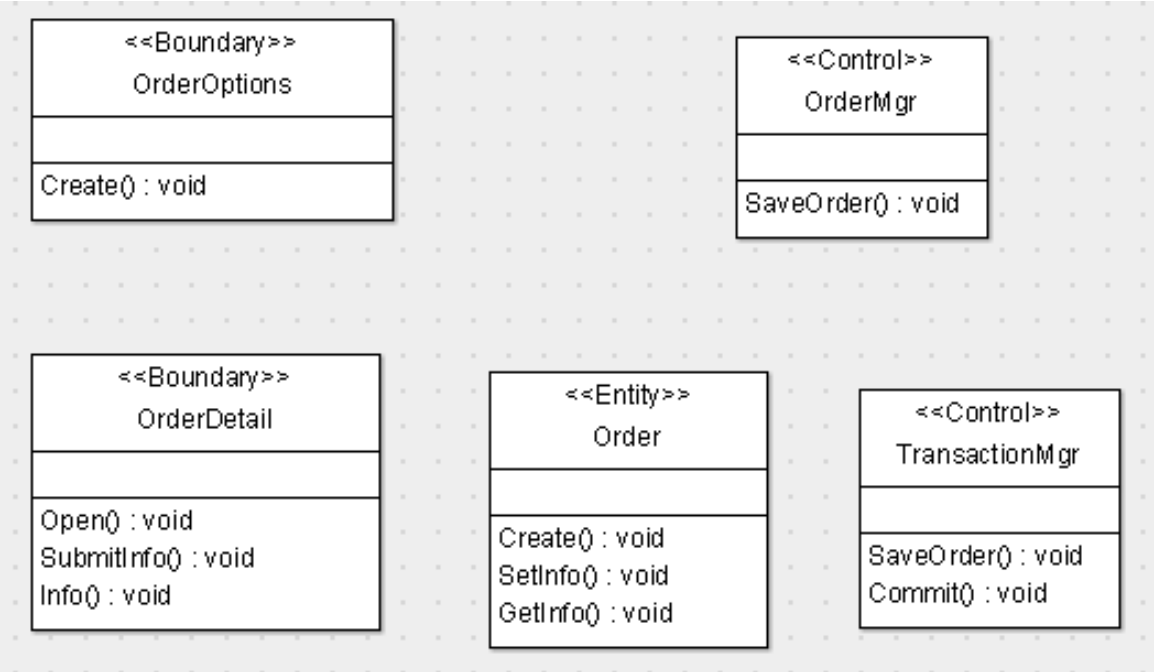


Рис. П4.3. Стереотипы классов для варианта использования «Ввести новый заказ»

Задание 7. Добавление стереотипов в классы:

1. Нажмите на класс ***OrderOptions***.
2. Введите в поле имя класса стереотип <<***Boundary***>>.
3. В раскрывающемся списке стереотипов теперь будет стереотип ***Boundary*** . Укажите его.
4. Нажмите с помощью мыши на классе ***OrderDetail***.
5. Из раскрывающегося списка Стереотип выберите стереотип ***Boundary.***
6. Свяжите классы ***OrderMgr*** и ***TransactionMgr*** со стереотипом ***Control***, а класс ***Order*** – со стереотипом ***Entity***. Теперь диаграмма классов должна выглядеть как на рисунке П4.3.

Задание 8. Объединение классов в пакеты.

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

Задание 9. Добавить диаграмму классов к каждому пакету.

Для этого необходимо выполнить следующие действия:

1. Дважды щелкните левой кнопкой на пакете ***Boundaries*** в окне диаграмм.
2. В появившемся окне выберите ***Да***.
3. Во вкладке ***Свойства,*** введите имя для новой диаграммы – ***MainB***.

4. Щелкните на диаграмму, чтобы открыть его.

5. В браузере выделите класс ***OrderOptions*** (потом ***OrderDetail***), щелкните ПКМ и выберите опцию ***Добавить в диаграмму***, а затем нажмите мышью в поле диаграммы.

6. Классы будут отображаться на диаграмме.

7. Выполните эти же шаги, чтобы поместить классы ***OrderMgr*** и ***TransactionMgr*** на главную диаграмму классов пакета ***Control*** (MainC).

8. Выполните эти же действия, чтобы поместить класс ***Order*** на главную диаграмму классов пакета ***Entities*** (MainE).

Задание 10. Для того чтобы создать диаграммы кооперации выполните следующие действия:

1. Нажмите на значок ***untitledModel*** в браузере.
2. Из меню ***Создать диаграмму***, чтобы выберите ***Диаграмму кооперацийй***.
3. Назовите эту диаграмму ***Ввод заказа***.
4. Нажмите на нее, чтобы открыть.

Задание 11. Для добавления объектов к диаграмме кооперацийй, выполните следующие действия:

1. Перетащите с диаграммы вариантов использования объект ***Продавец*** на рабочую область.
2. На панели инструментов нажмите кнопку ***ClassifierRole.***
3. Щелкните в любом месте диаграммы, чтобы разместить там новый объект.
4. Назовите объект «***Order Options Form***».
5. Помести на диаграмме остальные объекты: Order Detail Form, ***Order Manager***, ***Transaction Manager***, ***Order #1234***.

Задание 12. Добавления сообщений на диаграмму кооперацийй:

1. На панели инструментов нажмите кнопку ***Новая ассоциация***.
2. Свяжите ***Продавец*** с объектом ***Order Options Form***.
3. Соедините остальные объекты.
4. Нажмите на связь между ***Продавцом*** и ***OrderOptionsForm***. На панели инструментов нажмите кнопку ***Добавить сообщение***. Нажмите на связь между ***Продавцом*** и ***Order Options Form***. Выбрав сообщение, введите его имя ***Create ().***
5. Поместите на диаграмме оставшиеся сообщения: ***Open (), SubmitInfo (), Save (), SaveOrder(), SetInfo (), GetInfo ()***.
6. Нажмите на объекте для добавления к нему сообщения рефлесии. По бокам объекта будут показаны значки, среди которых необходимо выбрать .
7. Нажмите на связь рефлексии ***Transaction Manager***, чтобы ввести сообщение. На панели инструментов нажмите кнопку ***Добавить сообщение***. Назовите новое сообщение ***Commit ()*** (Сохранить информацию о заказе в базе данных) (рис. П4.4).

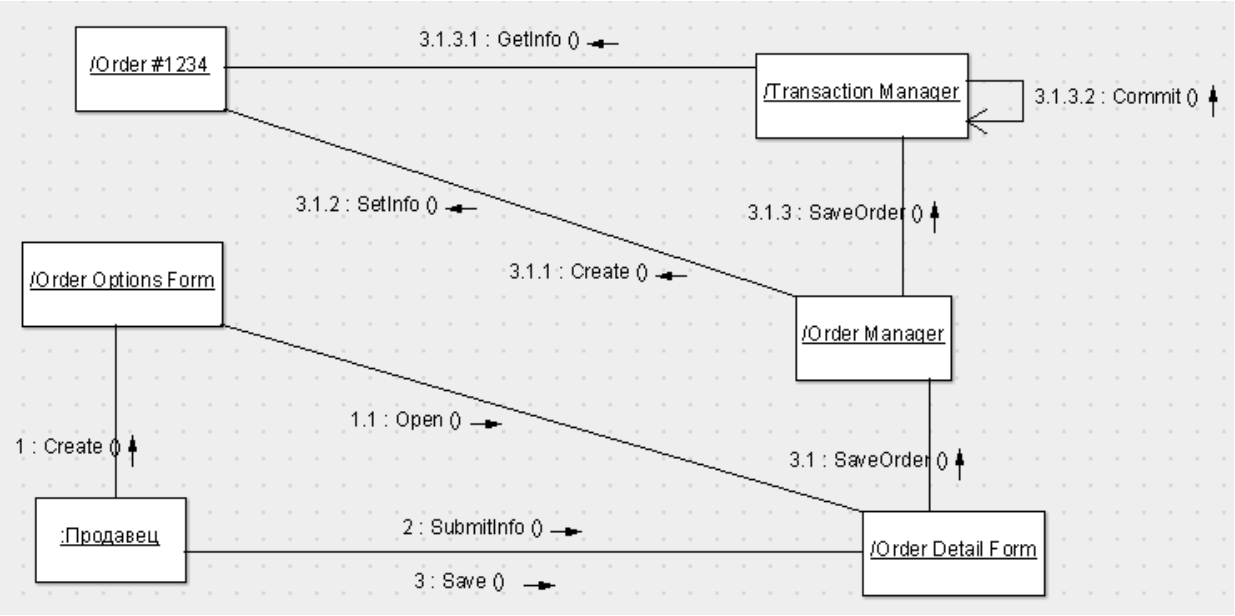


Рис. П4.4. Диаграмма кооперацій

Задание 13. Для создания диаграммы последовательностей, выполните следующие действия:

1. Щелкните ***ПКМ*** логическое представление браузера.
2. В открывшемся меню нажмите ***Create Diagram → Диаграмма последовательности***.
3. Назовите новую диаграмму «***Ввод заказ***».
4. Дважды щелкните на нем, чтобы открыть.

Задание 14. Для добавления на диаграмму актера и объектов, выполните следующие действия:

1. Перетяните актера ***Продавец*** из браузера на диаграмму. Для панели панель инструментов, нажмите кнопку ***New Classifier Role***. Имя объекта ***Order Options Form*** – Выбор варианта заказа.
2. Таким же образом добавьте все элементы, описанные в задании 11 пункте 5 (рис. П4.5).

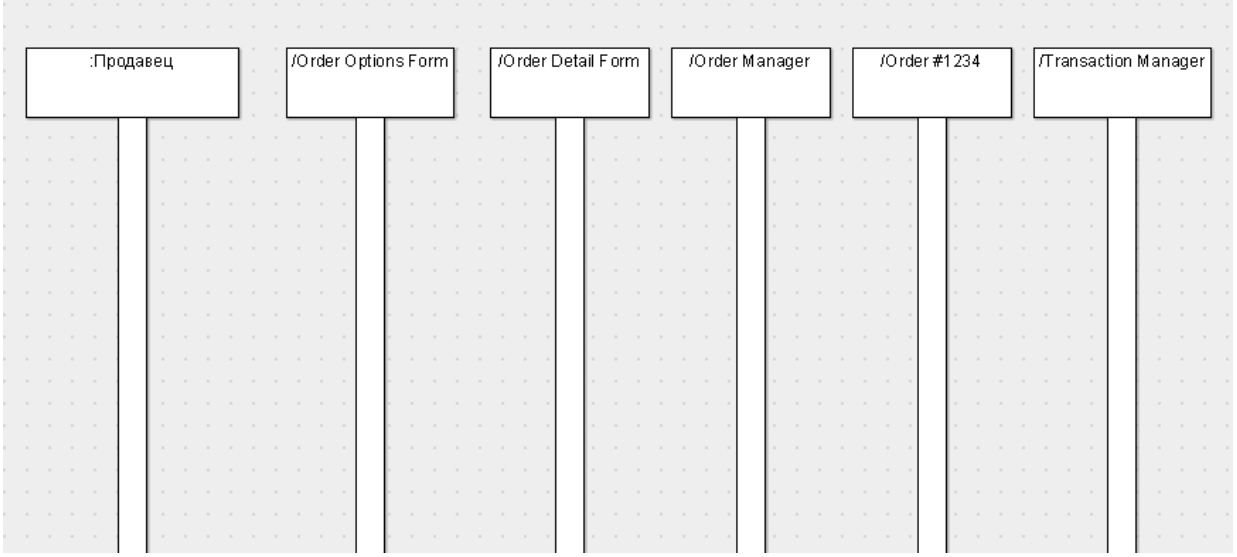


Рис. П4.5. Диаграмма последовательности

Задание 15. Для добавления сообщений на диаграмму, выполните следующие действия:

1. На панели инструментов нажмите кнопку ***Действие отправки***, перетащите от линии жизни актера к линии жизни объекта ***Order Options Form***. В ArgoUML в данном виде диаграмм сообщения автоматически не нумеруются, поэтому нумерация переносится из диаграммы кооперации. Выбрав сообщение, введите имя ***1: Create ().***
2. Добавьте оставшиеся сообщения, описанные в задании 12 пункте 5.

Задание 16. Сопоставление объектов с классами:

1. Нажмите на объекте ***Order Options Form***.
2. В панели ***Свойств*** раскройте список ***База***.
3. В открывшемся списке нажмите на кнопку с плюсом.
4. В открывшемся окне выберите класс ***OrderOptions*** и нажмите на кнопку со стрелкой для добавления в список.
5. Нажмите кнопку ***ОК***.
6. Сопоставьте остальные объекты с классами: ***Order Detail Form - OrderDetail, Order Manager – OrderMgr, Transaction Manager - TransactionMgr, Order #1234 - Order.***

Задание 17. Соотношение сообщений с операциями.

1. Нажмите на сообщение ***1: Create () заказ***.
2. В окне ***Свойств*** дважды щелкните в поле ***Действие***. В открывшемся окне введите ***Create ().***
3. Сопоставьте остальные сообщения.

Задание 18. Добавление нового класса:

1. Найдите в дереве (браузере) диаграмму классов «***Add New Order***». Нажмите на нее дважды, чтобы открыть.
2. Поместите на этой диаграмме новый класс ***OrderItem*** (Позиция заказа). Назначьте этому классу стереотип ***Entity***.
3. В дереве перетащите класса в пакет ***Entities***.

Задание 19. Добавление атрибутов:

1. Щелкните ПКМ на классе ***Order*** (Заказ).
2. В открывшемся меню выберите ***Добавить → Новый атрибут***.
3. Введите новый атрибут ***OrderNumber : Integer*** и нажмите ***Enter.*** Тип атрибута можно выбрать из раскрывающегося списка в поле ***Тип*** или ввести вручную.
4. Введите следующий атрибут ***CustomerName : String***.
5. Добавьте атрибуты ***OrderDate : Date и OrderFillDate : Date***.
6. Щелкните ПКМ на классе ***OrderItem***.
7. В открывшемся меню выберите ***Добавить → Новый атрибут***.
8. Введите новый атрибут ***ITEMID : Integer***.
9. Введите следующий атрибут ***ItemDescription : String***.

Задание 20. Добавления операций к классу ***OrderItem:***

1. Щелкните ***ПКМ*** на классе ***OrderItem***.
2. В открывшемся меню выберите ***Добавить → Новая операция***.
3. Введите новую операцию ***Create.***
4. Введите следующую операцию ***SetInfo***.
5. Введите следующую операцию ***GetInfo***.

Задание 21. Подробное описание операций с помощью диаграммы класса:

1. Нажмите на класс ***Order***, выбрав его таким образом.
2. Нажмите на этот класс в разделе операций.
3. Отредактируйте операцию ***Create()*** так, чтобы она выглядела таким образом: ***Create () : Boolean***. Для этого дважды нажмите на операцию ***Create*** в разделе операций класса.
4. Отредактируйте операцию ***SetInfo(),*** чтобы она выглядела следующим образом: ***SetInfo(OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date) : Boolean***
5. Отредактируйте операцию ***GetInfo (),*** чтобы она выглядела следующим образом: ***GetInfo () : String***

Задание 22. Подробное описание операций остальных классов:

1. С помощью браузера или диаграмм, введите следующую сигнатуру операций класса ***OrderDetail: Open () : Boolean, SubmitInfo() : Boolean, Save() : Boolean***.
2. С помощью браузера или диаграмм, введите следующую сигнатуру операций класса ***OrderOptions : Create () : Boolean*** .
3. С помощью браузера или диаграмм, следующую сигнатуру операций класса ***OrderMgr: SaveOrder(ORDERID: Integer): Boolean.***
4. С помощью браузера или диаграмм, введите следующую сигнатуру операций класса ***TransactionMgr: SaveOrder (ORDERID: Integer): Boolean, Commit (): Integer***.

Задание 23. Добавление отношения между классами.

Чтобы найти отношения, были изучены диаграммы последовательности. Все классы, которые там взаимодействуют, требовали определения соответствующих связей на диаграммах классов. После выявления связей, их добавили в модель. Добавьте отношения, как показано на рисунке П4.6.

Множественность указывается выбором из контекстного меню связи опции ***Multiplicity***. После добавления связей диаграмма классов должна выглядеть как на рис. П4.6.

Задание 24. Создание диаграммы состояний.

1. Найдите в браузере класс ***Order***.
2. Выделите этот класс.
3. В меню ***Создать диаграмму*** выберите ***Диаграмма состояний***.

Задание 25. Добавление начального и конечного состояний.

1. На панели инструментов нажмите кнопку ***New Initial*** (***Начальное состояние)***. Поместите это состояние на диаграмму.
2. На панели инструментов нажмите ***Конечное состояние***. Поместите на диаграмму.

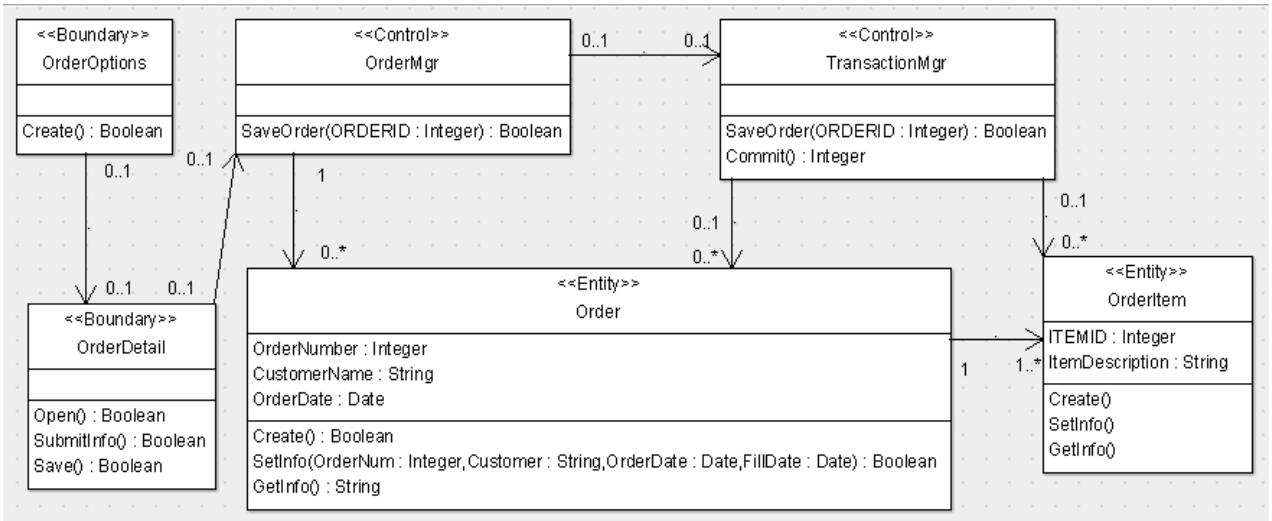


Рис. П4.6. Диаграмма классов Add New Order

Задание 26. Добавление ***Супер состояния***:

1. На панели инструментов нажмите кнопку ***Композитное состояние.***
2. Поместите это состояние на диаграмму.

Задание 27. Добавление оставшихся состояний.

1. На панели инструментов нажмите кнопку ***Простое состояние***. Поместите его на диаграмму. Назовите состояние ***Отменен.***
2. На панели инструментов нажмите кнопку ***Простое состояние***. Поместите его на диаграмму. Назовите состояние ***Выполнен.***
3. На панели инструментов нажмите кнопку ***Простое состояние***. Поместите это состояние внутрь ***Суперсостояния***. Назовите состояние ***Инициализация.***
4. На панели инструментов нажмите кнопку ***Простое состояние***. Поместите это состояние внутрь С***уперсостояния***. Назовите состояние ***Сборка заказа.***

Задание 28. Подробное описание состояний.

Для добавления в состояния действий на входе, выходе и деятельности исполнения на вкладке ***Свойства состояния*** выберите соответственно: действие при входе, действие при выходе, деятельность выполнения.

Задание 29. Добавление переходов:

1. На панели инструментов нажмите кнопку ***New Transition (Переход*).**
2. Нажмите на начальной точке. Проведите линию перехода к состоянию ***Инициализация.***
3. Повторить предыдущие шаги для добавления оставшихся переходов.

Задание 30. Подробное описание переходов

1. Нажмите на переход от состояния ***Инициализация*** к состоянию ***Сборка заказа***, открывая окно его свойств.
2. В поле ***Имя*** введите ***Выполнить заказ***.
3. Повторите этапы, добавив событие ***Отменить заказ*** для переключения между ***Супер состоянием*** и состоянием ***Отменен***.
4. Нажмите на переход от состояния ***Сборка заказа*** к состоянию ***Заполнены (Выполнено)***, открывая окно его свойств.
5. В поле ***Имя*** введите фразу ***Добавить Порядок пункта (Добавить в заказ на новое место)***.

Диаграмма состояний должна иметь вид как на рис. П4.7.

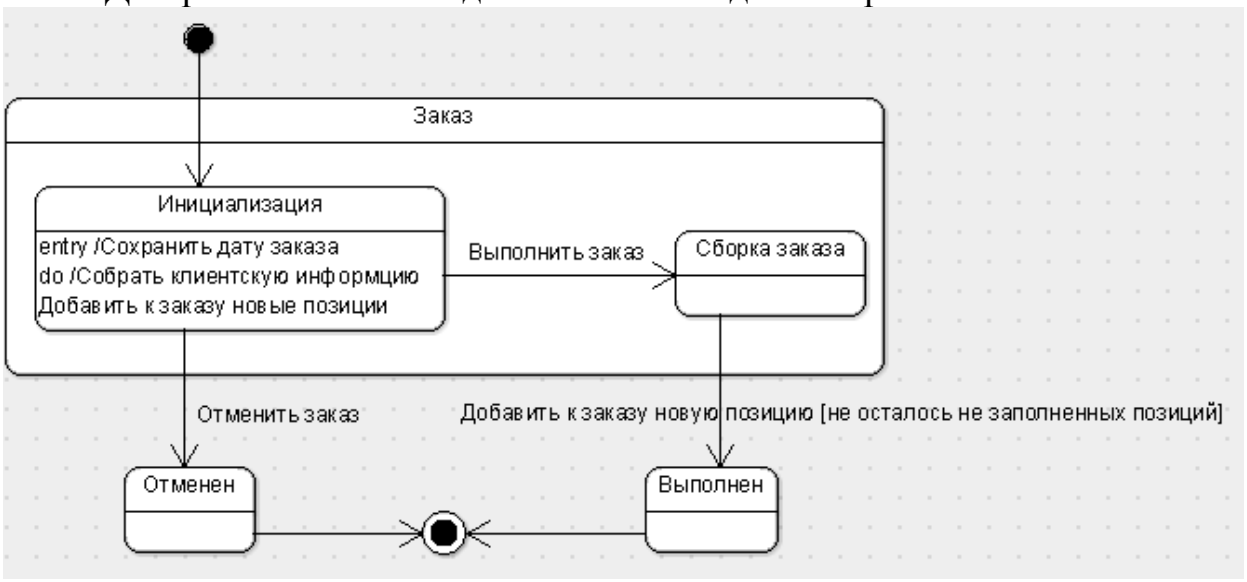


Рис. П4.7. Диаграмма состояний

Задание 31. Добавление компонентов и изображение зависимостей

1. Выполните команду ***Создать диаграмму → Диаграмма развертывания***.
2. Нажмите на название диаграммы в браузере. Введите имя ***Entities***.
3. На панели инструментов нажмите кнопку ***New Component*** (Компонент).
4. Щелкните в поле диаграммы. Введите имя компонента.
5. Добавить все компоненты.
6. На панели инструментов нажмите кнопку ***New Dependency*** (Зависимость). Проведите необходимые зависимости. Аналогично создайте диаграммы ***Control*** и ***Boundaries***.

Диаграмма компонентов будет иметь вид как на рис. П4.8–П4.10.

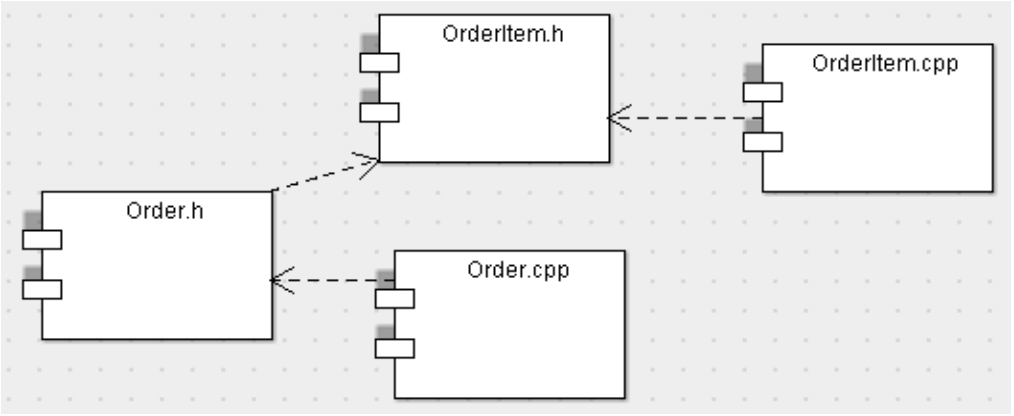


Рис. П4.8. Диаграмма компонентов Entities

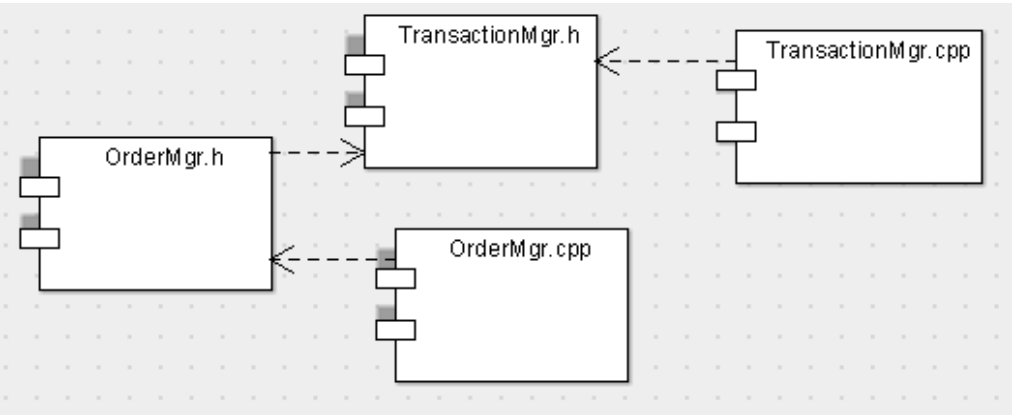


Рис. П4.9. Диаграмма компонентов Control

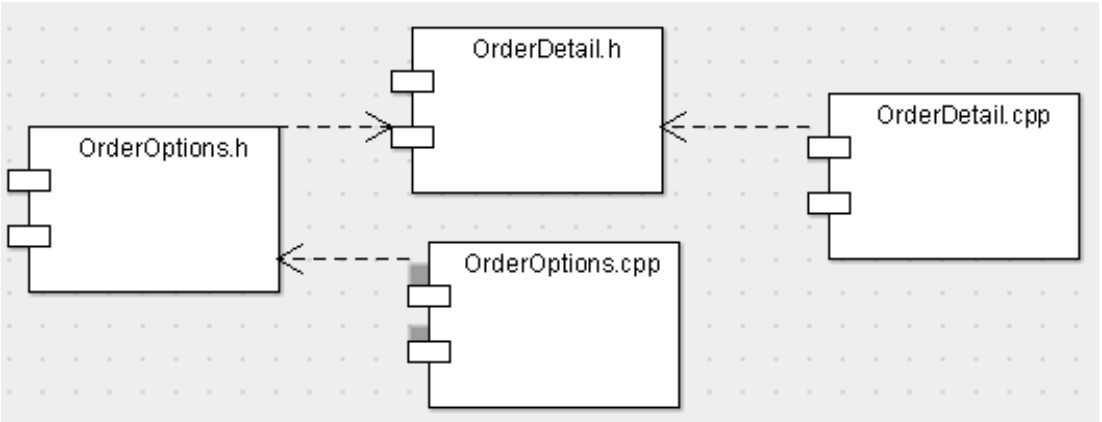


Рис. П4.10. Диаграмма компонентов Boundaries

Задание 32. Создание диаграммы компонентов системы

1. Выполните команду ***Создать диаграмму → Диаграмма развертывания***.
2. Нажмите на название диаграммы ***Диаграмма компонентов системы***.
3. Перетащите из дерева на рабочую область диаграммы компоненты ***OrderDetail.h, OrderOtions.s, OrderMgr.h, TransactionMgr.h, Order.h, OrderItem.h***.

4. Создайте дополнительные компоненты ***OrderClient.exe и OrderServer.exe.***

5. Создайте зависимости между компонентами, которые не были созданы автоматически.

Диаграмма компонентов системы примет вид как на рис. П4.12.

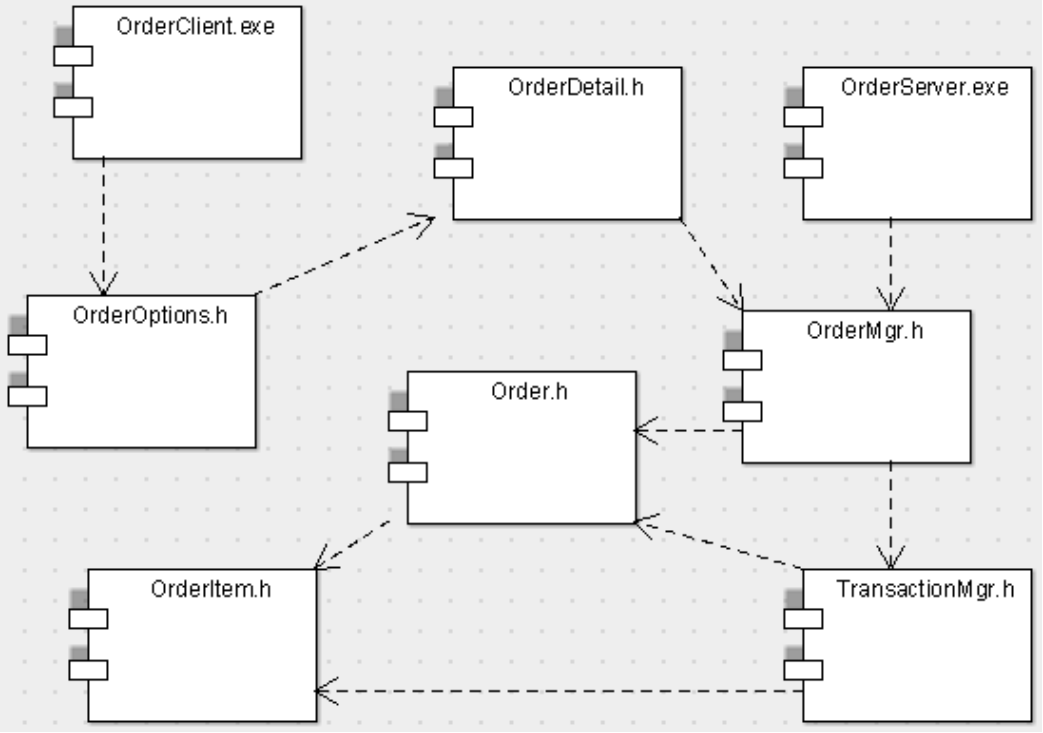


Рис. П4.12. Диаграмма компонентов системы

Задание 33. Добавление узлов на диаграмму размещения (развертывания):

1. Выберите пункт меню ***Создать диаграмму → Диаграмма развертывания.***
2. На панели инструментов нажмите кнопку ***New Node.***
3. Введите имя узла «***Сервер базы данных***».
4. Добавьте оставшиеся узлы.

Задание 34. Добавление связей:

1. На панели инструментов нажмите кнопку ***New Link.***
2. Нажмите на «***Сервер базы данных***».
3. Проведите линию до узла «***Сервер приложений***».
4. Добавьте остальные связи.

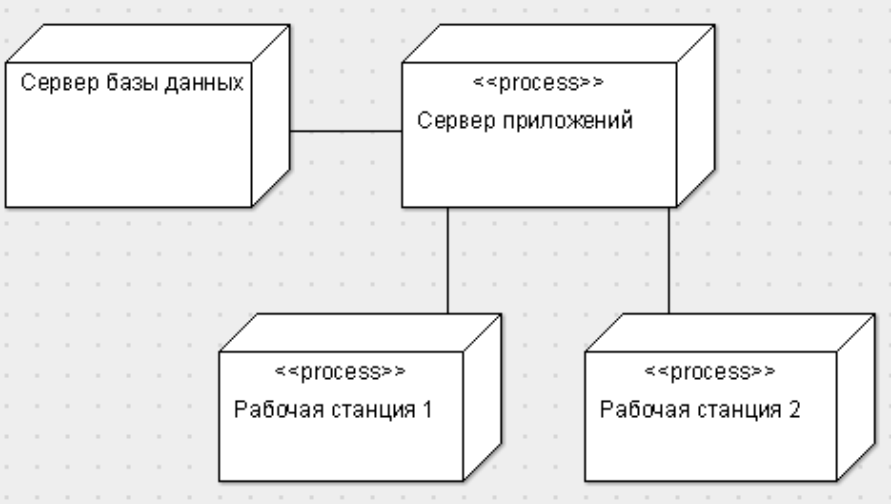


Рис. П4.13. Диаграмма размещений

Задание 35. Добавление процессов и операций

1. Щелкните ПКМ на «***Сервер приложений***». Из раскрывающегося списка выберите ***Apply Stereotypes → <<process>>.***
2. Выполните аналогичные действия для объектов ***Рабочая станция 1*** и ***Рабочая станция 2***.
3. Выделите «***Сервер приложений***». На панели ***Свойств*** в поле ***Операции*** нажмите ***ПКМ*** и выберите ***Новая операция***. В поле ***Имя*** введите ***OrderServerExe***.
4. Добавьте остальные процессы: ***Рабочая станция 1 – OrderClientExe, Рабочая станция 2 – ATMClientExe.***

Задание 36. По результатам выполнения практического задания создайте отчет, содержащий следующую информацию:

1. Тема. Цель. Оборудование.
2. Результат выполнения практического задания.
3. Ответы на контрольные вопросы. Вывод.

Контрольные вопросы

1. Для чего используется язык UML?
2. Опишите основные диаграммы UML.
3. Опишите возможности пакета ArgoUML.
4. Опишите последовательность действий при создании диаграмм классов, компонент, состояний, последовательностей, вариантов использования, размещения.