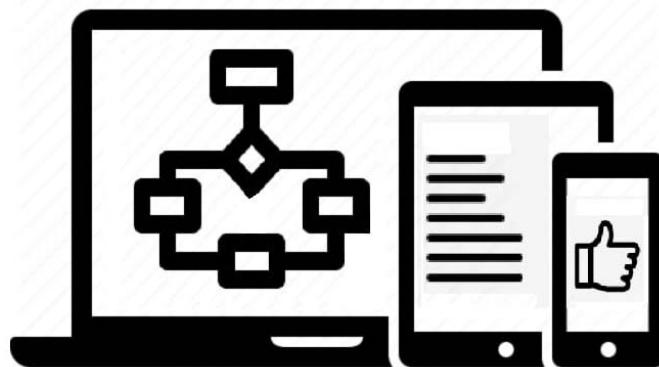


МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

*Методические рекомендации к лабораторным работам
для студентов направления подготовки
09.03.01 «Информатика и вычислительная техника»
дневной формы обучения*



Могилев 2022

УДК 004.41/.42
ББК 32.973-018
П79

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «30» сентября 2022 г., протокол № 3

Составители: канд. техн. наук, ст. преподаватель Ю. В. Вайнилович;
канд. техн. наук, доц. С. К. Крутолевич

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации содержат требования к лабораторным работам по дисциплине «Проектирование программного обеспечения».

Учебно-методическое издание

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Ответственный за выпуск	В. В. Кутузов
Корректор	Т. А. Рыжикова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский
университет, 2022

Содержание

Введение.....	4
1 Лабораторная работа № 1. Анализ текущего состояния предметной области.....	5
2 Лабораторная работа № 2. Разработка бизнес-требований с использованием инструмента Lean Canvas.....	10
3 Лабораторная работа № 3. Применение техники Impact Mapping стратегического планирования	11
4 Лабораторная работа № 4. Описание архетипов пользователей.....	12
5 Лабораторная работа № 5. Построение карты пользовательских историй	12
6 Лабораторная работа № 6. Написание пользовательских историй (User Story Writing)	13
7 Лабораторная работа № 7. Написание критериев приемки (Acceptance Criteria).....	14
8 Лабораторная работа № 8. Построение диаграммы вариантов использования. Спецификация вариантов использования	15
9 Лабораторная работа № 9. Построение диаграммы классов	18
10 Лабораторная работа № 10. Построение диаграмм взаимодействия информационной системы.....	20
11 Лабораторная работа № 11. Построение диаграмм последовательности информационной системы.....	24
12 Лабораторная работа № 12. Построение диаграммы компонентов.....	26
13 Лабораторная работа № 13. Построение диаграммы развертывания.....	29
14 Лабораторная работа № 14. Построение диаграммы состояний информационной системы.....	31
15 Лабораторная работа № 15. Разработка системной документации	33
16 Лабораторная работа № 16. Разработка руководства пользователя.....	34
Список литературы	34

Введение

Целью курса является формирование специалистов, умеющих обоснованно и результативно применять существующие и осваивать новые технологии разработки программного обеспечения.

Методические рекомендации предназначены для проведения лабораторных работ по дисциплине «Проектирование программного обеспечения».

Методические рекомендации содержат шестнадцать лабораторных работ.

Тему индивидуального задания выдает преподаватель на первой лабораторной работе.

При оформлении отчетов по лабораторным работам придерживаются следующих правил:

- отчёт должен быть отпечатан на принтере;
- в отчет включаются следующие разделы: титульный лист, цель и задачи работы, выполненные задания, выводы;
- отчет обязательно скрепляется.

1 Лабораторная работа № 1. Анализ текущего состояния предметной области

Цель работы

Выявить и описать высокоуровневые требования к информационной системе в соответствии с вариантом задания.

Порядок выполнения работы

1 Составить документ «Профиль клиента», в котором кратко описать информацию о компании (1–2 страницы).

2 Разработать видение информационной системы (ИС) (видение выполнения проекта и границы проекта – документ, который кратко описывает, в каких подразделениях и в какой функциональности будет внедряться ИС), в том числе:

- проанализировать проблемную ситуацию, определить позицию разрабатываемой информационной системы (АИС);

- сформулировать краткое описание АИС, определить его возможности;

- выработать и описать прочие требования к АИС.

Шаблон документа «Видение» содержит следующие основные разделы:

- введение;
- позиционирование;
- описания совладельцев и пользователей;
- краткий обзор изделия;
- возможности продукта;
- ограничения;
- показатели качества;
- старшинство и приоритеты;
- другие требования к изделию;
- требования к документации;
- приложение.

Во введении описываются цель документа, его контекст (связь и взаимовлияние с различными проектами), определения, сокращения, ссылки на другие документы, краткое содержание.

В разделе «Позиционирование» помещается определение решаемой проблемы (проблем), указывается целевой заказчик и исследуются деловые преимущества изделия перед аналогичными на рынке.

В описании совладельцев и пользователей, помимо собственно описания этих двух групп, исследуется демография рынка: целевые рыночные сегменты, размер и темпы роста рынка, существующие конкурентные предложения на рынке, репутация разработчика на рынке.

Краткий обзор изделий содержит резюме изделия, описание его перс-

пектив и ключевых возможностей, предположения и зависимости, указывается стоимость и ее калькуляция, рассматриваются вопросы лицензирования и инсталляции.

В разделе, посвященном возможностям продукта, они описываются более подробно, каждая – в отдельном параграфе.

В раздел «Ограничения» следует выносить существующие технические, технологические и другие обстоятельства, которые необходимо учитывать на данной стадии.

Раздел «Показатели качества» содержит описание наиболее существенных нефункциональных требований к системе (эффективности, надежности, отказоустойчивости и др.).

Раздел «Старшинство и приоритеты» ранжирует сформулированные ранее требования и возможности системы по степени важности, очередности реализации и т. п.

Раздел «Другие требования к изделию» описывает применяемые стандарты, системные требования, эксплуатационные требования, требования к окружающей среде.

В требованиях к документации приводятся ключевые характеристики руководства пользователя, интерактивной справки, руководства по установке и конфигурированию, файла Read Me.

В приложение выносятся атрибуты возможностей. RUP рекомендует следующий набор атрибутов: статус, выгода, объем работ, риск, стабильность, целевой выпуск, назначение, причина.

3 Оформить работу.

4 Осуществить защиту работы.

Теоретический материал

Источники требований.

Основным источником требований к информационной системе, безусловно, являются соображения, высказанные представителями Заказчика. В соответствии с иерархической моделью требований данная информация структурируется как минимум на два уровня: бизнес-требования и требования пользователей.

Проблема состоит в том, что требования формулируются к создаваемой, еще не существующей системе, т. е., по сути, решается начальная подзадача задачи проектирования АИС, а представители Заказчика далеко не всегда бывают компетентны в данном вопросе. Поэтому, наряду с требованиями, высказанными Заказчиком, целесообразно собирать и требования от других совладельцев системы: сотрудников аналитической группы исполнителя, внешних экспертов и т. д.

Результирующий, часто достаточно сырой материал рассматривается как документ «Требования совладельцев». На требования совладельцев обычно не накладывается никаких специальных ограничений.

Другим важным источником информации, помимо выявления требований,

являются артефакты, описывающие предметную область. Это могут быть документы с описанием бизнес-процессов предприятия, выполненные консалтинговым агентством, либо просто документы (должностные инструкции, распоряжения, своды бизнес-правил), принятые на предприятии. Одной из немногих методологий, в которой специально выделяется рабочий поток делового моделирования, является Rational Unified Process.

Еще одна альтернатива, используемая при выявлении требований, – так называемые «лучшие практики», широко применяемые в настоящее время в бизнес-консалтинге и при внедрении корпоративных информационных систем. Лучшие практики представляют собой описания моделей деятельности успешных компаний отрасли, используемые длительное время в сотнях и тысячах компаний по всему миру.

Таким образом, основными источниками, образующими «вход» процесса выявления требований, являются требования, высказанные совладельцами, как таковые или (и) артефакты, описывающие объект исследования. Однако это достаточно упрощенный взгляд: чтобы данные поступили «на вход», аналитики требований должны проделать немалую работу, связанную с подбором респондентов и информационных материалов, организацией интервью и т. д.

Стратегии выявления требований.

Интервью. Ключевой стратегией выявления требований было и остается интервью с экспертами.

В ставшей уже классической, но ничуть не утратившей актуальность монографии Д. Марко в процессе проведения интервью предлагается выделить три подчиненных процесса: подготовку, проведение интервью (опроса) и завершение.

Подготовка позволяет спланировать процесс опроса и выработать стратегию управления этим процессом. Важность подготовительного этапа возрастает, если респондент является «дефицитным» полезным ресурсом, например, президентом крупной компании.

В проведении опроса самое важное – правильно организовать и поддерживать поток информации от эксперта к вам. Рекомендуется потратить время на обдумывание верного начала опроса, при сборе информации по возможности использовать записи, заканчивать разговор плавно.

Следите за возникновением следующих ситуаций:

- Вы уже получили достаточно информации;
- Вы получаете большой объем неподходящей информации;
- обилие информации Вас подавляет;
- эксперт начинает уставать;
- у Вас с экспертом часто возникают конфликты.

Любая из этих причин – достаточное основание для завершения беседы.

Анкетирование. Анкетирование – самый малозатратный для аналитика способ извлечения информации, он же и наименее эффективный. Обычно применяется как дополнение к другим стратегиям выявления требований.

Недостатки анкетирования очевидны: респонденты часто бывают неспособны либо слабо мотивированы в том, чтобы хорошо и информативно заполнить анкету. Велика вероятность получить неполную или вовсе ложную информацию. Преимущество в том, что подготовка и анализ анкет требуют небольшого ресурса.

Рекомендуется формулировать в анкетах вопросы с замкнутым циклом ответов в одной из следующих трех форм.

Многоальтернативные вопросы. Эта форма анкеты известна всем, кто когда-либо проходил тестирование; может расширяться комментариями респондента в свободной форме.

Рейтинговые вопросы. Представляют predetermined набор ответов на сформулированные вопросы. Используются такие значения, как «абсолютно согласен», «согласен», «отношусь нейтрально», «не согласен», «абсолютно не согласен», «не знаю».

Вопросы с ранжированием. Эта форма предусматривает ранжирование (упорядочивание) ответов путем присваивания им порядковых номеров, процентных значений и т. п.

Наблюдение. Наблюдение за работой моделируемой организационной системы – полезная стратегия получения информации (хотя, строго говоря, по результатам наблюдения можно получить модель ОС, а не модель АТ).

Различают пассивное и активное наблюдение. При активном наблюдении аналитик работает как участник команды, что позволяет улучшить понимание процессов.

Через наблюдение, а возможно и участие, аналитики получают информацию о происходящих день за днем операциях из первых рук. Во время наблюдения за работой системы часто возникают вопросы, которые никогда бы не появились, если бы аналитик только читал документы или разговаривал с экспертами.

Недостатком этой стратегии является то, что наблюдатель, как и всякий «измерительный прибор», вносит помехи в результаты измерений: сотрудники организации, находясь «под колпаком», могут начать вести себя принципиально по-другому, чем обычно.

Самостоятельное описание требований. Документы – хороший источник информации, потому что они чаще всего доступны и их можно «опрашивать» в удобном для себя темпе. Чтение документов – прекрасный способ получить первоначальное представление о системе и сформулировать вопросы к экспертам.

Если опытный аналитик уже исследовал большое число систем такого же типа, что и на предприятии внедрения, он обладает фундаментальными знаниями в соответствующей предметной области относительно определенного класса систем. Авторы методологии SADT рекомендуют проводить самоопрос с тем, чтобы получить максимальную пользу от своих знаний.

По результатам анализа документов и собственных знаний аналитик может составить описание требований и предложить его представителям Заказчика в качестве информации к размышлению либо основы для формирования

технического задания.

Недостаток этой стратегии – опасность пропуска знаний, специфичных для объекта исследования (в случае самоопроса), либо неформализованных знаний, эмпирических правил и процедур, широко используемых на практике, но не вошедших в документы.

Совместные семинары. Совместные семинары, сохраняя все преимущества режима интервью, приносят дополнительные бонусы: работа в группе более продуктивна, группы быстрее обучаются, более склонны к квалифицированным заключениям, позволяют исключить многие ошибки.

Эта стратегия, очевидно, одна из самых затратных, однако она окупается за счет меньшего количества ошибок и отказе от формализации в пользу живого общения, выработке общего языка и пр. Некоторые методологии (например, ХР) зиждутся на постоянном тесном контакте между Заказчиком и Исполнителем, и, если такой возможности нет, ХР-проект просто не сможет состояться.

«Разъясняющие встречи», или «запланированный мозговой штурм», – термин, пришедший из общей практики менеджмента и базирующийся на идеях сотрудничества заинтересованных лиц для совместного анализа путей решения проблем, определения и предупреждения рисков и т. п.

Как и проведение интервью, организация семинара требует соблюдения правил.

Прототипирование. Прототипирование – ключевая стратегия выявления требований в большинстве современных методологий. Программный прототип – «зеркало», в котором видно отражение того, как понял Исполнитель требования Заказчика. Процесс выявления требований путем прототипирования тем более интенсивен, чем это «зеркало кривее». Документальный способ выявления требований всегда уступает живому общению. Анализ того, что сделано в виде интерфейсов пользователя, дает еще больший эффект. Подключается правополушарный канал восприятия, который, как известно, работает у большинства людей на порядок эффективнее, чем вербальный.

Контрольные вопросы

- 1 Назовите основные источники требований к разрабатываемому программному обеспечению.
- 2 Назовите виды требований. Охарактеризуйте каждый из них.
- 3 Назовите стратегии выявления требований. Опишите каждую из них.
- 4 Дайте определение понятию «прототипирование».

2 Лабораторная работа № 2. Разработка бизнес-требований с использованием инструмента Lean Canvas

Цель работы

Приобрести навыки анализа предметной области, изучить инструмент бизнес-моделирования Lean Canvas, построить модель Lean Canvas для предметной области.

Порядок выполнения работы

- 1 Создать модель с использованием электронных шаблонов.
- 2 Ключевые показатели сформулировать согласно критериям SMART.
- 3 Оформить работу.
- 4 Осуществить защиту работы.

Теоретический материал

Перед выполнением практического задания ознакомиться с следующими материалами.

- 1 Что такое Lean Canvas? (<https://habr.com/ru/company/productstar/blog/508994/>).
- 2 Lean Canvas: инструкция по применению (<https://scrumtrek.ru/blog/product-management/9113/lean-canvas-что-это/>).
- 3 Цели SMART: 10 примеров + инструкция от ТОП-менеджера (<https://in-scale.ru/blog/celi-smart/>).

Контрольные вопросы

- 1 В каких случаях использование инструмента бизнес-моделирования Lean Canvas наиболее актуально?
- 2 Опишите порядок заполнения модели Lean Canvas.
- 3 Опишите назначение каждого блока модели Lean Canvas.
- 4 Каковы достоинства и недостатки системы постановки целей SMART?
- 5 Опишите критерии SMART-целей.
- 6 Приведите примеры правильных и неправильных формулировок SMART-целей.

3 Лабораторная работа № 3. Применение техники Impact Mapping стратегического планирования

Цель работы

Приобрести навыки анализа предметной области, изучить инструмент бизнес-моделирования Impact Mapping, построить ментальную карту с использованием техники Impact Mapping для предметной области.

Порядок выполнения работы

- 1 Построить ментальную карту (mind map) с использованием электронных шаблонов. Для построения ментальной карты с использованием техники Impact Mapping рекомендуется использовать шаблон miro.com. Ключевые показатели сформулировать согласно критериям SMART.
- 2 Оформить отчет.
- 3 Защитить лабораторную работу.

Теоретический материал

Перед выполнением практического задания ознакомиться с следующими материалами:

- 1 Impact Mapping на практике (<https://habr.com/ru/post/246401/>).
- 2 Impact Mapping – инструкция по применению (<https://scrumtrek.ru/blog/product-management/3326/impact-mapping-guide/>).

Контрольные вопросы

- 1 Что такое Impact Mapping?
- 2 В каких случаях использование техники бизнес-моделирования Impact Mapping наиболее актуально?
- 3 Опишите этапы построения карты влияния с использованием техники Impact Mapping.
- 4 Каковы достоинства и недостатки системы постановки целей SMART?
- 5 Опишите критерии SMART-целей.
- 6 Приведите примеры правильных и неправильных формулировок SMART-целей.

4 Лабораторная работа № 4. Описание архетипов пользователей

Цель работы

Приобрести навыки описания архетипов пользователей, применить методики Personas для описания архетипов пользователей.

Порядок выполнения работы

- 1 Описать архетипы потребителей, выявленных в лабораторных работах № 1 и 2.
- 2 Оформить отчет.
- 3 Защитить лабораторную работу.

Теоретический материал

Перед выполнением практического задания ознакомиться с следующими материалами.

- 1 Как использовать метод архетипов в брендинге. (<https://yulialos.com/archetypes/kak-ispolzovat/>).
- 2 Об использовании персон (персонажей) пользователей при разработке продуктов (<https://habr.com/ru/post/349740/>).
- 3 Пользовательские персоны: Курс «Создание программного продукта и управление его развитием» (<https://habr.com/ru/company/acronis/blog/514756/>).

Контрольные вопросы

- 1 Кто такие User Personas?
- 2 Откуда берутся персоны?
- 3 Кто и зачем использует метод Personas?

5 Лабораторная работа № 5. Построение карты пользовательских историй

Цель работы

Приобрести навыки использования технологии User Story Mapping при проектировании программного обеспечения.

Порядок выполнения работы

- 1 Разработать карту пользовательских историй.

- 2 Оформить отчет.
- 3 Защитить лабораторную работу.

Теоретический материал

Перед выполнением практического задания ознакомиться с следующими материалами:

- 1 User Story Mapping – инструкция по применению (<https://scrumtrek.ru/blog/product-management/3498/user-story-mapping-guide/>).
- 2 User Story Mapping: как построить карту пользовательских историй (<https://blog.calltouch.ru/user-story-mapping-kak-postroit-kartu-polzovatelskih-istorij/>).

Контрольные вопросы

- 1 Что такое User Story Mapping и для чего используется?
- 2 Каковы преимущества USM?
- 3 Из чего состоит User Story Mapping?
- 4 Опишите последовательность построения User Story Mapping.

6 Лабораторная работа № 6. Написание пользовательских историй (User Story Writing)

Цель работы

Приобрести навыки описания пользовательских историй.

Порядок выполнения работы

- 1 Описать не менее 10 пользовательских историй в соответствии с критериями INVEST.
- 2 Оформить отчет.
- 3 Защитить лабораторную работу.

Теоретический материал

Перед выполнением лабораторной работы ознакомиться с следующими материалами.

- 1 Что такое INVEST-критерии для задач (<https://zen.yandex.ru/media/id/5b86398716027100aaeb711f/что-такое-investkriterii-dlia-zadach-5d47cfeffc69ab00c41b5a2d>).
- 2 Martin, Robert. 2009. Clean Code: A Handbook of Agile Software Craftsmanship. Boston: MA: Pearson Education.

Контрольные вопросы

- 1 Что такое User Story?
- 2 Опишите правила составления User Story.
- 3 Сколько User Story может быть у одного актера?
- 4 Опишите критерии INVEST.

7 Лабораторная работа № 7. Написание критериев приемки (Acceptance Criteria)

Цель работы

Приобрести навыки разработки критериев приемки.

Порядок выполнения работы

- 1 Для каждой user story разработать критерии приемки. Для пяти user story критерии приемки должны быть разработаны в формате Gherkin.
- 2 Оформить отчет.
- 3 Защитить лабораторную работу.

Теоретический материал

Перед выполнением практического задания ознакомиться с следующими материалами.

- 1 Хороший Gherkin как путь к хорошей автоматизации (<https://www.software-testing.ru/library/testing/testing-tools/3245-writing-good-gherkin-enables-good-test-automation>).
- 2 Martin, Robert. 2009. Clean Code: A Handbook of Agile Software Craftsmanship. Boston: MA: Pearson Education.

Контрольные вопросы

- 1 Что такое acceptance criteria?
- 2 Для чего нужны acceptance criteria?
- 3 Что такое Gherkin?
- 4 Опишите базовый синтаксис Gherkin.
- 5 Опишите типичные ошибки «плохих» user story.

8 Лабораторная работа № 8. Построение диаграммы вариантов использования. Спецификация вариантов использования

Цель работы

Выявить и описать требования пользователей (ранг – специалист, потенциальный пользователь) к информационной системе в соответствии с вариантом задания, определить основных актёров и сформулировать варианты использования.

Порядок выполнения работы

1 Выявить актёров. Необходимо выявить максимально возможное количество актёров (это могут быть потенциальные пользователи системы, внешние системы). Осуществить классификацию актёров, разбить их по группам; сформулировать окончательный реестр актёров. Каждому актёру сопоставить краткое (в один абзац) описание.

2 Выявить варианты использования. Для каждого из актёров необходимо выявить максимально возможное количество вариантов использования.

3 Проанализировать взаимосвязи между вариантами использования. При анализе исключать дублирующиеся, выявлять отношения расширения, включения, родовидовые.

4 Составить диаграмму вариантов использования.

5 Составить результирующий реестр функциональных требований (один вариант использования – одно требование). Присвоить вариантам использования порядковые номера.

6 Конкретизировать описания вариантов использования. Каждому варианту использования сопоставить краткое описание (текст в 1–2 абзаца). В тексте отразить его наименование, основное действующее лицо, прочих актёров, связи с другими вариантами использования (если есть), краткое описание функционирования.

7 Оформить работу.

8 Осуществить защиту работы.

Теоретический материал

Диаграмма вариантов использования UML, Use case Diagram – одно из самых простых представлений системы. Ее базовые «строительные элементы» – актёры и варианты использования.

Исполнитель (действующее лицо, Actor) – личность, организация или система, взаимодействующая с ИС; различают внешнего исполнителя (который использует или используется системой, т. е. порождает прецеденты деятельности) и внутреннего исполнителя (который обеспечивает реализацию

прецедентов деятельности внутри системы). На диаграмме исполнитель представляется стилизованной фигуркой человека.

Прецедент (вариант использования) – законченная последовательность действий, инициированная внешним объектом (личностью или системой), которая взаимодействует с ИС и получает в результате некоторое сообщение от ИС. На диаграмме представляется овалом с надписью, отражающей содержание действия.

Диаграмма вариантов использования задумана так, чтобы дать наиболее общее представление о функциональности системы (ее компоненты), не вдаваясь в детали взаимосвязей функций.

Для включения в диаграмму выбранные прецеденты должны удовлетворять следующим критериям:

- прецедент должен описывать, ЧТО нужно делать, а не КАК;
- прецедент должен описывать действия с точки зрения ИСПОЛНИТЕЛЯ;
- прецедент должен возвращать исполнителю некоторое СООБЩЕНИЕ;
- последовательность действий внутри прецедента должна представлять собой одну НЕДЕЛИМУЮ цепочку.

Поэтому основной вид отношения, используемый в диаграмме, – ассоциация между актером и вариантом использования (рисунок 8.1).

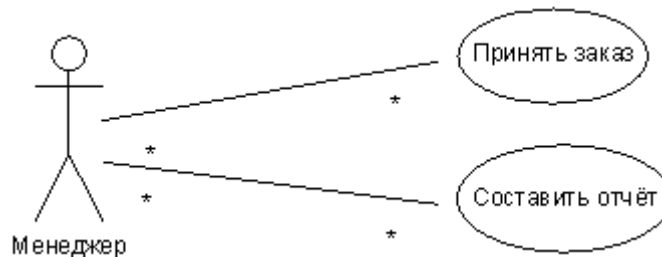


Рисунок 8.1 – Отношение ассоциации

Другие виды отношений – отношение включения (include), расширения (extend) и обобщения/генерализации.

Включение (рисунок 8.2) служит для обозначения подчиненных вариантов использования (когда один или более вариантов использования содержат вызовы одной и той же функциональности).



Рисунок 8.2 – Отношение включения

Расширение (рисунок 8.3) в точности соответствует точке расширения, используемой при описании варианта использования.

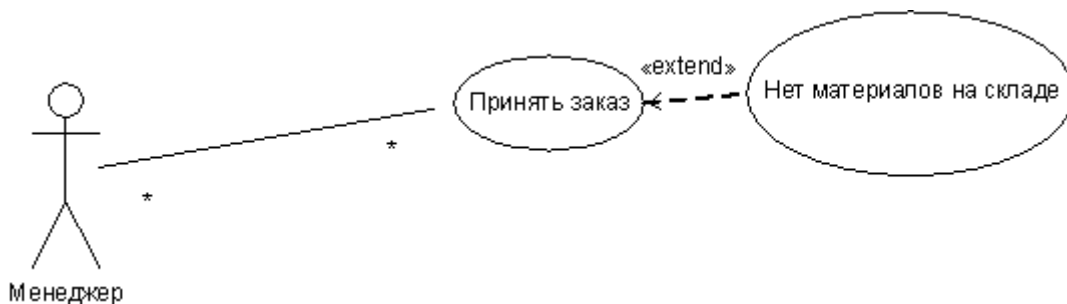


Рисунок 8.3 – Отношение расширения

Отношение обобщения (рисунок 8.4) может применяться как к актёрам, так и к вариантам использования с целью указания специализации одних относительно других.



Рисунок 8.4 – Отношение обобщения

Контрольные вопросы

- 1 Поясните назначение языка UML.
- 2 Перечислите основные принципы объектно-ориентированного анализа и проектирования, положенные в основу языка UML.
- 3 Назовите и охарактеризуйте виды отношений между объектами.
- 4 Назовите основные элементы диаграммы вариантов использования.
- 5 Что в диаграммах вариантов использования называется актером? Как графически изображается актер?
- 6 Что в диаграммах вариантов использования называется вариантом использования? Как графически изображается вариант использования?
- 7 Какие виды отношений между отдельными элементами определены на диаграммах вариантов использования?
- 8 Приведите графическое представление различных видов отношений на диаграммах вариантов использования.
- 9 Поясните применение отношения ассоциации на диаграммах вариантов использования.
- 10 Как графически обозначается мощность ассоциации на диаграммах вариантов использования?

11 Поясните применение отношения включения на диаграммах вариантов использования.

12 Поясните применение отношения расширения на диаграммах вариантов использования.

13 Поясните применение отношения обобщения на диаграммах вариантов использования.

9 Лабораторная работа № 9. Построение диаграммы классов

Цель работы

Выявить классы предметной области, разработать диаграммы классов документов и отчетов, разработать модель базы данных.

Порядок выполнения работы

1 На основе информации, полученной на этапах моделирования бизнес-процессов (лабораторные работы №1–3), выявить классы предметной области.

2 Разработать диаграмму классов документов и отчетов на основе альбома документов (лабораторная работа № 3).

3 Разработать диаграмму реляционной модели данных.

4 Привести таблицу соответствия полей модели данных и диаграммы классов документов и отчетов. Таблица показывает, что все данные, необходимые для формирования документов и отчетов, хранятся в базе данных.

5 Оформить работу.

6 Осуществить защиту работы.

Теоретический материал

Классы в UML изображаются на диаграммах классов, которые позволяют описать систему в статическом состоянии – определить типы объектов системы и различного рода статические связи между ними.

Классы отображают типы объектов системы.

Между классами возможны отношения:

– зависимости, которые описывают существующие между классами отношения использования;

– обобщения, связывающие обобщенные классы со специализированными;

– ассоциации, отражающие структурные отношения между объектами классов.

Зависимостью называется отношение использования, согласно которому изменение в спецификации одного элемента (например, класса «товар») может повлиять на использующий его элемент (класс «строка заказа»). Часто зависимости показывают, что один класс использует другой в качестве аргумента.

Обобщение – это отношение между общей сущностью (родителем – класс «клиент») и ее конкретным воплощением (потомком – классы «корпоративный клиент» или «частный клиент»). Объекты класса-потомка могут использоваться всюду, где встречаются объекты класса-родителя, но не наоборот. При этом он наследует свойства родителя (его атрибуты и операции). Операция потомка с той же сигнатурой, что и у родителя, замещает операцию родителя; это свойство называют полиморфизмом. Класс, у которого нет родителей, но есть потомки, называется корневым; класс, у которого нет потомков, называется листовым.

Ассоциация – это отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа («клиент» может сделать «заказ»). Если между двумя классами определена ассоциация, то можно перемещаться от объектов одного класса к объектам другого. При необходимости направление навигации может задаваться стрелкой. Допускается задание ассоциаций на одном классе. В этом случае оба конца ассоциации относятся к одному и тому же классу. Это означает, что с объектом некоторого класса можно связать другие объекты из того же класса. Ассоциации может быть присвоено имя, описывающее семантику отношений. Каждая ассоциация имеет две роли, которые могут быть отражены на диаграмме. Роль ассоциации обладает свойством множественности, которое показывает, сколько соответствующих объектов может участвовать в данной связи.

Если приходится моделировать отношение типа «часть – целое», то используется специальный тип ассоциации – агрегирование. В такой ассоциации один из классов имеет более высокий ранг (целое – класс «заказ») и состоит из нескольких меньших по рангу классов (частей – класс «строка заказа»).

В UML используется и более сильная разновидность агрегации – композиция, в которой объект-часть может принадлежать только единственному целому. В композиции жизненный цикл частей и целого совпадает, любое удаление целого обязательно захватывает и его части.

Для ассоциаций можно задавать атрибуты и операции, создавая по обычным правилам UML классы ассоциаций.

Контрольные вопросы

- 1 Что такое логическое представление системы?
- 2 Что такое диаграмма классов?
- 3 Какие нотации используются в диаграмме классов?
- 4 Что такое класс?
- 5 Какие классы используются в диаграмме классов?
- 6 Какие связи возможны между классами?
- 7 Что такое n-арная ассоциация?
- 8 В чем разница между агрегацией и композицией?
- 9 Что такое класс ассоциации?

10 Лабораторная работа № 10. Построение диаграмм взаимодействия информационной системы

Цель работы

Описать логику процедур, бизнес-процессы и потоки работ.

Порядок выполнения работы

- 1 Для двух основных вариантов использования разработать диаграммы взаимодействия.
- 2 Оформить работу. В работе должны быть представлены диаграммы взаимодействия и их описание.
- 3 Осуществить защиту работы.

Теоретический материал

В контексте языка UML деятельность (activity) представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения.

На диаграмме деятельности применяют один основной тип сущностей – действие и один тип отношений – переходы (передачи управления и данных). Также используются такие конструкции, как развилки, слияния, соединения, ветвления, которые похожи на сущности, но таковыми на самом деле не являются, а представляют собой графический способ изображения некоторых частных случаев многоместных отношений.

Состояние действия (action state) (рисунок 10.1) 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.

Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования. Никаких дополнительных или неявных ограничений при записи действий не накладывается. Рекомендуется в качестве имени простого действия использовать глагол с пояснительными словами. Если же действие может быть представлено в некотором формальном виде, то целесообразно записать его на том языке программирования, на

котором предполагается реализовывать конкретный проект.

Иногда возникает необходимость представить на диаграмме деятельности некоторое сложное действие, которое, в свою очередь, состоит из нескольких более простых действий. В этом случае можно использовать специальное обозначение так называемого состояния поддеятельности (subactivity state) (рисунок 10.2). Такое состояние является графом деятельности и обозначается специальной пиктограммой в правом нижнем углу символа состояния действия. Данная конструкция может применяться к любому элементу языка UML, который поддерживает «вложенность» своей структуры. При этом пиктограмма может быть дополнительно помечена типом вложенной структуры.

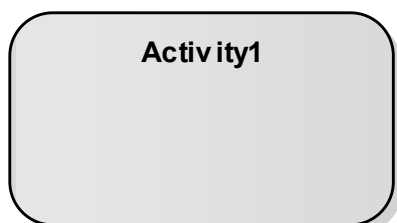


Рисунок 10.1 – Состояние действия



Рисунок 10.2 – Сложное действие

Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния. Они имеют такие же обозначения, как и на диаграмме состояний (рисунок 10.3). При этом каждая деятельность начинается в начальном состоянии и заканчивается в конечном состоянии. Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз. В этом случае начальное состояние будет изображаться в верхней части диаграммы, а конечное – в ее нижней части.



Рисунок 10.3 – Начальное и конечное состояния

Деятельность – это протяженное по времени составное действие. Составное! То есть составленное из более простых действий. Вот эти самые простые (атомарные) действия, а вернее, последовательность их выполнения, частенько изображают внутри деятельности в виде маленькой диаграммы активностей (рисунок 10.4). Это слегка напоминает матрешку – одна (а часто и не одна) диаграмма внутри другой.

Переходы. При построении диаграммы деятельности используются только нетриггерные 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 (рисунок 10.4).

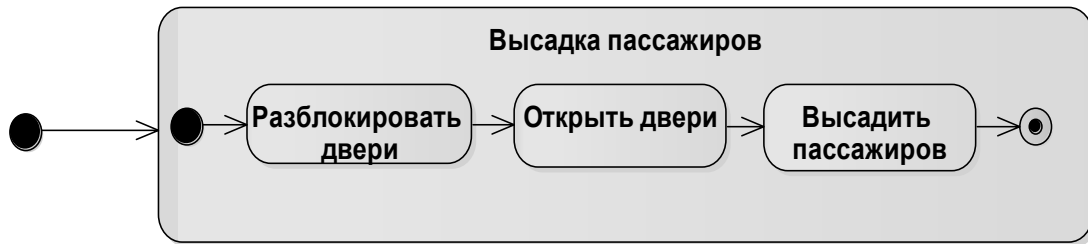


Рисунок 10.4 – Вложенные состояния

Ветвление (развилка). Если из состояния действия выходит единственный переход, то он может быть никак не помечен. Если же таких переходов несколько, то выполняться может только один из них. В этом случае для каждого из переходов должно быть явно записано сторожевое условие в прямых скобках. Условие же истинности должно выполняться 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 (рисунок 10.5). 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.

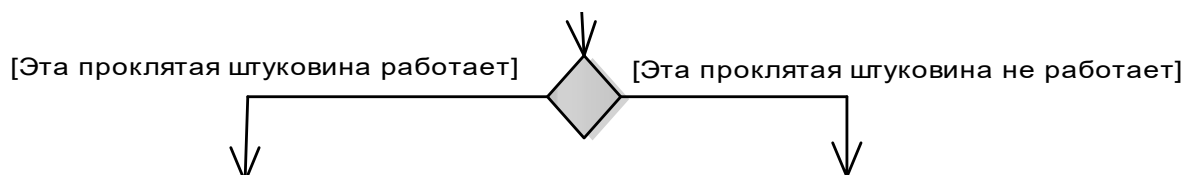


Рисунок 10.5 – Ветвление

Параллельные процессы (переход «разделение – слияние»). Один из недостатков обычных блок-схем алгоритмов связан с проблемой изображения параллельных ветвей отдельных вычислений. Поскольку распараллеливание вычислений существенно повышает общее быстродействие программных систем, необходимы графические примитивы для представления параллельных процессов. В языке UML для этой цели используется специальный символ для разделения и слияния параллельных вычислений или потоков управления. Таким символом является прямая черточка (рисунок 10.6). 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, а слияние (concurrent join) – несколько входящих переходов и один выходящий.

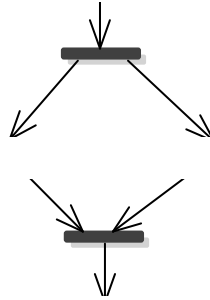


Рисунок 10.6 – Параллельные процессы

Дорожки. Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделированием бизнес-процессов. Действительно, деятельность любой компании (фирмы) также представляет собой не что иное, как совокупность отдельных действий, направленных на достижение требуемого результата. Однако применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение 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 виду 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.

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.

Объекты. В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый их результат. Действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Поскольку в таком ракурсе объекты играют определенную

роль в понимании процесса деятельности, иногда возникает необходимость явно указать их на диаграмме.

Для графического представления объектов используется 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.

Контрольные вопросы

- 1 Дайте определение понятия «диаграмма деятельности».
- 2 Опишите назначение диаграммы деятельности.
- 3 Дайте определение понятий «состояние деятельности» и «состояние действия». Графическое изображение состояния.
- 4 Приведите пример ветвления и параллельных потоков управления процессами на диаграмме деятельности.
- 5 Какие переходы используются на диаграмме деятельности?
- 6 Что представляет собой дорожка на диаграмме деятельности?

11 Лабораторная работа № 11. Построение диаграмм последовательности информационной системы

Цель работы

Изобразить участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются.

Порядок выполнения работы

- 1 Для двух основных вариантов использования (для которых не разрабатывалась диаграмма взаимодействия в лабораторной работе № 8) разработать диаграммы последовательности. Каждая диаграмма последовательности должна показывать как основной поток событий, так и альтернативный (то есть содержать фреймы взаимодействия alt либо opt, остальные – по мере

надобности).

2 Оформить работу. В работе должны быть представлены диаграммы последовательности и их описание.

3 Осуществить защиту работы.

Теоретический материал

Диаграммы последовательности показывают взаимодействие, представляя каждого участника вместе с его линией жизни (lifeline), которая идет вертикально вниз и упорядочивает сообщения на странице; сообщения также следует читать сверху вниз.

Каждая линия жизни имеет полосу активности, которая показывает интервал активности участника при взаимодействии. Она соответствует времени нахождения в стеке одного из методов участника.

Для обозначения циклов и условий используются фреймы взаимодействий (interaction frames), представляющие собой средство разметки диаграммы взаимодействия.

В основном фреймы состоят из некоторой области диаграммы последовательности, разделенной на несколько фрагментов.

Для отображения цикла применяется оператор loop с единственным фрагментом. Для условной логики можно использовать оператор alt и помещать условие в каждый фрагмент. Будет выполнен только тот фрагмент, защита которого имеет истинное значение. Для единственной области существует оператор opt.

Общепринятые операторы для фреймов взаимодействия приведены в таблице 11.1.

Таблица 11.1 – Общепринятые операторы для фреймов взаимодействия

Оператор	Значение
alt	Несколько альтернативных фрагментов (alternative); выполняется только тот фрагмент, условие которого истинно
opt	Необязательный (optional) фрагмент; выполняется, только если условие истинно. Эквивалентно alt с одной веткой
par	Параллельный (parallel); все фрагменты выполняются параллельно
loop	Цикл (loop); фрагмент может выполняться несколько раз, а защита обозначает тело итерации
region	Критическая область (critical region); фрагмент может иметь только один поток, выполняющийся за один прием
neg	Отрицательный (negative) фрагмент; обозначает неверное взаимодействие
ref	Ссылка (reference); ссылается на взаимодействие, определенное на другой диаграмме. Фрейм рисуется, чтобы охватить линии жизни, вовлеченные во взаимодействие. Можно определять параметры и возвращать значение
sd	Диаграмма последовательности (sequence diagram); используется для очерчивания всей диаграммы последовательности, если это необходимо

Контрольные вопросы

- 1 Может ли диаграмма последовательностей содержать объект с линией жизни, но без фокуса управления?
- 2 Чем отличаются представления кооперации на уровне спецификации и на уровне примеров?
- 3 В чем разница между активными и пассивными объектами?
- 4 Чем асинхронное сообщение отличается от синхронного?
- 5 Что такое мультиобъект?
- 6 Что такое композитный объект и как он связан с понятием кооперации?
- 7 Как можно избежать усложнения диаграммы взаимодействия с разветвленным потоком управления?

12 Лабораторная работа № 12. Построение диаграммы компонентов

Цель работы

Показать разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

Порядок выполнения работы

- 1 Разработать диаграмму компонентов ИС.
- 2 Оформить работу.
- 3 Осуществить защиту работы.

Теоретический материал

Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

Для представления физических сущностей в языке UML применяется специальный термин – компонент (component). Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели.

В языке UML выделяют три вида компонентов:

1) компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций. Такими компонентами могут быть динамически подключаемые библиотеки с расширением dll, веб-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp;

2) компоненты – рабочие продукты. Как правило, это файлы с исходными текстами программ, например, с расширениями h или src для языка C++;

3) компоненты исполнения, представляющие исполняемые модули – файлы с расширением exe.

Поскольку компонент как элемент физической реализации модели представляет отдельный модуль кода, иногда его дополнительно специфицируют. Один из способов спецификации различных видов компонентов – явное указание стереотипа компонента перед его именем.

В языке UML для компонентов определены следующие стереотипы.

Библиотека (library) – определяет первую разновидность компонента, который представляется в форме динамической или статической библиотеки.

Таблица (table) – определяет первую разновидность компонента, который представляется в форме таблицы базы данных.

Файл (file) – определяет вторую разновидность компонента, который представляется в виде файлов с исходными текстами программ.

Документ (document) – определяет вторую разновидность компонента, который представляется в форме документа.

Исполняемый (executable) – определяет третий вид компонента, который может исполняться в узле.

Строго говоря, эти дополнительные обозначения не специфицированы в языке UML. Однако их применение упрощает понимание диаграммы компонентов, существенно повышая наглядность физического представления.

Интерфейсы. Интерфейс (interface) служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры.

Их основное назначение – совмещать функциональность нескольких компонентов при решении специальных задач.

По отношению к компонентам интерфейс может быть обеспеченным (реализованным, предоставляемым) и требуемым (запрашиваемым).

Если компонент реализует интерфейс, то для данного компонента это обеспеченный интерфейс.

Если компонент вызывает операции интерфейса, то для данного компонента это требуемый интерфейс.

Обеспеченный интерфейс изображается в виде маленького круга, рядом с которым записывается его имя. В качестве имени может быть существительное, которое характеризует соответствующую информацию или сервис (например, «датчик», «сирена», «видеокамера»), но чаще строка текста (например, «запрос к базе данных», «форма ввода», «устройство подачи звукового сигнала»). Если имя записывается на английском языке, то оно должно начинаться с заглавной буквы I, например, ISecureInformation, ISensor.

Другое обозначение интерфейса – в виде прямоугольника класса со стереотипом «интерфейс» и возможными секциями атрибутов и операций. Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

Если компонент реализует интерфейс, то для данного компонента это обеспеченный интерфейс и данный факт показывается с помощью отношения реализации.

Если компонент вызывает операции интерфейса, то для данного компонента это требуемый интерфейс и данный факт показывается с помощью отношения зависимости.

Порты. Передача информации, вызов операций и взаимодействие между компонентами может осуществляться при помощи специальных элементов – портов.

Порт предполагает дополнительную точку взаимодействия между компонентом и окружающей средой. При этом взаимодействие описывается в форме специального протокола, т. е. доступ к функциональности через порт не является общедоступным, а может содержать целый ряд дополнительных требований. Причем удовлетворение соответствующих требований, как раз фиксируемых в форме протокола, служит обязательным условием получения или предоставления соответствующей функциональности.

Наличие имени у порта является необязательным. В этом случае он ассоциируется с интерфейсом, с которым связан.

Собирающий соединитель. Собирающий соединитель – специальное отношение, которое связывает компонент с окружающими его портами в контексте предоставляемых и требуемых сервисов.

Необходимость данного элемента определяется тем, что функциональность системы собирается из функциональности составляющих ее компонентов.

Отношение зависимости. Отношения между компонентами могут носить различный характер. Мы рассмотрели отношения между компонентами в контексте окружающих интерфейсов. Другой способ отображения отношений между компонентами – это зависимость.

Зависимость 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 (при этом факт наличия зависимости соответствует факту наличия интерфейсов в контексте предоставляемых либо требуемых).

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. В другом случае зависимость может отражать наличие в независимом компоненте описаний классов, которые используются в зависимом компоненте для создания соответствующих объектов. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

Отношение реализации. На диаграмме компонентов могут быть представлены отношения зависимости между компонентами и реализованными в них классами. Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы.

Разумеется, изменения в структуре описаний классов могут привести к изменению компонента.

Контрольные вопросы

- 1 Дайте определение понятию «диаграмма развертывания».
- 2 Опишите назначение диаграммы развертывания.
- 3 Перечислите существующие отношения между компонентами диаграммы.

13 Лабораторная работа № 13. Построение диаграммы развертывания

Цель работы

Представить физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения.

Порядок выполнения работы

- 1 Разработать диаграмму развертывания.
- 2 Оформить работу.
- 3 Осуществить защиту работы.

Теоретический материал

Диаграмма развертывания представляет физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения.

Главными элементами диаграммы являются узлы, связанные информационными путями. Узел (node) – это то, что может содержать программное обеспечение. Узлы бывают двух типов. Устройство (device) – это физическое оборудование: компьютер или устройство, связанное с системой. Среда выполнения (execution environment) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процесс-контейнер.

Изображения узлов могут расширяться, чтобы включить некоторую дополнительную информацию о спецификации узла. Если необходимо явно указать компоненты, которые размещаются на отдельном узле, то это можно сделать двумя способами.

1 Артефакты (artifacts). Артефакты являются физическим олицетворением программного обеспечения; обычно это файлы.

Таковыми файлами могут быть исполняемые файлы (такие как файлы .exe,

двоичные файлы, файлы DLL, файлы JAR, сборки или сценарии) или файлы данных, конфигурационные файлы, HTML-документы и т. д.

Перечень артефактов внутри узла указывает на то, что на данном узле артефакт разворачивается в запускаемую систему.

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

Для спецификации различных видов компонентов допускается явное указание стереотипа компонента перед его именем. В языке UML для компонентов определены следующие стереотипы.

1 Библиотека (library) – определяет разновидность компонента, который представляется в форме динамической или статической библиотеки.

2 Таблица (table) – определяет разновидность компонента, который представляется в форме таблицы базы данных.

3 Файл (file) – определяет разновидность компонента, который представляется в виде файлов с исходными текстами программ.

4 Документ (document) – определяет разновидность компонента, который представляется в форме документа.

5 Исполнимый (executable) – определяет вид компонента, который может исполняться в узле.

В качестве дополнения к имени узла могут использоваться различные стереотипы, которые явно специфицируют назначение этого узла. Хотя в языке UML стереотипы для узлов не определены, в литературе встречаются следующие их варианты: «процессор», «датчик», «модем», «сеть», «консоль» и др., которые самостоятельно могут быть определены разработчиком. Более того, на диаграммах развертывания допускаются специальные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции.

Информационные пути между узлами представляют обмен информацией в системе. Можно сопровождать эти пути информацией об используемых информационных протоколах.

Среда выполнения (execution environment) обычно является частью общего узла или устройства, на которой эта среда выполнения находится. Среды выполнения могут быть вложенными (например, среда исполнения «база данных» может быть вложена в среду выполнения «операционная система»).

Контрольные вопросы

- 1 Дайте определение понятию «узел». Графическое изображение узла.
- 2 Назовите основные стереотипы узлов.
- 3 Перечислите виды связей между узлами.

14 Лабораторная работа № 14. Построение диаграммы состояний информационной системы

Цель работы


Показать поведение ИС в течение ее жизни, начиная от открытия главного окна и заканчивая его закрытием и выходом из системы.

Порядок выполнения работы

- 1 Разработать диаграмму состояний ИС.
- 2 Оформить работу.
- 3 Осуществить защиту работы.

Теоретический материал

Диаграмма состояний (state machine diagrams) - это известная технология описания поведения системы. В том или ином виде диаграмма состояний существует с 1960 г., и на ранних этапах развития объектно-ориентированного программирования они применялись для представления поведения системы.

На диаграмме состояния графическим примитивом  показывают, в каком состоянии может находиться система. На диаграмме также представляются правила, согласно которым система переходит из одного состояния в другое. Эти правила представляются в виде переходов – линий, связывающих состояния. Переход (transition) означает перемещение из одного состояния в другое. Каждый переход имеет свою метку – событие, которое может вызвать изменение состояния.

На рисунке 14.1 диаграмма состояния начинается с состояния «Состояние1». На диаграмме это обозначено с помощью начального псевдосостояния (initial), которое не является состоянием, но имеет стрелку, указывающую на начальное состояние.

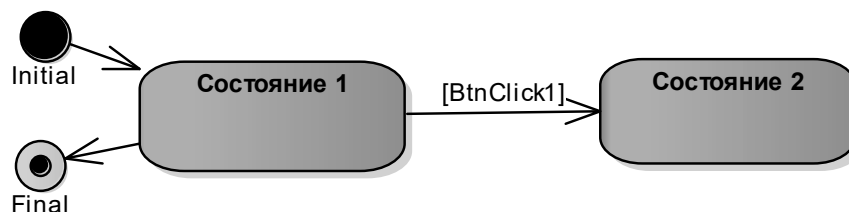


Рисунок 14.1 – Пример диаграммы состояний

Конечное состояние (final) означает, что система закончила работу.

Внутренние активности в диаграмме состояний.

Состояния могут реагировать на события без совершения перехода, используя внутренние активности (internal activities), и в этом случае событие и

активность размещаются внутри окна состояния.

На рисунке 14.2 представлено состояние с внутренними активностями системы. Внутренняя активность подобна самопереходу (self-transition) - переходу, который возвращает в то же состояние. Синтаксис внутренних активностей построен по схеме: событие/процедура.

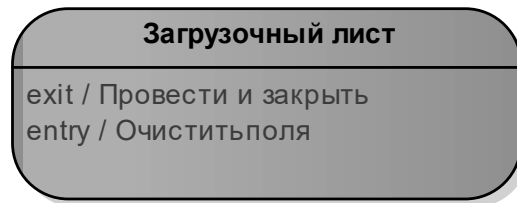


Рисунок 14.2 – Пример состояния с внутренними активностями

На рисунке также показаны специальные активности: входная и выходная. Входная активность выполняется каждый раз, когда система входит в состояние; выходная активность – каждый раз, когда система покидает состояние. Однако внутренние активности не инициируют входную и выходную активности; в этом состоит различие между внутренними активностями и самопереходами.

Состояния активности в диаграмме состояний

В состояниях на рисунках 14.1 и 14.2 система молчит и ожидает следующего события, прежде чем что-нибудь сделать. Однако возможны состояния, в которых объект проявляет некоторую активность.

Ведущаяся активность обозначается символом `do/`; отсюда термин `do-activity` (проявлять активность) (рисунок 14.3).

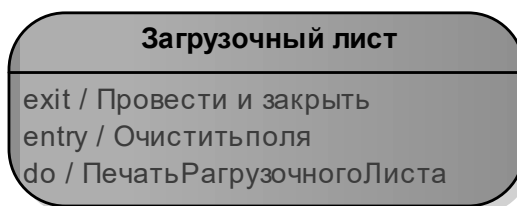


Рисунок 14.3 – Пример состояния активности

Контрольные вопросы

- 1 Дайте определение понятия «диаграмма состояний».
- 2 Опишите назначение диаграммы состояний.
- 3 Дайте определение понятия «конечный автомат».
- 4 Дайте определение понятиям «состояния», «действие», «псевдосостояние». Графическое изображение состояния.
- 5 Что представляет собой переход? Какие бывают переходы, в чем их различие?
- 6 Дайте определение понятия «событие».

7 Дайте определения следующих понятий: «составное состояние», «последовательные подсостояния», «параллельные подсостояния», «несовместимое подсостояние», «историческое состояние», «параллельный переход», «состояние синхронизации».

15 Лабораторная работа № 15. Разработка системной документации

Цель работы

Разработать документацию на программный продукт.

Порядок выполнения работы

- 1 Разработать руководство программиста.
- 2 Оформить работу.
- 3 Осуществить защиту работы.

Теоретический материал

Руководство программиста разрабатывается в соответствии с ГОСТ 19.504–79 *Руководство программиста. Требования к содержанию и оформлению.*

Контрольные вопросы

- 1 Назовите причины, по которым требуется писать руководство программиста.
- 2 Какие существуют стандарты по разработке документации на программный продукт?
- 3 Опишите структуру руководства программиста в соответствии с ГОСТ 19.504–79 *Руководство программиста. Требования к содержанию и оформлению.*

16 Лабораторная работа № 16. Разработка руководства пользователя

Цель работы

Разработать руководство пользователя информационной системой.

Порядок выполнения работы

- 1 Разработать руководство пользователя.
- 2 Оформить работу.
- 3 Осуществить защиту работы.

Теоретический материал

Руководство пользователя разрабатывается в соответствии с ГОСТ 19.505–79.

Контрольные вопросы

- 1 Назовите причины, по которым требуется писать руководство пользователя системы.
- 2 Какие существуют стандарты по разработке документации на программный продукт?
- 3 Опишите структуру руководства пользователя в соответствии с ГОСТ 19.505–79.

Список литературы

- 1 **Арлоу, Д.** UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу. – Санкт-Петербург: Символ-Плюс, 2016. – 624 с.
- 2 **Буч, Г.** Введение в UML от создателей языка / Г. Буч, Дж. Рамбо, И. Якобсон. – Москва: ДМК-Пресс, 2015. – 498 с.
- 3 **Леоненков, А.** Самоучитель UML 2 / А. Леоненков. – Санкт-Петербург : БХВ, 2007. – 576 с.
- 4 **Бедердинова, О. И.** Моделирование информационных систем на платформе SOFTWARE IDEAS MODELER : учебное пособие / О. И. Бедердинова, Л. В. Кремлева, С. В. Протасова. – Москва: ИНФРА-М, 2019. – 166 с.
- 5 **Гагарина, Л. Г.** Технология разработки программного обеспечения учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова; под ред. Л. Г. Гагариной. – Москва: ФОРУМ; ИНФРА-М, 2019. – 400 с.