МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания для практических занятий

По дисциплине: «Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем»

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность (профиль) Цифровые технологии химических производств

Невинномысск, 2024

УДК 681.3
ББК 39.973.202-018.2
М 54

Автор-составитель:
канд. техн. наук, доцент Э.Е. Тихонов

М -54 Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем: методические указания по освоению дисциплины направления 09.03.02 Информационные системы и технологии / сост. Э.Е. Тихонов - Невинномысск: НТИ, 2024. – 140 с.

Методические указания по освоению дисциплины «Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем» предназначены для студентов очной и заочной форм обучения направления 09.03.02 Информационные системы и технологи

Методические указания составлены на основании Федеральный государственного образовательного стандарта высшего образования по направлению подготовки 09.03.02 Информационные системы и технологии (уровень бакалавриата), утвержденного Министерством образования и науки РФ «12» марта 2015 г. № 219. С изменениями в соответствии с приказом Министерством образования и науки РФ « О внесении изменений в Федеральные Государственные образовательные стандарты высшего образования» «09» сентября 2015 г. № 999.

ББК 39.973.202-018.2

Тематический план лабораторных работ

№ Темы дисциплины	Наименование тем дисциплины, их краткое содержание	Объем часов	Из них в интерактивной форме
7 семестр			
Тема 1. Методологические основы проектирования			
1	Разработка технического задания	2	лабораторная работа
Тема 2. Каноническое проектирование			
2	Разработка эскизного проекта	2	лабораторная работа
Тема 3. Автоматизированное проектирование			
3	Оценка качественных показателей ПС	2	лабораторная работа
Тема 4. Типовое проектирование			
4	Тестирование программных систем	2	лабораторная работа
Тема 5. Прототипное проектирование			
5	Составление технологической документации	2	лабораторная работа
Тема 6. Управление проектом			
6	Составление пользовательской документации	2	лабораторная работа
7	Оформление документов сертификации	2	лабораторная работа
Тема 7. Проектирование элементов ИС			
8	Составление лицензионного соглашения	2	лабораторная работа
9	Создание диаграммы вариантов использования	2	лабораторная работа
10	Создание диаграмм взаимодействия	2	лабораторная работа
11	Создание диаграмм классов	2	лабораторная работа
12	Создание диаграмм классов (учет новых требований)	2	лабораторная работа
13	Создание диаграмм классов (добавление связей между классами)	2	лабораторная работа
14	Создание диаграммы состояний	2	лабораторная работа
15	Создание UML диаграмм в Microsoft Visio	2	лабораторная работа
16	Изучение методов построения графических моделей средствами Visual Studio	2	лабораторная работа
17	Создание диаграмм в Visual Studio	4	лабораторная работа
	Итого за 7 семестр	**36**	
	Итого	**36**	

* - с применением дистанционных образовательных технологий

Содержание

1. Организационно–методический раздел

Цели и задачи освоения дисциплины

Целью освоения учебной дисциплины является:

−подготовка выпускников к проектно-конструкторской деятельности в области создания и внедрения аппаратных и программных средств объектов профессиональной деятельности в соответствии с техническим заданием и с использованием средств автоматизации проектирования;

−формирование у студентов фундаментальных теоретических знаний по вопросам методики и практики проектирования сложных программных средств для информационных систем, а также обучение студентов современным программным средствам для проектирования программного обеспечения, основанным на использовании CASE-технологии.

Задачи:

−изучение методологии и инструментальных средств разработки программных систем;

−использование предметно-ориентированной среды разработки;

−знакомство с языком UML.

−изучение вопросов организации информационного обеспечения ИС. Освоение методологии проектирования баз и хранилищ данных: анализ предметной области, концептуальное, логическое и физическое проектирование.

−разработка интерфейса (приложений) с использованием с использованием Visual Studio.

−знакомство с автоматизированным проектированием ИС с использованием CASE-средств StarUML.

3 Перечень лабораторных работ

1. Лабораторная работа №1 Разработка технического задания
2. Лабораторная работа № 2 Разработка эскизного проекта
3. Лабораторная работа № 3 Оценка качественных показателей ПС
4. Лабораторная работа № 4 Тестирование программных систем
5. Лабораторная работа № 5 Составление технологической документации
6. Лабораторная работа № 6 Составление пользовательской документации
7. Лабораторная работа № 7 Оформление документов сертификации
8. Лабораторная работа № 8 Составление лицензионного соглашения

Дополнительные лабораторные работы

9. Лабораторная работа №1. «Исследование предметной области. Диаграммы прецедентов и диаграммы действий»
10. Лабораторная работа №2. «Разработка, моделирование и анализ структуры информационной системы. Диаграммы классов»
11. Лабораторная работа №3. «Динамика поведения информационной системы. Диаграммы взаимодействия»
12. Лабораторная работа №4. «Динамика поведения информационной системы. Диаграммы состояний. Физическая модель системы. Диаграммы реализации»
13. Лабораторная работа №5. «Генерация программного кода в StarUML»
14. Лабораторная работа №6. Создание диаграммы вариантов использования.
15. Лабораторная работа №7. Создание диаграмм взаимодействия.
16. Лабораторная работа №8. Создание диаграмм классов.
17. Лабораторная работа №9. Создание диаграмм классов (учет новых требований).
18. Лабораторная работа №10. Создание диаграмм классов (добавление связей между классами).
19. Лабораторная работа №11. Создание диаграммы состояний.
20. Лабораторная работа №12. Создание диаграммы компонентов.

1. Лабораторная работа №13 Создание UML диаграмм в Microsoft Visio
2. Лабораторная работа №14 Изучение методов построения графических моделей средствами Visual Studio
3. Лабораторная работа №15. Создание диаграмм в Visual Studio
4. Лабораторная работа №16 Демонстрационный пример. Моделирование требований пользователей
5. Лабораторная работа №17 Проектирование готовых проектов. Задания для самостоятельной работы

4. Самостоятельная работа студента

Самостоятельная работа студентов (СРС) является основным способом овладения учебным материалом во время, освобожденное от основных обязательных учебных занятий.

На самостоятельное изучение выносятся темы, включенные в учебные планы, но не рассматриваемые на теоретических и практических занятиях.

Вид занятия самостоятельной внеаудиторной деятельности студентов может быть теоретическим (работа с учебником, дополнительной литературой), практическим (работа в домашних условиях, в учебных кабинетах), и комбинированным (подготовка текста, беседы, доклада, реферата). Содержание СРС включает разнообразные формы деятельности: работа с литературой, оформление рефератов, буклетов, бесед.

Разнообразны и формы контроля самостоятельной внеаудиторной деятельности студентов. Так, учебный материал, предусмотренный рабочим планом для освоения студентами в процессе самостоятельной деятельности, выносится на итоговый контроль (дифференцированный зачет, экзамен) наряду с учебным материалом, который отрабатывается при проведении занятий; контроль качества самостоятельного изучения материала проводится во время консультаций, индивидуального собеседования преподавателя со студентом, при проверке рефератов, буклетов и т. д.

Самостоятельная работа больше всего способствует реализации требований квалификационной характеристики. Она выполняет не только учебные, но и воспитательные, и развивающие функции. Самостоятельная работа воспитывает обязательность, собранность, пунктуальность, систематичность в работе.

5. Учебно-методическое обеспечение дисциплины

Перечень основной учебной литературы, необходимой для освоения дисциплины

1. Ляхов, В. Ф. (СевКавГТУ). Прикладная информатика (в экономике) : проектирование информационных систем : учеб. пособие / В. Ф. Ляхов ; Мин-во образования и науки Рос. Федерации, ГОУ ВПО Сев. Кав. гос. техн. ун-т, Ч. 2. - Ставрополь : Изд-во СевКавГТУ, 2021. - 140 с. : ил. - Библиогр.: с. 139(11 назв.). - ISBN 5-9296-0342-1
2. Рочев, К. В. Информационные технологии. Анализ и проектирование информационных систем Электронный ресурс / Рочев К. В. : учебное пособие. - 2-е изд., испр. - Санкт-Петербург : Лань, 2019. - 128 с. - ISBN 978-5-8114-3801-3
1.

Перечень дополнительной учебной литературы, необходимой для освоения дисциплины

Абрамов, Г.В. Проектирование информационных систем Электронный ресурс : учебное пособие / Л.А. Коробова / И.Е. Медведкова / Г.В. Абрамов ; ред. И.А. Авцинов. - Проектирование

информационных систем,2020-09-27. - Воронеж : Воронежский государственный университет инженерных технологий, 2021. - 172 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-89448-953-7

Липаев, В. В. Системное проектирование сложных программных средств для информационных систем. - М. : СИНТЕГ, 2022. - 224 с. - (Информатизацияция России на пороге XXI века). - Библиогр.: с. 209-211. - ISBN 5-89638-019-4

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. http://www.iprbookshop.ru/
2. http://ru.wikipedia.org/wiki/
3. Научная электронная библиотека www.elibrary.ru

6. Методические рекомендации преподавателю

Курс , соответствующий данной программе, содержит лекции, лабораторные занятия и самостоятельную работу студентов.

Целью лекций является изложение теоретического материала и иллюстрация его примерами новейших информационных технологий, компьютерной техники, программного обеспечения.

Целью лабораторных занятий является закрепление теоретического материала лекций и выработка умения применять теоретические знания при решении практических заданий.

Основным теоретическим определениям должны сопутствовать пояснения, касающиеся применения действующих информационных технологий в России. Курс лекций должен строится на основе формулировок теоретических положений дисциплины, так как только при таком подходе студенты приобретают базовые знания, необходимые для дальнейшего изучения дисциплин общепрофессионального и специального циклов в целом.

В течение каждого семестра необходимо провести по две аттестации. Аттестация может проходить в форме письменного опроса по теории или в виде тестов. Студент, сдавший на удовлетворительно и выше две аттестации, а также сдавший все на момент аттестации лабораторные работы, получает зачет или допуск к экзамену.

7. Методические указания по выполнению лабораторных работ

Лабораторная работа №1 Разработка технического задания

Цель работы: ознакомление с процедурой разработки технического задания на создание программного продукта с применением ГОСТ 19.102-77 «Стадии разработки программ и программной документации».

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На данной стадии выполняются следующие работы:

1.Обоснование необходимости разработки программ:

- Постановка задачи;
- Сбор исходных материалов;
- Выбор и обоснование критериев эффективности и качества;
- Обоснование необходимости проведения научно- исследовательских работ.

2.Выполнение научно-исследовательских работ:

- Определение структуры входных и выходных данных;
- Предварительный выбор методов решения задач;
- Обоснование целесообразности применения ранее разработанных программ;
- Определение требований к техническим средствам;
- Обоснование принципиальной возможности решения поставленных задач;

3.Разработка и утверждение технического задания:

- Определение требований к программе;
- Разработка технико-экономического обоснования разработки программы;
- Определение стадий, этапов и сроков разработки программы и документации на нее;
- Выбор языков программирования;
- Определение необходимости проведения научно- исследовательской работы на последующих стадиях;
- Согласование и утверждение технического задания.

Результатом выполнения данной стадии является техническое задание, оформленное в соответствии с ОС ТУСУР.

Лабораторная работа № 2 Разработка эскизного проекта

Цель работы: ознакомление с процедурой разработки эскизного проекта на программный продукт, с применением ГОСТ 19.105 -78 «Пояснительная записка к техническому проекту» и ГОСТ 19.404 – 79 «Пояснительная записка. Требования к содержанию и оформлению».

Конкретное содержание работ на стадии эскизного проекта и их объем определяет степень сложности разрабатываемого ПП. Результатом выполнения данной стадии является полное описание архитектуры ПП. Как правило, это описание делается на нескольких уровнях иерархии. На верхнем уровне детализации выделяются основные подсистемы, которым присваиваются имена, устанавливаются связи между подсистемами, их функции, получаемые путем декомпозиции предполагаемых функций ПП. Затем процедура декомпозиции выполняется для каждой подсистемы, выделяются модули, составляющие данную подсистему. В конечном итоге, получается иерархически организованная система, состоящая из уровней, каждый из которых представляет собой совокупность взаимосвязанных модулей.

В качестве примера ниже приводится фрагмент расширенного описания работ стадии эскизного проекта:

- разработка плана совместных работ на разработку ПП;
- разработка и обоснование математической модели системы и описание результатов моделирования;
- разработка и обоснование алгоритмов и временных графиков функционирования ПП по всем режимам работы;
- разработка и обоснование ресурсов памяти для реализации алгоритмов;
- разработка перечня документов на ПП;
- разработка и обоснование структуры БД, внешних входных и выходных данных;
- разработка и обоснование алгоритмов информационного обеспечения;
- разработка и обоснование набора тестов для проверки ПП;
- разработка и обоснования организации работ по развитию ПП;
- Оформление пояснительной записки.

Результатом выполнения данной работы является эскизный проект, оформленный в соответствии с ОС ТУСУР.

Лабораторная работа № 3 Оценка качественных показателей ПС

Цель работы: в лабораторной работе тестируем и оцениваем качественные показатели ПС.

Методика оценки качественных показателей ПС основана на составлении метрики ПС. В лабораторной работе необходимо выполнить следующее:

1. Выбрать стандарт для оценки качества ПС, аргументировать свой выбор. Далее отобрать показатели качества (не менее 5) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1).

Показатели качества	Сущность показателя	Экспертная оценка (вес) w_i	Оценка, установленная ксперимент $r_{эi}$

2. Установить веса показателей w ($\sum w_i = 1$).

3. Для каждого показателя установить конкретную численную оценку r от 0 до 1, исходя из следующего:

- 0 – свойство в ПС присутствует, но качество его неприемлемо;
- 0.5 - 1 – свойство в ПС присутствует и обладает приемлемым качеством;
- 1 – свойство в ПС присутствует и обладает очень высоким качеством.
- Возможно, присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПС.

$$K = \sum w_i \cdot r_i$$

Результатом выполнения данной работы является отчет с перечнем проведенных тестов и рассчитанное среднее значение оценки качества ПС.

Лабораторная работа № 4 Тестирование программных систем

Цель работы: экспериментальное определение фактических (достигнутых) характеристик свойств испытываемой программной системы.

Тестирование является завершающим этапом разработки программного продукта. Ему предшествует этап статической и динамической отладки программ. В узком смысле цель тестирования состоит в обнаружении ошибок, цель же отладки – не только в обнаружении, но ив устранении ошибок. Цели у отладки и испытания разные. Полностью отлаженная программа может не обладать определенными потребительскими свойствами и тем самым быть непригодной к использованию по своему назначению. Не может служить альтернативой испытанию и проверка работоспособности программы на контрольном примере, так как программа, работоспособная в условиях контрольного примера, может оказаться неработоспособной в других условиях применения. Попытки охватить контрольным примером все предполагаемые условия функционирования сводятся в конечном счете к тем же испытаниям.

Методы тестирования:

- ВОСХОДЯЩЕЕ ТЕСТИРОВАНИЕ – программа собирается и тестируется снизу вверх.
- НИСХОДЯЩЕЕ ТЕСТИРОВАНИЕ – программа собирается и тестируется сверху вниз. Изолировано тестируется только головной модуль.
- МЕТОД БОЛЬШОГО СКАЧКА – каждый модуль тестируется автономно. По окончании тестирования модулей они интегрируются в систему все сразу.
- МЕТОД САНДВИЧА – представляет собой компромисс между восходящим и нисходящим подходами. При использовании этого метода одновременно начинают восходящее и нисходящее тестирование, собирая программу как снизу, так и сверху и встречаясь, в конце концов, где-то в середине. Точка встречи зависит от конкретной тестируемой программы и должна быть заранее определена при изучении ее структуры.

Результатом выполнения данной работы является разработанный план тестирования ПС, отчет о тестировании и Акт о тестировании ПС.

Лабораторная работа № 5 Составление технологической документации

Цель работы: ознакомление с процедурой составления технологической документации к разработанному программному продукту.

Документация по сопровождению ПП (system documentation)

описывает ПП с точки зрения ее разработки.

В случае необходимости модернизации ПП к этой работе привлекается специальная команда разработчиков-сопроводителей. Этой команде придется иметь дело с такой же документацией, которая определяла деятельность команды первоначальных (основных) разработчиков ПП.

Команда разработчиков-сопроводителей должна будет изучать технологическую документацию, чтобы понять строение и процесс разработки модернизируемого ПП, и внести необходимые изменения, повторяя в значительной степени технологические процессы, с помощью которых создавался первоначальный ПП.

Документация по сопровождению ПП можно разбить на две

группы:

(1) документация, определяющая строение программ и структур данных ПП и технологию их разработки;

(2) документацию, помогающую вносить изменения в ПП.

Документы установления достоверности ПП включают, прежде всего, документацию по тестированию (схема тестирования и описание комплекта тестов), но могут включать и результаты других видов проверки ПС, например, доказательства свойств программ.

Документация второй группы содержит руководство по сопровождению ПП (system maintenance guide), которое описывает известные проблемы вместе с ПП, описывает, какие части системы являются аппаратно- и программно-зависимыми, и как развитие ПП принято в расчет в его строении (конструкции).

Результатом выполнения данной работы является технологическая документация к разработанному ПП, оформленная в соответствии ОС ТУСУР.

Лабораторная работа № 6 Составление пользовательской документации

Цель работы: ознакомление с процедурой составления пользовательской (эксплутационной) документации к программному продукту.

Эксплуатационная документация должна обеспечивать отчуждаемость ПС от их первичных разработчиков, адекватно отражать требуемое внешнее качество и качество в использовании, а также возможность освоения и эффективного применения ПС достаточно квалифицированными специалистами. Она применяется непосредственными пользователями в соответствии с функциональным назначением ПС, а также заказчиками, покупателями и поставщиками программных продуктов. Состав этой документации формируется с использованием части технологических документов с учетом требований заказчиков или потенциальных пользователей ПС. Содержание эксплуатационных документов должно предотвращать или исключать возможность некорректного использования комплекса программ пользователями за пределами условий эксплуатации, при которых поставщиком гарантируются требуемые и утвержденные характеристики качества функционирования ПС. При формировании эксплуатационных документов ПС, кроме базовых стандартов жизненного цикла могут использоваться ряд ведомственных нормативных документов и фирменных руководств.

Эксплуатационная документация включает в себя:

- Документация администрирования при применении ПС;

- Документация операторов-пользователей при применении программного средства;
- Документация обучения специалистов применению ПС.

Документация администрирования при эксплуатации информационной системы должна обеспечивать поддержку первичной инсталляции, штатного функционирования и восстановления программ и данных после сбоев и отказов. Управляющая деятельность администратора состоит в манипулировании управляемыми объектами и должна описываться, анализироваться и регламентироваться совокупностью требований и документов. Для этого необходима полная документация о компонентах информационной системы (компьютерах, сетевых устройствах), которые имеют свои особенности в управлении с помощью специальных программных компонентов, поддерживающих администрирование и управление системой. К основным функциям системы администрирования, ***документы для которых подлежат разработке и оцениванию,*** относятся:

- планирование использования памяти и производительности вычислительной системы в рабочем режиме применения ПС, оперативное управление и распределение ресурсов информационной системы;
- инсталляция и генерация рабочих версий ПС для оперативных пользователей;
- управление и учет внешней среды при выполнении адаптации и реконфигурации конкретного ПС;
- выявление, регистрация и накопление данных о сбоях и дефектах функционирования программ и данных;
- управление средствами защиты информации и санкционированного доступа пользователей, анализ попыток взлома системы защиты, восстановление программ и информации баз данных при искажениях;
- сбор и обобщение статистики о качестве функционирования ПС в составе системы обработки информации.

Документация операторов-пользователей должна обеспечивать корректную и квалифицированную эксплуатацию комплекса программ во всем диапазоне его характеристик, предписанных требованиями заказчика и зафиксированных метриками в использовании. Объектами разработки и оценивания являются документы на процедуры и компоненты интерфейса с внешней средой и с пользователями, определяющие инициализацию соответствующих операций, их ход и результаты, а также комфортность их выполнения. Должно быть предусмотрено достаточное качество идентификации ошибочных действий и ситуаций, а также стандартизированной формы сообщений об ошибках пользователей.

Приобретение, поставка, разработка, функционирование и сопровождение программных средств в значительной степени зависит от квалификации специалистов. Поэтому эксплуатационной документацией обязательно должно поддерживаться эффективное обучение персонала с целью его подготовки к приобретению, поставке, применению и сопровождению программного средства. ***Процесс обучения специалистов,*** контроль и учет результатов обучения с оцениванием достигнутой ими квалификации должен гарантировать, что соответствующие категории обученного персонала готовы для выполнения запланированных действий и решения задач с определенным программным средством.

Результатом выполнения данной работы является пользовательская документация к разработанному ПП, оформленная в соответствии с ОС ТУСУР.

Лабораторная работа № 7 Оформление документов сертификации

Цель работы : ознакомление с процедурой разработки и оформления документов сертификации программного продукта.

Правила заполнения бланка сертификата соответствия:

В приложени 1-2 приведены образцы заявки на сертификацию и Сертификата. В графах сертификата указываются следующие сведения:

Позиция 1 — Наименование и код органа по сертификации, выдавшего сертификат, в соответствии с аттестатом аккредитации (прописными буквами) и адрес (строчными буквами). Если наименование органа не помещается в одну строку, то допускается адрес писать под обозначенной строкой. В случае если орган использует печать организации, на базе которой он образован, после наименования органа, выдавшего сертификат, в скобках (строчными буквами) указывается наименование этой организации, а адрес — под реквизитом "подпись" позиции 15. Наименование органа (организации) должно быть идентичным наименованию в печати.

Позиция 2 — Регистрационный номер сертификата формируется в соответствии с правилами ведения Государственного реестра.

Позиция 3 — Срок действия сертификата устанавливается органом по сертификации, выдавшим сертификат, по правилам, изложенным в порядке сертификации однородной продукции. При этом дата пишется: число — двумя арабскими цифрами, месяц — прописью, год.

Позиция 4 — Наименование, тип, вид, марка (как правило, прописными буквами) в соответствии с нормативным документом на продукцию; номер технических условий или иного документа, устанавливающего требования к продукции, номер изделия, размер партии, при серийном производстве указать: "серийное производство"; номер накладной (договора, контракта, паспорта и т. д.) — для партии (единичного изделия).

Позиция 5 — Общероссийский классификатор продукции (ОКП). Код продукции (6 старших разрядов) по классификатору продукции.

Позиция 6 — 9-разрядный код продукции по классификатору товарной номенклатуры внешней экономической деятельности (заполняется обязательно для импортируемой и экспортируемой продукции). Толкование содержания позиции и определение кодов ТН ВЭД, анализ классификационных признаков и лексических средств их выражения осуществляются органами Государственного таможенного комитета Российской Федерации.

Позиция 7 - При обязательной сертификации *в* первой строке указываются свойства, на соответствие которым она проводится, например: "безопасности". Во второй строке — обозначение нормативных документов, на соответствие которым проведена сертификация - Если продукция сертифицирована на все требования нормативного документа (документов), первая строка текстом не дополняется.

Позиция 8 — Если сертификат выдан изготовителю, указывается наименование предприятия-изготовителя. Если сертификат выдан продавцу, подчеркивается слово "продавец", указываются наименование и адрес предприятия, которому выдан данный сертификат, а также, начиная со слова "изготовитель" наименование и адрес предприятия — изготовителя продукции. Наименования и адреса предприятий указываются в соответствии с заявкой.

Позиция 9 - При наличии указываются регистрационный номер в Государственном реестре сертификата системы качества или производства со сроком действия, номер и дата акта (протокола) о проверке производства или другие документы, подтверждающие стабильность производства, например, выданные зарубежной организацией и учтенные органон по сертификации.

Позиция 10 - Строка после слов "Сертификат выдан на основании" не заполняется.

Позиции 11,12,13 — Указываются все документы об испытаниях или сертификации, учтенные органом сертификации при выдаче сертификата в том числе:

1. Протоколы испытаний в аккредитованной лаборатории (поз.11, 12, 13 заполняются в соответствии с графами таблицы).

2 .Протоколы испытаний в не аккредитованной испытательной лаборатории (в позиции 13 указываются наименование и дата Решения Госстандарта России о разрешении проведения испытаний в указанной лаборатории).

3 .Документы, выданные органами и службами государственных органов управления: Госсанэпиднадзора, Госкомэкологии РФ, государственной ветеринарной службы РФ и другие (в поз. 11— наименование органа, выдавшего документ, в поз. 12, 13 — реквизиты документов).

4 .Документы, выданные зарубежными органами: сертификаты (протоколы испытаний) (в

поз. 11 указываются наименование органа и его адрес, в поз. 13 - наименование и дата утверждения сертификата (протокола испытаний), срок действия сертификата).

5 .При выдаче сертификата на основании заявления-декларации в поз. 11 и 12 указываются реквизиты заявления-декларации, а также документов, приведенных в декларации.

Позиция 14 — В случае выдачи заявителю лицензии на право маркирования продукции знаком соответствия в данной позиции указывается: "Маркирование продукции производится знаком соответствия по ГОСТ Р 50.460 – 92".

Позиция 15 — Указывается место нанесения знака соответствия на изделии, таре, упаковке либо сопроводительной документации в соответствии с порядком сертификации однородной продукции.

Позиция 16 — Подпись, инициалы, фамилия руководителя органа, выдавшего сертификат, печать органа или организации, на базе которой образован орган, на обеих сторонах сертификата.

Позиция 17 - Дата регистрации в Государственном реестре.

Исправления, подчистки, поправки на сертификате не допускаются.

Результатом выполнения данной работы является оформленные заявка на проведение сертификации продукции в Системе добровольной сертификации и Сертификат соответствия ГОСТ Р на разработанный ПП.

Лабораторная работа № 8 Составление лицензионного соглашения

Цель работы: ознакомление с процедурой составления лицензионного соглашения конечного пользователя программного продукта.

Лицензионное соглашение (License agreement, Лицензионный договор) - договор, по которому одна сторона (лицензиар) предоставляет право на использование изобретения или иного технического достижения (лицензию), а другая сторона (лицензиат) выплачивает за это определенное вознаграждение.

Лицензионные договоры на программное обеспечение содержат следующие пункты:

1. ДЕКЛАРАТИВНАЯ ЧАСТЬ (RECITALS)

А. Доверитель является эксклюзивным обладателем мировых прав, на программное обеспечение и сопутствующую документацию,

относящуюся к *(тип программного обеспечения),* как изложено далее и описано в Приложении "А" к настоящему Соглашению (далее именуемое "Программное обеспечение Доверителя").

В. Доверитель желает передать Получателю эксклюзивную, оплаченную и бессрочную лицензию на Программное обеспечение Доверителя в <странах>, при сохранении эксклюзивных прав в

<странах>.

ДОВЕРИТЕЛЬ И ПОЛУЧАТЕЛЬ УПОМИНАЮТСЯ В ДАЛЬНЕЙШЕМ ПО ОТДЕЛЬНОСТИ КАК "СТОРОНА", А ВМЕСТЕ

- КАК "СТОРОНЫ".

НАСТОЯЩИМ УДОСТОВЕРЯЕТСЯ, ЧТО, принимая во

внимание исходные предпосылки и взаимные обязательства, изложенные ниже, Стороны договорились о нижеследующем:

2. ОПРЕДЕЛЕНИЯ (DEFINITIONS)

Нижеследующие подпункты (обозначенные большими буквами) имеют в настоящем разделе следующие значения.

"Права Доверителя" означают все права Доверителя на Программное обеспечение Доверителя. В целях настоящего Соглашения, если контекст такового не требует чего-либо другого, Программное обеспечение Доверителя включает в себя любые Изменения Доверителя

(как определено в разделе 3).

Под "Информацией" понимается любая спецификация, документация, программное обеспечение, листинги программ, схемы, чертежи, данные, клиентские списки и отчеты, финансовая документация, деловая информация и любая другая информация подобного рода в электронном или зрительно читаемом виде, которая является собственностью и/или коммерческой тайной Доверителя, Получателя или Аффилированного лица Получателя.

Термин "Лицензионные права" означает все права Доверителя на Лицензионной территории, включая, но не ограничиваясь, права на использование, модификацию и подготовку производных произведений на основе Программного обеспечения Доверителя, а также право сублицензирования Программного обеспечения Доверителя для использования (третьими лицами) на Лицензионной Территории.

Под "Лицензионной территорией" понимается территория .

"Документация по программному обеспечению" означает документы, поименованные далее в Приложении "А".

3. ПЕРЕДАЧА (GRANT)

Доверитель настоящим передает Получателю эксклюзивную, полностью оплаченную, бессрочную лицензию на "Лицензионные права" в соответствии с настоящим Соглашением.

4. УСЛУГИ ПО ПОДДЕРЖКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (SOFTWARE SUPPORT SERVICES)

Доверитель обязуется оказывать или предпримет меры по оказанию Получателю разумных услуг по обслуживанию поддержке Программного обеспечения Доверителя. Такие услуги будут оказаны Получателю по обычным действующим ставкам.

5. РОЯЛТИ (ROYALTY)

В виде компенсации за переданные права на Программное обеспечение Получатель выплачивает Доверителю роялти из расчета

_ % от чистой сумму продаж, произведенных Получателем или собранных им.

6. ПРЕКРАЩЕНИЕ (TERMINATION)
7. СУДЕБНЫЙ ЗАПРЕТ (INJUNCTIVE RELIEF)
8. КОНФИДЕНЦИАЛЬНОСТЬ (CONFIDENTIALITY)
8. ГАРАНТИИ ДОВЕРИТЕЛЯ (GRANTOR WARRANTIES)
9. ГАРАНТИИ ПОЛУЧАТЕЛЯ (GRANTEE WARRANTIES) 10.ГАРАНТИЯ ВОЗМЕЩЕНИЯ УЩЕРБА, НАНОСИМОГО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ ДОВЕРИТЕЛЯ (GRANTOR'S INTELLECTUAL PROPERTY INDEMNITY)
11. ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДОВЕРИТЕЛЯ (PROTECTION OF THE GRANTOR SOFTWARE)
12. ГАРАНТИЯ ВОЗМЕЩЕНИЯ УЩЕРБА (INDEMNITY)
13. ОТКАЗ ОТ ОТВЕТСТВЕННОСТИ (DISCLAIMER)
14. УПРАВЛЯЮЩИЙ ЗАКОН И ТЕРРИТОРИАЛЬНАЯ ПОДСУДНОСТЬ (GOVERNING LAW AND VENUE) 15.ОТСУТСТВИЕ ПРАВА НА ОТКАЗ (NO WAIVER)
16. СОВОКУПНОСТЬ СРЕДСТВ СУДЕБНОЙ ЗАЩИТЫ (REMEDIES CUMULATIVE)
17. ДЕЛИМОСТЬ СОГЛАШЕНИЯ (SEVERABILITY)
18. ЭКЗЕМПЛЯРЫ (COUNTERPARTS)
19. ОТНОШЕНИЯ СТОРОН (RELATIONSHIP OF PARTIES)
20. ПЕРЕУСТУПКА (ASSIGNMENT)
21. ИЗВЕЩЕНИЯ (NOTICE)
22. ОБЯЗАТЕЛЬНОЕ ДЕЙСТВИЕ (BINDING EFFECT)
23. ЗАГОЛОВКИ РАЗДЕЛОВ (SECTION HEADINGS)
24. ИЗДЕРЖКИ ЗА ПРИНУДИТЕЛЬНОЕ ИСПОЛНЕНИЕ (EXPENSES FOR ENFORCEMENT)

25. ПОЛНОТА СОГЛАШЕНИЯ (ENTIRE AGREEMENT).

Результатом выполнения данной работы является разработанное Лицензионное соглашение на разработанный ПП.

ЛАБОРАТОРНАЯ РАБОТА № 6
«Создание диаграммы вариантов использования»

Цель работы:

Создать диаграмму Вариантов Использования (Use Case) для собственной информационной системы, подобно тому, как показано в примере. Требуемые для этого действия подробно перечислены далее. Готовая диаграмма Вариантов Использования (Use Case) должна выглядеть как на рисунке 1.

Этапы выполнения лабораторной работы

1 Создать диаграммы Вариантов Использования, вариантов использования и Действующих лиц.

1.1 Дважды щелкните на Главной диаграмме Вариантов Использования (Main) в браузере, чтобы открыть ее.

1.2 С помощью кнопки Use Case (Вариант Использования) панели инструментов поместите на диаграмму новый вариант использования. Назовите этот новый вариант использования "Ввести новый заказ".

1.3 Повторите этапы, чтобы поместить на диаграмму остальные варианты использования: Изменить существующий заказ, Напечатать инвентарную опись, Обновить инвентарную опись, Оформить заказ, Отклонить заказ

1.4 С помощью кнопки Actor (Действующее лицо) панели инструментов поместите на диаграмму новое действующее лицо. Назовите его "Продавец".

1.5 Повторите предыдущие шаги, поместив на диаграмму остальных действующих лиц: Управляющий магазином, Клерк магазина, Бухгалтерская система

2 Указать абстрактные варианты использования.

2.1 Щелкните правой кнопкой мыши на варианте использования "Отклонить заказ" на диаграмме. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). Пометьте контрольный переключатель Abstract (Абстрактный), чтобы сделать этот вариант использования абстрактным.

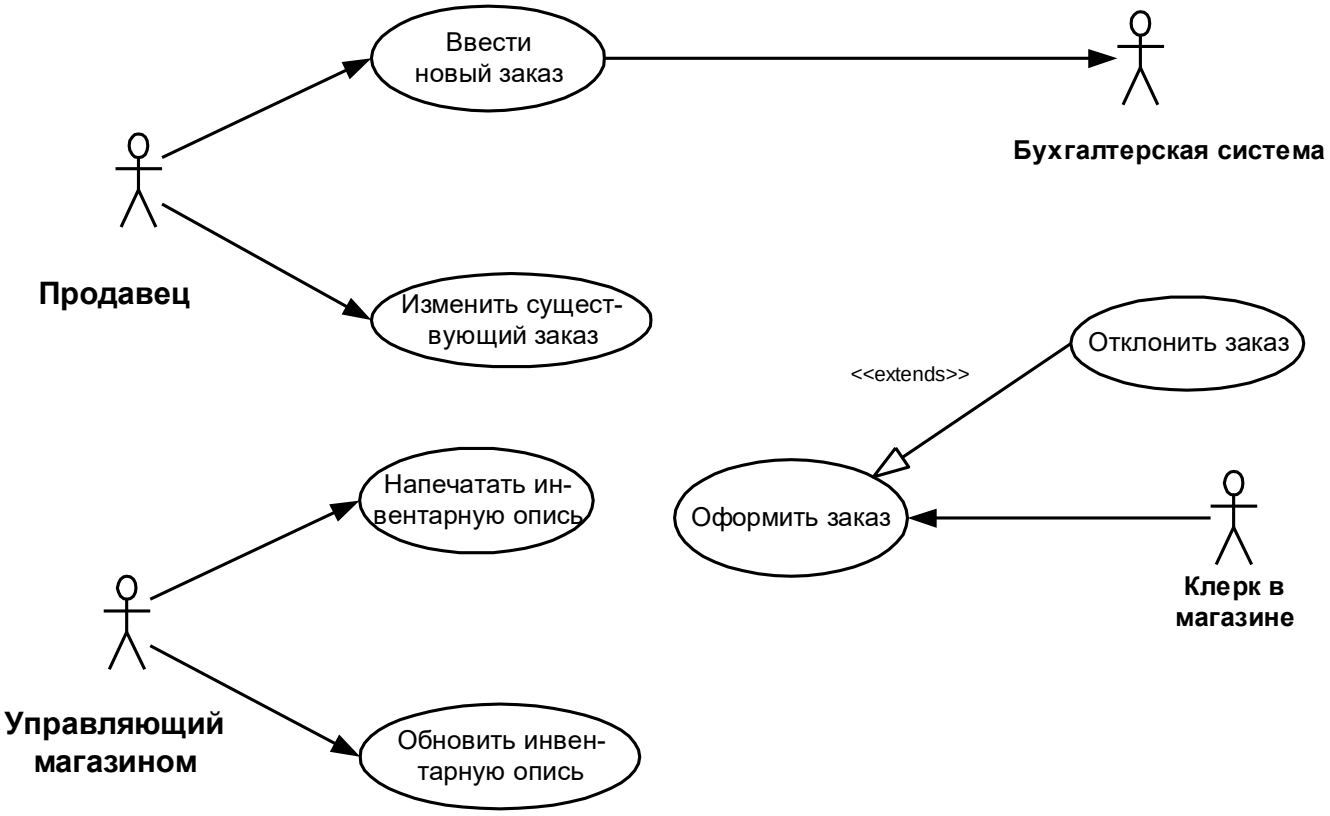


Рисунок 1 – Диаграмма Вариантов Использования для системы обработки заказов

3 Добавить ассоциации.

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

Повторите этот этап, чтобы поместить на диаграмму остальные ассоциации.

4 Добавить связь расширения.

4.1 С помощью кнопки Generalization панели инструментов нарисуйте связь между вариантом использования "Отклонить заказ" и вариантом использования "Оформить заказ". Стрелка должна протянуться от первого варианта использования ко второму. Связь расширения означает, что вариант использования "Отклонить заказ" при необходимости дополняет функциональные возможности варианта использования "Оформить заказ".

4.2 Щелкните правой кнопкой мыши на новой связи между вариантами использования "Отклонить заказ" и "Оформить заказ". В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке стереотипов введите слово extends (расширение), затем нажмите ОК. Слово <<extends>> появится на линии данной связи.

5 Добавить описания к вариантам использования.

5.1 Выделите в браузере вариант использования "Ввести новый заказ".

5.2 В окне документации введите следующее описание к этому варианту использования: Этот вариант использования дает клиенту возможность ввести новый заказ в систему. С помощью окна документации введите описания ко всем остальным вариантам использования.

6 Добавить описания к действующему лицу.

6.1 Выделите в браузере действующее лицо Продавец. В окне документации введите для этого действующего лица следующее описание: Продавец - это служащий, доставляющий и старающийся продать продукцию. С помощью окна документации введите описания к оставшимся действующим лицам.

7 Прикрепление файла к варианту использования.

7.1 Для описания главного потока событий варианта использования "Ввести новый заказ" создайте файл OrderFlow.doc, содержащий следующий текст:

Продавец выбирает пункт "Создать новый заказ" из имеющегося меню.

Система выводит форму "Подробности заказа".

Продавец вводит номер заказа, заказчика и то, что заказано.

Продавец сохраняет заказ.

Система создает новый заказ и сохраняет его в базе данных.

7.2 Щелкните правой кнопкой мыши на варианте использования "Ввести новый заказ". В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). Перейдите на вкладку файлов. Щелкните правой кнопкой мыши на белом поле и из открывшегося меню выберите пункт Insert File (Ввести файл). Укажите файл OpenFlow.doc и нажмите на кнопку Open (Открыть), чтобы прикрепить файл к варианту использования.

Лабораторная работа № 7 «Создание диаграмм взаимодействия»

Цель работы:

Разработать диаграммы Последовательности и Кооперативные диаграммы, описывающие введение нового заказа в информационную систему обработки заказов.

Замечания к выполнению задания. Изучив диаграмму Use Case, четко определена область применения системы. Далее необходимо проанализировать ее составные части. Высший приоритет среди пользователей имеет вариант использования "Ввести новый заказ", он же связан с наибольшим риском. В связи с этим необходимо рассмотреть его в первую очередь. Для этого необходимо четко определить поток событий, который будет реализовываться в варианте использования (необходимо описание сценариев). В результате описание может выглядеть следующим образом:

- Продавец вводит новый заказ.
- Продавец пытается ввести заказ, но товара нет на складе.
- Продавец пытается ввести заказ, но при его сохранении в базе данных произошла ошибка.

Создайте диаграмму Последовательности и Кооперативную диаграмму, отражающую ввод нового заказа в систему обработки заказов. Готовая диаграмма Последовательности должна выглядеть как на рисунке 5.

Это только одна из диаграмм, необходимых для моделирования варианта использования "Ввести новый заказ". Она соответствует успешному варианту хода событий. Для описания того, что случится, если возникнет ошибка, или если пользователь выберет другие действия из предложенных, придется разработать другие диаграммы. Каждый альтернативный поток варианта использования может быть промоделирован с помощью своих собственных диаграмм Взаимодействия.

Этапы выполнения лабораторной работы.

1 Настройка.

1.1 В меню модели выберите пункт Tools > Options (Инструменты > Параметры). Перейдите на вкладку диаграмм.

1.2 Контрольные переключатели Sequence Numbering, Collaboration Numbering и Focus of Control должны быть помечены. Нажмите ОК, чтобы выйти из окна параметров.

2 Создание диаграммы Последовательности.

2.1 Щелкните правой кнопкой мыши на Логическом представлении броузера. В открывшемся меню выберите пункт New > Sequence Diagram. Назовите новую диаграмму "Ввод заказа". Дважды щелкните на ней, чтобы открыть ее.

3 Добавление на диаграмму действующего лица и объектов.

3.1 Перетащите действующее лицо Продавец (Salesperson) с броузера на диаграмму.

3.2 На панели инструментов нажмите кнопку Object (Объект).

3.3 Щелкните мышью в верхней части диаграммы, чтобы поместить туда новый объект. Назовите объект "Order Options Form -- Выбор варианта заказа".

3.4 Повторите этапы, чтобы поместить на диаграмму все остальные объекты:

"Order Detail Form" -- Форма Детали заказа

"Order N1234" -- Заказ №1234.

4 Добавление сообщений на диаграмму.

4.1 На панели инструментов нажмите кнопку Object Message (Сообщение объекта).

4.2 Проведите мышью от линии жизни актера Продавец к линии жизни объекта Выбор варианта заказа.

4.3 Выделив сообщение, введите его имя "Create New Order" -- Создать новый заказ.

4.4 Повторите этапы, чтобы поместить на диаграмму дополнительные сообщения:

Open form -- Открыть форму (между Выбором варианта заказа и Деталями заказа)

Enter order number, customer, order items -- Ввести номер заказа, заказчика и число заказываемых предметов (между Продавцом и Деталями заказа)

Save the order -- Сохранить заказ (между Продавцом и Деталями заказа)

Create new, blank order -- Создать пустой заказ (между Деталями заказа и Заказом №1234)

Set the order number, customer, order items -- Ввести номер заказа, заказчика и число заказываемых предметов (между Деталями заказа и Заказом №1234).

Save the order -- Сохранить заказ (между Деталями заказа и Заказом №1234)

Завершен первый этап работы. Готовая диаграмма Последовательности представлена на рисунке 2.

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

5 Добавление на диаграмму дополнительных объектов.

5.1 На панели инструментов нажмите кнопку Object. Щелкните мышью между объектами Детали заказа и Заказ №1234, чтобы поместить туда новый объект.

5.2 Введите имя объекта - Order Manager (Управляющий заказами).

5.3 На панели инструментов нажмите кнопку Object. Новый объект расположите справа от Заказа №1234. Введите его имя - Transaction Manager (Управляющий транзакциями).

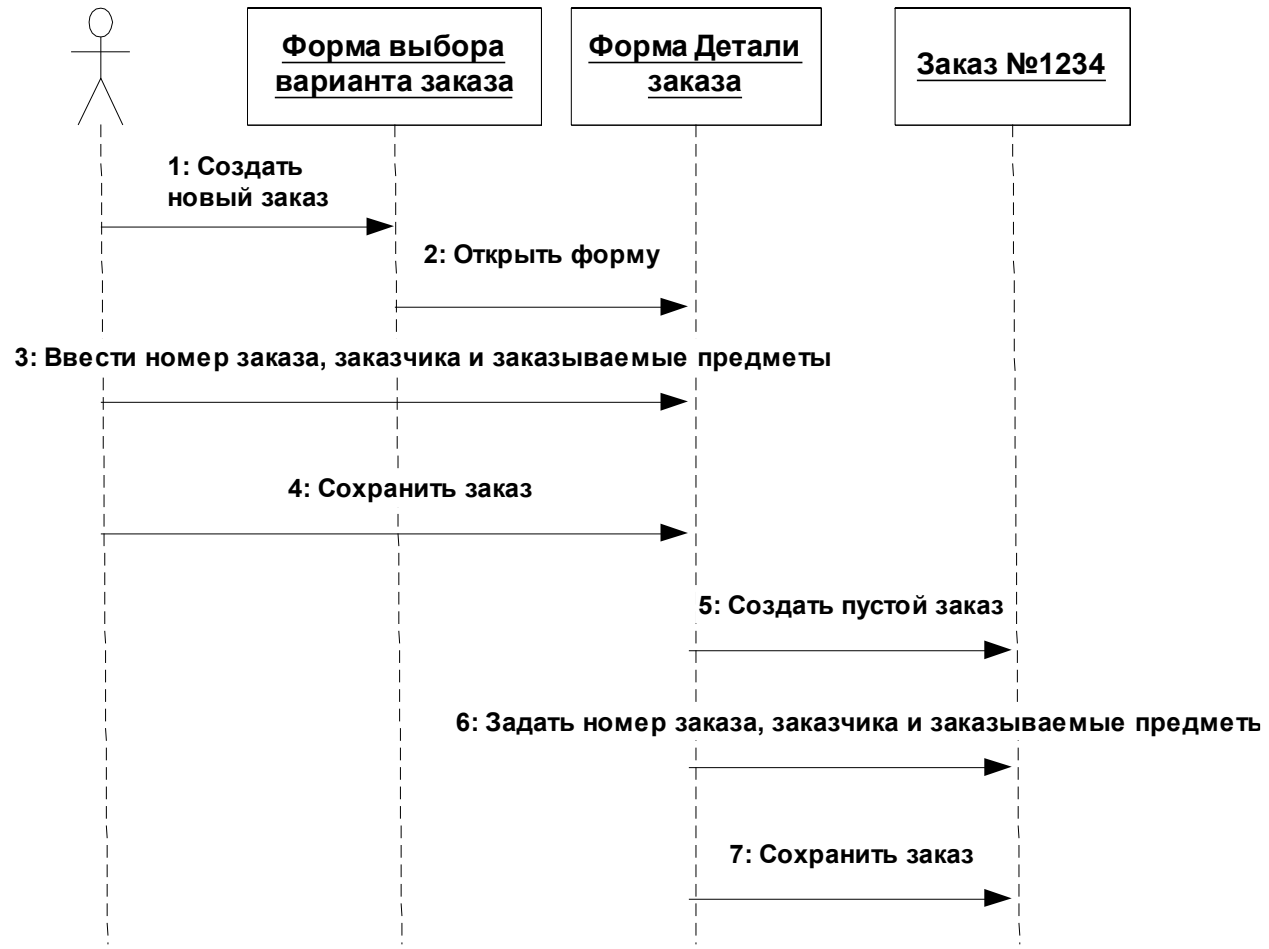


Рисунок 2 – Диаграмма Последовательности ввода нового заказа после завершения первого этапа работы.

6 Назначение ответственностей объектам.

6.1 Выделите сообщение 5 (Создать пустой заказ). Нажмите комбинацию клавиш CTRL + D, чтобы удалить это сообщение.

6.2 Повторите этапы, чтобы удалить два последних сообщения:

Вести номер заказа, заказчика и число заказываемых предметов

Сохранить заказ

6.3 На панели инструментов нажмите кнопку Object Message. Поместите на диаграмму новое сообщение, расположив его под сообщением 4 между Деталями заказа и Управляющим заказами. Назовите его Save the order (Сохранить заказ).

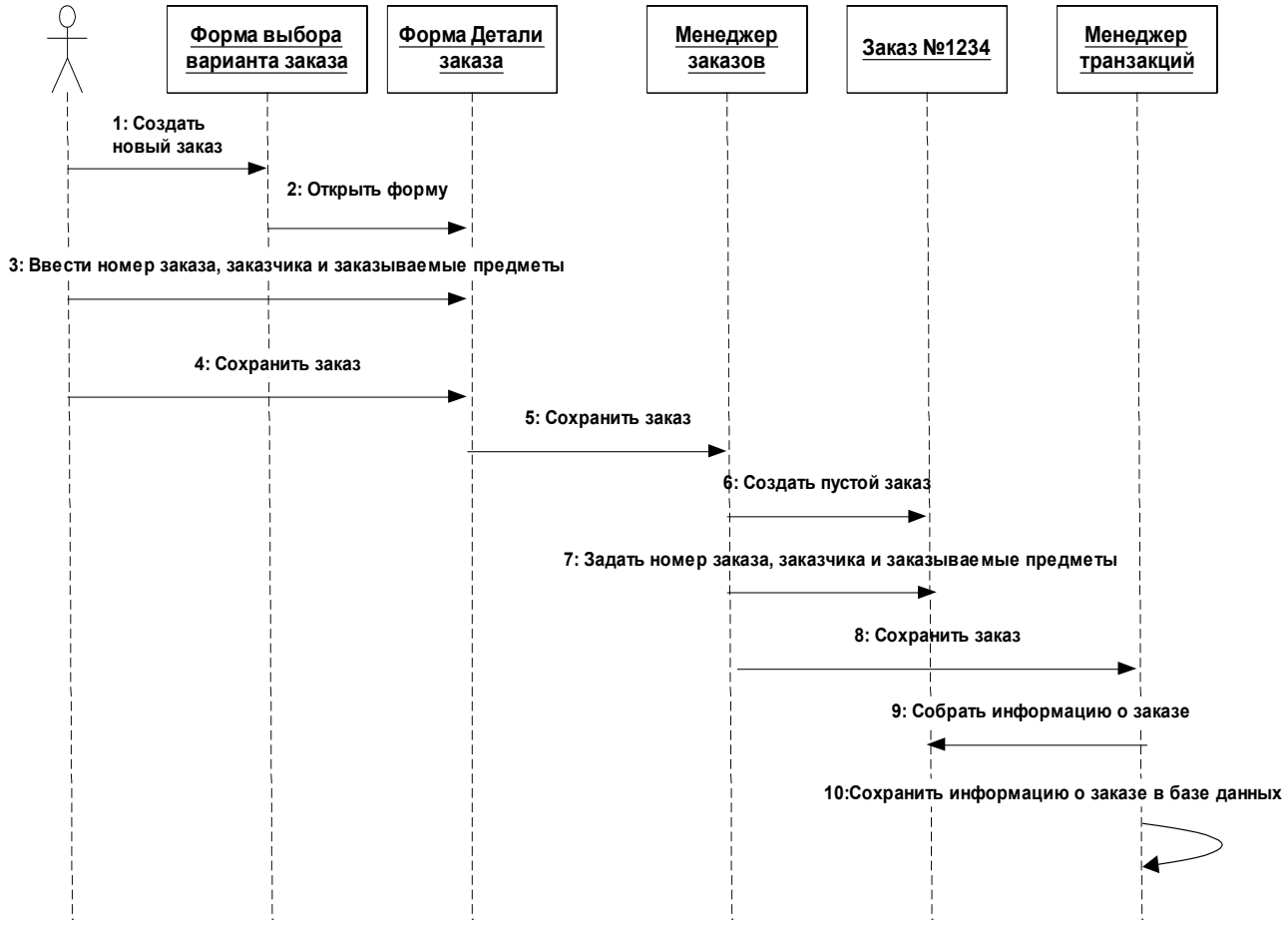


Рисунок 3 – Диаграмма Последовательности с новыми объектами

6.4 Повторите этапы 4 - 6, добавив сообщения с шестого по девятое и назвав их:

Create new, blank order (Создать новый заказ) - между Управляющим заказами и Заказом №1234.

Set the order number, customer, order items (Вести номер заказа, заказчика и число заказываемых предметов) - между Управляющим заказами и Заказом №1234.

Save the order (Сохранить заказ) - между Управляющим заказами и Управляющим транзакциями.

Collect order information (Информация о заказе) - между Управляющим транзакциями и Заказом №1234.

6.5 На панели инструментов нажмите кнопку Message to Self (Сообщение себе).

6.6 Щелкните на линии жизни объекта Управляющий транзакциями ниже сообщения 9, добавив туда рефлексивное сообщение. Назовите его Save the order information to the database (Сохранить информацию о заказе в базе данных).

Теперь диаграмма Последовательности должна выглядеть как на рисунке 3.

7 Соотнесение объектов с классами.

7.1 Щелкните правой кнопкой мыши на объекте Выбор варианта заказа. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов. В поле имени введите имя OrderOptions (Выбор заказа). Щелкните на кнопке ОК. Вы вернетесь к окну спецификации объекта.

7.2 В списке классов выберите теперь класс OrderOptions. Щелкните на кнопке ОК, чтобы вернуться к диаграмме. Теперь объект называется Order Options Form : OrderOptions (Выбор варианта заказа : OrderOptions).

7.3 Для соотнесения остальных объектов с классами повторите этапы:

Класс OrderDetail соотнесите с объектом Детали заказа.

Класс OrderMgr - с объектом Управляющий заказами.

Класс Order - с объектом Заказ №1234.

Класс TransactionMgr - с объектом Управляющий транзакциями.

После завершения этих действий ваша диаграмма должна выглядеть как на рисунке 4.

8 Соотнесение сообщений с операциями.

8.1 Щелкните правой кнопкой на сообщении 1, Создать новый заказ. В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции. В поле имени введите имя операции - Create (Создать). Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться на диаграмму.

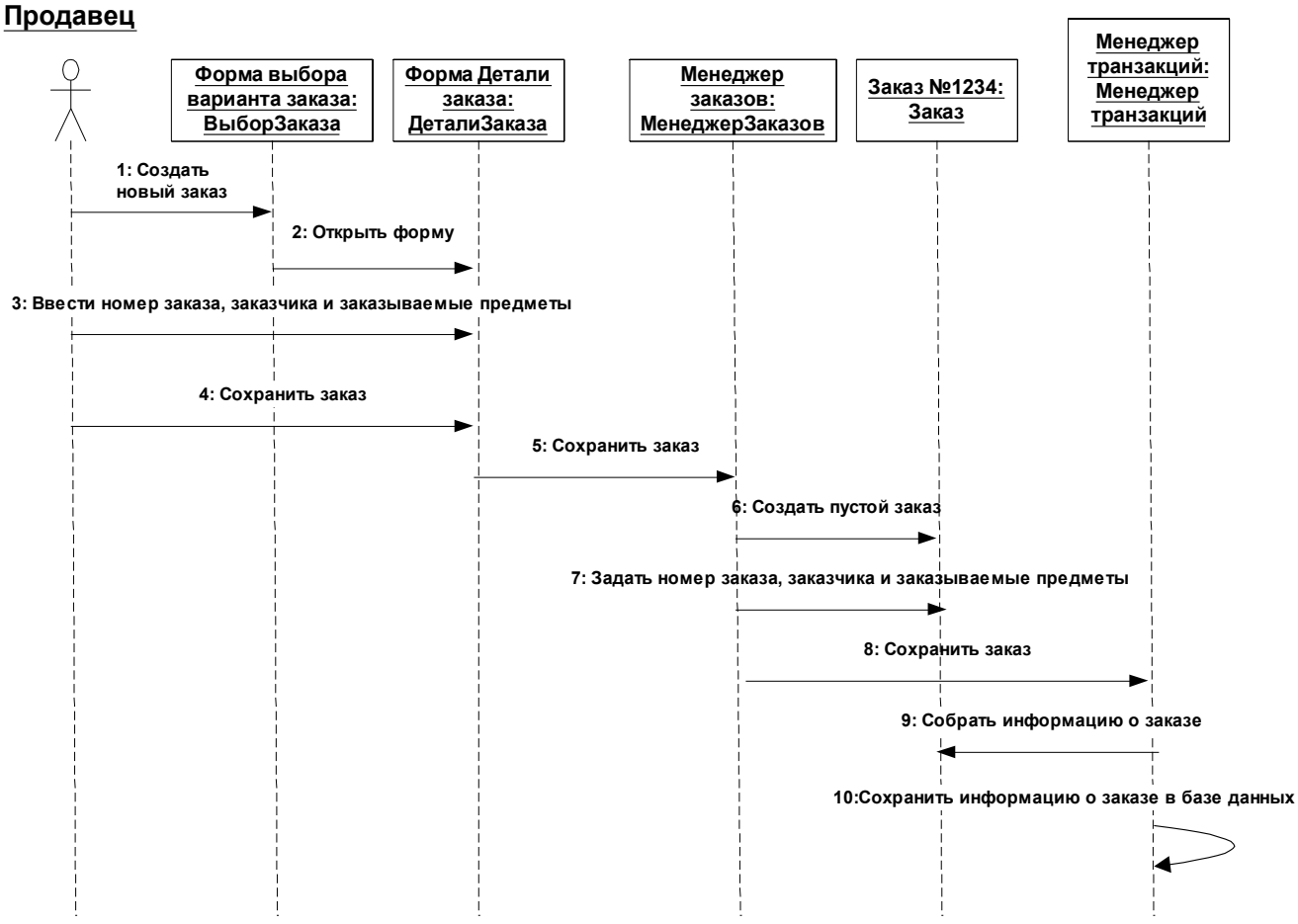


Рисунок 4 – Диаграмма Последовательности с именами классов.

8.2 Еще раз щелкните правой кнопкой мыши на сообщении 1. В открывшемся меню выберите новую операцию Create(). Повторите сообщения с 1 по 6, пока не соотнесете с операциями все остальные сообщения:

Сообщение 2:Открыть соотнесите с операцией Open()

Сообщение 3: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SubmitInfo().

Сообщение 4:Сохранить заказ - с операцией Save().
Сообщение 5:Сохранить заказ - с операцией SaveOrder().
Сообщение 6:Создать пустой заказ - с операцией Create().
Сообщение 7: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SetInfo().
Сообщение 8:Сохранить заказ - с операцией SaveOrder().
Сообщение 9:Информация о заказе - с операцией GetInfo().
Сообщение 10:Сохранить информацию о заказе в базе данных - с операцией Commit.

Ваша диаграмма должна выглядеть как на рисунке 5.

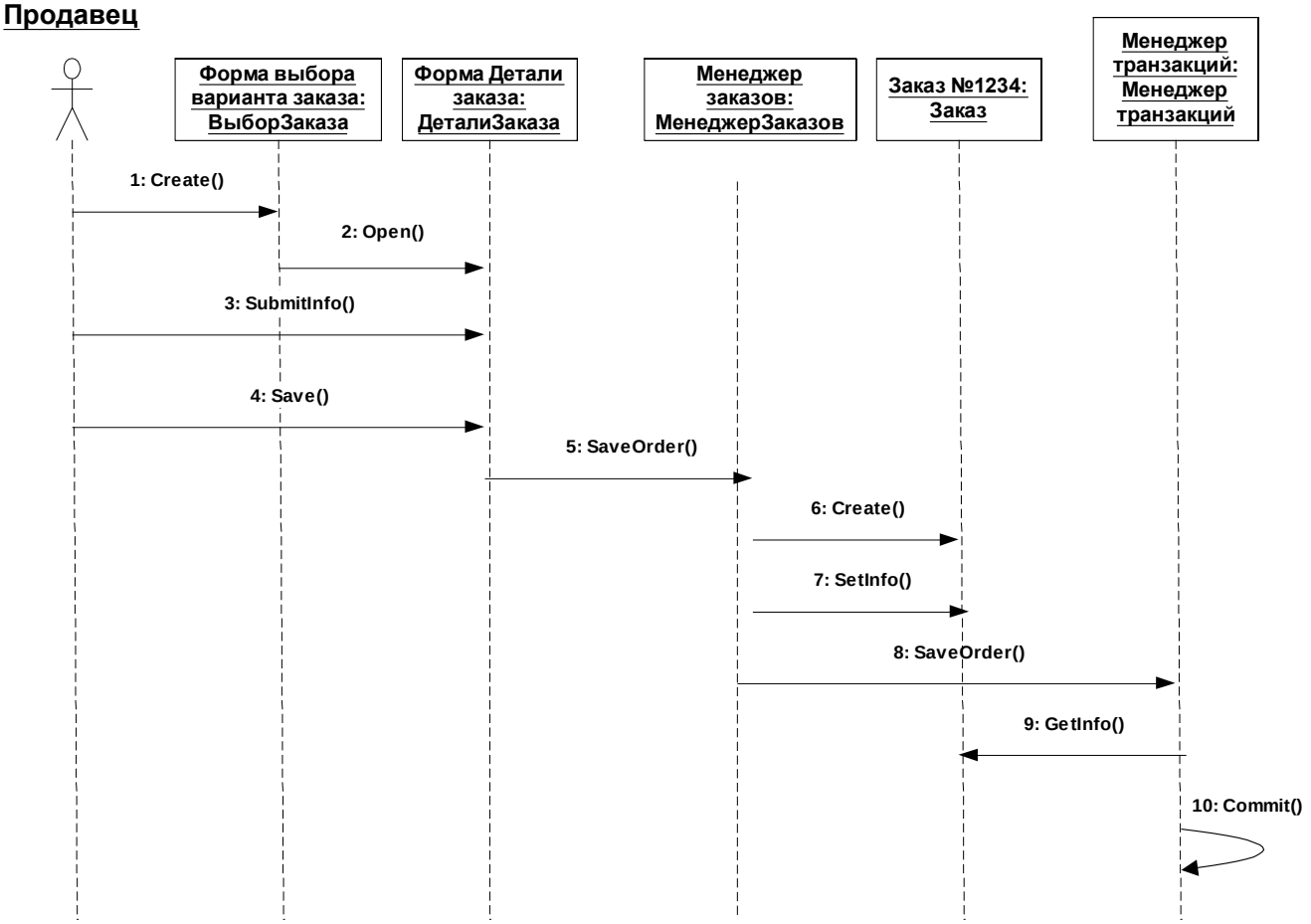


Рисунок 5 – Диаграмма Последовательности с показанными на ней операциями.

9 Создание Кооперативной диаграммы.

Для создания Кооперативной диаграммы достаточно просто нажать клавишу F5 или, если вы хоти сами проделать все требуемые операции, воспользуйтесь приводимым далее планом.

9.1 Щелкните правой кнопкой мыши на Логическом представлении в браузере. В открывшемся меню выберите пункт New > Collaboration Diagram. Назовите эту диаграмму Ввод заказа. Щелкните на ней дважды, чтобы открыть ее.

10 Добавление действующего лица и объектов на диаграмму.

10.1 Перетащите действующее лицо Продавец (Salesperson) с броузера на диаграмму.
10.2 На панели инструментов нажмите кнопку Object (Объект).

10.3 Щелкните мышью где-нибудь внутри диаграммы, чтобы поместить туда новый объект.
10.4 Назовите объект "Order Options Form" -- Выбор варианта заказа.
Повторите этапы 10.3 и 10.4, чтобы поместить на диаграмму все остальные объекты:
"Order Detail Form" -- Форма Детали заказа
"Order N1234" -- Заказ №1234.

11 Добавление сообщений на диаграмму.

11.1 На панели инструментов нажмите кнопку Object Link (Связь объекта).
11.2 Проведите мышью от действующего лица Продавец к объекту Выбор варианта заказа.
11.3 Повторите этапы 11.1 и 11.2, соединив связями следующие объекты:
Действующее лицо Продавец и объект Детали Заказа.
Объект Выбор варианта заказа и объект Детали заказа.
Объект Детали заказа и объект Заказ №1234.
11.4 На панели инструментов нажмите кнопку Link Message (Сообщение связи).
11.5 Щелкните на связи между Продавцом и Выбором варианта заказа.
11.6 Выделив сообщение, введите его имя "Create New Order -- Создать новый заказ".
11.7 Повторите этапы с 11.4 по 11.6, поместив на диаграмму все остальные сообщения, как показано ниже:
Open form -- Открыть форму (между Выбором варианта заказа и Деталями заказа)
Enter order number, customer, order items -- Ввести номер заказа, заказчика и число заказываемых предметов (между Продавцом и Деталями заказа)
Save the order -- Сохранить заказ (между Продавцом и Деталями заказа)
Create new, blank order -- Создать пустой заказ (между Деталями заказа и Заказом №1234)
Set the order number, customer, order items -- Ввести номер заказа, заказчика и число заказываемых предметов (между Деталями заказа и Заказом №1234).
Save the order -- Сохранить заказ (между Деталями заказа и Заказом №1234)
Теперь, как и раньше, надо продолжить работу и поместить на диаграмму дополнительные элементы, а также рассмотреть ответственности объектов.

12 Добавление на диаграмму дополнительных объектов.

12.1 На панели инструментов нажмите кнопку Object.
12.2 Щелкните мышью где-нибудь на диаграмме, чтобы поместить туда новый объект.
12.3 Введите имя объекта - Order Manager (Управляющий заказами).
12.4 На панели инструментов нажмите кнопку Object.
12.5 Поместите на диаграмму еще один объект.
12.6 Введите его имя - Transaction Manager (Управляющий транзакциями).

13 Назначение ответственностей объектам.

13.1 Выделите сообщение 5 (Создать пустой заказ). Выделяйте слова, а не стрелку.
13.2 Нажмите комбинацию клавиш CTRL + D, чтобы удалить это сообщение.
13.3 Повторите этапы 13.1 и 13.2, чтобы удалить сообщения 6 и 7:
Вести номер заказа, заказчика и число заказываемых предметов
Сохранить заказ
13.4 Выделите связь между объектами Детали заказа и Заказ №1234.
13.5 Нажмите комбинацию клавиш CTRL + D, чтобы удалить эту связь.

13.6 На панели инструментов нажмите кнопку Object Link (Связь объекта).

13.7 Нарисуйте связь между Деталями Заказа и Управляющим заказами.

13.8 На панели инструментов нажмите кнопку Object Link (Связь объекта).

13.9 Нарисуйте связь между Управляющим заказами и Заказом №1234.

13.10 На панели инструментов нажмите кнопку Object Link (Связь объекта).

13.11 Нарисуйте связь между Заказом №1234 и Управляющим транзакций.

13.12 На панели инструментов нажмите кнопку Object Link (Связь объекта).

13.13 Нарисуйте связь между Управляющим заказами и Управляющим транзакций.

13.14 На панели инструментов нажмите кнопку Link Message (Сообщение связи).

13.15 Щелкните на связи между объектами Детали заказа и Управляющим заказами, чтобы ввести новое сообщение.

13.16 Назовите это сообщение Save the order (Сохранить заказ).

13.17 Повторите этапы 13.14 – 13.16, добавив сообщения с шестого по девятое и назвав их:

Create new, blank order (Создать новый заказ) - между Управляющим заказами и Заказом №1234.

Set the order number, customer, order items (Вести номер заказа, заказчика и число заказываемых предметов) - между Управляющим заказами и Заказом №1234.

Save the order (Сохранить заказ) - между Управляющим заказами и Управляющим транзакциями.

Collect order information (Информация о заказе) - между Управляющим транзакциями и Заказом №1234.

13.18 На панели инструментов нажмите кнопку Message to Self (Сообщение себе).

13.19 Щелкните на объекте Управляющий транзакциями, добавив к нему рефлексивное сообщение.

13.20 На панели инструментов нажмите кнопку Link Message (Сообщение связи).

13.21 Щелкните на рефлексивной связи Управляющего транзакциями, чтобы ввести туда сообщение.

13.22 Назовите новое сообщение Save the order information to the database (Сохранить информацию о заказе в базе данных).

14 Соотнесение объектов с классами (если при разработке описанной выше диаграммы Последовательности сами классы вы уже создали).

14.1 Найдите в браузере класс OrderOptions.

14.2 Перетащите его на объект Выбор варианта заказа на диаграмме.

14.3 Повторите этапы 14.1 и 14.2, соотнеся остальные объекты и соответствующие им классы:

Класс OrderDetail соотнесите с объектом Детали заказа.

Класс OrderMgr - с объектом Управляющий заказами.

Класс Order - с объектом Заказ №1234.

Класс TransactionMgr - с объектом Управляющий транзакциями.

15 Соотнесение объектов с классами (если вы не создавали описанную выше диаграмму Последовательности).

15.1 Щелкните правой кнопкой мыши на объекте Выбор варианта заказа.

15.2 В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

15.3 В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов.

15.4 В поле имени введите имя OrderOptions (Выбор заказа).

15.5 Щелкните на кнопке ОК. Вы вернетесь к окну спецификации объекта.

15.6 В списке классов выберите теперь класс OrderOptions.

15.7 Щелкните на кнопке ОК, чтобы вернуться к диаграмме. Теперь объект называется Order Options Form : OrderOptions (Выбор варианта заказа : OrderOptions).

15.8 Для соотнесения остальных объектов с классами повторите этапы с 15.1 по 15.7:

Класс OrderDetail соотнесите с объектом Детали заказа.
Класс OrderMgr - с объектом Управляющий заказами.
Класс Order - с объектом Заказ №1234.
Класс TransactionMgr - с объектом Управляющий транзакциями.

16 Соотнесение сообщений с операциями (если при разработке описанной выше диаграммы Последовательности сами операции вы уже создали).

16.1 Щелкните правой кнопкой на сообщении 1, Создать новый заказ.

16.2 В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

16.3 В раскрывающемся списке имен укажите имя операции - Create (Создать). Нажмите на кнопку ОК.

16.4 Повторите этапы с для соотнесения с операциями остальных сообщений:

Сообщение 2:Открыть соотнесите с операцией Open()
Сообщение 3: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SubmitInfo().
Сообщение 4:Сохранить заказ - с операцией Save().
Сообщение 5:Сохранить заказ - с операцией SaveOrder().
Сообщение 6:Создать пустой заказ - с операцией Create().
Сообщение 7: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SetInfo().
Сообщение 8:Сохранить заказ - с операцией SaveOrder().
Сообщение 9:Информация о заказе - с операцией GetInfo().
Сообщение 10:Сохранить информацию о заказе в базе данных - с операцией Commit().

17 Соотнесение сообщений с операциями (если вы не создавали описанную выше диаграмму Последовательности).

17.1 Щелкните правой кнопкой на сообщении 1, Создать новый заказ.

17.2 В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции.

17.3 В поле имени введите имя операции - Create (Создать).

17.4 Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться на диаграмму.

17.5 Еще раз щелкните правой кнопкой мыши на сообщении 1.

17.6 В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

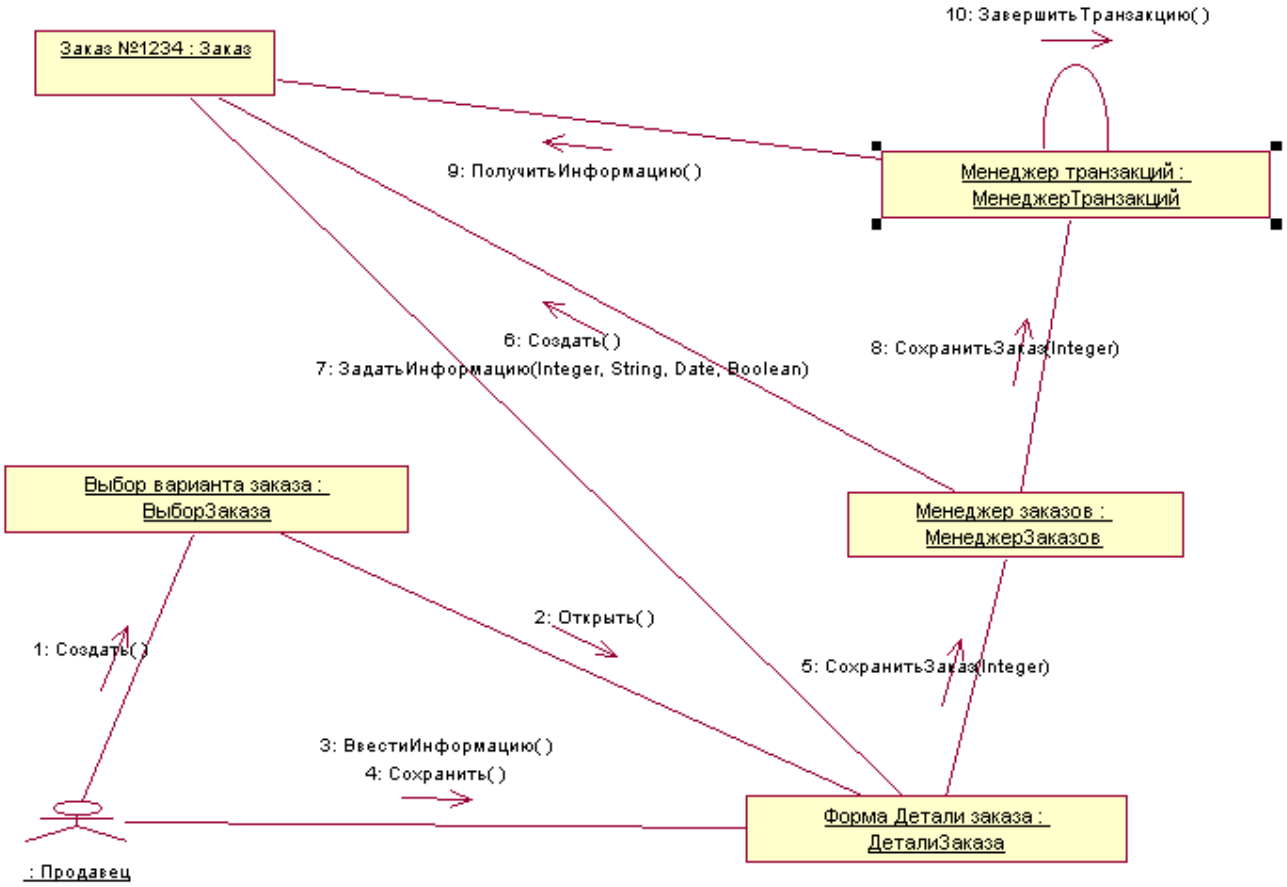


Рисунок 6 – Кооперативная диаграмма с показанными на ней операциями

17.7 В раскрывающемся списке Name (имя) укажите имя новой операции. Нажмите на кнопку ОК.

Повторите этапы с первого по седьмой, чтобы создать новые операции и соотнести с ними остальные сообщения:

Сообщение 2:Открыть соотнесите с операцией Open()
Сообщение 3: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SubmitInfo().
Сообщение 4:Сохранить заказ - с операцией Save().
Сообщение 5:Сохранить заказ - с операцией SaveOrder().
Сообщение 6:Создать пустой заказ - с операцией Create().
Сообщение 7: Ввести номер заказа, заказчика и число заказываемых предметов - с операцией SetInfo().
Сообщение 8:Сохранить заказ - с операцией SaveOrder().
Сообщение 9:Информация о заказе - с операцией GetInfo().
Сообщение 10:Сохранить информацию о заказе в базе данных - с операцией Commit.

Ваша диаграмма должна выглядеть как на рисунке 6.

Лабораторная работа № 8 «Создание диаграмм классов»

Цель работы:

Сгруппировать в пакеты классы, созданные во время выполнения предыдущего задания. Создать несколько диаграмм Классов, на которых необходимо показать классы и пакеты системы.

Замечание (Создание диаграммы Классов). Объедините классы в пакеты. Создайте диаграмму Классов для отображения пакетов, диаграммы Классов для представления классов в

каждом пакете и диаграмму Классов для представления всех классов варианта использования "Ввести новый заказ".

Этапы выполнения лабораторной работы

1 Настройка.

1.1 В меню модели выберите пункт Tools > Options (Инструменты > Параметры).

1.2 Перейдите на вкладку диаграмм. Убедитесь, что помечен контрольный переключатель Show Stereotypes (Показать стереотипы). Убедитесь, что помечены контрольные переключатели Show All Attributes (Показать все атрибуты) и Show All Operations (Показать все операции).

1.3 Убедитесь, что не помечены переключатели Suppress Attributes (Подавить вывод атрибутов) и Suppress Operations (Подавить вывод операций).

2 Создание пакетов.

2.1 Щелкните правой кнопкой мыши на Логическом представлении броузера.

2.2 В открывшемся меню выберите пункт New > Package (Создать > пакет).

2.3 Назовите новый пакет Entities (Сущности).

2.4 Повторите этапы с первого по третий, создав пакеты Boundaries (границы) и Control (управление).

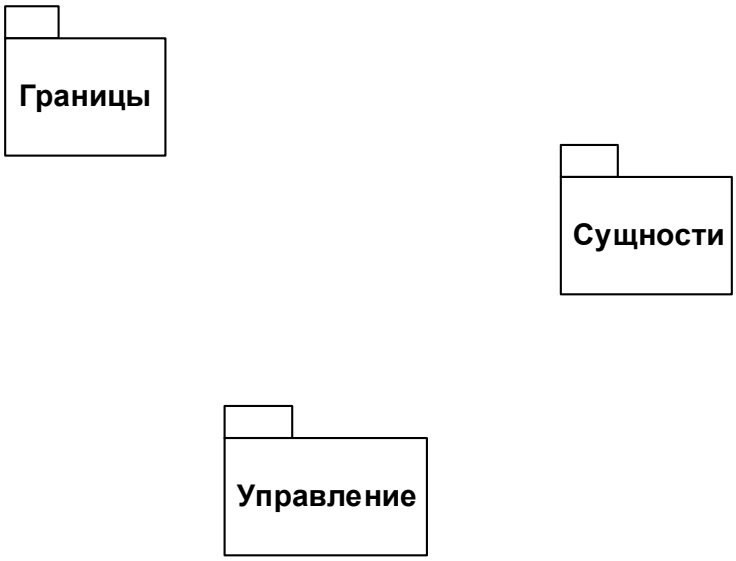


Рисунок 7 - Главная диаграмма Классов системы обработки заказов

3 Создание Главной диаграммы Классов.

3.1 Дважды щелкните на Главной диаграмме Классов прямо под Логическим представлением броузера, чтобы открыть ее.

3.2 Перетащите пакет Entities из броузера на диаграмму.

3.3 Перетащите пакеты Boundaries и Control из броузера на диаграмму.

Главная диаграмма Классов должна выглядеть как на рисунке 7.

4 Создание диаграммы Классов для сценария "Ввести новый заказ" со всеми классами.

4.1 Щелкните правой кнопкой мыши на Логическом представлении броузера.

4.2 В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмму Классов).

4.3 Назовите новую диаграмму Классов Add New Order (Введение нового заказа).

4.4 Щелкните в браузере на этой диаграмме дважды, чтобы открыть ее.

4.5 Перетащите из броузера все классы (OrderOptions, OrderDetail, Order, OrderMgr и TransactionMgr).

Диаграмма Классов должна выглядеть как на рисунке 8.

Рисунок 8 - Диаграмма Классов Add New Order

5 Добавление стереотипов к классам.

5.1 Щелкните правой кнопкой мыши на классе OrderOptions диаграммы. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). В поле стереотипа введите слово Boundary. Нажмите на кнопку ОК.

5.2 Щелкните правой кнопкой мыши на классе OrderDetail диаграммы. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке в поле стереотипов теперь будет стереотип Boundary. Укажите его. Нажмите на кнопку ОК.

Повторите этапы, связав классы OrderMgr и TransactionMgr со стереотипом Control, а класс Order - со стереотипом Entity.

Теперь диаграмма Классов должна выглядеть как на рисунке 9.

6 Объединение классов в пакеты.

6.1 Перетащите в браузере класс OrderOptions на пакет Boundaries.

6.2 Перетащите класс OrderDetail на пакет Boundaries.

6.3 Перетащите классы OrderMgr и TransactionMgr на пакет Control.

6.4 Перетащите класс Order на пакет Entities.

7 Добавление диаграмм Классов к каждому пакету.

7.1 Щелкните правой кнопкой на пакете Boundaries броузера. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмму Классов).

7.2 Введите имя новой диаграммы - Main (Главная). Дважды щелкните мышью на этой диаграмме, чтобы открыть ее. Перетащите на нее из броузера классы OrderOptions и OrderDetail. Закройте диаграмму.

7.3 Щелкните правой кнопкой на пакете Entities броузера. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмму Классов).

7.4 Введите имя новой диаграммы - Main (Главная). Дважды щелкните мышью на этой диаграмме, чтобы открыть ее. Перетащите на нее из броузера класс Order. Закройте диаграмму.

7.5 Щелкните правой кнопкой на пакете Control броузера. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмму Классов).

7.6 Введите имя новой диаграммы - Main (Главная). Дважды щелкните мышью на этой диаграмме, чтобы открыть ее. Перетащите на нее из броузера классы OrderMgr и TransactionMgr.

Закройте диаграмму.

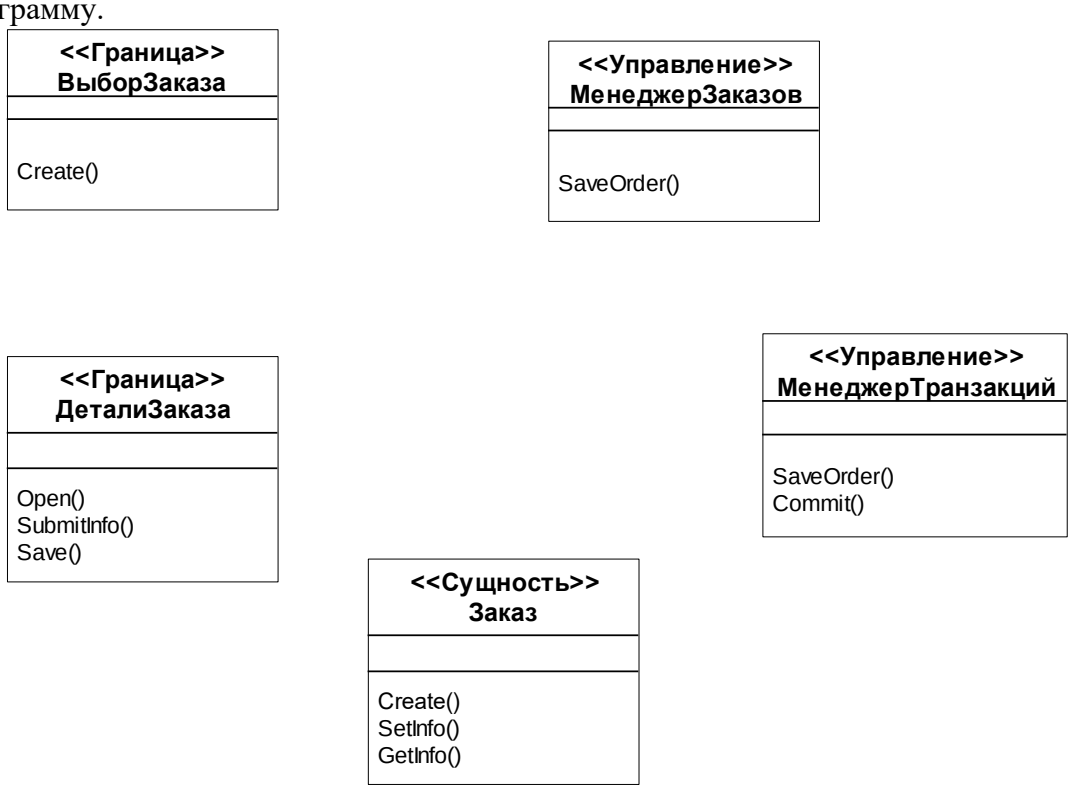


Рисунок 9 - Стереотипы классов для варианта использования Ввести новый заказ

Лабораторная работа № 9 «Создание диаграмм классов (учет новых требований)»

Цель работы:

В Лабораторной № 2 было создано несколько операций для классов задачи. В предыдущем упражнении нанесены классы на диаграмму. В этой лабораторной к описаниям операций необходимо добавить детали, включая параметры и типы возвращаемых значений. Кроме того, у классов необходимо определить атрибуты.

Замечания. 1. После того, как разработана диаграмма Классов для варианта использования "Ввести новый заказ", заполняем ее подробностями. В качестве языка программирования был выбран C++, что позволяет добавить к классам параметры операций, типы данных и типы возвращаемых значений.

Для определения атрибутов необходимо обратиться к потоку событий. В результате, к классу Order диаграммы Классов были добавлены атрибуты Order Number (номер заказа) и Customer Name (Имя клиента). Так как в одном заказе можно указать большое количество товаров, и у каждого из них имеются свои собственные данные и поведение, возможно моделировать их как самостоятельные классы, а не как атрибуты класса Order.

Чтобы привести модель в соответствие с новыми идеями, необходимо обновить диаграмму Последовательностей, как показано на рисунке 10.

2. По заданию надо отслеживать дату заказа и дату его выполнения. Кроме того, так как появились новые поставщики, слегка изменилась процедура инвентаризации. Первоначально документировались новые требования относительно дат, а также рассматривались изменения в процедуре инвентаризации "на высоком уровне". Поскольку мы работали над вариантом использования "Ввести новый заказ", нас больше всего интересовало, как эти процедурные изменения повлияют на данный вариант использования. Работа с вариантом использования "Провести инвентаризацию" будет реализована позже, тогда мы сможем определиться с деталями соответствующих процедур. Хотя они чрезвычайно сильно повлияют на вариант использования

"Провести инвентаризацию", но совсем не отразятся на варианте использования "Ввести новый заказ".

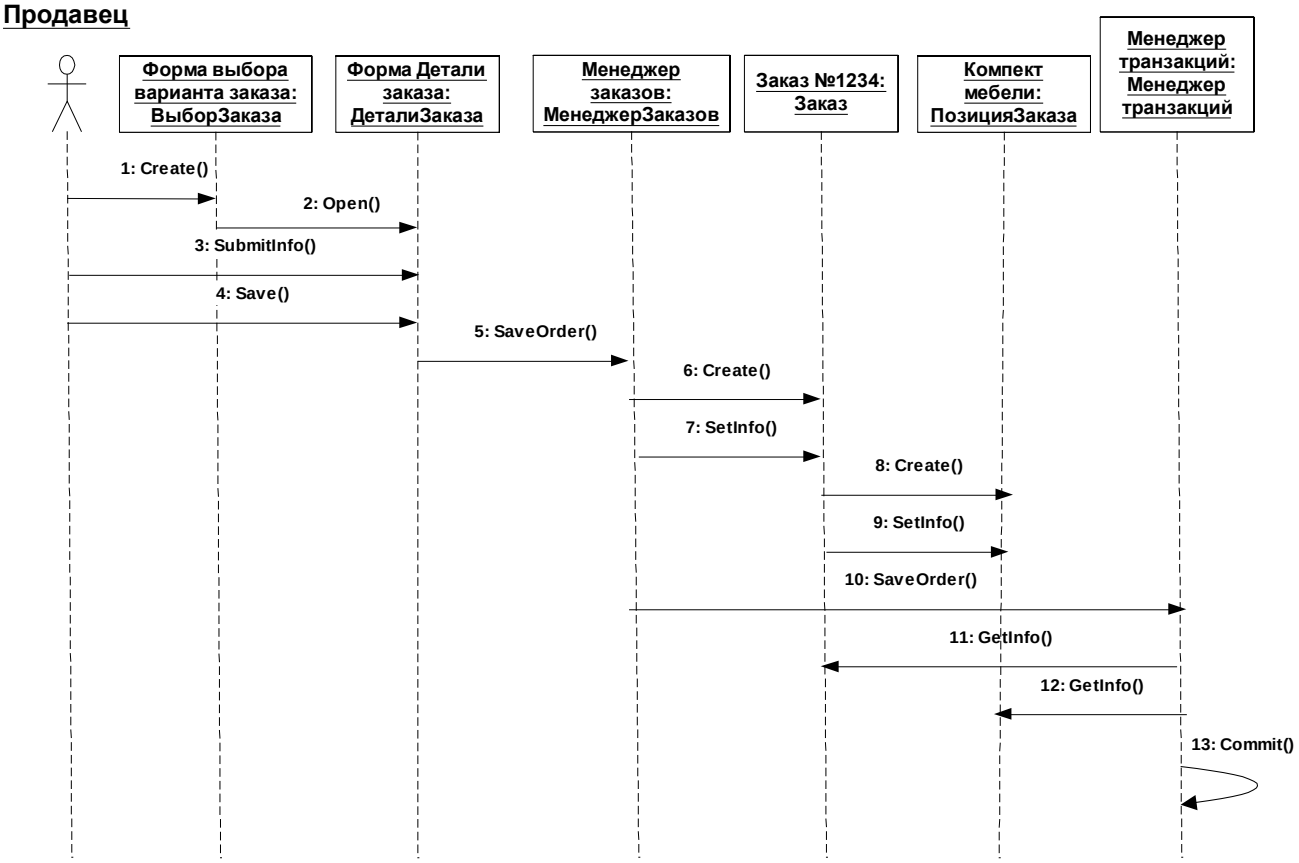


Рисунок 10 - Обновленная диаграмма Последовательностей

Новые требования, связанные с датами, приводят к необходимости ввести пару новых атрибутов в класс Order. После этого модель станет опять соответствовать последним предъявленным к системе требованиям. Поэтому добавим атрибуты и операции к классам диаграммы Классов "Ввести новый заказ". Для атрибутов и операций используем специфические для языка особенности. Установим параметры так, чтобы показывать все атрибуты, все операции и их сигнатуры. Видимость покажем с помощью нотации UML.

Этапы выполнения лабораторной работы

1 Настройка.

1.1 В меню модели выберите пункт Tools > Options.

1.2 Перейдите на вкладку Diagram.

- Убедитесь, что переключатель Show Visibility помечен.
- Убедитесь, что переключатель Show Stereotypes помечен.
- Убедитесь, что переключатель Show Operation Signatures помечен.
- Убедитесь, что переключатели Show All Attributes и Show All Operations помечены.
- Убедитесь, что переключатели Suppress Attributes и Suppress Operations не помечены.

1.3 Перейдите на вкладку Notation. Убедитесь, что переключатель Visibility as Icons не помечен.

2 Добавление нового класса.

2.1 Найдите в браузере диаграмму Классов варианта использования "Ввести новый заказ".

2.2 Щелкните на ней дважды, чтобы ее открыть.

2.3 Нажмите кнопку Class панели инструментов.

2.4 Щелкните мышью внутри диаграммы, чтобы поместить там новый класс.

2.5 Назовите его OrderItem (ПозицияЗаказа).

2.6 Назначьте этому классу стереотип Entity.

2.7 В браузере перетащите класс в пакет Entities.

3 Добавление атрибутов.

3.1 Щелкните правой кнопкой мыши на классе Order (Заказ). В открывшемся меню выберите пункт New Attribute (Создать атрибут). Введите новый атрибут OrderNumber : Integer (НомерЗаказа). Нажмите клавишу Enter.

3.2 Введите следующий атрибут CustomerName : String (НаименованиеЗаказчика).

3.3 Повторите этапы, добавив атрибуты OrderDate : Date (ДатаЗаказа) и OrderFillDate : Date (ДатаЗаполненияЗаказа).

3.4 Щелкните правой кнопкой мыши на классе OrderItem. В открывшемся меню выберите пункт New Attribute (Создать атрибут). Введите новый атрибут ItemID : Integer (ИдентификаторПредмета). Нажмите клавишу Enter.

3.5 Введите следующий атрибут ItemDescription : String (ОписаниеПредмета).

4 Добавление операций к классу OrderItem.

4.1 Щелкните правой кнопкой мыши на классе OrderItem. В открывшемся меню выберите пункт New Operation (Создать операцию). Введите новую операцию Create. Нажмите клавишу Enter.

4.2 Введите следующую операцию SetInfo. Нажмите клавишу Enter.

4.3 Введите следующую операцию GetInfo.

5 Подробное описание операций с помощью диаграммы Классов.

5.1 Щелкните мышью на классе Order, выделив его таким способом.

5.2 Щелкните на этом классе еще один раз, чтобы переместить курсор внутрь.

5.3 Отредактируйте операцию Create(), чтобы она выглядела следующим образом: Create() : Boolean.

5.4 Отредактируйте операцию SetInfo(), чтобы она выглядела следующим образом: SetInfo(OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date) : Boolean.

5.5 Отредактируйте операцию GetInfo(), чтобы она выглядела следующим образом: GetInfo() : String

6 Подробное описание операций с помощью браузера.

6.1 Найдите в браузере класс OrderItem. Чтобы раскрыть этот класс, щелкните на значке "+" рядом с ним. В браузере появятся его атрибуты и операции.

6.2 Дважды щелкните на операции GetIInfo(), чтобы открыть окно ее спецификации. В раскрывающемся списке Return class (возвращаемый класс) укажите String. Щелкните на кнопке ОК, закрыв окно спецификации операции.

6.3 Дважды щелкните в браузере на операции SetInfo класса OrderItem, чтобы открыть окно ее спецификации. В раскрывающемся списке Return class укажите Boolean.

6.4 Перейдите на вкладку Detail (Подробно). Щелкните правой кнопкой мыши на белом поле в области аргументов, чтобы добавить туда новый параметр.

6.5 В открывшемся меню выберите пункт Insert. Rose добавит туда аргумент под названием argname. Щелкните один раз на этом слове, чтобы выделить его, и измените имя аргумента на ID. Щелкните на колонке Type, открыв раскрывающийся список типов. В нем выберите тип Integer. Щелкните на колонке Default, чтобы добавить значение аргумента по умолчанию. Введите туда число 0. Нажмите на кнопку ОК, закрыв окно спецификации операции.

6.6 Дважды щелкните на операции Create() класса OrderItem, чтобы открыть окно ее спецификации. В раскрывающемся списке Return class укажите Boolean. Нажмите на кнопку ОК, закрыв окно спецификации операции.

7 Подробное описание операций с помощью любого из описанных методов.

7.1 Используя броузер или диаграмму Классов, введите следующую сигнатуру операций класса OrderDetail:

Open() : Boolean
SubmitInfo() : Boolean
Save() : Boolean

7.2 Используя броузер или диаграмму Классов, введите следующую сигнатуру операций класса OrderOptions:

Create() : Boolean

7.3 Используя броузер или диаграмму Классов, введите следующую сигнатуру операций класса OrderMgr:

SaveOrder(OrderID : Integer) : Boolean

7.4 Используя броузер или диаграмму Классов, введите следующую сигнатуру операций класса TransactionMgr:

SaveOrder(OrderID : Integer) : Boolean

Commit() : Integer.

Лабораторная работа №10 «Создание диаграмм классов (добавление связей между классами)»

Цель работы:

Определить связи между классами, участвующими в варианте использования "Ввести новый заказ".

Замечание. После добавления к классам атрибутов и операций мы уже почти готовы к генерации кода. Сначала, однако, необходимо изучить связи между классами.

Чтобы найти связи, нужно изучить диаграммы Последовательности. Все взаимодействующие там классы нуждаются в определении соответствующих связей на диаграммах Классов. После обнаружения связей необходимо добавить их в модель.

Этапы выполнения лабораторной работы

1 Настройка.

1.1 Найдите в браузере диаграмму Классов "Ввод нового заказа" Дважды щелкните на ней, чтобы открыть ее. Проверьте, имеется ли на панели инструментов диаграммы кнопка Unidirectional Association. Если ее нет, продолжайте настройку. Если есть, приступайте к выполнению самого упражнения.

1.2 Щелкните правой кнопкой мыши на панели инструментов диаграммы и в открывшемся меню выберите пункт Customize. Добавьте на панель кнопку, называющуюся Create A Unidirectional Association.

2 Добавление ассоциаций.

2.1 Нажмите кнопку панели инструментов Unidirectional Association.

2.2 Нарисуйте ассоциацию от класса ВыборЗаказа (OrderOptions) к классу ДеталиЗаказа (OrderDetail).

2.3 Повторите этапы 2.1 и 2.2, создав еще ассоциации:

От класса OrderDetail к классу МенеджерЗаказов (OrderMgr)
От класса OrderMgr к классу Заказ (Order)
От класса OrderMgr к классу МенеджерТранзакций (TransactionMgr)
От класса TransactionMgr к классу Order
От класса TransactionMgr к классу ПозицияЗаказа (OrderItem)
От класса Order к классу OrderItem

2.4 Щелкните правой кнопкой мыши на однонаправленной ассоциации между классами OrderOptions и OrderDetail, со стороны класса OrderOptions. В открывшемся меню выберите пункт Multiplicity > Zero or One.

2.5 Щелкните правой кнопкой мыши на другом конце однонаправленной ассоциации.В открывшемся меню выберите пункт Multiplicity > Zero or One.

2.6 Повторите эти этапы, добавив на диаграмму значения множественности для остальных ассоциаций, как показано на рисунке 11.

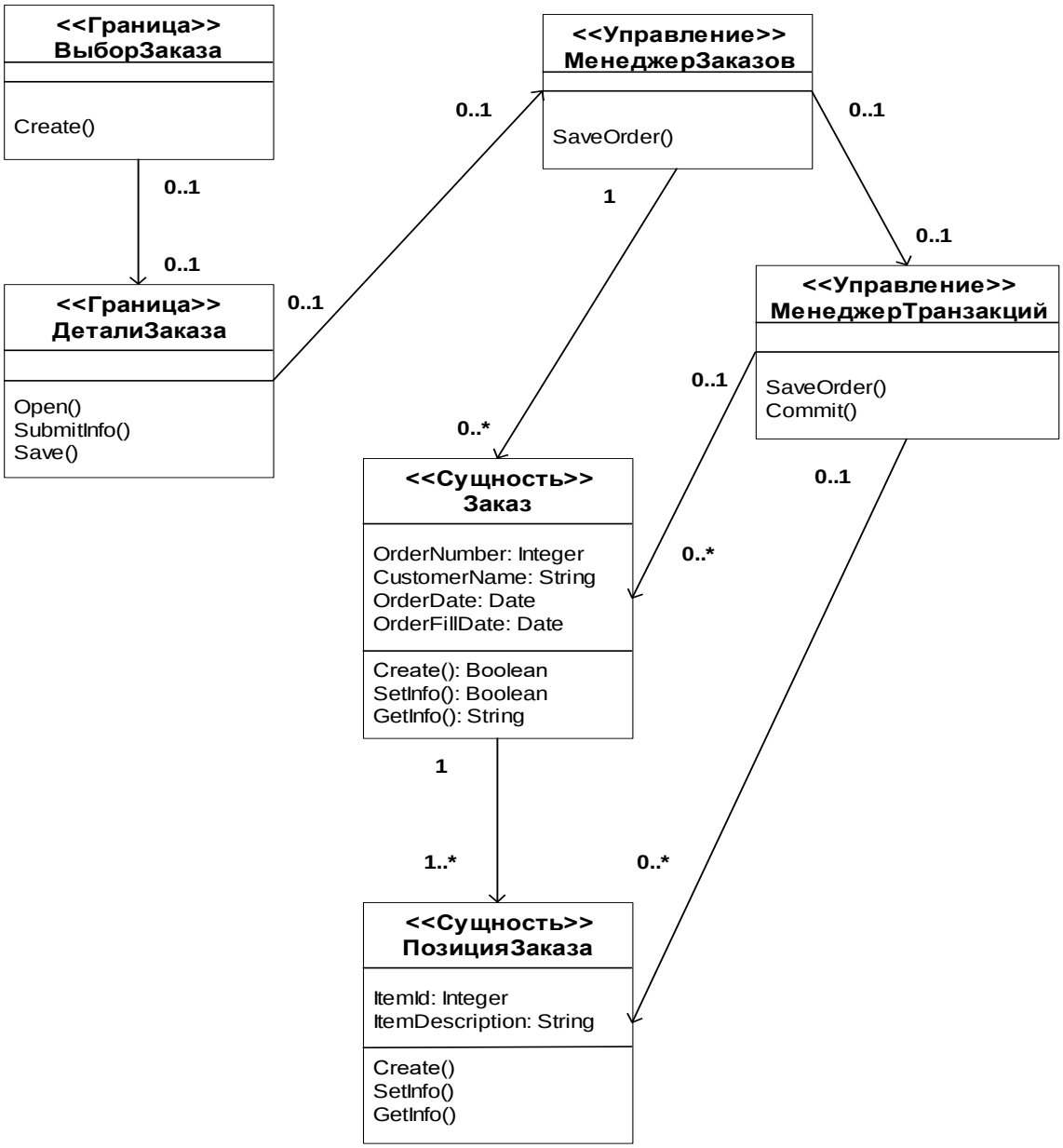


Рисунок 11 - Ассоциации сценария «Ввести новый заказ».

Лабораторная работа №11 «Создание диаграммы состояний»

Цель работы:

Создать диаграмму Состояний для класса Order.

Замечание. Проектируя класс Order, необходимо обратить внимание на его поведение. Многие требования к классу значительно изменялись при изменении состояния его экземпляра. Например, заказы, выполнение которых было приостановлено, вели себя не так, как выполненные заказы, а те, в свою очередь, не так, как отмененные заказы.

Чтобы убедиться, что проект удовлетворяет всем этим требованиям, создадим диаграмму Состояний для класса Order. С помощью этой диаграммы окончательно становиться понятно, как надо писать код для этого класса.

Создание диаграммы Состояний

Разработайте диаграмму Состояний для класса Order, показанную на рисунке 12.

Этапы выполнения лабораторной работы

1 Создание диаграммы.

1.1 Найдите в браузере класс Order.

1.2 Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт Open State Diagram.

2 Добавление начального и конечного состояний.

2.1 На панели инструментов нажмите кнопку Start State (Начальное состояние).

2.2 Поместите это состояние на диаграмму.

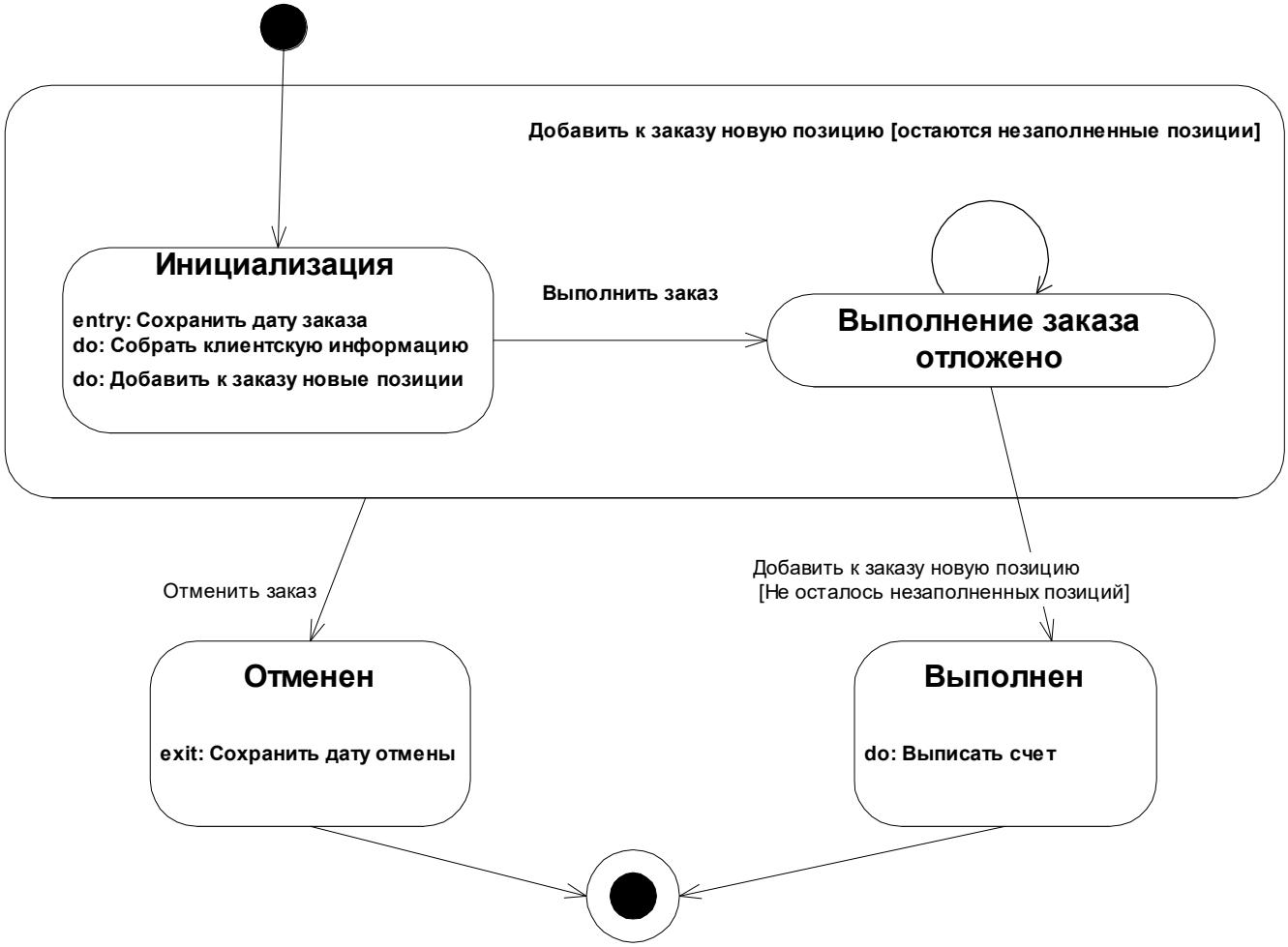
2.3 На панели инструментов нажмите кнопку End State (Конечное состояние).

2.4 Поместите это состояние на диаграмму.

3 Добавление суперсостояния.

3.1 На панели инструментов нажмите кнопку State (Состояние).

3.2 Поместите это состояние на диаграмму.



РРисунок 12 - Диаграмма Состояний для класса Order

4 Добавление оставшихся состояний.

4.1 На панели инструментов нажмите кнопку State (Состояние). Поместите это состояние на диаграмму. Назовите состояние Cancelled (Отменен).

4.2 На панели инструментов нажмите кнопку State (Состояние). Поместите это состояние на диаграмму. Назовите состояние Filled (Выполнен).

4.3 На панели инструментов нажмите кнопку State (Состояние). Поместите это состояние на диаграмму внутрь суперсостояния. Назовите состояние Initialization (Инициализация).

4.4 На панели инструментов нажмите кнопку State (Состояние). Поместите это состояние на диаграмму внутрь суперсостояния. Назовите состояние Pending (Выполнение заказа приостановлено).

5 Подробное описание состояний.

5.1 Дважды щелкните на состоянии Initialization (Инициализация).

5.2 Щелкните правой кнопкой мыши на окне Actions (Действия).

5.3 В открывшемся меню выберите пункт Insert (Вставить).

5.4 Дважды щелкните мышью на новом действии.

5.5 Назовите его Store Order Date (Сохранить дату заказа).

5.6 Убедитесь, что в окне When (Когда) указан пункт On Entry (На входе).

5.7 Повторите этапы 5.3 – 5.7, добавив следующие действия:

Collect Customer Info (Собрать клиентскую информацию), в окне When указать пункт Do

Add Order Items (Добавить к заказу новые графы), в окне When указать Do

5.8 Нажмите на кнопки ОК два раза, чтобы закрыть спецификацию.

5.9 Дважды щелкните на состоянии Cancelled (Отменен).

5.10 Повторите этапы 5.2 – 5.7, добавив действие Store Cancellation Data (Сохранить дату отмены), указать пункт On Exit (на выходе). Нажмите на кнопки ОК два раза, чтобы закрыть спецификацию.

5.11 Дважды щелкните на состоянии Filled (Выполнен).

5.12 Повторите этапы 5.2 – 5.7, добавив действие Bill Customer (Выписать счет), указать пункт Do. Нажмите на кнопки ОК два раза, чтобы закрыть спецификацию.

6 Добавление переходов.

6.1 На панели инструментов нажмите кнопку Transition (Переход).

6.2 Щелкните мышью на начальном состоянии.

6.3 Проведите линию перехода к состоянию Initialization (Инициализация).

6.4 Повторите этапы с первого по третий, создав следующие переходы:

От состояния Initialization (Инициализация) к состоянию Pending (Выполнение заказа приостановлено)

От состояния Pending (Выполнение заказа приостановлено) к состоянию Filled (Выполнен)

От суперсостояния к состоянию Cancelled (Отменен)

От состояния Cancelled (Отменен) к конечному состоянию

От состояния Filled (Выполнен) к конечному состоянию

6.5 На панели инструментов нажмите кнопку Transition to Self (Переход к себе).

6.6 Щелкните на состоянии Pending (Выполнение заказа приостановлено).

7 Подробное описание переходов.

7.1 Дважды щелкните на переходе от состояния Initialization (Инициализация) к состоянию Pending (Выполнение заказа приостановлено), открыв окно его спецификации.

7.2 В поле Event (Событие) введите фразу Finalize order (Выполнить заказ).

7.3 Щелкните на кнопке ОК, закрыв окно спецификации.

7.4 Повторите этапы с первого по третий, добавив событие Cancel Order (Отменить заказ) к переходу между суперсостоянием и состоянием Cancelled (Отменен).

7.5 Дважды щелкните на переходе от состояния Pending (Выполнение заказа приостановлено) к состоянию Filled (Выполнен), открыв окно его спецификации.

7.6 В поле Event (Событие) введите фразу Add Order Item (Добавить к заказу новую позицию).

7.7 Перейдите на вкладку Detail (Подробно).

7.8 В поле Condition (Условие) введите No unfilled items remaining (Не осталось незаполненных позиций). Щелкните на кнопке ОК, закрыв окно спецификации.

7.9 Дважды щелкните мышью на рефлексивном переходе (Transition to Self) состояния Pending (Выполнение заказа приостановлено).

7.10 В поле Event (Событие) введите фразу Add Order Item (Добавить к заказу новую позицию).

7.11 Перейдите на вкладку Detail (Подробно).

7.12 В поле Condition (Условие) введите Unfilled items remaining (Остаются незаполненные позиции). Щелкните на кнопке ОК, закрыв окно спецификации.

Лабораторная работа №12 «Создание диаграммы компонентов»

Цель работа:

Создать диаграмму Компонентов системы обработки заказов. По мере реализации других вариантов использования на диаграмму следует добавлять новые компоненты.

Замечания. Завершив анализ и проектирование системы, разработаны диаграммы Компонентов. Выбрав в качестве языка программирования C++, для каждого класса созданы соответствующие этому языку компоненты.

На рисунке 13 показана главная диаграмма Компонентов всей системы. Внимание на ней уделяется пакетам создаваемых компонентов.

На рисунке 14 показаны все компоненты пакета Entities. Эти компоненты содержат классы пакета Entities Логического представления системы.

На рисунке 15 показаны компоненты пакета Control. Они содержат классы пакета Control Логического представления системы.

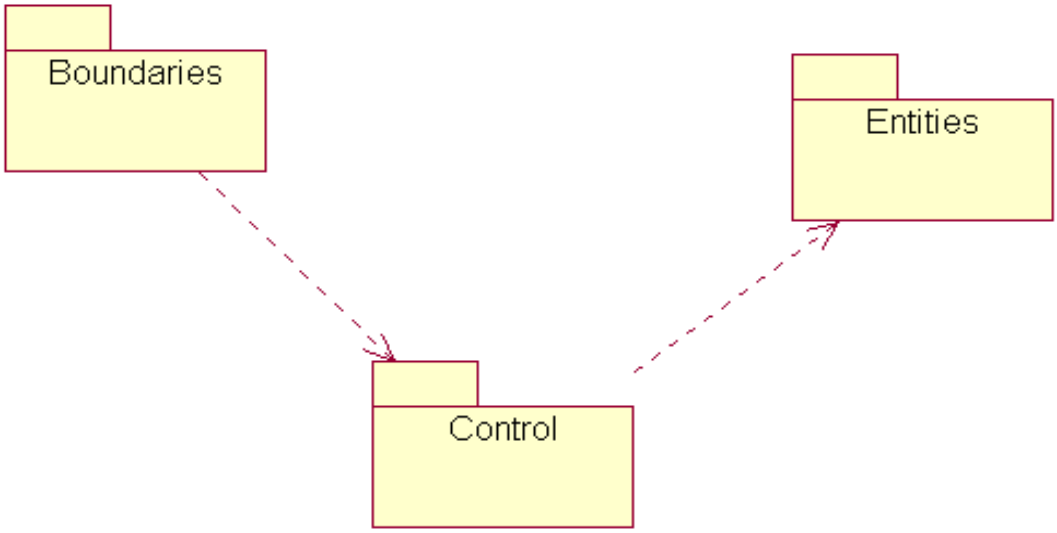


Рисунок 13 - Главная диаграмма Компонентов системы

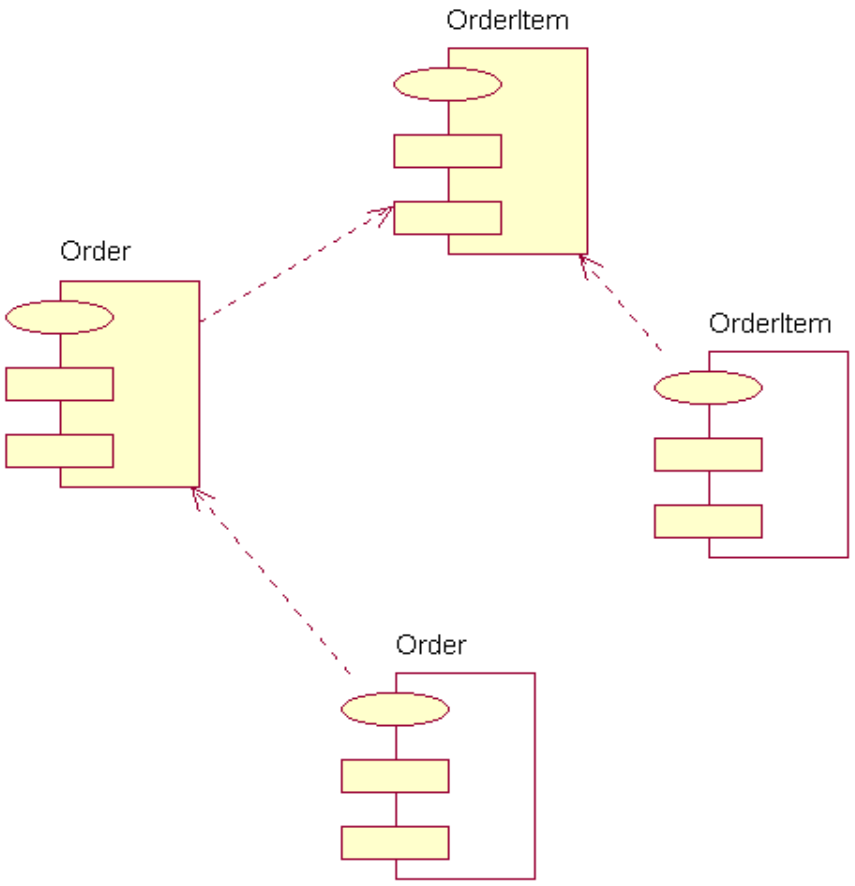


Рисунок 14 - Диаграмма Компонентов пакета Entities

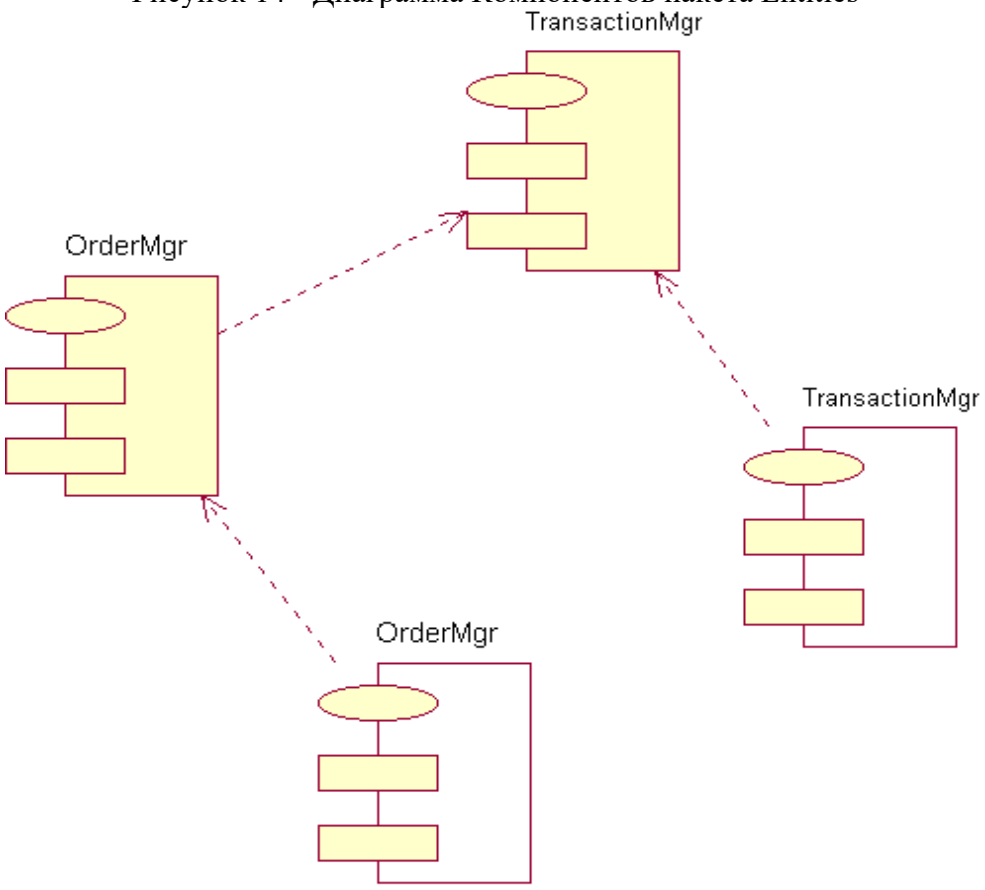


Рисунок 15 - Диаграмма Компонентов пакета Control

На рисунке 16 показаны компоненты пакета Boundaries. Они также соответствуют классам одноименного пакета Логического представления системы.

На рисунке 17 показаны все компоненты системы. Мы назвали эту диаграмму диаграммой Компонентов системы. На ней можно видеть все зависимости между всеми компонентами проектируемой системы.

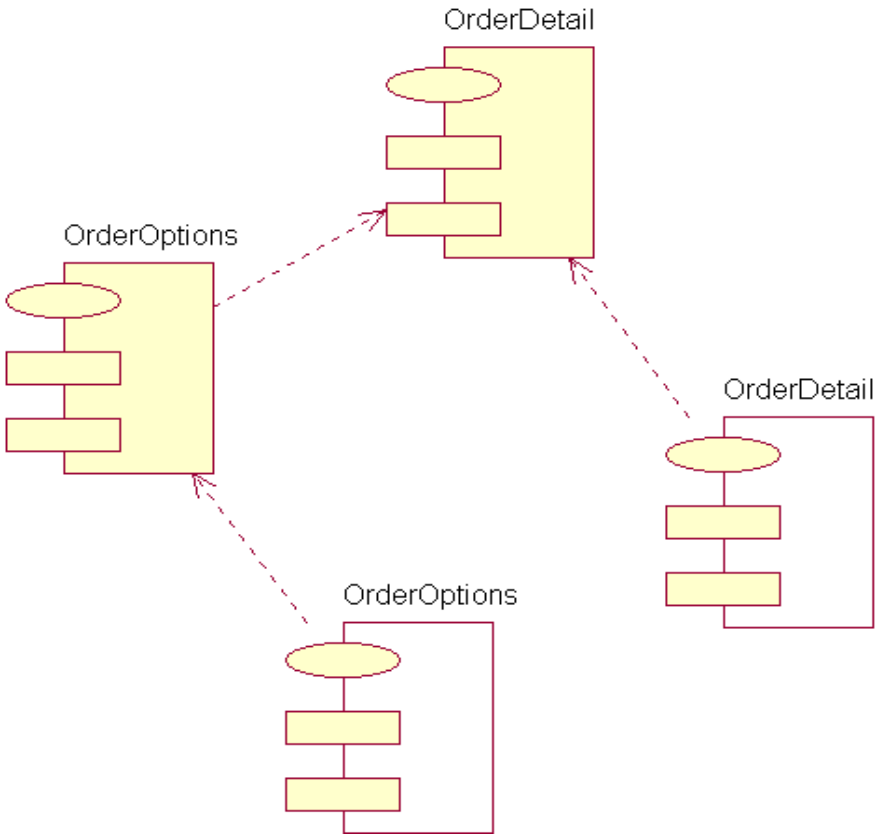


Рисунок 16 - Диаграмма Компонентов пакета Boundaries.

Этапы выполнения лабораторной работы

1 Создание пакетов компонентов.

1.1 Щелкните правой кнопкой мыши на представлении компонентов в браузере.

1.2 В открывшемся меню выберите пункт New > Package (Создать > пакет).

1.3 Назовите этот пакет Entities (Сущности).

1.4 Повторите этапы с первого по третий, создав пакеты Boundaries (Границы) и Control (Управление).

2 Добавление пакетов на Главную диаграмму Компонентов.

2.1 Откройте Главную диаграмму Компонентов, дважды щелкнув на ней.

2.2 Перетащите пакеты Entities, Boundary и Control из броузера на Главную диаграмму.

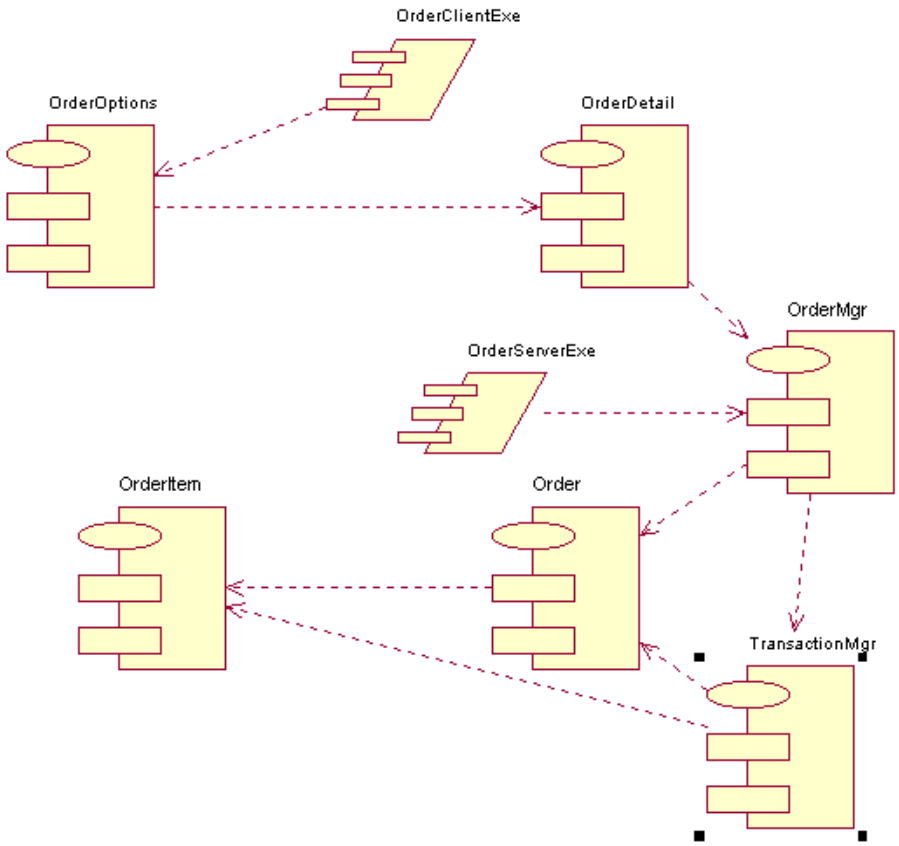


Рисунок 17 - Диаграмма Компонентов системы

3 Рисование зависимостей между пакетами.

3.1 На панели инструментов нажмите кнопку Dependency (Зависимость).

3.2 Щелкните мышью на упаковке Boundaries Главной диаграммы Компонентов.

3.2 Проведите линию зависимости до упаковки Control.

3.3 Повторите этапы 3.1 – 3.3, проведя еще зависимость от пакета Control до пакета Entities.

4 Добавление компонентов к пакетам и рисование зависимостей.

4.1 Дважды щелкните мышью на пакете Entities Главной диаграммы Компонентов, открыв Главную диаграмму Компонентов этого пакета.

4.2 На панели инструментов нажмите кнопку Package Specification (Спецификация пакета).

4.3 Поместите спецификацию пакета на диаграмму.

4.4 Введите имя спецификации пакета OrderItem.

4.5 Повторите этапы 4.2 – 4.4, добавив спецификацию пакета Order.

4.6 На панели инструментов нажмите кнопку Package Body (Тело пакета).

4.7 Поместите его на диаграмму.

4.8 Введите имя тела пакета OrderItem.

4.9 Повторите этапы 6 - 8, добавив тело пакета Order.

4.10 На панели инструментов нажмите кнопку Dependency (Зависимость).

4.11 Щелкните мышью на теле пакета OrderItem.

4.12 Проведите линию зависимости от него к спецификации пакета OrderItem.

4.13 Повторите этапы 4.10 – 4.12, добавив линию зависимости между телом пакета Order и спецификацией пакета Order.

4.14 Повторите этапы 4.10 – 4.12, добавив линию зависимости от спецификации пакета Order к спецификации пакета OrderItem.

4.15 С помощью описанного метода создайте следующие компоненты и зависимости:

Для пакета Boundaries:

Спецификацию пакета OrderOptions

Тело пакета OrderOptions

Спецификацию пакета OrderDetail
Тело пакета OrderDetail
Зависимости в пакете Boundaries:
От тела пакета OrderOptions до спецификации пакета OrderOptions
От тела пакета OrderDetail до спецификации пакета OrderDetail
От спецификации пакета OrderOptions до спецификации пакета OrderDetail

Для пакета Control:
Спецификацию пакета OrderMgr
Тело пакета OrderMgr
Спецификацию пакета TransactionMgr
Тело пакета TransactionMgr

Зависимости в пакете Control:
От тела пакета OrderMgr до спецификации пакета OrderMgr
От тела пакета TransactionMgr до спецификации пакета TransactionMgr
От спецификации пакета OrderMgr до спецификации пакета TransactionMgr

5 Создание диаграммы Компонентов системы.

5.1 Щелкните правой кнопкой мыши на представлении Компонентов в браузере.
5.2 В открывшемся меню выберите пункт New > Component Diagram
5.3 Назовите новую диаграмму System.
5.4 Дважды щелкните на этой диаграмме.

6 Размещение компонентов на диаграмме Компонентов системы.

6.1 Если это еще не было сделано, разверните в браузере пакет компонентов Entities, чтобы открыть его.
6.2 Щелкните мышью на спецификации пакета Order в пакете компонентов Entities.
6.3 Перетащите эту спецификацию на диаграмму.
6.4 Повторите этапы 2 и 3, поместив на диаграмму спецификацию пакета OrderItem.
6.5 С помощью этого метода поместите на диаграмму следующие компоненты:
Из пакета компонентов Boundaries:
Спецификацию пакета OrderOptions
Спецификацию пакета OrderDetail

Из пакета компонентов Control:
Спецификацию пакета OrderMgr
Спецификацию пакета TransactionMgr

6.4 На панели инструментов нажмите кнопку Task Specification (Спецификация задачи).
6.5 Поместите спецификацию задачи на диаграмму и назовите ее OrderClientExe.
6.6 Повторите этапы 6.6 и 6.7 для спецификации задачи OrderServerExe.

7 Добавление оставшихся зависимостей на диаграмму Компонентов системы.

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

7.1 На панели инструментов нажмите кнопку Dependency (Зависимость).
7.2 Щелкните на спецификации пакета OrderDetail.
7.3 Проведите линию зависимости к спецификации пакета OrderMgr.
7.4 Повторите этапы 7.1 – 7.3, создав следующие зависимости:
От спецификации пакета OrderMgr к спецификации пакета Order
От спецификации пакета TransactionMgr к спецификации пакета OrderItem
От спецификации пакета TransactionMgr к спецификации пакета Order
От спецификации задачи OrderClientExe к спецификации пакета OrderOptions
От спецификации задачи OrderServerExe к спецификации пакета OrderMgr

8 Соотнесение классов с компонентами.

8.1 В Логическом представлении броузера найдите класс Order пакета Entities.
8.2 Перетащите этот класс на спецификацию пакета компонента Order в представлении

Компонентов броузера. В результате класс Order будет соотнесен со спецификацией пакета компонента Order.

8.3 Перетащите класс Order на тело пакета компонента Order в представлении Компонентов броузера. В результате класс Order будет соотнесен с телом пакета компонента Order.

8.4 Повторите этапы 8.1 – 8.3, соотнеся с классами следующие компоненты:

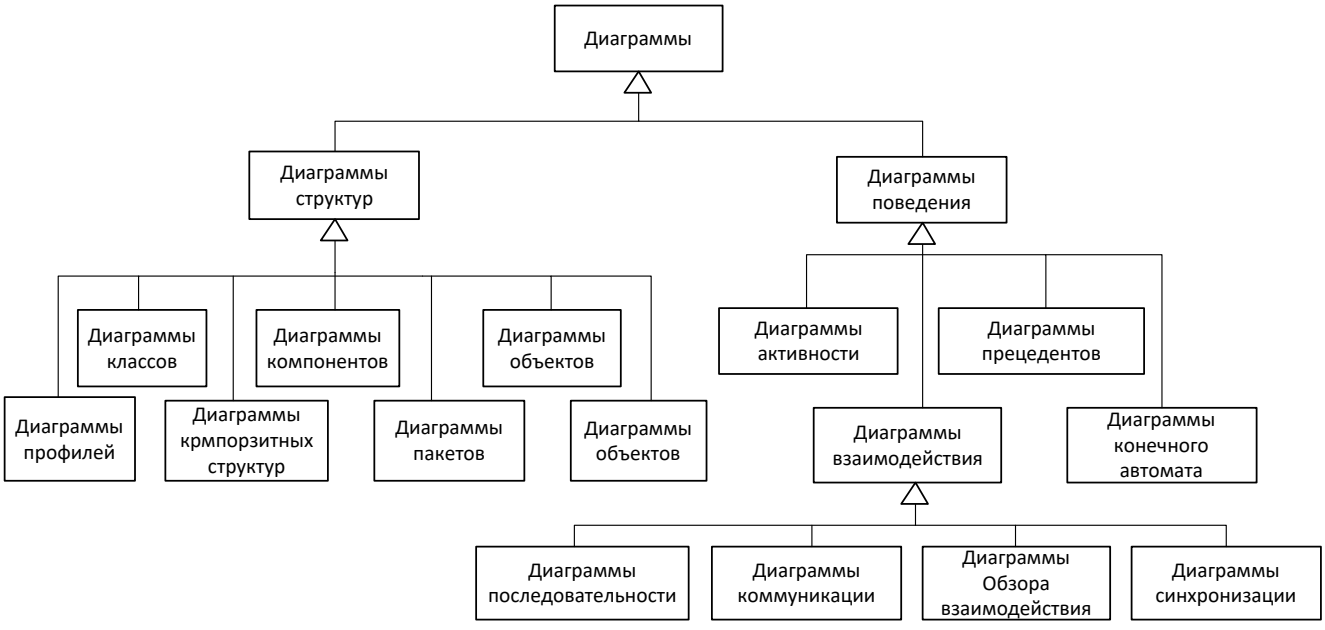
Класс OrderItem со спецификацией пакета OrderItem
Класс OrderItem с телом пакета OrderItem
Класс OrderOptions со спецификацией пакета OrderOptions
Класс OrderOptions с телом пакета OrderOptions
Класс OrderDetail со спецификацией пакета OrderDetail
Класс OrderDetail с телом пакета OrderDetail
Класс OrderMgr со спецификацией пакета OrderMgr
Класс OrderMgr с телом пакета OrderMgr
Класс TransactionMgr со спецификацией пакета TransactionMgr
Класс TransactionMgr с телом пакета TransactionMgr

Лабораторная работа 13. Создание UML диаграмм в Microsoft Visio 2010

UML (Unified Modeling Language) - унифицированный язык моделирования – это язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML моделью*. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Существует 13 официальных диаграмм UML 2.0, каждая из которых представляет собой различное представление разных аспектов системы:

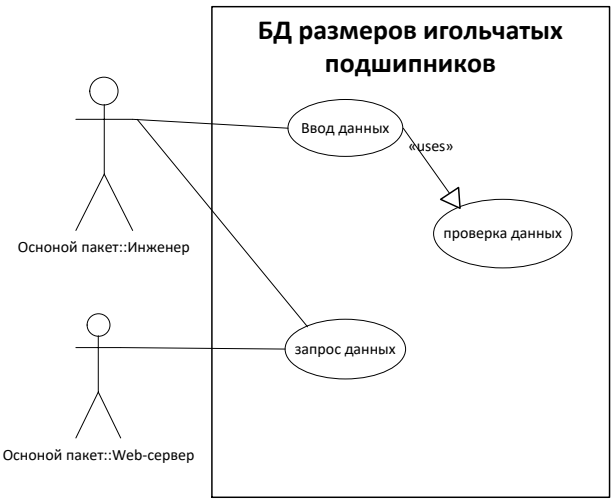


Рассмотрим более подробно некоторые виды диаграмм.

Диаграммы прецедентов.

Диаграмма прецедентов (Диаграмма вариантов использования, Use case diagram) — диаграмма, на которой отражены отношения, существующие между участниками и вариантами использования.

Основная задача — представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.
Каждый прецедент(use case) характеризует определенный тип использования системы участником; у каждого прецедента есть имя, и он может иметь текстовое пояснение.
Участник (актёр, actor) - представляет кого угодно (что угодно) не относящегося к системе, это тот, кто использует систему и реагирует на её действия (человек, железо, время, другая система); единственное действие участника это приём и передача информации системе.
Ограничения (boundary, граница системы) - это классификатор (система/подсистема/класс), функциональность которого мы описываем с помощью прецедентов; ограничения определяют границы системы/подсистемы.
Отношения(relationships) - ассоциация между участником и прецедентом показывает направление информации между внешним элементом и прецедентом.
Если зависимость между прецедентами содержит стереотип **<<включает>> (include), значит, первый прецедент включает действия второго (в Visio используется стереотип <<uses>>). Так же присутствуют связи <<расширяет>> - при выполнении расширяемого) прецедента *(стрелка указывает на него)* выполнения расширителя не обязательно, но возможно, <<обобщение>> (аналогично наследованию классов). Так же прецеденты могут быть соединены ассоциациями(линия или стрелка), имя которых должно раскрывать суть взаимодействия прецедентов.**



В выше приведенном примере рассмотрена система – база данных размеров игольчатых подшипников. С данной системой работают 2 участника: инженер и Web-сервер (участник не обязательно человек, а любая другая система). Инженер может совершить 2 действия с системой: ввести новые данные, при вводе которых система выполнит их проверку, либо запросить данные.

Web-сервер может выполнить только запрос на информацию для отображения ее на сайте каталога игольчатых подшипников.

Диаграммы последовательности.

Диаграммы последовательности (sequence diagram) отображают динамику взаимодействия объектов во времени. Объекты на диаграмме располагаются слева направо. Время идет сверху вниз.

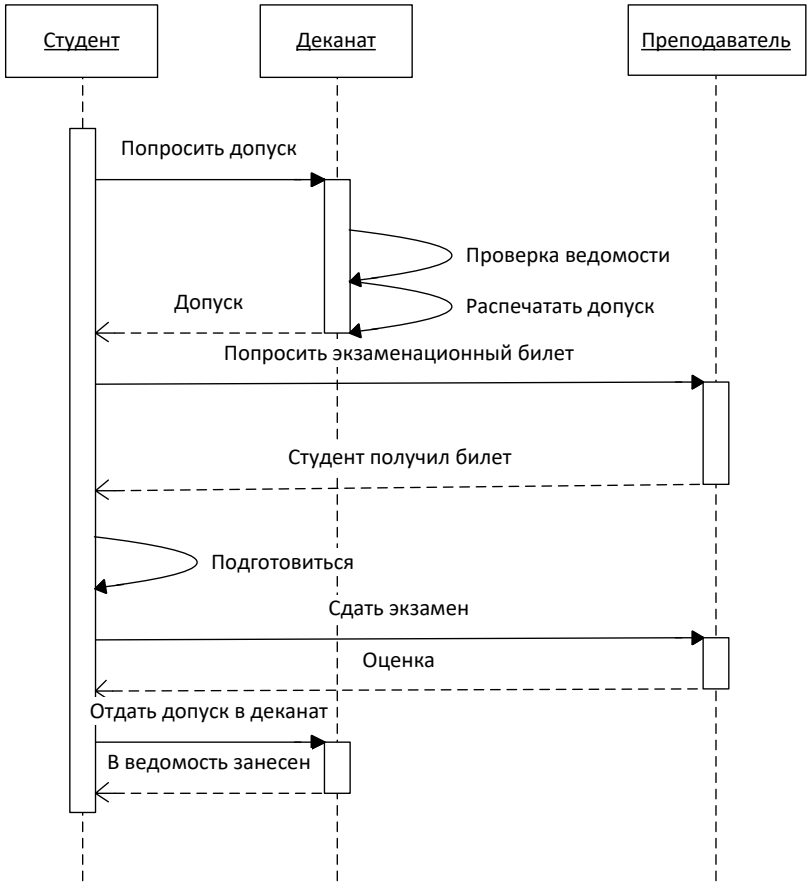
Основные элементы диаграммы последовательности.

Линия жизни объекта (object lifeline) изображается 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.

Фокус управления(focus of control).. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название фокуса управления. Фокус управления изображается в форме вытянутого узкого прямоугольника, верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а его нижняя сторона - окончание фокуса управления (окончание активности). Прямоугольник располагается ниже обозначения соответствующего объекта и может заменять его линию жизни, если на всем ее протяжении он является активным.
Сообщения. В UML каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой. Сообщение представляет собой законченный фрагмент информации, который отправляется одним объектом другому. Прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено. В языке UML различаются несколько разновидностей сообщений, каждое из которых имеет свое графическое изображение:

- первая разновидность сообщения является наиболее распространенной и используется для вызова процедур, выполнения операций или обозначения отдельных вложенных потоков управления. Начало этой стрелки всегда соприкасается с фокусом управления или линией жизни того объекта-клиента, который инициирует это сообщение. Конец стрелки соприкасается с линией жизни того объекта, который принимает это сообщение и выполняет в ответ определенные действия. Принимающий объект, как правило, получает фокус управления, становясь активным;
- вторая разновидность сообщения используется для обозначения простого потока управления. Каждая такая стрелка указывает на выполнение одного шага потока. Такие сообщения, обычно, являются асинхронными, то есть могут возникать в произвольные моменты времени. Передача такого сообщения, как правило, сопровождается получением фокуса управления принявшим его объектом;
- третья разновидность явно обозначает асинхронное сообщение между двумя объектами в некоторой процедурной последовательности. Примером такого сообщения может служить прерывание операции при возникновении исключительной ситуации. В этом случае информация о такой ситуации передается вызывающему объекту для продолжения процесса дальнейшего взаимодействия;
- четвертая разновидность сообщения используется для возврата из вызова процедуры. Примером может служить простое сообщение о завершении некоторых вычислений без предоставления результата расчетов объекту-клиенту. В процедурных потоках управления эта стрелка может быть опущена, поскольку ее наличие неявно предполагается в конце активизации объекта. Считается, что каждый вызов процедуры имеет свою пару - возврат вызова. Для непроцедурных потоков управления, включая параллельные и асинхронные сообщения, стрелка возврата должна указываться явным образом.

Пример:
Студент просит допуск в деканате. Деканат проверяет по ведомостям, допущен ли студент, распечатывает ему допуск. С допуском студент идет на экзамен, получает у преподавателя экзаменационный билет, садится и готовится. После подготовки рассказывает свой билет, получает оценку и относит допуск в деканат. Деканат вносит изменения в ведомости.
Обратите внимание, что диаграмма последовательности охватывает только один сценарий выполнения. В данном сценарии студент допущен до экзамена и сдаст его.



Пример диаграммы последовательности.

Диаграммы классов.

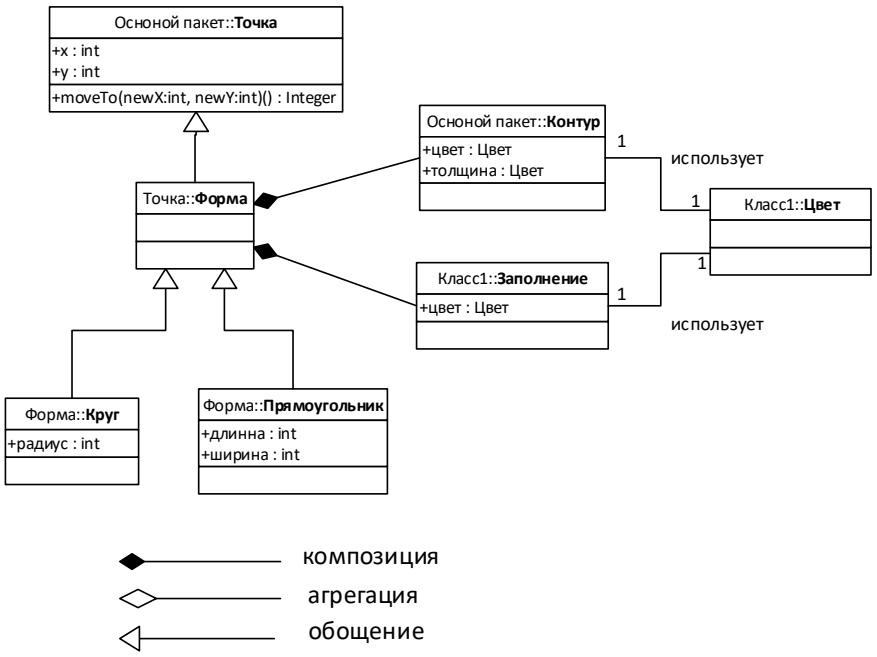
Диаграмма классов (class diagramm) описывает статическую структуру системы. На ней отображены классы, их методы и атрибуты, а так же связи между ними.

Основной элемент диаграммы классов – это класс. Он обозначается прямоугольником, горизонтально разделенным на 3 части. В верхней записывается имя класса, в центральной атрибуты, в нижней - методы. Если требуется изобразить интерфейсный класс, то он будет содержать только 2 поля – для имени и методов, так как у такого класса нет атрибутов.

Классы могут находится в следующих типах отношений:

- Ассоциация показывает как объекты одного класса связаны с объектами другого. Обозначаются линиями, идущими от одного класса к другому. Самыми распространенными являются однонаправленные и двунаправленные ассоциации. Иногда концах линий пишут мультипликаторы, чтобы показать количество объектов, участвующих в ассоциации(1 .. 1, 1 .. n, итд).
- Агрегация — это разновидность ассоциации при отношении между целым и его частями. Как тип ассоциации агрегация может быть именованной. Агрегация встречается, когда один класс является коллекцией или контейнером других. Если контейнер будет уничтожен, то его содержимое — нет.
- Композиция — более строгий вариант агрегации. Известна также как агрегация по значению. Композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет также уничтожено.

- Обобщение на диаграммах классов используется, чтобы показать связь между классом-родителем и классом-потомком. Оно вводится на диаграмму, когда возникает разновидность какого-либо класса (например: животное - рептилия), а также в тех случаях, когда в системе обнаруживаются несколько классов, обладающих сходным поведением



Пример диаграммы классов.

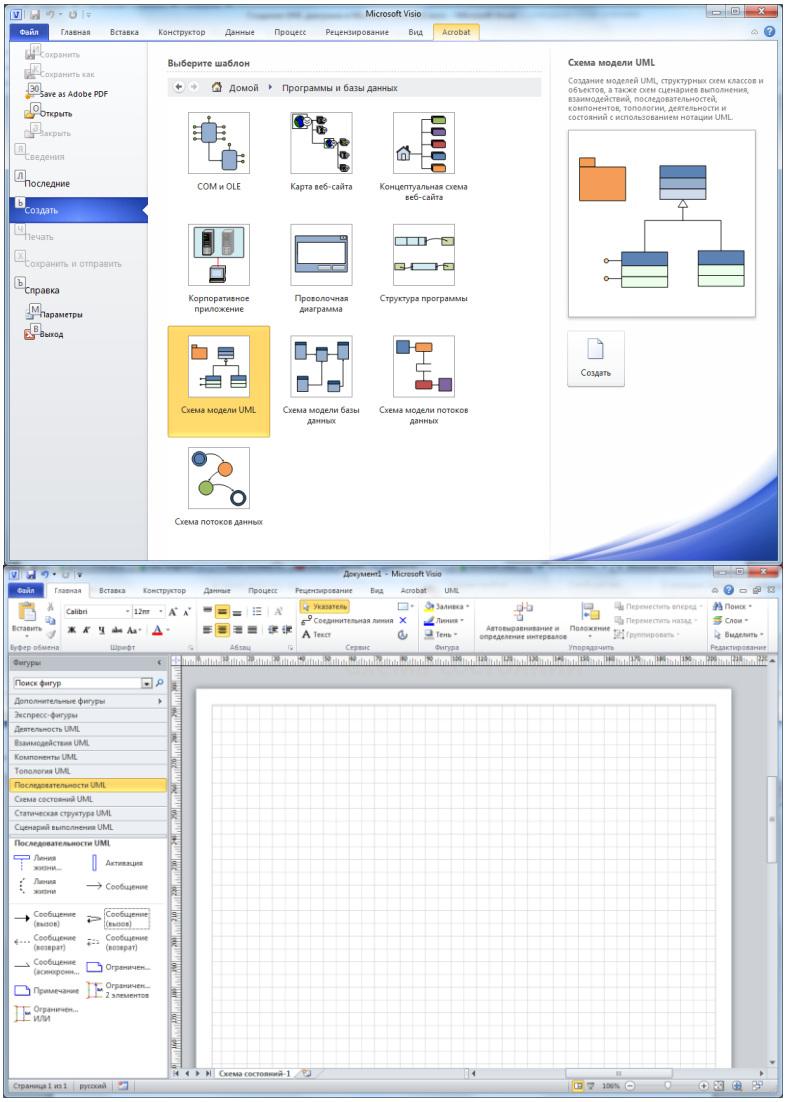
Базовый класс - «Точка». Его атрибуты и методы наследуются через класс «Форма» классам «Круг» и «Прямоугольник». Класс «Форма» является классом агрегатом для классов «Контур» и «Заполнение». Последние использую класс «Цвет» для хранения цвета.

Microsoft Visio 2010

Microsoft Visio — векторный графический редактор, редактор диаграмм и блок-, /предлагает богатые средства визуализации для наглядного представления любой информации, комплексных систем и многоступенчатых процессов.

Рассмотрим основные шаги по построению диаграмм на примере диаграммы прецедентов из изложенного ранее примера.

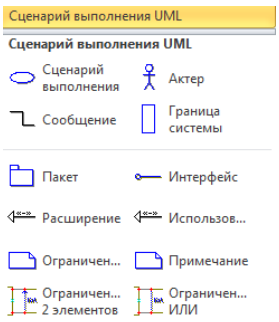
1) В меню «Файл» выберите «Создать». На экране отобразится большое количество категорий шаблонов, доступных в Visio. Выбор нужных шаблонов позволяет сократить время на настройку среды под требуемый тип диаграмм. Выберите категорию «Программы и базы данных» и в ней – шаблон «Схема модели UML»



Интерфейс Visio 2010

Visio 2010 обладает интуитивно интерфейсом, выполненным в стиле Ribbon. В верхней части экрана расположена лента инструментов, которые расположены на соответствующих вкладках. В левой части экрана находится панель компонентов, которые можно использовать в своих работах. В Visio имеется большая библиотека объектов, для различных типов диаграмм, схем таблиц. Так же присутствует поддержка сторонних объектов, созданных пользователями. Так как выбран был шаблон UML, то слева располагаются вкладки с элементами UML диаграмм.

2) В начале необходимо обозначить границы системы. Этот элемент, как и все остальные, необходимые для построения диаграммы прецедентов расположены во вкладке «Сценарий выполнения UML»



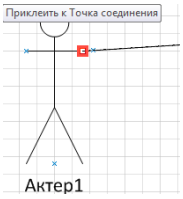
Необходимо перетащить объект «Граница системы» на область рисования, и отмасштабируйте его. Так же нужно изменить его имя с «Система» на «База данных размеров игольчатых подшипников»

3) Далее надо добавить остальные объекты: три сценария выполнения и двух актеров и расположить их на диаграмме. Не забудьте дать им соответствующие имена.
4) Настало время связей. Перетащите объект «Сообщение» и нажмите на него два раза.

Окончания ассоциаций:

Имя окончания	Агрегат	Видимость	Кратность	IsNavigable
Конец1	нет	private	*	☐
Конец2	нет	private	*	☐

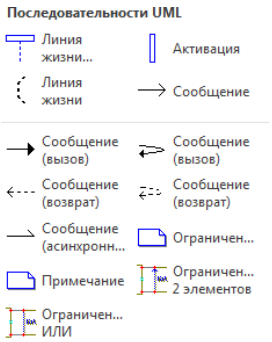
В его свойствах необходимо стереть имена окончаний и кратность. Таким образом, получилась чистая линия. Потяните за один из концов линии и подведите ее к синему крестику на любом их актеров. Как только точка соединения станет красной, отпустите мышь. Теперь этот конец линии соединен с актером, и будет перемещаться вместе с ним. Соедините другой конец линии с прецедентом.



Для того чтобы сделать связь со стереотипом «uses» используйте инструмент «использование», работа с ним проходит так же как с обычной связью.
После того как все объекты соединены диаграмма готова.
Для построения диаграмм классов и последовательностей используется тот же принцип соединения определенными связями базовых конструкций. Ниже описаны основные из них:

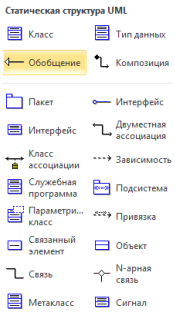
1) Диаграмма последовательности. Для этой диаграммы объекты берется из категории «Последовательности UML»:

- Линия жизни. Содержит имя объекта, и пунктирную линию, обозначающую продолжительность жизни объекта в система. Линия содержит точки, к которым крепятся стрелки и фокусы управления (активности).
- Активация. Располагается на линии жизни, отображая активность объекта.



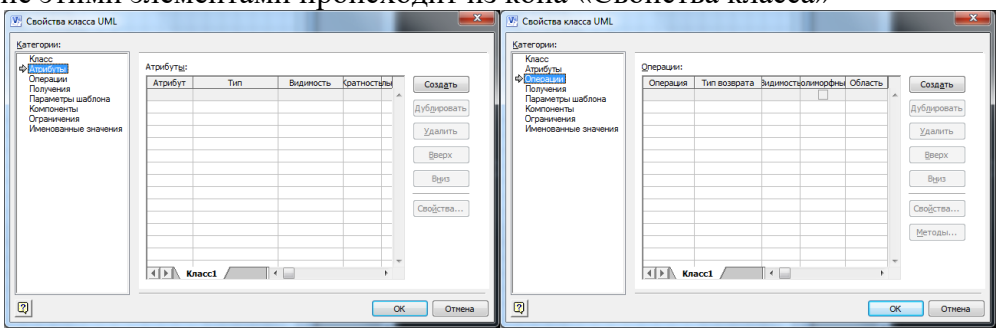
- Сообщения (вызов). Используются для отправки сообщений другому объекту, или самому себе. Имеют 2 точки, которыми можно манипулировать для задания положения стрелки – начало и конец.
- Сообщения (возврат). Возвращают управления вызывающему классу. Управляются так же, как и другие стрелки.

2) Диаграмма классов. Инструменты для диаграммы классов находятся во вкладке «Статические структуры UML»



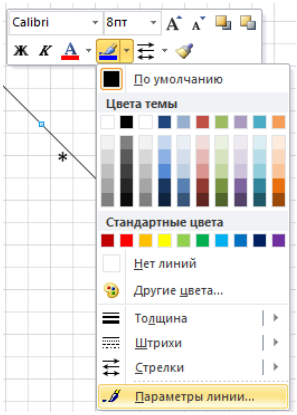
Основные компоненты:

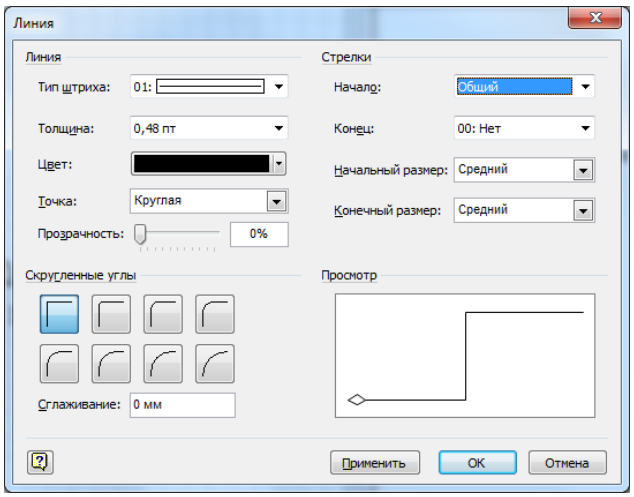
- Класс. Содержит 3 поля: для имени класса, для атрибутов класса и для методов класса. Управление этими элементами происходит из кона «Свойства класса»



Во вкладке «атрибуты» можно задать имена атрибутов, спецификатор доступа (видимость) и тип, который можно выбрать из базовых и из ранее созданных классов. Аналогично для методов класса во вкладке «операции».

- Стрелка «Обобщение». Соединяет класс родитель и класс наследника.
- Стрелка «Композиция». Соединяет целое и его части. При необходимости можно задать в свойствах мультипликатор (кратность). Для получения из композиции агрегации, необходимо зайти в параметры линии стрелки и изменить фигуру начала с «общий» на «составной».





Если поставить тип начала «нет», то данные стрелки можно использовать для простых ассоциаций.

После завершения диаграммы, ее можно сохранить в формате Visio или в виде растового изображения, скопировать в любой графический редактор, или в приложение MC Office. В последнем, будет возможность обратного переноса диаграммы в Visio, при необходимости доработки.

Лабораторная работа 14. Изучение методов построения графических моделей средствами Visual Studio 2010 Ultimate

Цель

Изучить средства Visual Studio Ultimate для построения основных диаграмм UML и познакомиться с их нотацией.

Общие сведения о средствах Visual Studio для моделирования UML

Visual Studio Ultimate предоставляет шаблоны для пяти часто используемых UML-диаграмм:

- вариантов использования,
- активности,
- классов,
- компонентов,
- последовательностей.

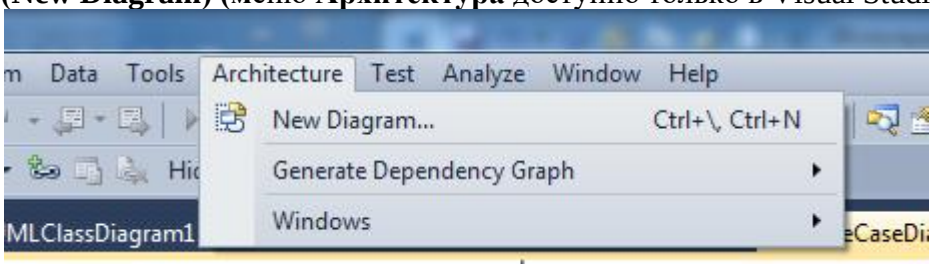
Кроме того, можно создавать схемы слоев, которые помогают определить структуру системы. UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

Все проекты моделирования содержат общую UML-модель и несколько UML-диаграмм. Каждая диаграмма является представлением части модели. UML-модель содержит все элементы, отображаемые на UML-схемах, и может просматриваться с помощью обозревателя моделей UML.

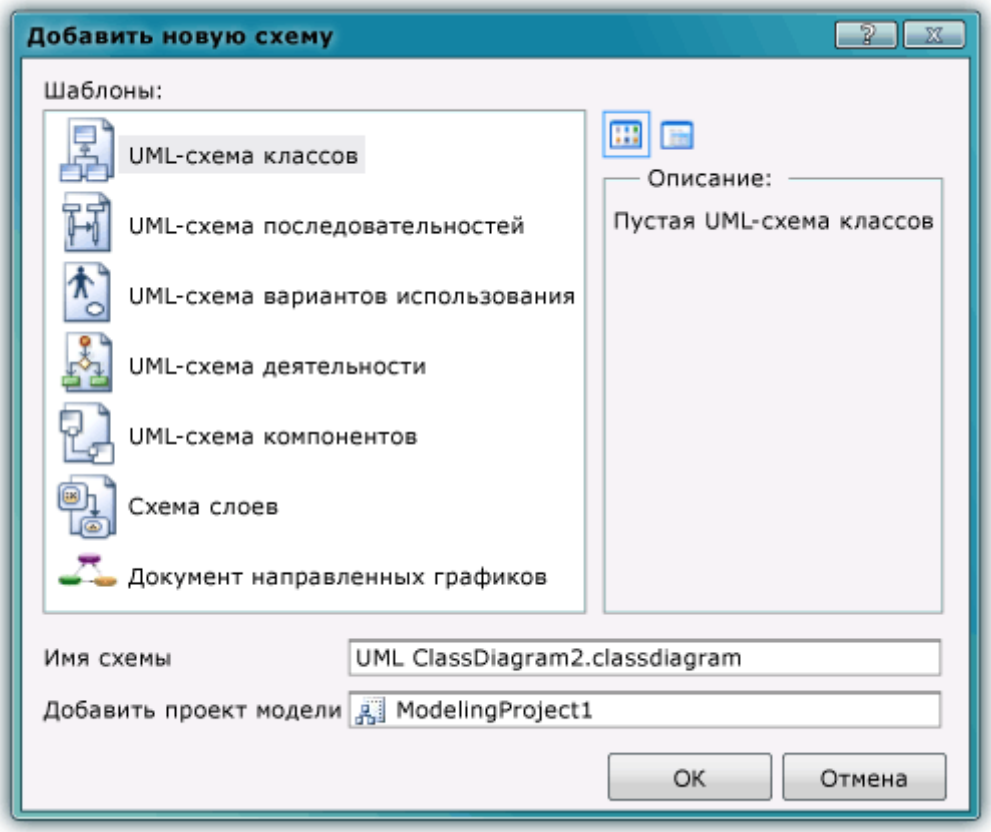
Создание диаграмм в проекте моделирования

Создание диаграммы и добавление ее в проект

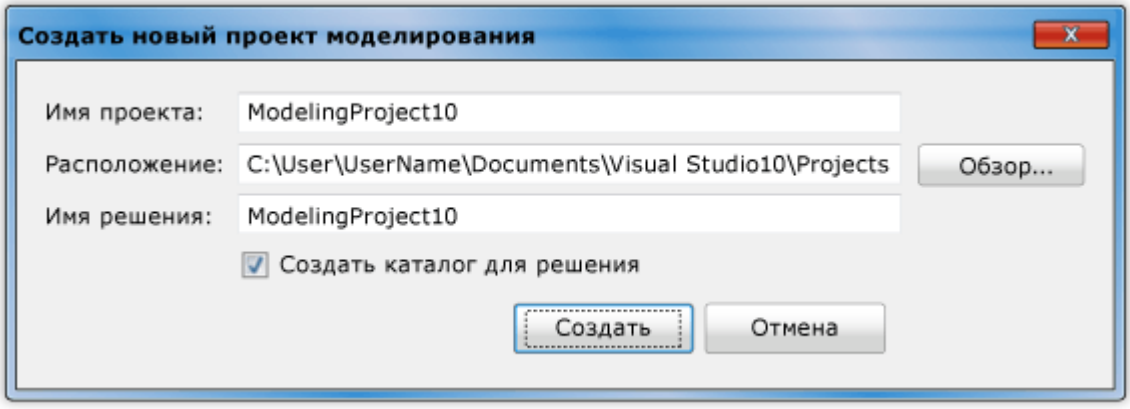
1. Запустите Visual Studio Ultimate и в меню **Архитектура (Architecture)** выберите команду **Создать схему (New Diagram) (м**еню **Архитектура** доступно только в Visual Studio Ultimate):



2. В диалоговом окне **Добавить новую схему** (**Add New Diagram**) выберите требуемый тип диаграммы моделирования:



3. Введите имя новой схемы.
4. В окне **Добавить проект модели** выполните одно из двух действий:
   - с помощью выпадающего списка выберите проект моделирования, который уже существует в решении, и нажмите кнопку **ОК**.
   - Выберите **Создать новый проект моделирования (Create a new modeling project)** и нажмите кнопку **ОК**.
     - В диалоговом окне **Создание нового проекта моделирования** (рис. П 1.3) введите имя и расположение нового проекта, затем нажмите кнопку **Создать**.



Если решение открыто, новый проект добавляется в решение.
Если решение не открыто, можно ввести имя нового решения.
Если проект моделирования уже есть, можно использовать следующую процедуру.

Добавление схемы в существующий проект моделирования

1. В **обозревателе решений** щелкните узел проекта моделирования.

Проект моделирования содержит папку определения модели **ModelDefinition**.

2. В меню **Проект** выберите команду **Добавить новый элемент**.
3. В диалоговом окне **Добавление нового элемента - <*имя проекта*>** в разделе **Шаблоны** щелкните тип схемы моделирования, например, **Схема компонентов UML**.
4. Введите имя схемы и нажмите кнопку **Добавить**.

Открытая схема моделирования отображается в проекте моделирования.

Внимание

Не следует добавлять, копировать или перетаскивать файлы существующей схемы слоев в другой проект моделирования или в другие местоположения в решении. Это приведет к исчезновению элементов из скопированных схем или к ошибкам при открытии схем. Файл схемы должен открываться в том проекте моделирования, в котором он создан. Это связано с тем, что UML-схема является представлением модели, принадлежащей проекту моделирования. Чтобы скопировать файл схемы, создайте новую схему, а затем скопируйте элементы из исходной схемы в новую схему

Создание пустого проекта моделирования

1. В меню **Файл** последовательно выберите пункты **Создать** и **Проект**.
2. В диалоговом окне **Создать проект** в разделе **Установленные шаблоны** щелкните пункт **Проекты моделирования**.
3. В среднем окне щелкните **Проект моделирования**.
4. Назовите проект и укажите расположение в полях **Имя** и **Расположение**.
5. В поле **Решение** выберите **Добавить в решение**, чтобы добавить новый проект в открытое решение, или **Создать новое решение**, чтобы закрыть открытые решения и добавить проект в новое решение.

Удаление схемы моделирования из проекта

Можно полностью удалить схему или временно исключить ее из проекта, а затем восстановить.

Полное удаление схемы из проекта

- В **обозревателе решений** щелкните правой кнопкой мыши основной файл, представляющий схему и нажмите кнопку **Удалить**.

Схема удаляется из проекта и файловой системы. Элементы, отображаемые на схеме, не удаляются из **Обозревателя моделей UML**.

Примечание

Все схемы содержат два файла, один из которых является дочерним по отношению к другому. Например, если имеется схема компонентов с именем CD1, следует удалить файл с именем CD1.componentdiagram. Его дочерний файл с именем CD1.componentdiagram.layout будут удален автоматически.

Временное исключение схемы из проекта

- В **обозревателе решений** щелкните файл схемы правой кнопкой мыши и выберите **Исключить из проекта**.

Схема удаляется из проекта. Она не удаляется из файловой системы.

Примечание

Элементы, отображаемые на схеме, не удаляются из **Обозревателя моделей UML**.

Восстановление временно исключенной из проекта схемы

1. В **обозревателе решений** щелкните узел проекта моделирования.

Примечание

Проект моделирования содержит папку определения модели **ModelDefinition**.

2. В меню **Проект** выберите команду **Добавить существующий элемент**.
3. В диалоговом окне **Добавление существующего элемента** найдите файл схемы, выберите его и нажмите кнопку **Добавить**.

Открытая схема моделирования отображается в проекте моделирования.

Примечание

С каждой схемой связана пара файлов в файловой системе. Не выбирайте файл с расширением .layout. Кроме того, Visual Studio Ultimate не поддерживает добавление существующих UML-схем к нескольким проектам моделирования. Все файлы схем должны открываться в проекте моделирования, в котором они созданы. Это связано с тем, что UML-схема показывает представление модели, принадлежащее проекту моделирования.

Представление диаграмм согласно Visual Studio Ultimate

Диаграмма вариантов использования (прецедентов)

В Visual Studio Ultimate в *диаграмме вариантов использования* обобщены сведения о том, кто использует приложение или систему, и какие действия с этим приложением или системой они могут выполнять.

Схема вариантов использования является основным инструментом, используемым для описания пользовательских требований. Она описывает отношения между требованиями, пользователями и основными компонентами. Однако схема вариантов использования не описывает требования подробно, их можно представить на отдельных схемах или в документах, которые можно связать с каждым вариантом использования.

Чтение схем вариантов использования

В следующих таблицах описаны элементы, которые можно использовать на схеме вариантов использования, и их основные свойства.

Субъекты, варианты использования и подсистемы

Фигура	Элемент	Описание и основные свойства
1	**Субъект**	Представляет пользователя, организацию или внешнюю систему, взаимодействующую с используемым приложением или системой. Субъект — это вид типа. • **Путь к рисунку** — путь к файлу изображения, который следует использовать вместо значка субъекта по умолчанию. Значок должен быть файлом ресурсов в проекте Visual Studio.

Фигура	Элемент	Описание и основные свойства
2	**Вариант использования**	Представляет действия, выполненные одним или несколькими субъектами для достижения конкретной цели. Вариант использования — это вид типа. • **Предметы** — подсистема, в которой отображается вариант использования.
3	**Ассоциация**	Указывает, что субъект принимает участие в варианте использования.
4	**Подсистема или компонент**	Система или приложение, с которым ведется работа, либо часть системы или приложения. Может представлять собой что угодно — от крупной сети до одного класса в приложении. Варианты использования, поддерживаемые системой или компонентом, отображаются внутри прямоугольника. Чтобы более ясно очертить область действия системы, рекомендуется показать некоторые варианты использования за пределами прямоугольника. Подсистема на схеме вариантов использования, по сути, имеет тот же тип, что и компонент на схеме компонентов. • **Является неявно создаваемым экземпляром** — если значение false, выполняющая система имеет один или несколько объектов, напрямую соответствующих этой подсистеме. Если значение true, подсистема является конструкцией в проектируемой системе, которая отображается в выполняющей системе только при создании экземпляров ее составных частей.

Структурирование вариантов использования

Фигура	Элемент	Описание
5	**Включение**	Включающий вариант использования вызывает включенный. Включение используется, чтобы показать, как разбить вариант использования на несколько более мелких шагов. Включенный вариант использования находится на

Фигура	**Элемент**	Описание
		окончании с наконечником стрелки. Обратите внимание, что на схеме не показана последовательность шагов. Для подробного описания этих шагов можно воспользоваться схемой деятельности, схемой последовательностей или другим документом.
6	**Расширение**	Расширяющий вариант использования добавляет цели и шаги в расширяемый вариант использования. Расширения работают только при определенных условиях. Расширенный вариант использования находится на окончании с наконечником стрелки. Обратите внимание, что на схеме не показаны конкретные условия, при которых применяются расширения: их можно записать в комментарии или другом документе.
7	**Наследование**	Устанавливает отношение между специализированным и обобщенным элементом. Обобщенный элемент находится на окончании с наконечником стрелки. Специализированный вариант использования наследует цели и субъекты своего обобщения и может добавлять более конкретные цели и шаги для их достижения. Специализированный субъект наследует варианты использования, атрибуты и ассоциации своего обобщения и может добавлять дополнительные объекты.
8	**Зависимость**	Указывает, что конструкция источника зависит от конструкции целевого объекта.
9	**Комментарий**	Используется для добавления общих примечаний на схеме.
10	**Артефакт**	Артефакт предоставляет ссылку на другую схему или документ. Его можно создать, перетащив файл из Обозревателя решений. С помощью инструмента "Зависимость" артефакт можно связать с любым другим элементом на схеме. Как правило, артефакт используется для связи варианта использования со схемой последовательностей, страницей OneNote, документом Word или презентацией PowerPoint, которая подробно его описывает. Документ может либо представлять собой элемент в решении Visual Studio, либо документ в расположении с общим доступом, например на сайте SharePoint. • **Гиперссылка**. URL-адрес или путь к файлу схемы или документа. Дважды щелкните артефакт, чтобы открыть файл или веб-страницу, с которой он связан.
11 (не показана)	**Пакеты**	Варианты использования, субъекты и подсистемы могут содержаться внутри пакетов. Фигуры пакетов не отображаются на схеме, но можно задать свойство схемы **LinkedPackage**. Элементы, которые впоследствии будут созданы на схеме, помещаются в этот пакет.

Диаграмма последовательностей

В Visual Studio Ultimate *схема последовательностей* показывает взаимодействие, которое представляет последовательность сообщений между экземплярами классов, компонентами, подсистемами и субъектами. Время увеличивается вниз по диаграмме, на которой показывается переход управления от одного участника к другому. Чтобы создать UML-схему последовательностей, в меню **Архитектура** щелкните **Создать схему**.

На рисунке показан пример экземпляров и событий вместо классов и методов. На рисунке могут появляться несколько экземпляров одного и того же типа, а также несколько вхождений одного сообщения.

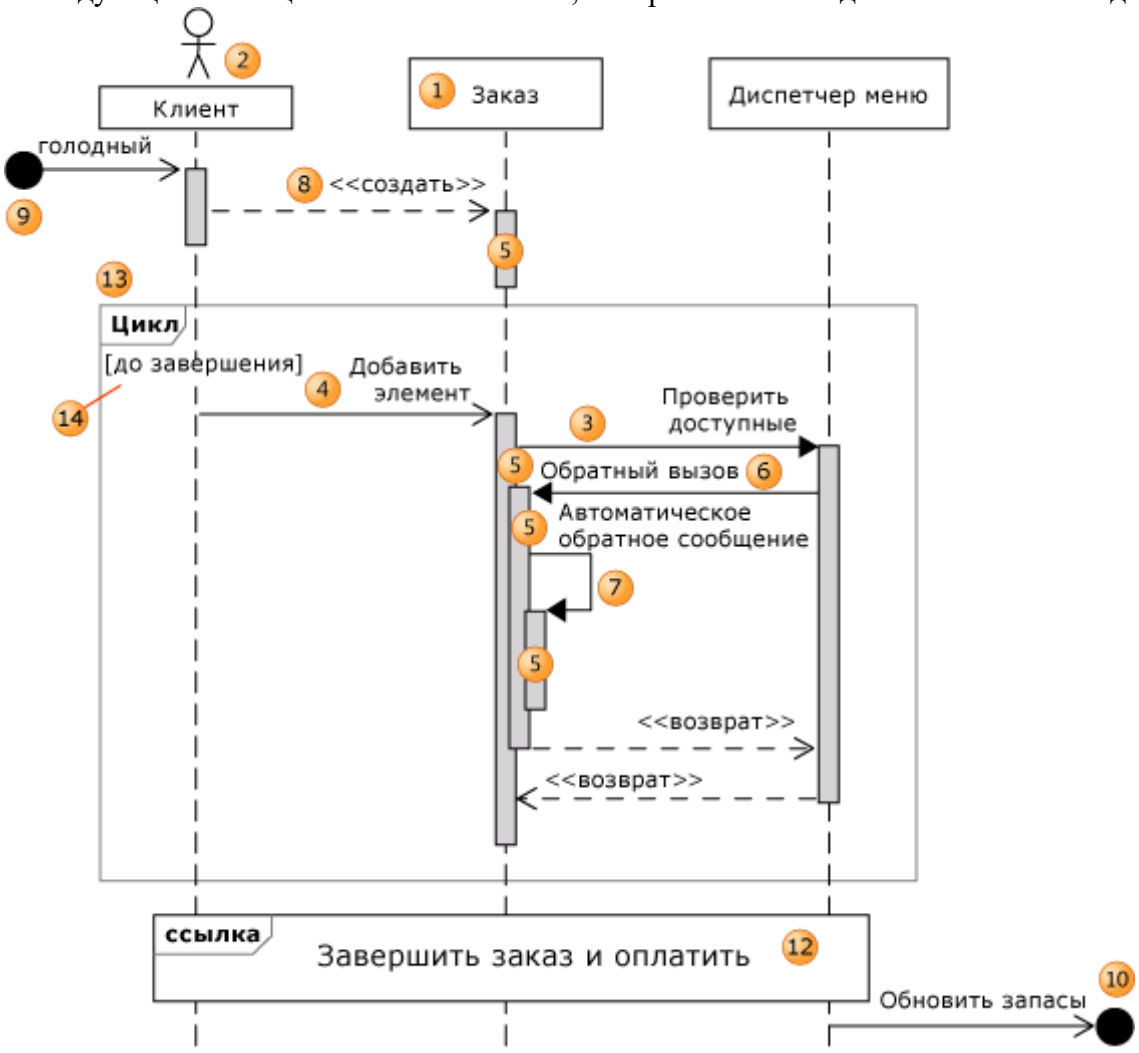
Существует два вида схем последовательностей:

- **Схемы последовательностей, основанные на коде** могут быть созданы из кода программы .NET и помещены в любой проект.
- **Схемы последовательностей UML** образуют часть модели UML и существуют только в пределах UML-проекта моделирования.

Два вида схем последовательностей похожи, хотя некоторые из свойств их элементов различаются.

Чтение схем последовательностей

В следующей таблице описаны элементы, которые можно видеть на схеме последовательностей.



Фигура	Элемент	Описание
1	**Линия жизни**	Вертикальная линия, которая представляет последовательность событий, происходящих в участнике во время взаимодействия,

Фигура	Элемент	Описание
		когда время направлено вниз по этой линии. Этот участник может быть экземпляром класса, компонента или субъекта.
2	**Субъект**	Участник, являющийся внешним по отношению к разрабатываемой системе. Можно заставить символ субъекта отображаться в верхней части линии жизни, задав ее свойство **Субъект**.
3	**Синхронное сообщение**	Отправитель ожидает ответа на синхронное сообщение перед тем, как продолжить. На рисунке показан вызов и возврат. Синхронные сообщения используются для представления обычных вызовов функций внутри программы, а также других видов сообщений, которые применяются аналогичным образом.
4	**Асинхронное сообщение**	Сообщение, не требующее ответа перед продолжением работы отправителя. Асинхронное сообщение показывает только вызов от отправителя. Используется для представления связи между отдельными потоками или создания нового потока.
5	**Вхождение выполнения**	Вертикальный затененный прямоугольник, который появляется на линии жизни участника и представляет период, когда участник выполняет операцию. Выполнение начинается, когда участник получает сообщение. Если инициируемое сообщение было синхронным сообщением, выполнение заканчивается стрелкой возврата к отправителю.
6	**Сообщение обратного вызова**	Сообщение, возвращающееся обратно участнику, который ожидает возврата из предыдущего вызова.Результирующее вхождение выполнения отображается поверх существующего.
7	**Исходное сообщение**	Сообщение от участника самому себе. Результирующее вхождение выполнения отображается поверх отправляющего выполнения.
8	**Создайте сообщение**	Сообщение, создающее участника. Если участник получает сообщение о создании, он должен быть первым, кто его получает.
9	**Найти сообщение**	Асинхронное сообщение от неизвестного или не указанного участника.
10	**Потерянное сообщение**	Асинхронное сообщение неизвестному или не указанному участнику.
11	**Комментарий**	Примечание можно подключить к любой точке линии жизни.
12	**Использование взаимодействия**	Заключает последовательность сообщений, которые определены в другой схеме.

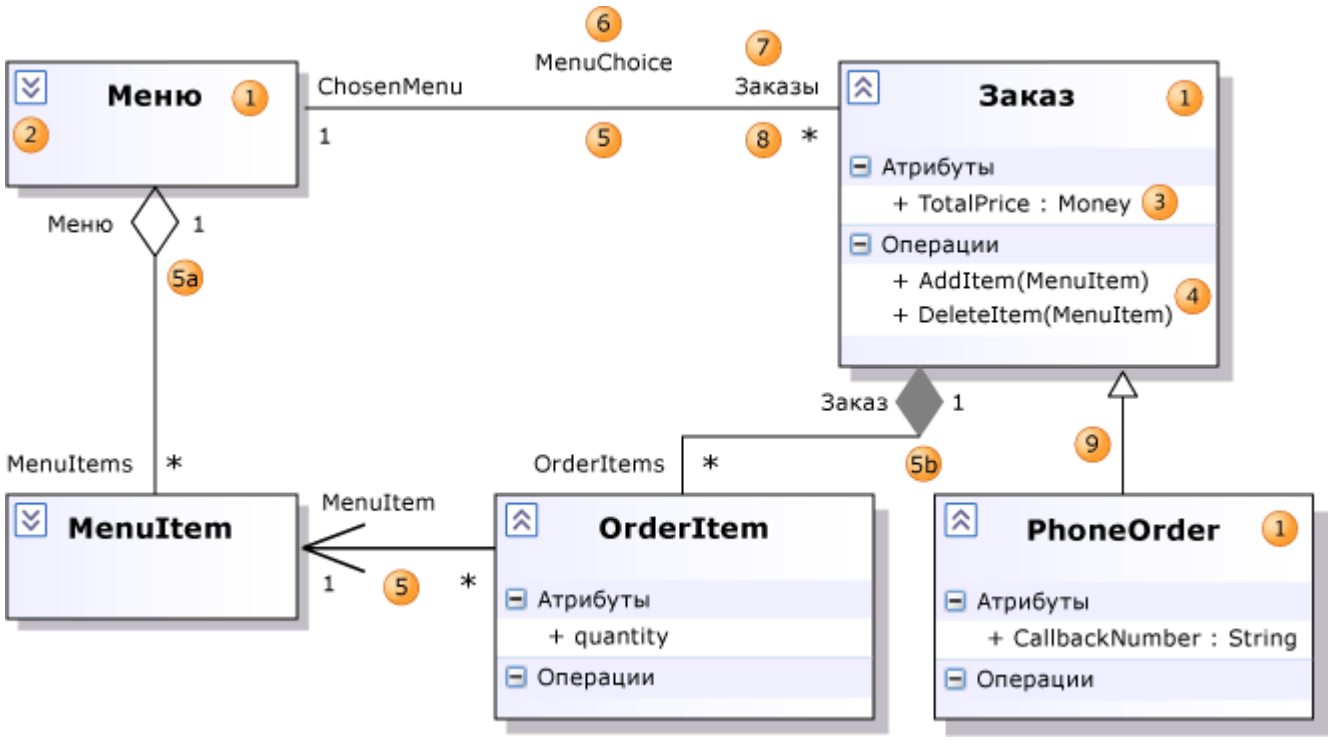
Фигура	Элемент	Описание
		Чтобы создать **использование взаимодействия**, щелкните инструмент и выполните перетаскивание поверх линий жизни, которые требуется включить.
13	**Объединенный фрагмент**	Коллекция фрагментов. Каждый фрагмент может включать одно или несколько сообщений. Существует несколько видов объединенных фрагментов. Чтобы создать фрагмент, щелкните сообщение правой кнопкой мыши, наведите указатель на пункт **Разместить во фрагменте**, после чего выберите тип фрагмента.
14	**Фрагмент условия**	Может использоваться для установки условия, зависящего от того, будет ли найден фрагмент. Чтобы задать условие, выберите фрагмент, выберите условие и введите значение.
	Взаимодействие	Коллекция сообщений и линий жизни, которая отображается в схеме последовательности. Чтобы просмотреть свойства взаимодействия, необходимо выбрать его в **Обозревателе UML-модели**.
	Схема последовательностей	На рисунке отображается взаимодействие. Чтобы просмотреть свойства, щелкните пустую область схемы. **Примечание** Имена схемы последовательностей, отображаемого взаимодействия и файла, который содержит схему, могут различаться.

Диаграмма классов

UML-схема классов описывает структуры объектов и сведений, используемые для внутренней организации приложения и для взаимодействия с пользователями. Кроме того, схема предоставляет сведения об этих структурах безотносительно какой-либо конкретной реализации. Ее классы и отношения могут реализовываться несколькими способами, например в таблицах базы данных, XML-узлах или сочетаниях программных объектов.

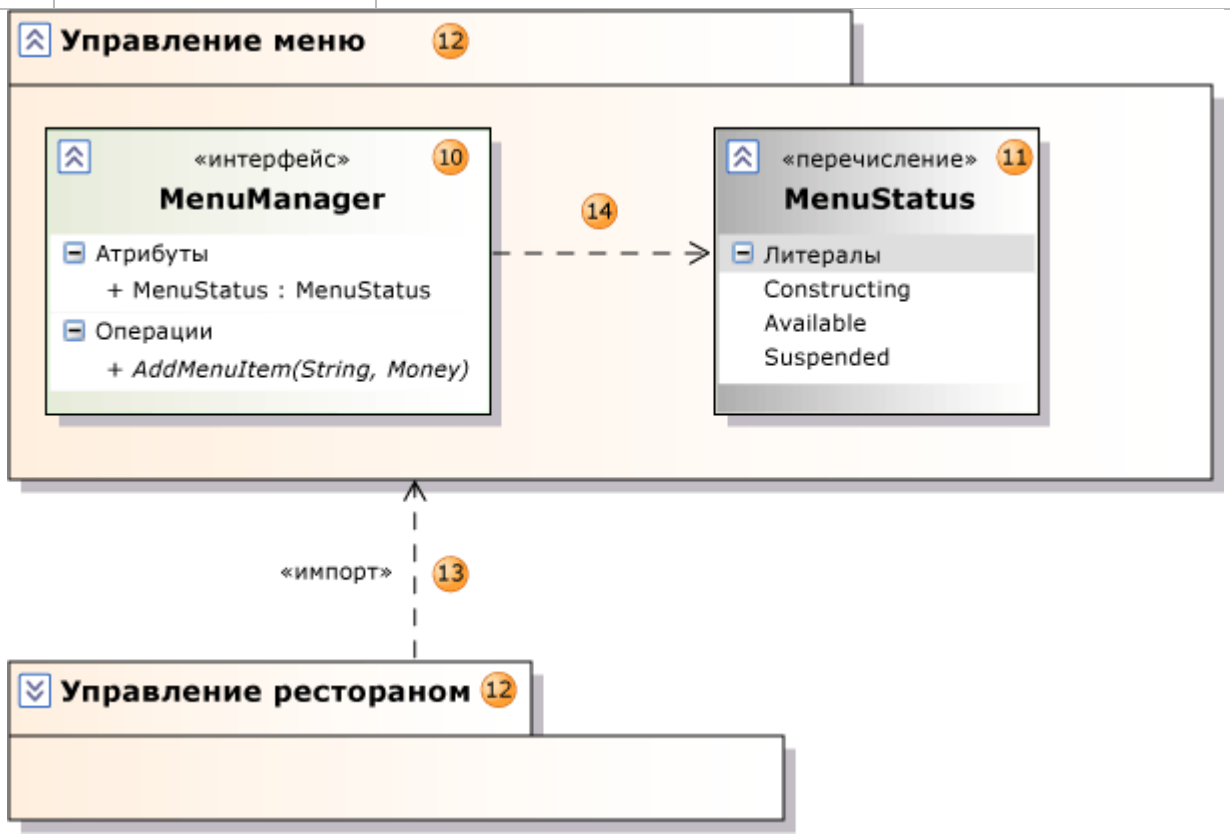
Чтение схем классов

В этом разделе в таблице описаны элементы, которые можно увидеть на UML-схеме классов.



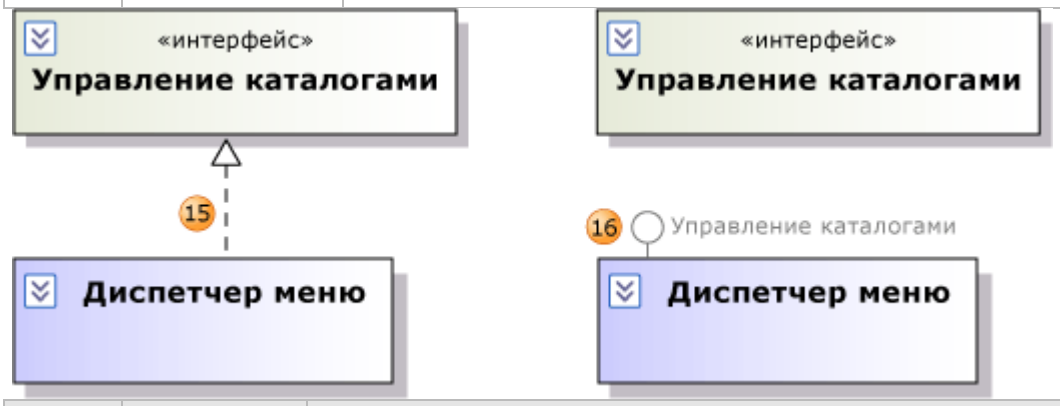
Фигура	Элемент	Описание
1	**Класс**	Определение объектов, совместно обладающих данными характеристиками структуры и поведения.
1	Классификатор	Общее имя для класса, интерфейса или перечисления. Компоненты, варианты использования и субъекты также являются классификаторами.
2	Элемент управления "свернуть/развернуть"	Если подробностей классификатора не видно, щелкните расширитель в верхней левой части классификатора. Иногда также нужно щелкнуть [+] для каждого сегмента.
3	**Атрибут**	Типизированное значение, прикрепленное к каждому экземпляру классификатора. Чтобы добавить атрибут, щелкните раздел **Атрибуты** и нажмите **ВВОД**. Введите сигнатуру атрибута
4	**Операция**	Метод или функция, которую можно выполнить с помощью экземпляров классификатора. Чтобы добавить операцию, щелкните раздел **Операции** и нажмите **ВВОД**. Введите сигнатуру операции.
5	**Ассоциация**	Отношение между членами двух классификаторов.
5а	**Агрегат**	Ассоциация, представляющая отношение совместного владения. Свойству **Агрегат** роли-владельца присвоено значение **Сделано общим**.
5б	**Композиция**	Ассоциация, представляющая отношение целого и части. Свойству **Агрегат** роли-владельца присвоено значение

Фигура	Элемент	Описание
		Составной.
6	**Имя ассоциации**	Имя ассоциации. Имя может оставаться пустым.
7	**Имя роли**	Имя роли, т. е. одного из окончаний ассоциации. Может использоваться для ссылки на связанный объект.
8	**Количество элементов**	Указывает, сколько объектов на этом окончании можно связать с объектами на другом окончании. Каждый заказ в этом примере должен быть связан только с одним меню. ***** означает, что ограничения числа ссылок, которые можно создать, не существует.
9	**Обобщение**	*Конкретный* классификатор наследует часть своего определения от *общего* классификатора. Общий классификатор находится на окончании соединителя с указателем стрелки. Атрибуты, ассоциации и операции наследуются конкретным классификатором. Воспользуйтесь инструментом **Наследование**, чтобы создать обобщение между двумя классификаторами.



Фигура	Элемент	Описание
10	**Интерфейс**	Определение части внешне видимого поведения объекта
11	**Перечисление**	Классификатор, состоящий из набора строковых литералов.

Фигура	Элемент	Описание
12	**Пакет**	Группа классификаторов, ассоциаций, действий, линий жизни, компонентов и пакетов. Логическая схема классов показывает, что членами данного пакета являются классификаторы и пакеты. Область видимости имен ограничивается пакетами, так что Класс1 в Пакет1 отличается от Класс1 вне этого пакета. Имя пакета отображается как часть свойств **Полное имя** его содержимого. Свойство **Связанный пакет** любой UML-схемы можно настроить так, чтобы оно ссылалось на пакет. В этом случае все элементы, создаваемые на этой схеме, станут частью пакета. Они отображаются в пакете в **Проводнике по моделям UML**.
13	**Импорт**	Отношение между пакетами, указывающее, что один пакет включает все определения другого.
14	**Зависимость**	Определение или реализация зависимого классификатора может измениться, если изменяется классификатор на окончании с наконечником стрелки.



Фигура	Элемент	Описание
15	**Реализация**	Класс реализует операции и атрибуты, определенные интерфейсом. Воспользуйтесь инструментом **Наследование**, чтобы создать реализацию между классом и интерфейсом.
16	**Реализация**	Альтернативное представление того же отношения. Метка на символе обозначения указывает на интерфейс. Чтобы создать эту презентацию, выделите существующее отношение реализации. Рядом с ассоциацией появляется тег действия. Щелкните тег действия и выберите **Показывать без описания операций**.

Диаграмма деятельности

На *схеме активности* бизнес-процесс или программный процесс показан как рабочий процесс, состоящий из ряда действий. Эти действия могут выполняться людьми, программными компонентами или компьютерами.
Схему активности можно использовать для описания процессов нескольких типов, таких как в следующих примерах.

- Бизнес-процесс или рабочий процесс, в котором участвуют пользователи и система.
- Шаги в тестовом случае.

- Программный протокол, т. е. разрешенная последовательность взаимодействий между компонентами.
- Программный алгоритм.

В этом разделе описаны элементы, которые можно использовать в схемах активности.

Чтение схем активности

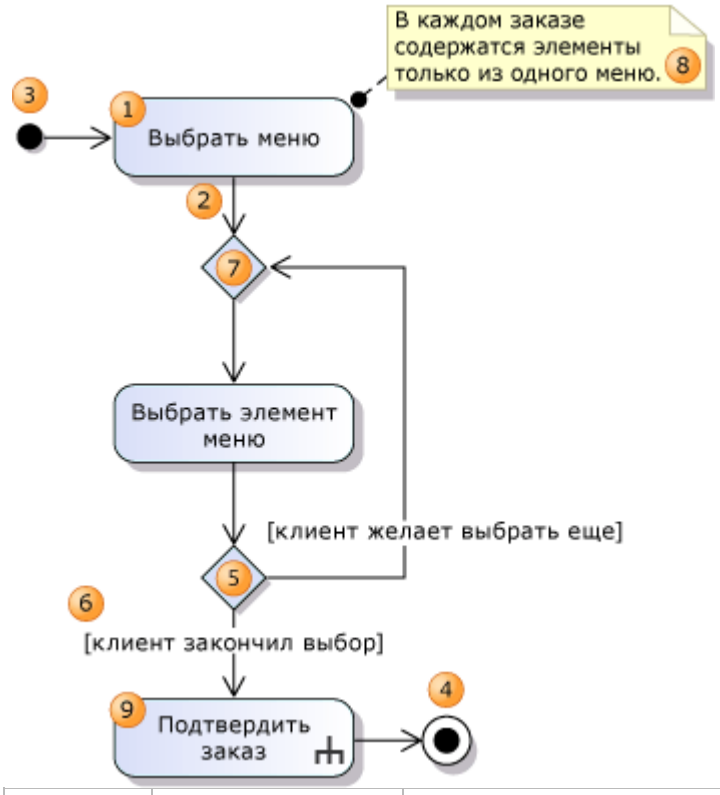
В таблицах в следующих разделах описаны элементы, которые можно использовать на схеме активности, и их основные свойства.

Действия и другие элементы, отображаемые на схеме активности, представляют собой одно действие. Эти действия можно просмотреть в обозревателе моделей UML. Он создается при добавлении первого элемента в схему.

Чтобы прочитать схему, представьте, что токен или поток управления проходит вдоль соединителей от одного действия к другому.

Простые потоки управления

Последовательность действий можно показать с помощью ветвей и циклов.

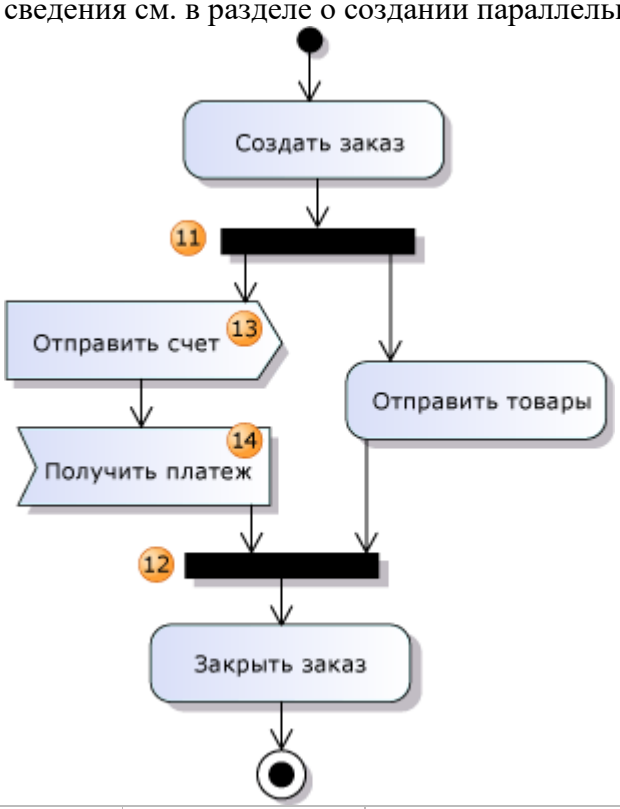


Фигура	Элемент	Описание и основные свойства
1	**Действие**	Шаг в действии, в котором пользователи программы выполняют какие-либо задачи. Действие может начинаться, когда токен был получен всеми его входящими потоками. Когда действие завершено, токены отправляются во все исходящие потоки. • **Body** — задает действие в подробностях. • **Language** — язык выражения в свойстве Body. • **Local Postconditions** — ограничения, которые должны быть удовлетворены по завершении выполнения. Цель, достигаемая действием. • **Local Preconditions** — ограничения, которые должны быть удовлетворены до начала выполнения.

Фигура	Элемент	Описание и основные свойства
2	**Поток управления**	Соединитель, который показывает поток управления между действиями. Чтобы интерпретировать схему, представьте, что токен переходит от одного действия к другому. Чтобы создать поток управления, используйте средство **Соединитель**.
3	**Начальный узел**	Указывает первый шаг или шаги в действии. В начале действия токен переходит из начального узла.
4	**Конечный узел действия**	Окончание действия. По прибытии токена действие завершается.
5	**Узел решений**	Условная ветвь в потоке. Имеет один вход и два или более выходов. Входящий токен появляется только на одном из выходов.
6	**Условие**	Условие, которое задает, может ли токен проходить вдоль соединителя. Чаще всего используются на исходящих потоках узла решений. Чтобы задать условие, щелкните поток правой кнопкой мыши, выберите **Свойства** и задайте свойство **Условие**.
7	**Узел слияния**	Требуется для слияния потоков, разделенных узлом решений. Имеет два или более входов и один выход. Токен на любом входе отображается на выходе.
8	**Комментарий**	Предоставляет дополнительные сведения об элементах, с которыми связан.
9	**Действие вызова поведения**	Действие, которое определяется более подробно на другой схеме активности. • **IsSynchronous** — если значение true, действие ожидает завершения активности. • **Behavior** — вызванное действие.
(не показана)	**Действие вызова операции**	Действие, которое вызывает операцию для экземпляра класса.
	Действия	Поток работ, описываемый схемой активности. Чтобы просмотреть свойства действия, необходимо выбрать его в**Обозревателе моделей UML**. • **Is Read Only** — если значение true, действие не должно изменять состояние ни одного объекта. • **Is Single Execution** — если значение true, одновременно возможно только одно выполнение этой схемы.
	UML-схема активности	Эта схема отображает действие. Чтобы просмотреть ее свойства, щелкните пустую область схемы.

Фигура	Элемент	Описание и основные свойства
		Примечание Имена схемы активности, файла, который содержит схему, и действия, отображаемого на схеме, могут различаться.

Параллельные потоки
Можно описать последовательности действий, выполняемых одновременно. Дополнительные сведения см. в разделе о создании параллельных потоков.

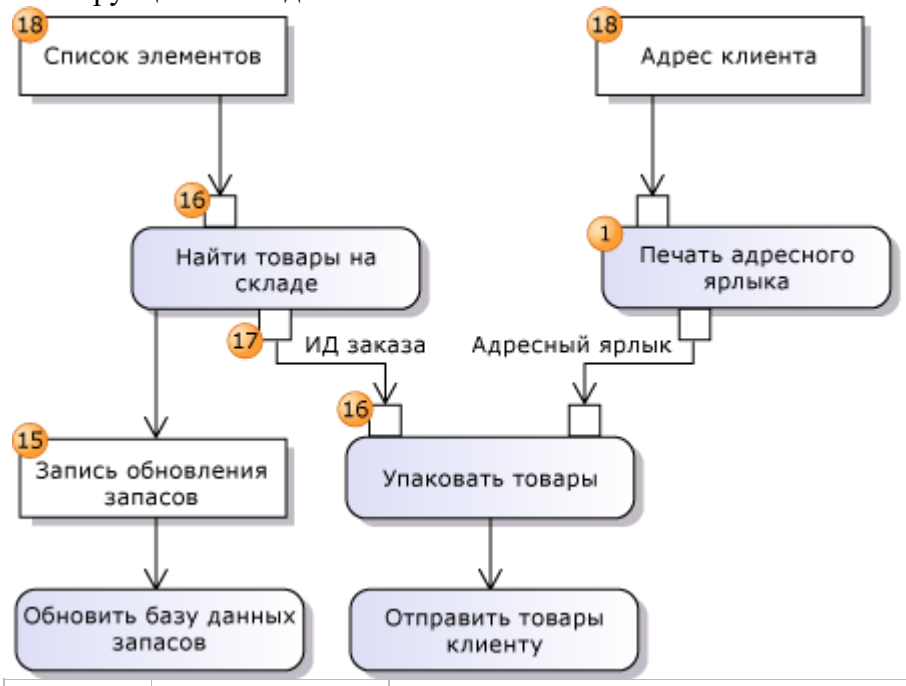


Фигура	Элемент	Описание
11	**Вилочный узел**	Разделяет единый поток на параллельные потоки. Каждый входящий токен создает токен на каждом исходящем соединителе.
12	**Узел присоединения**	Объединяет параллельные потоки в один поток. Если каждый входящий поток имеет ожидающий токен, создается токен на выходе.
13	**Действие отправки сигнала**	Действие, которое отправляет сообщение или сигнал другому действию или параллельному потоку того же действия. Тип и содержимое сообщения видны из названия действия или задаются в дополнительных комментариях. Действие может отправлять данные в сигнале, который можно передать действию в потоке объектов или закреплении ввода (16).
14	**Действие события принятия**	Действие, которое ожидает сообщения или сигнала, чтобы продолжиться. Тип сообщения, которое может быть получено действием, виден из названия или задается в дополнительных комментариях. Если действие не имеет входящего потока управления, оно создает

Фигура	Элемент	Описание
		токен всякий раз при получении сообщения. Действие может получать данные в сигнале, который можно передать в потоке объектов или закреплении вывода (17). • **IsUnmarshall** — если значение true, может существовать несколько типизированных закреплений вывода, и данные распаковываются в них. Если значение false, все данные отображаются в одном закреплении.

Потоки данных

Можно описать поток данных из одного действия в другое. Дополнительные сведения об элементах, упомянутых в данном разделе, см. в подразделе о создании потоков данных раздела "Инструкции по созданию схемы активности".



Фигура	Элемент	Описание
15	**Узел объекта**	Представляет данные, передаваемые в потоке. • **Ordering** — способ хранения нескольких токенов. • **Selection** — вызывает процесс фильтрации данных, который можно определить на другой схеме. • **Upper Bound** — 0 означает, что данные должны передаваться в потоке напрямую; * означает, что данные можно хранить в потоке. • **Type** — тип хранимых и передаваемых объектов.
16	**Закрепление ввода**	Представляет данные, которые действие может получать при выполнении. • **Type** — тип переданных объектов.
17	**Закрепление вывода**	Представляет данные, которые действие создает при выполнении. • **Type** — тип переданных объектов.
18	**Узел**	Узел объекта, через который действие может получать или создавать

	параметра действия	данные. Используется, если представленное схемой действие вызывается из другого действия, либо если схема описывает операцию или функцию. • **Type** — тип переданных объектов.
(не показана)	**Поток объектов**	Соединитель, который показывает поток данных между действиями и узлами объекта. Чтобы создать поток объектов, нужно использовать средство **Соединитель** для связи закрепления ввода или вывода либо узла объекта с другим элементом. • **Selection** — вызывает процесс фильтрации данных, который можно определить на другой схеме. • **Transformation** — вызывает преобразующий данные процесс, который можно определить на другой схеме. • **IsMulticast** — указывает на возможность существования нескольких получающих объектов или компонентов. • **IsMultiReceive** — указывает на возможность получения входных данных из нескольких объектов или компонентов.

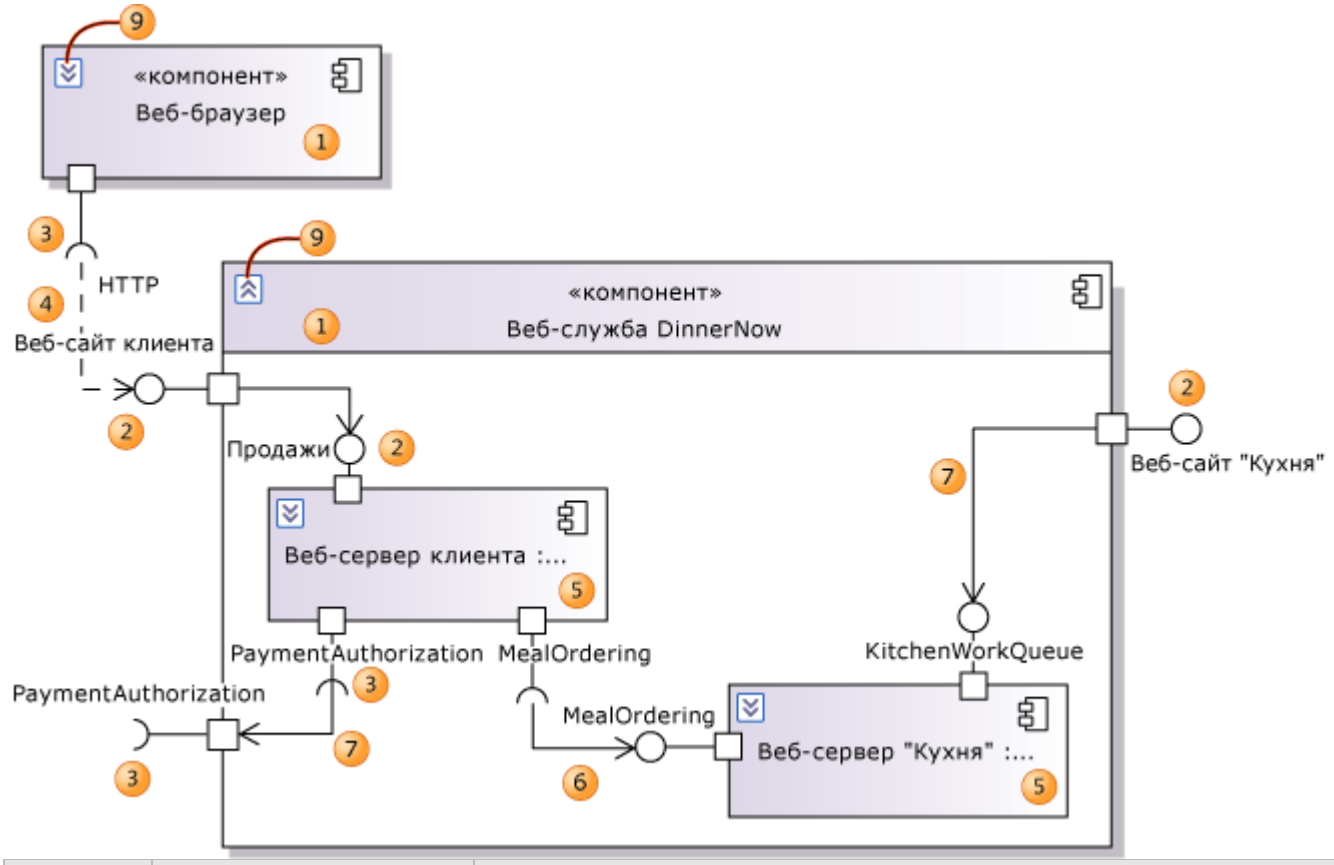
Диаграмма компонентов

В Visual Studio Ultimate на *схеме компонентов* показаны части конструкции программной системы. Схема компонентов помогает визуализировать высокоуровневую структуру системы и поведение служб, предоставляемых и потребляемых этими элементами через интерфейсы.

Схему компонентов можно использовать, чтобы описать конструкцию системы, реализуемую на любом языке и в любом стиле. Нужно только определить части конструкции, взаимодействующие с другими частями через ограниченный набор входных и выходных каналов. Можно использовать компоненты любого масштаба, взаимосвязанные любым способом.

Чтение схем компонентов

В следующей таблице описаны элементы, которые можно использовать на схеме компонентов, и их основные свойства.



Фигура	Элемент	Описание и основные свойства
1	**Компонент**	Допускающий повторное использование функциональный элемент системы. Компонент предоставляет и потребляет поведение через интерфейсы и может использовать другие компоненты. Можно скрывать или отображать внутренние части компонента с помощью элемента управления "развернуть/свернуть" (9). Компонент — это вид класса. • **Является неявно создаваемым экземпляром**. Если значение true (по умолчанию), компонент существует только как артефакт конструкции. Во время выполнения существует только ее часть.
2	**Предоставленный порт интерфейса**	Представляет группу сообщений или вызовов, реализуемых компонентом и доступных для использования другими компонентами или внешними системами. Порт — это свойство компонента, имеющее в качестве типа интерфейс.
3	**Требуемый порт интерфейса**	Представляет группу сообщений или вызовов, отправляемых компонентом другим компонентам или внешним системам. Компонент предназначен для соединения с компонентами, которые предоставляют хотя бы эти операции. Порт имеет в качестве типа интерфейс.
4	**Зависимость**	Может использоваться для указания, что требуемый интерфейс одного компонента может соответствовать предоставленному интерфейсу другого компонента.

		Зависимости также можно использовать в более общем случае при работе с элементами модели, чтобы показать, что конструкция одного зависит от конструкции другого.
5	**Часть**	Атрибут компонента, тип которого, как правило, является другим компонентом. Часть используется при внутреннем проектировании ее родительского компонента. Графически части изображаются вложенными в родительский компонент. Чтобы создать часть существующего типа компонента, перетащите компонент из Проводника по моделям UML в компонент-владелец. Чтобы создать часть нового типа, выберите инструмент **Компонент** и щелкните компонент-владелец. Например, компонент Car имеет части engine:CarEngine, backLeft:Wheel, frontRight:Wheel и т. д. Несколько частей могут иметь один и тот же тип, и разные компоненты могут иметь части одного типа. • **Тип**. Тип части, определяемый в другом месте модели. Как правило, типом является другой компонент. • **Количество элементов**. По умолчанию используется значение 1. Можно задать значение **0..1**, чтобы указать, что часть может иметь значение **null**, или задать значение *****, чтобы указать, что часть является коллекцией экземпляров данного типа. Также в качестве значения можно задать любое выражение, которое можно оценить в числовом диапазоне.
6	**Сборка части**	Соединение между требуемыми портами интерфейса одной части и предоставленными портами интерфейса другой.Реализация сборки части для разных компонентов может различаться. Соединенные части должны иметь один родительский компонент.
7	**Делегирование**	Связывает порт с интерфейсом одной из частей компонента. Указывает, что сообщения, отправленные компоненту, обрабатываются этой частью, или что сообщения, отправленные этой частью, отсылаются из родительского компонента.
8	**Обобщение**	Указывает, что один компонент наследуется от другого. Части и интерфейсы наследуются.
9	Элемент управления "развернуть/свернуть"	Позволяет скрывать или отображать внутренние части компонента.
(не показана)	**Комментарий**	Для дополнительных примечаний. Комментарий можно связать с неограниченным числом элементов на схеме, воспользовавшись инструментом **Соединительная линия**.

Лабораторная работа 15. Создание диаграмм в Visual Studio Ultimate

Создание диаграммы вариантов использования

В Visual Studio Ultimate можно создать *схему вариантов использования*, чтобы обобщить сведения о том, кто использует приложение или систему, и какие действия с этим приложением или системой они могут выполнять.

С помощью схемы вариантов использования можно вести обсуждения и передавать сведения о следующем.

- Сценарии взаимодействия системы или приложения с людьми, организациями или внешними системами.
- Цели, которых с помощью схем смогут достичь соответствующие субъекты.
- Область функционирования системы.

На схеме вариантов использования не отображаются подробности вариантов использования, а только обобщаются сведения о некоторых отношениях между вариантами использования, субъектами и системами. В частности, на схеме не отображается порядок выполнения действий для достижения целей в каждом варианте использования. Эти подробности можно описать в других схемах и документах, которые можно связать с каждым вариантом использования.

В предоставленных описаниях вариантов использования имеется несколько терминов, имеющих отношение к области активности, в которой функционирует система, например "Продажи", "Меню", "Клиент" и т. д. Важно четко определить эти термины и их отношения.

Варианты использования позволяют описать только функциональные требования к системе. Прочие требования (например, бизнес-правила, требования к качеству обслуживания и ограничения реализации) необходимо представлять отдельно. Архитектура и подробности внутренней структуры также нужно описывать отдельно.

В этом разделе в примерах описывается веб-сайт, на котором клиенты могут заказывать еду из местных ресторанов.

- *Субъект* (1) — это класс лиц, организаций, устройств или внешних программных компонентов. взаимодействующих с системой. Примерами субъектов являются следующие: Клиент, Ресторан, Датчик температуры, Устройство авторизации кредитных карт.
- *Вариант использования* (2) представляет действия, совершаемые одним или несколькими субъектами для достижения определенной цели.Примерами вариантов использования являются следующие: Заказ еды, Обновление меню, Обработка платежа.

На схеме вариантов использования они ассоциированы (3) с субъектами, выполняющими их.

- *Система (4)* — это любой объект в разработке. Системой может быть небольшой программный компонент, субъектами которой являются другие программные компоненты, полное приложение или крупный распределенный набор приложений, развернутых на нескольких компьютерах и устройствах. Примерами подсистем являются следующие: "Веб-сайт для заказа еды", "Бизнес по доставке еды", "Веб-сайт, версия 2".

Схема вариантов использования может показывать, какие варианты использования поддерживаются системой или ее подсистемами.

Основные этапы создания схем вариантов использования

Создание новой схемы вариантов использования

1. В меню **Архитектура** выберите пункт **Создать схему**.
2. В разделе **Шаблоны** щелкните **UML-схема вариантов использования**.
3. Назовите схему.
4. В области **Добавить в проект моделирования** выделите существующий проект моделирования в решении или выберите **Создать новый проект моделирования** и нажмите кнопку **ОК**.

Создание схемы вариантов использования

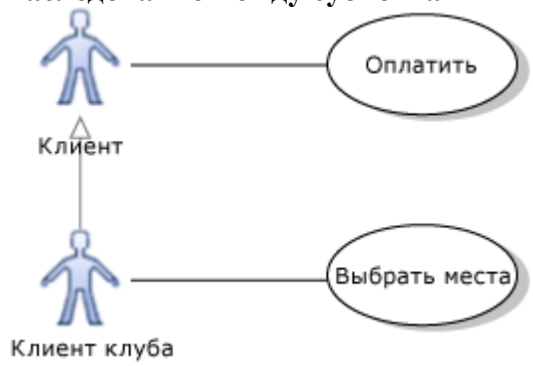
1. Перетащите границы **Подсистемы** из панели элементов на схему, чтобы представить всю систему или ее основные компоненты.
    - Можно создать схему вариантов использования без границ системы, если не нужно описывать, какие варианты использования поддерживает система или ее компоненты.
    - При необходимости перетащите угол изображения системы, чтобы увеличить его.
    - Переименуйте систему соответственно.
2. Перетащите **Субъекты** из панели элементов на схему (разместите их за пределами границ любой системы).
    - Субъекты представляют классы пользователей, организаций и внешних систем, взаимодействующих с данной системой.
    - Переименуйте их. Например: "Клиент", "Ресторан", "Организация, выдавшая кредитную карту".
3. Перетащите **варианты использования** из панели элементов в соответствующие системы.
    - Варианты использования представляют действия, выполняемые субъектами с помощью системы.
    - Переименуйте их, используя названия, которые будут понятны этим субъектам. Не используйте названия, имеющие отношение к коду.Например: "Заказ еды", "Оплата еды", "Доставка еды".
    - Начните с самых крупных транзакций, таких как Заказ еды, а затем переходите к более мелким взаимодействиям, таким как Выбор пункта меню.
    - Поместите каждый вариант использования в систему или крупную подсистему, обеспечивающую его реализацию (игнорируя различные виды и компоненты, используемые только для связи с пользователем).
    - Можно создать вариант использования за пределами границы системы, чтобы показать, что он не поддерживается системой (возможно, в определенной версии или выпуске).
4. Нажмите кнопку **Ассоциация** на панели элементов, затем последовательно выберите вариант использования и субъекта, участвующего в варианте использования. Свяжите каждый субъект с соответствующим вариантом использования подобным образом.
5. Структурируйте варианты использования с помощью отношений **Включение**, **Расширение** и **Обобщение**. Чтобы создать каждую из этих ссылок, последовательно щелкните инструмент, исходный вариант использования и целевой вариант использования.
6. Опишите варианты использования более подробно.
7. Создайте отдельные схемы для различных подсистем или разных групп связанных вариантов использования. Все схемы в одном проекте моделирования являются представлениями одной модели.

Создание субъектов и вариантов использования

Основной целью схемы вариантов использования является показать, кто взаимодействует с системой и каких целей они при этом достигают.

- Создайте **Субъекты**, чтобы представить классы людей, организаций, других систем, программ или устройств, взаимодействующих с данной системой или подсистемой.
    - Для каждого отдельного набора целей определите субъекты по типу или роли. При этом физические лица или сущности могут совпадать с ними. Например, Ресторан и Клиент — отдельные субъекты, хотя иногда сотрудник ресторана может выступать в качестве клиента.
- Создайте **варианты использования** для каждой цели, которой субъект стремится достичь в системе.
    - Назовите и опишите варианты использования словами, понятными субъекту, а не терминами реализации.
- Используйте **Ассоциации**, чтобы связать субъектов с вариантами использования.

Наследование между субъектами



Между субъектами можно создать связь **Обобщение**. Специализированный субъект, такой как Клиент Клуба в приведенном примере, наследует варианты использования обобщенного субъекта, в данном случае Клиента. Указатель стрелки должен указывать на более общего субъекта (Клиента). При создании связи сначала укажите более специализированного субъекта. Специализированный субъект может иметь дополнительные собственные варианты использования, не доступные другим субъектам.

Внимание

Не рекомендуется создавать циклы отношений обобщения, так как это приводит к тому, что субъект обобщает сам себя. Циклы могут стать причиной ошибок.

Другие значки субъектов

Вместо того чтобы использовать стандартные контурограммы, для представления субъекта можно использовать настраиваемые значки. Например, можно изменить внешний вид значка, чтобы он напоминал устройство, ресторан, банк и т. д.

Изменение внешнего вида субъекта

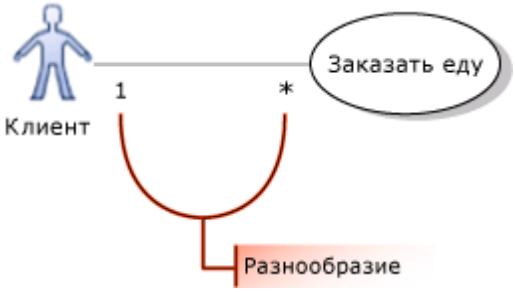
1. Щелкните субъект правой кнопкой мыши и выберите **Свойства**.

Появится окно **Свойства**.

2. Задайте свойство **Путь к изображению**, указав расположение файла изображения.
    - Можно использовать любой из нескольких допустимых форматов изображений, включая .gif, .jpg, и .bmp.
    - Используйте файл, включенный в систему управления версиями решения или проекта, чтобы файл оставался доступным при перемещении или копировании решения.
3. Чтобы реплицировать этот вид на других схемах вариантов использования, скопируйте субъект и вставьте его в другую схему.
    - Изменение изображения применимо только к представлению на определенной схеме. Оно не применимо к базовому элементу модели.Если перетащить субъект из Проводника по моделям UML на другую схему, он отобразится в виде стандартной контурограммы.

Количество элементов между субъектами и вариантами использования

Ассоциация между субъектом и вариантом использования может показывать *количество элементов* на каждом окончании.



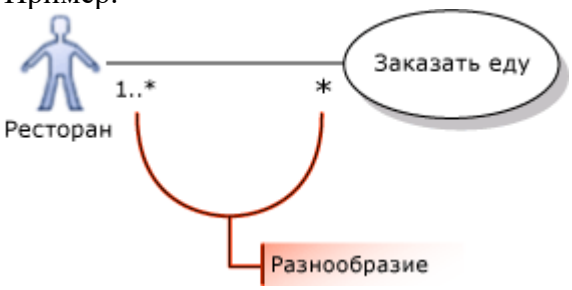
Примечание

Если количество элементов обоих окончаний ассоциации равно **1**, они не отображаются на схеме вариантов использования.

По умолчанию количество элементов равно **1**. В строгой интерпретации модели равное 1 количество элементов означает, что, например, каждый заказ размещается только одним клиентом, и что каждый клиент размещает только один заказ за раз.

Количество элементов можно изменить.

Пример.



- Чтобы указать, что несколько субъектов одного класса могут участвовать в одном вхождении варианта использования, на окончании субъекта в ассоциации укажите количество элементов **1..***.

На иллюстрации показано, что в выполнении одного заказа могут участвовать несколько ресторанов.

- Чтобы указать, что каждый субъект может одновременно участвовать в нескольких вхождениях варианта использования, на окончании варианта использования в ассоциации укажите количество элементов *****.

На иллюстрации показано, что каждый ресторан может одновременно работать над несколькими заказами.

Задание количества элементов в ассоциации

1. Щелкните ассоциацию правой кнопкой мыши и выберите **Свойства**.
2. Разверните свойство **Первая роль** или **Вторая роль**.

Роль — это элемент на одном окончании ассоциации.

3. Выберите из следующего списка значение для свойства Multiplicity.
   - **1** — чтобы указать, что только один экземпляр этой роли может участвовать в каждой связи.
   - **1..*** — чтобы указать, что в каждой связи может участвовать один или несколько экземпляров этой роли.
   - **0..1** — чтобы указать, что участие не является обязательным.
   - ***** — чтобы указать, что в связи участвует 0 или более экземпляров этой роли.

Примечание

Многие команды не размещают сведения о количестве элементах на схемах вариантов использования, оставляя значение по умолчанию 1. В этом случае эти сведения предоставляются в отдельных описаниях вариантов использования. В этом случае количества элементов на схемах вариантов использования скрыты.

Использование субъекта или варианта использования на нескольких схемах

Можно показать одни и те же субъекты и варианты использования на нескольких схемах. Пример.

- На разных схемах можно описать разные варианты использования, в которых участвует один субъект.
- Одну схему можно использовать, чтобы показать субъекты и подсистемы, с которыми связан вариант использования, а другую — чтобы показать структуру варианта использования, состоящую из включенных и расширенных вариантов использования.

Отображение одного субъекта или варианта использования на разных схемах

1. Создайте субъект или вариант использования на одной схеме.
2. Создайте другую схему вариантов использования.
3. Перетащите субъект или вариант использования из **Проводника по моделям** на новую схему.

Примечание

Если разместить на новой схеме субъект и вариант использования, которые уже связаны друг с другом, ассоциация между ними автоматически отобразится на новой схеме.

Описание вариантов использования в подробностях

Вариант использования представляет следующее.

- Цель субъекта при использовании системы, например Покупка еды.
- Один или более *сценариев*, т. е. последовательностей шагов, совершаемых для достижения цели, например: {Заказ еды, Оплата, Доставка}. Помимо успешных сценариев может быть несколько сценариев исключений или сбоев, например Кредитная карта отклонена.

При описании вариантов использования можно использовать разные уровни детализации. На ранних этапах разработки достаточно имени схемы вариантов использования. Впоследствии можно создать более подробные описания сценариев.

В Visual Studio Ultimate можно описать вариант использования несколькими способами, которые можно использовать по отдельности или вместе.

- Свяжите вариант использования с другой схемой или схемами проекта.
    - Схема активности позволяет составлять более сложное описание процесса, используя циклы, ветви и параллельные потоки. На схеме активности также можно показать поток данных между разными частями процесса.
    - Схема последовательностей позволяет описывать сложные ряды взаимодействий между разными субъектами. Также эту схему можно использовать, чтобы показать, что происходит в системе в ответ на каждый вариант использования.
- Свяжите вариант использования со страницей OneNote или параграфом, подробно описывающим вариант использования.
- Свяжите вариант использования с документом Word, в котором сценарии варианта использования описываются с помощью текста, снимков экрана и других средств.

Связывание варианта использования со схемой или файлом в одном решении

1. Создайте схему, например схему последовательностей или схему действий, чтобы проиллюстрировать сценарий варианта использования.
2. Вернитесь к схеме вариантов использования.
3. Перетащите схему или файл из обозревателя решений на пустую часть схемы вариантов использования.
4. Используйте инструмент **Зависимость**, чтобы соединить артефакт с вариантом использования.

Связывание с файлом решения, таким как документ Word или презентация PowerPoint

1. Создайте документ, в котором сценарий варианта использования описывается с помощью текста, снимков экрана и других средств.
2. Добавьте документ в решение.
    a. Переместите документ Word в папку Windows, в которой сохранено решение.
    b. В обозревателе решений щелкните решение правой кнопкой мыши, выберите команду **Добавить** и щелкните **Существующий элемент**.

c. Перейдите к документу Word и щелкните **Добавить**.

Документ Word отображается в папке решения в обозревателе решений.

3. Перетащите документ Word из обозревателя решений на пустую часть схемы вариантов использования.

Появляется новый артефакт.

4. Используйте инструмент **Зависимость**, чтобы соединить артефакт с вариантом использования.

Связывание с общим документом, элементом OneNote или веб-страницей

1. Получите URL-адрес общего элемента. Это может быть, например, путь к сетевому файлу, начинающийся с \\, или URL-адрес веб-страницы или сайта SharePoint, начинающийся с http://, или ссылка на раздел, страницу или параграф OneNote, начинающийся с onenote:.
2. Выберите **Артефакт** на панели элементов и щелкните на поверхности схемы вариантов использования.
3. Выбрав новый артефакт, введите или вставьте URL-адрес в свойство **Гиперссылка**.

Примечание

Дважды щелкните артефакт, чтобы открыть схему или документ, с которым он связан.

Связывание вариантов использования с рабочими элементами.

Если в проекте используется Visual Studio Team Foundation Server 2010, и установлен Сред. Командный обозреватель, можно связать каждый вариант использования с рабочим элементом в Team Foundation.

Это позволяет выполнять следующие действия.

- Описать вариант использования в связанном рабочем элементе. В частности, если в проекте используется формальный шаблон процессов Visual Studio, можно создать ссылку на рабочий элемент варианта использования. Этот тип рабочего элемента предоставляет поля для описания целей и сценариев варианта использования.
- Свяжите тестовые случаи с этим вариантом использования, чтобы иметь возможность получать отчеты о том, в какой степени разрабатываемый код реализует вариант использования.
- Свяжите задачи с вариантом использования, чтобы иметь возможность отслеживать ход разработки.

Структурирование вариантов использования

Нужно попытаться описать поведение системы с помощью небольшого числа основных вариантов использования. Каждый крупный вариант использования определяет основную цель, достигаемую субъектом, например приобретение продукции или (с точки зрения поставщика) предоставление продукции на продажу.

Четко представив эти цели, можно переходить к более подробному описанию каждой из этих целей и рассматривать различия в основных целях.

Старайтесь не разбивать варианты использования на слишком много компонентов. Варианты использования позволяют описать работу пользователей с системой, а не ее внутреннее функционирование. Кроме того, зачастую рекомендуется создавать первоначальные рабочие версии кода, а не тратить время на подробное структурирование вариантов использования.

На схеме вариантов использования можно обобщить сведения об отношениях между основными вариантами использования и более детализированными вариантами.

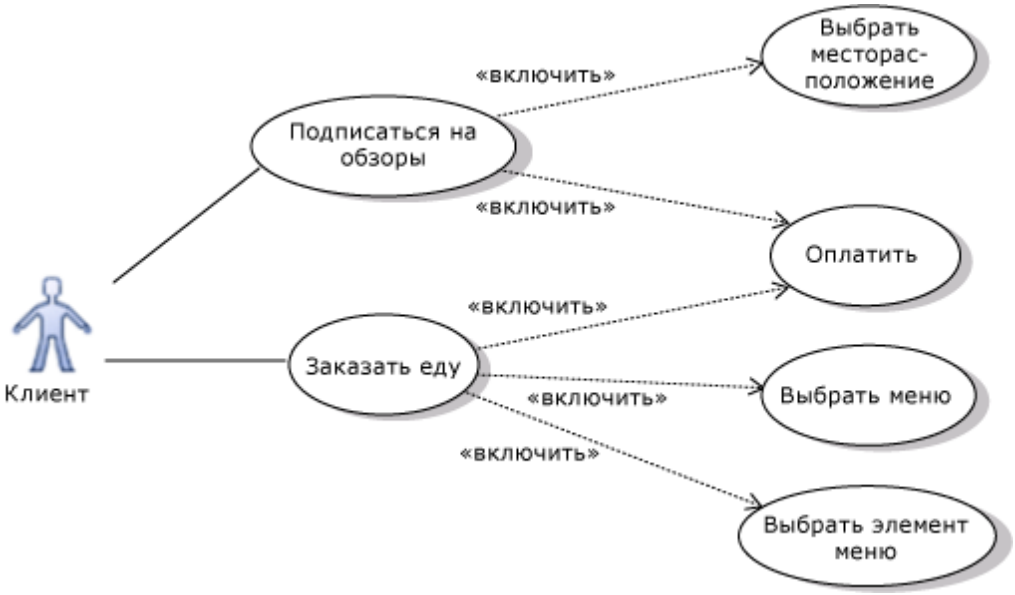
Отображение подробностей варианта использования с помощью отношений включения

Используйте отношение **Включение**, чтобы показать, что один вариант использования описывает некоторые подробности другого. На иллюстрации вариант Заказ еды включает варианты Оплата, Выбор меню и Выбор пункта меню. Каждый из включенных вариантов (более подробные варианты использования) представляют действия, которые, возможно, придется совершить субъекту или субъектам для достижения общей цели, описанной во включающем варианте использования. Стрелка должна указывать на более подробный, включенный вариант использования.

Внимание

Не рекомендуется создавать циклы отношений включения, так как это приводит к тому, что субъект включает сам себя. Циклы могут стать причиной ошибок.

Включенные варианты использования можно использовать совместно. В данном примере варианты использования Заказ еды и Подписка на обзоры включают вариант использования Оплата.



Цель и сценарии включенного варианта использования должны иметь смысл независимо друг от друга, чтобы их можно было включать в варианты использования, создаваемые позже.

Разделение вариантов использования на включающие и включенные части позволяет достичь следующих целей.

- Структурировать описания вариантов использования по уровню детализации.
- Избежать дублирования общих сценариев в разных вариантах использования.

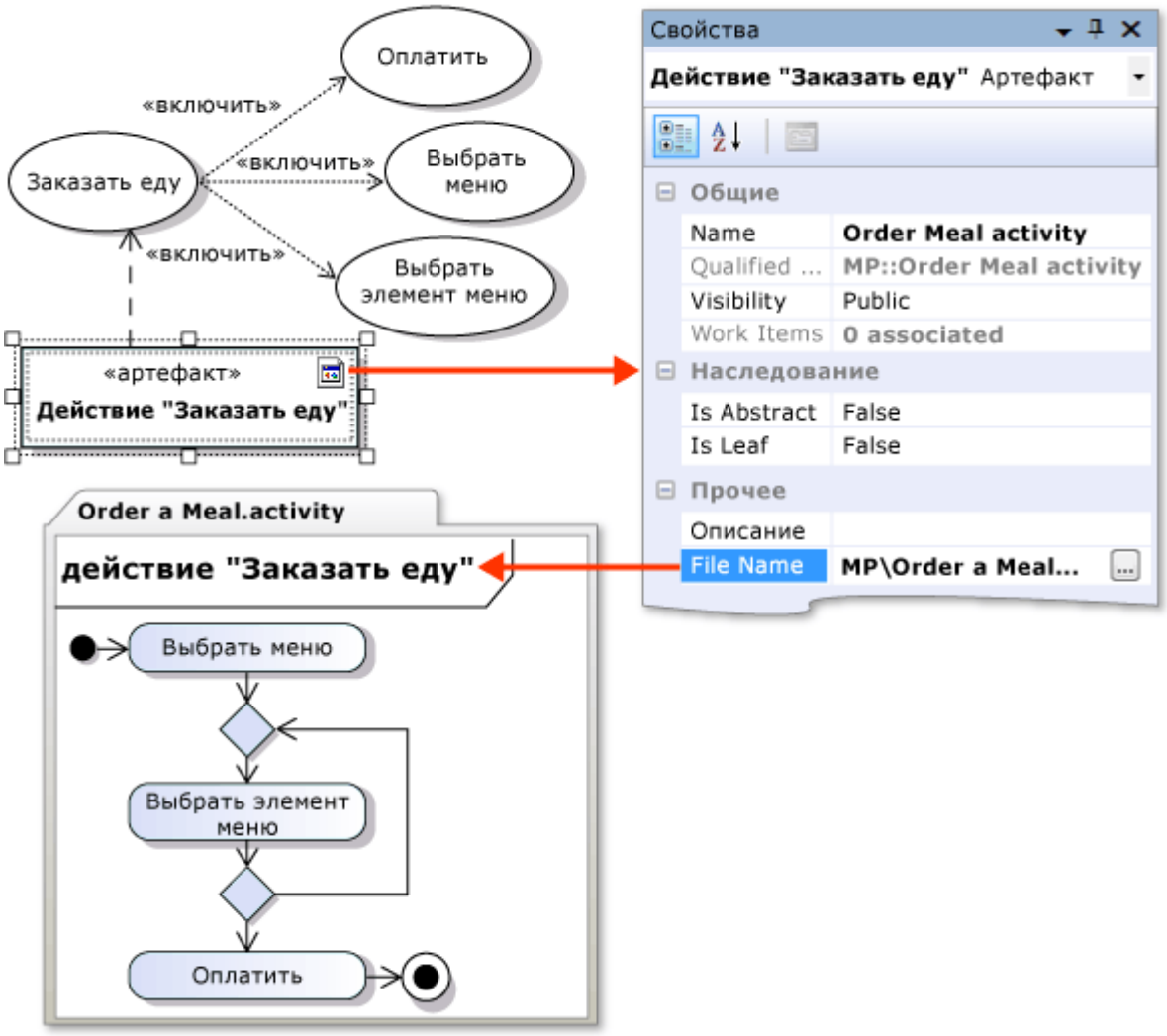
Подробное определение порядка совершения шагов

Схема вариантов использования никак не описывает последовательность совершения шагов и не сообщает, обязательно ли выполнять то или иное действие во всех вариантах использования.

Чтобы прояснить порядок совершения шагов, можно использовать **артефакт**, чтобы прикрепить отдельный документ к включающему варианту использования. В следующем примере схема активности прикреплена к варианту использования "Заказ еды". Кроме того, можно использовать текстовый документ, включающий список шагов или последовательность снимков экрана.

Обратите внимание на следующие соглашения о допустимых именах при использовании схемы активности.

- Имя целого действия совпадает с именем включающего варианта использования.
- Действия на схеме активности имеют те же имена, что и включенные варианты использования.



Совместная работа с целями с помощью отношений обобщения

Используйте отношение обобщения, чтобы показать, что *специализированный* вариант использования — это конкретный способ достижения целей, выраженных в другом, *общем* варианте использования. Стрелка должна указывать на более общий вариант использования.

Например, вариант Оплата обобщает варианты Оплата с помощью кредитной карты и Оплата наличными.

Специализированные варианты использования помогают показать различные способы достижения одной цели с использованием системы.

Считается, что специализированные варианты использования наследуют цели и субъекты общего варианта использования. Общий вариант использования не обязательно должен иметь собственные сценарии; специализации этого варианта описывают различные пути достижения целей.

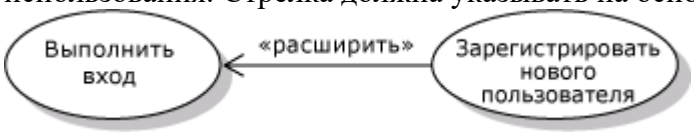
Реструктуризация общих целей из двух или более вариантов использования

1. Создайте и назовите новый общий вариант использования.

2. Создайте отношение **Обобщение**, чтобы большая стрелка указывала на новый общий вариант использования.
    a. Щелкните **Обобщение** на панели элементов.
    b. Щелкните специализированный вариант использования (Оплата с помощью кредитной карты в этом примере).
    c. Щелкните общий вариант использования (Оплата в этом примере).
3. Если описаны цели для специализированных вариантов использования, переместите общие части в описание общего варианта использования.
4. Субъекты, совместно используемые в разных специализированных вариантах использования, можно переместить в общий вариант использования.

Разделение различающихся вариантов с помощью отношений расширения

Используйте связь "Расширение", чтобы показать, что один вариант использования в определенных обстоятельствах может добавлять функциональные возможности другому варианту использования. Стрелка должна указывать на основной, расширенный вариант использования.



Например, вариант использования Вход в систему стандартного веб-сайта может включать вариант Зарегистрировать нового пользователя, но только если пользователь еще не имеет учетной записи.

Разделение варианта использования на основные и расширенные части.

1. Создайте и назовите новый расширенный вариант использования.
2. Создайте отношение **Расширение** со стрелкой, указывающей на расширенный вариант использования.
    a. Щелкните **Расширение** на панели элементов.
    b. Щелкните расширенный вариант использования (Зарегистрировать нового пользователя в примере).
    c. Щелкните расширенный вариант использования (Вход в систему в этом примере).
3. Если уже созданы сценарии расширенного варианта использования, переместите соответствующие шаги в сценарий расширения.
4. Описание расширения (Зарегистрировать нового пользователя в этом примере) также должно включать подробности о месте расширения в сценариях основного варианта использования и обстоятельствах использования этого расширения. Можно сказать, что это описание модифицирует описание основного варианта.

Расширенный вариант использования представляет шаги сценария, которые иначе являлись бы частью сценариев основного варианта использования.Сценарий и цели расширения всегда используются в контексте основного варианта использования, следовательно, они не обязательно должны иметь смысл отдельно от него.

Разделение расширений может оказаться полезным при описании следующих ситуаций.

- Имеются дополнительные субъекты, участвующие только в расширенном варианте использования. Например, администратор должен утвердить регистрацию клиента на веб-сайте.
- Отдельная подсистема обрабатывает расширенный вариант использования.
- Расширение доступно только в определенных версиях системы. Каждую версию можно показать как отдельную подсистему на схеме вариантов использования.

Использование границ подсистем

Используйте границу подсистемы, чтобы показать, какие варианты использования находятся в области действия системы.

Создание границы подсистемы

1. На панели элементов выберите **Подсистема**, затем щелкните схему.

Подсистема появится на схеме.

2. Перетащите углы подсистемы, чтобы изменить ее размер.
3. Перетащите существующие варианты использования в подсистему или из нее, чтобы скорректировать ее содержимое.

- или -

Чтобы создать новый вариант использования непосредственно в подсистеме, щелкните **Вариант использования** на панели элементов, затем щелкните внутри подсистемы.

Варианты использования за пределами области системы

Часто имеет смысл включить в схему варианты использования, которые являются частью бизнеса, но не обрабатываются разрабатываемой системой.Это позволяет разработчикам лучше понять контекст, в котором они работают. Например, вариант "Доставка еды" можно показать как вариант использования, включающий субъекты "Ресторан" и "Клиент", и указать, что он не входит в область ответственности веб-сайта по заказу еды.

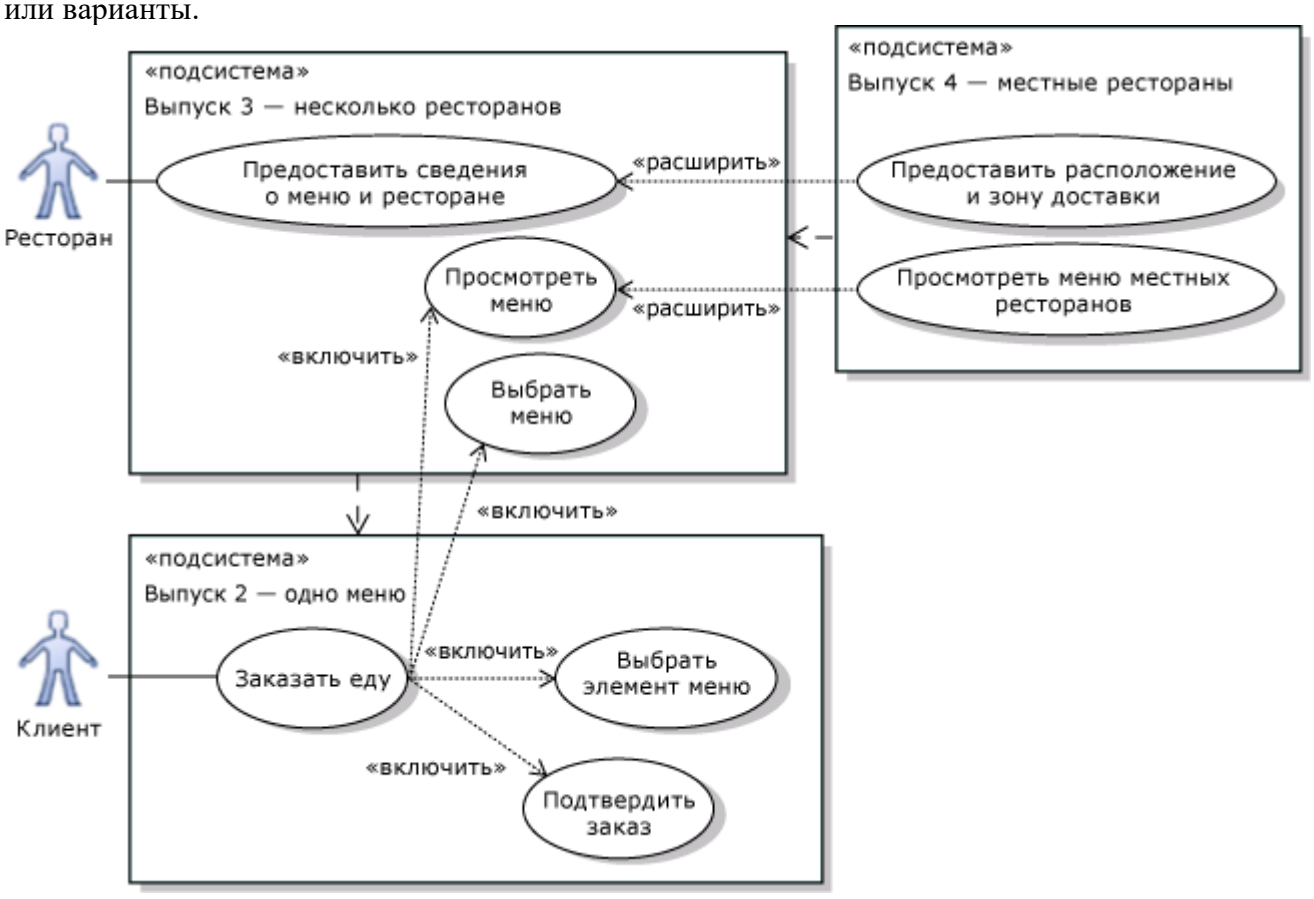
Несколько подсистем

Можно создать несколько границ подсистем, чтобы показать, как разные компоненты системы обрабатывают разные варианты использования.Например, вариант использования Добавление отзыва о ресторане может обрабатываться на отдельном форуме веб-сайта. Помните, что схема вариантов использования описывает только видимые пользователю элементы. Если необходимо описать внутреннее разделение функций в системе, следует использовать схему компонентов.

Версии системы

Разные границы подсистемы можно использовать, чтобы показать разные версии системы. Например, вариант использования "Оплата" может входить только в состав версии 2 веб-сайта (но не версии 1). Это подразумевает, что система помогает клиентам размещать заказы. Однако клиенты должны оплачивать эти заказы ресторану напрямую.

Используйте отношение **Зависимость**, чтобы связать подсистемы, представляющие разные версии или варианты.



Создание диаграммы последовательностей

В Visual Studio Ultimate можно создать *схему последовательностей*, чтобы отобразить взаимодействие. Взаимодействие — это последовательность сообщений между типичными экземплярами классов, компонентов, подсистем или субъектов.

Существует два вида схем последовательностей.

- Основанные на коде схемы последовательностей можно создавать из программного кода .NET.
- UML-схемы последовательностей являются частью UML-проектов моделирования.

Этот раздел посвящен UML-схемам последовательностей.

Схемы последовательностей можно использовать в разных целях и на разных уровнях детализации программы. Как правило, схема последовательностей создается в следующих случаях.

- Если используется схема вариантов использования, которая обобщает сведения о пользователях системы и их целях, можно создать схемы последовательностей, чтобы описать, как основные компоненты системы взаимодействуют для достижения цели каждого варианта использования.
- Если определены сообщения, поступающие в интерфейс компонента, можно создать схемы последовательностей, чтобы описать, как внутренние части компонента взаимодействуют для достижения результата, требуемого для каждого входящего сообщения.

Создание схем последовательностей имеет несколько преимуществ.

- Можно легко увидеть, как задачи распределяются между компонентами.
- Можно определить шаблоны взаимодействия, затрудняющие обновление программы.

Отношение к другим схемам

UML-схемы последовательностей можно использовать с другими схемами несколькими способами.

Линии жизни и типы

Линии жизни, создаваемые в схеме последовательностей, могут представлять типичные экземпляры компонентов или классов в системе. Можно создавать линии жизни из типов, а типы — из линий жизни, а также отображать типы на UML-схемах классов и UML-схемах компонентов.

Типы параметров

С помощью UML-схемы классов можно также описать типы параметров и возвращаемые значения в сообщениях, обмен которыми ведется между линиями жизни.

Подробности варианта использования

Вариант использования представляет цель пользователя, а также последовательность шагов для достижения этой цели. Последовательность шагов можно описать несколькими способами. Во-первых, можно создать схему последовательностей, показывающую взаимодействия между пользователями и основными компонентами системы. .

Исходный код

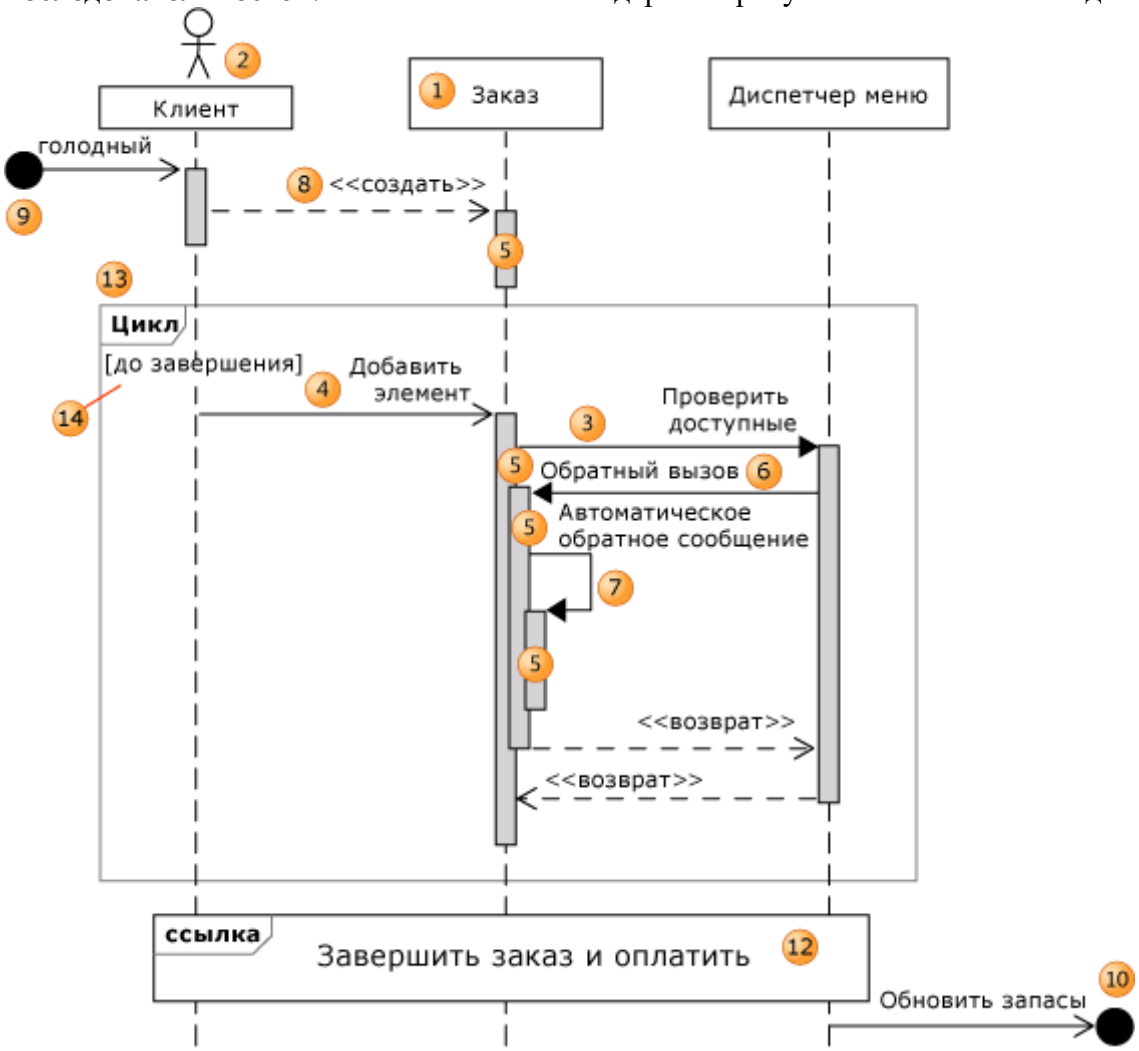
Можно создать схему последовательностей из исходного кода. Можно пересмотреть схему, чтобы поэкспериментировать с разными вариантами проектирования. Также при необходимости можно скопировать содержимое в схему последовательностей проекта моделирования.

Основные этапы создания схем последовательностей

Создание схемы последовательностей

1. В меню **Архитектура** выберите пункт **Создать схему**.
2. В разделе **Шаблоны** щелкните **UML-схема последовательностей**.
3. Назовите схему.
4. В области **Добавить в проект моделирования** выделите существующий проект моделирования в решении или выберите **Создать новый проект моделирования** и нажмите кнопку **ОК**.

Новая схема последовательностей отображается на панели элементов **Схемы последовательностей**. Панель элементов содержит требуемые элементы и соединители.



Создание схемы последовательностей

1. Перетащите **Линии жизни** (1) из **Панели элементов** на схему, чтобы представить экземпляры классов, компонентов, субъектов или устройств.

Примечание

Кроме того, линию жизни можно создать, перетащив существующий класс, интерфейс, субъект или компонент из **Проводника по моделям UML** на схему. Так создается линия жизни, представляющая экземпляр выбранного типа.

2. Создайте сообщения, чтобы показать, как линии жизни взаимодействуют для достижения конкретной цели.

Чтобы создать сообщение (3, 4, 6, 7) щелкните инструмент создания сообщений. Затем щелкните отправляющую линию жизни в том месте, где необходимо начать сообщение, и щелкните получающую линию жизни.

Вхождение выполнения (5) отображается на получающей линии жизни. Вхождение выполнения представляет период времени, в течение которого экземпляр выполняет метод. Можно создать другие сообщения, начинающиеся с вхождения выполнения.

3. Чтобы показать сообщение, поступающее из неизвестного источника события (9) или передает данные неизвестным получателям (10), создайте асинхронное сообщение из или в пустое пространство на схеме. Эти сообщения называются *найденные сообщения* (9) и *утерянные сообщения* (10).

Примечание

Чтобы переместить группу линий жизни, имеющих утерянные или найденные сообщения, выполните следующие действия, чтобы выделить линии жизни перед перемещением: нарисуйте

прямоугольник вокруг этих линий жизни, либо нажмите и удерживайте клавишу **CTRL** и последовательно щелкните каждую линию жизни. Если для выбора и перемещения всех линий жизни использовать команду **Выделить все** или сочетание клавиш **CTRL+A**, утерянные или найденные сообщения, прикрепленные к этим линиям жизни, не будут перемещены. В этом случае сообщения можно переместить отдельно.

4. Создайте схемы последовательностей для каждого основного сообщения одному и тому же компоненту или системе.

Изменение порядка сообщений

- Перетащите сообщение вверх или вниз по соответствующей линии жизни. Можно перетаскивать сообщения на другие сообщения, а также в блок выполнения или из него.

- или -

- Щелкните сообщение и используйте клавиши **СТРЕЛКА ВВЕРХ** и **СТРЕЛКА ВНИЗ**, чтобы скорректировать положение сообщений. Используйте сочетания клавиш **SHIFT+СТРЕЛКА ВВЕРХ** и **SHIFT+СТРЕЛКА ВНИЗ**, чтобы изменить последовательность сообщений.

Перемещение или копирование последовательностей сообщений на схеме последовательностей

1. Щелкните сообщение (3, 4) правой кнопкой мыши и выберите **Копировать**.
2. Щелкните правой кнопкой мыши вхождение выполнения (5) или линию жизни, из которой необходимо отправить новое сообщение, и выберите**Вставить**. При необходимости нового отправителя можно изобразить на другой схеме.

Копия сообщения и все его дочерние сообщения добавляются в окончание вхождения выполнения или в окончание линии жизни.

Примечание

Вставленное сообщение всегда отображается на окончании вхождения выполнения или линии жизни. Вставив сообщение, можно перетащить его на прежнее место.

Оптимизация размещения элементов на схеме последовательностей

- Щелкните правой кнопкой мыши пустую область схемы и выберите **Изменить порядок размещения**.
- Чтобы отменить операцию, последовательно щелкните **Изменить** и **Отменить**.

Изменить пакет, владеющий взаимодействием

1. В **Проводнике по моделям UML** найдите взаимодействие, отображаемое на схеме последовательностей.

Примечание

Взаимодействие не отобразится в **Проводнике по моделям UML**, пока в схему последовательностей не будет добавлена первая линия жизни.

2. Перетащите взаимодействие в пакет.

- или -

Щелкните взаимодействие правой кнопкой мыши и выберите **Вырезать**. Щелкните пакет правой кнопкой мыши и выберите **Вставить**.

Создание и использование простых схем последовательностей

Наиболее простая и часто используемая форма схемы последовательностей содержит только линии жизни и сообщения. Схема этого вида позволяет ясно показать типичную последовательность взаимодействий между объектами в проектируемой системе или между системой и ее пользователями.Часто этого достаточно, чтобы обсуждать проектируемую систему и передавать сведения о ней.

При создании простой схемы последовательностей не следует забывать о следующем.

Типы сообщений

Для создания сообщений можно использовать три различных инструмента.

- Используйте инструмент **Синхронная работа**, чтобы описать взаимодействие, в ходе которого отправитель ожидает, пока получатель даст ответ (3).

Стрелка **<<return>>** отображается в конце вхождения выполнения. Она обозначает, что контроль над взаимодействием возвращается отправителю.

- Используйте инструмент **Асинхронная работа**, чтобы описать взаимодействие, в ходе которого отправитель может продолжать выполнять действия немедленно, не дожидаясь получателя (4).
- Используйте инструмент **Создать**, чтобы описать взаимодействие, в ходе которого получатель (8) создается отправителем.

Сообщение о создании должно быть первым сообщением, которое получит получатель.

Создание заметок о взаимодействиях

Чтобы описать последовательность более подробно, можно разместить **Комментарий** в любом месте схемы.

Используя **Ссылки комментария**, можно связать комментарий с линиями жизни, выполнениями, использованиями взаимодействия и фрагментами.

Внимание

При необходимости прикрепить комментарий к определенной точке последовательности, нужно связать его с вхождением выполнения, использованием взаимодействия или фрагментом. Не связывайте комментарий с линией жизни, потому что в этом случае комментарий не будет прикреплен к правильной точке последовательности.

Используйте комментарий в следующих целях.

- Отметить, что было достигнуто на ключевых точках последовательности. Это позволяет другим пользователям видеть цели взаимодействий.
- Описать общую цель всей последовательности. Прикрепить комментарий к начальному вхождению выполнения или не прикреплять его ни к чему.Пример. "Клиент выбрал пункты из меню, и ему была предоставлена информация о стоимости этих пунктов".
- Описать обязанности каждой линии жизни. Прикрепить комментарий к линии жизни. Пример. "Менеджер по обработке заказов собирает сведения о выбранных клиентом пунктах меню".
- Создавать примечания об исключениях и других вариантов последовательностей, которые могут быть выполнены в качестве альтернативы типичной последовательности, изображенной ниже. Пример. "Клиент может пропустить остальные элементы последовательности".
    - В качестве более формальной альтернативы этого вида примечаний можно использовать фрагменты.

Определение области действия схемы

Очень важно четко обозначить, что должно отображаться на схеме.

Инициирующее событие

Каждая схема должна показывать последовательность взаимодействий, порождаемых одним инициирующим событием. Таким событием может стать следующее.

- Пользователь, инициирующий вариант использования (например, открывающий веб-страницу для покупки еды).
- Сообщение от одного системного компонента другому, например, с запросом о доступности пунктов, которые желает приобрести клиент.
- Событие, инициируемое изменением состояния, например, если уровень запасов определенного товара падает ниже порогового значения.

Уровень детализации

На схемах последовательностей можно отображать разные уровни детализации. Можно определить уровень детализации в двух разных измерениях практически независимо друг от друга.

Линии жизни могут представлять один из уровней детализации.

- Объекты в существующем или разрабатываемом программном коде.

- Компоненты и их субкомпоненты, как правило, без видов, посредников и других соединительных механизмов.
- Система и внешние субъекты

Сообщения могут представлять один из уровней детализации.

- Программные сообщения в программном коде в API или веб-интерфейсе.
- Транзакции или субтранзакции, например между пользователями и системой или между кодом и базой данных.
- Варианты использования — основные виды взаимодействий между пользователями и системой.

При анализе существующего кода и описании новой проектируемой системы часто имеет смысл создавать и анализировать представления с меньшей детализацией.

Описание вариантов

На схеме отображается одна, типичная последовательность событий. Если необходимо показать альтернативные возможности, такие как сценарии сбоев, можно воспользоваться одним из описанных ниже методов.

- Создать отдельные схемы последовательностей для описания этих сценариев.
- Использовать описание структур управления с помощью фрагментов, чтобы показать циклы, альтернативные варианты и т. д.

Оценка конструкции

Схему можно использовать для оценки распределения задач между ее объектами или компонентами. Необходимо провести реструктуризацию, если наблюдаются следующие явления.

- Создается впечатление, что одна линия жизни выполняет все функции, вызывая все остальные элементы схемы, в то время как другие линии жизни лишь пассивно отвечают на эти вызовы.
- Многие сообщения пересекают линии жизни. Каждая линия жизни должна отправлять сообщения небольшому числу соседних элементов и не должна взаимодействовать с соседями этих соседних элементов. Как правило, имеется возможность расположить линии жизни так, чтобы сообщения пересекали линии жизни всего в нескольких местах; в местах пересечения целевая линия жизни не должна обмениваться сообщениями, пересекающими какие-либо другие линии жизни.
- Некоторые линии жизни выполняют несколько разных видов задач. Область ответственности каждой линии жизни должна описываться в одном кратком предложении. В этом предложении должны обобщаться сведения о том, что линия жизни делает в ответ на каждое получаемое сообщение.

Классы и линии жизни

Линии жизни на схемах последовательностей показывают экземпляры классов или интерфейсов компонентов.

Создание линий жизни из типов

Из уже определенных классов (например, на схеме классов) можно создавать новые линии жизни.

Создание линии жизни из существующего типа

- Перетащите класс, компонент или интерфейс из Проводника по моделям UML на схему последовательностей.

- или -

1. На соответствующей схеме щелкните класс, компонент или интерфейс правой кнопкой мыши и выберите **Создать линию жизни**.
2. В диалоговом окне **Создать линию жизни** выберите схему последовательностей и нажмите кнопку **ОК**.

Отобразится новая линия жизни с именованным экземпляром. Типом этой линии жизни является перемещенный тип.

Примечание

Это действие можно повторять неограниченное количество раз. Это позволяет создать линии жизни с разными именами экземпляров.

Изменение типа линии жизни

1. Щелкните линию жизни правой кнопкой мыши и выберите **Свойства**.
2. В окне **Свойства** задайте значение для свойства **Тип**. Можно выбрать тип из раскрывающегося меню или указать новое имя.

Создание классов из линий жизни

Создав одну или несколько схем последовательностей, можно обобщить линии жизни, создавая из них классы или интерфейсы.

Создание класса или интерфейса из линии жизни

1. Щелкните линию жизни правой кнопкой мыши и выберите **Создать класс** или **Создать интерфейс**.

В Проводнике по моделям UML отображается новый класс или интерфейс.

2. Создайте в классе или интерфейсе операции для каждого получаемого линией жизни сообщения.
    a. Выделите все сообщения, которые необходимо включить.
    b. Щелкните одно из сообщений правой кнопкой мыши и выберите **Создать метод**.

Новый класс или интерфейс имеет операции для каждого выделенного сообщения.

Имя операции отображается под стрелкой каждого сообщения и в свойстве сообщения **Операция**.

Если сообщение включает параметры в форме "(параметр : тип)", они отобразятся в списке параметров новой операции.

Примечание

Если в схему последовательностей добавляются новые сообщения, этот шаг необходимо повторить.

3. Чтобы просмотреть подробные сведения о новом классе или интерфейсе, добавьте его в схему классов или компонентов.
    a. Откройте или создайте схему классов или компонентов.
    b. Перетащите новый класс или интерфейс из **Проводника по моделям UML** на схему классов.

Класс или интерфейс появится на схеме классов.

- или -

    c. Перетащите новый интерфейс из **Проводника по моделям UML** на компонент или порт на схеме компонентов.

Интерфейс отображается в компоненте в качестве обозначения без описания операций.

Создание классов параметров

В сообщения на схеме последовательностей можно включить параметры. UML-схему классов можно использовать для описания типов параметров.

[Создание последовательностей взаимодействия с возможностью повторного использования]

Для описания последовательности можно использовать отдельную схему, которая содержит подробные сведения, которые необходимо отделить от других, либо сведения, относящиеся к нескольким схемам.

На одной схеме можно создать прямоугольник использования взаимодействия (12), указывающий на подробные сведения на другой схеме.

Дважды щелкните использование взаимодействия, чтобы открыть связанную с ним схему последовательностей.

Создание последовательности взаимодействий с возможностью повторного использования из существующих линий жизни

1. Щелкните **Использование взаимодействия** на **Панели элементов**.
2. На схеме последовательностей удерживайте нажатой кнопку мыши, перетаскивая по схеме линии жизни, которые необходимо включить в последовательность с возможностью

повторного использования. Начните с выбора положения по вертикали для вставки использования взаимодействия.

Использование взаимодействия отображается напротив выбранных линий жизни на схеме последовательностей.

3. Дважды щелкните имя использования взаимодействия и переименуйте его, чтобы описать результат использования последовательности с возможностью повторного использования на этой схеме.

- или -

Напишите имя в качестве вызова функции, укажите параметры.

4. Свяжите использование взаимодействия с другой схемой последовательностей. Щелкните использование взаимодействия правой кнопкой мыши и выполните одно из следующих действий.

Щелкните **Создать новую последовательность**, чтобы создать новую схему последовательностей.

- или -

Щелкните **Связать с последовательностью**, чтобы связать использование с существующей схемой.

Visual Studio создает связь между использованием взаимодействия и новой последовательностью взаимодействия.

В решении отображается новая схема последовательностей. На ней отображаются линии жизни, с помощью которых было создано использование взаимодействия.

Примечание

Отображаются только те линии жизни, с помощью которых было создано использование взаимодействия. На новой схеме не будут отображаться линии жизни, созданные после использования взаимодействия, даже если использование взаимодействия включает эти линии.

Создание последовательности с возможностью повторного использования из существующих сообщений

- Щелкните сообщение, которое необходимо переместить, правой кнопкой мыши и выберите **Перенести на схему**.

Visual Studio:

- заменяет выделенное сообщение и любые дочерние сообщения использованием взаимодействия;
- перемещает замещенные сообщения на новую схему последовательностей;
- создает связь между использованием взаимодействия и новой схемой последовательности.

Переход к последовательности, на которую ссылается использование взаимодействия

- Дважды щелкните использование взаимодействия.

- или -

Щелкните использование взаимодействия правой кнопкой мыши и выберите **Перейти к последовательности**.

Создание заполнителя с использованием взаимодействия

Можно создать использование взаимодействия, не связывая его с другой схемой. Его можно использовать в качестве заполнителя части последовательности, подробные сведения о которой еще предстоит уточнить. Используйте имя использования взаимодействия, чтобы обозначить желаемый результат.

Сворачивание групп линий жизни

Можно свернуть несколько линий жизни, чтобы группа отображалась, как одна линия. Так можно визуально представить группу объектов как один компонент. Сообщения и использования взаимодействий между линиями жизни свернутой группы скрыты. Сообщения и последовательности взаимодействий, включающие другие линии жизни, отображаются.

Сворачивание группы линий жизни

1. Выберите две или более линий жизни.
2. Щелкните одну из них правой кнопкой мыши и выберите **Свернуть**.

Несколько линий жизни заменяются одной.
Сообщения и использования взаимодействия, включающие только членов этой группы, не отображаются.

3. Щелкните имя, чтобы переименовать группу.

Примечание
Если развернуть группу, имя группы будет утеряно.
Разворачивание свернутой группы

- Щелкните свернутую линию жизни правой кнопкой мыши и выберите **Развернуть**.

Примечание
Имя группы будет утеряно, равно как и любые ссылки группы на комментарии или рабочие элементы.
Описание структур управления с помощью фрагментов

Для определения циклов, ветвей и параллельной обработки на схеме последовательностей можно использовать объединенные фрагменты (13). Эти сведения можно отобразить и на схеме активности. Схема активности позволяет показать сообщения, которыми обмениваются субъекты, менее наглядно, но иногда с помощью схемы активности можно более эффективно представить циклы, ветви и параллелизм.
Создание объединенного фрагмента

1. Выделите сообщение или последовательность сообщений, начинающихся в одном вхождении выполнения или на одной линии жизни.

Примечание
Выделите стрелки сообщений, а не вхождения выполнения, на которые указывают сообщения.

2. Щелкните правой кнопкой мыши одно из сообщений, выберите **Разместить во фрагменте**, затем щелкните требуемый тип фрагмента.

Отображается новый фрагмент. В нем содержатся выбранные сообщения.
Если тип объединенного фрагмента допускает наличие нескольких фрагментов, отображается также пустой фрагмент.

3. Чтобы задать условие для фрагмента, щелкните границу фрагмента правой кнопкой мыши и выберите **Свойства**. Задайте значение для свойства **Условие**.

Условие используется для определения требований к ветви или циклу.

4. Чтобы добавить новый фрагмент в вид, допускающий наличие нескольких фрагментов, щелкните границу фрагмента правой кнопкой мыши и выберите **Добавить**. Щелкните **Операнд взаимодействия до** или **Операнд взаимодействия после**.
5. Чтобы добавить во фрагмент новые сообщения, используйте инструменты создания сообщений, либо скопируйте и ставьте их во фрагмент.

Создание схем последовательностей из кода

В файле кода Visual C# или Visual Basic можно создать схему последовательностей из определения метода.
Созданная схема последовательностей во многом схожа со схемой последовательностей, созданной в проекте моделирования. Однако элементы в созданной схеме последовательностей не отображаются в Проводнике по моделям UML.
Создание схемы последовательностей из кода

1. В Visual Studio откройте файл кода, который содержит определение метода.
2. Щелкните правой кнопкой мыши в любом месте определения метода и выберите **Создать схему последовательностей**.

Примечание
После создания схемы любые изменения, вносимые в схему, не отображаются в коде, а любые

изменения, вносимые в код, не отображаются на схеме. Чтобы отобразить эти изменения, необходимо создать новую схему последовательностей. Можно создавать новую схему последовательностей неограниченное число раз, используя тот же метод.
Можно перейти от линий жизни и сообщений к определениям в коде классов и методов, которые они представляют.

Переход от созданных линий жизни и сообщений к коду

- Щелкните созданную линию жизни или сообщение правой кнопкой мыши и выберите **Перейти к определению**.

Копирование созданных последовательностей в UML-модель

Можно копировать линии жизни, сообщения и другие части созданной последовательности на схему последовательностей проекта моделирования.

Копирование созданной схемы последовательностей в модель UML или из нее

1. На схеме последовательностей выделите элементы, которые необходимо скопировать, например линии жизни и сообщения. Если необходимо скопировать все элементы схемы, щелкните **Выделить все** в меню **Правка**.
2. В меню **Правка** выберите **Копировать**.
3. Создайте или откройте схему последовательностей в проекте моделирования.
4. В меню **Правка** выберите **Вставить**.

Копии выделенных элементов отображаются на схеме.

Примечание

Иногда необходимо изменить цвет вставленных элементов. Выделите их и выберите цвет в окне **Свойства**.

Создание диаграммы классов

В Visual Studio Ultimate для описания типов данных и их связей отдельно от реализации можно использовать *UML-схему классов*. Схема позволяет сконцентрироваться на логических аспектах классов, а не их реализации. .
UML-схему классов можно использовать в разных целях.

- Для предоставления описания типов, используемых в системе и передаваемых между компонентами, независимо от реализации.

Например, тип "Заказ еды" может реализовываться в бизнес-слое в коде .NET, в интерфейсах между компонентами в XML, в базе данных в SQL и в пользовательском интерфейсе в HTML. Несмотря на то что подробности этих реализаций различаются, отношение между типом "Заказ еды" и другими типами, такими как "Меню" и "Оплата", сохраняется. UML-схема классов позволяет обсуждать эти отношения отдельно от реализаций.

- Для более точного определения набора терминов, используемых для обмена сведениями между приложением и его пользователями, а также в описаниях потребностей пользователей.

В качестве примера можно привести описания функциональности пользователей (user story), варианты использования и описания других требований в приложении, обеспечивающем работу ресторана. В этом описании можно найти такие термины как "Меню", "Заказ", "Еда", "Цена", "Оплата" и т. д. Можно создать UML-схему классов, определяющую отношения между этими терминами. Это позволит снизить риск возникновения несоответствий в описаниях требований, пользовательском интерфейсе и справочной документации.

Отношение к другим схемам

UML-схема классов, как правило, создается одновременно с другими схемами моделирования, чтобы предоставить описания используемых типов. В каждом случае физическое представление типов не подразумевается ни одной из схем.

Если создана	используйте UML-схему классов, чтобы описать следующее.

схема активности	тип данных, передаваемых через узел объекта. типы закреплений ввода и вывода и узлы параметров действий.
схема последовательностей	типы параметров и возвращаемые значения сообщений. типы линий жизни. Класс линии жизни должен включать операции для всех сообщений, которые он может получить.
схема компонентов	интерфейсы компонента с перечислением их операций. Полный компонент также можно описать как класс..
схема вариантов использования	типы, упомянутые в описаниях целей и шагов варианта использования..

Основные этапы создания схем классов

Создание UML-схемы классов

1. В меню **Архитектура** выберите пункт **Создать схему**.
2. В разделе **Шаблоны** щелкните **UML-схема классов**.
3. Назовите схему.
4. В области **Добавить в проект моделирования** выделите существующий проект моделирования в решении или выберите **Создать новый проект моделирования** и нажмите кнопку **ОК**.

Новая схема классов отображается на панели элементов **UML-схемы классов**. Панель элементов содержит требуемые элементы и отношения.

Создание UML-схемы классов

1. Чтобы создать тип, выберите инструмент **Класс**, **Интерфейс** или **Перечисление** на панели элементов, затем щелкните пустую область схемы.
2. Чтобы добавить атрибуты или операции в типы, а литералы — в перечисление, щелкните заголовок **Атрибуты**, **Операции** или **Литералы** в типе и нажмите ВВОД.

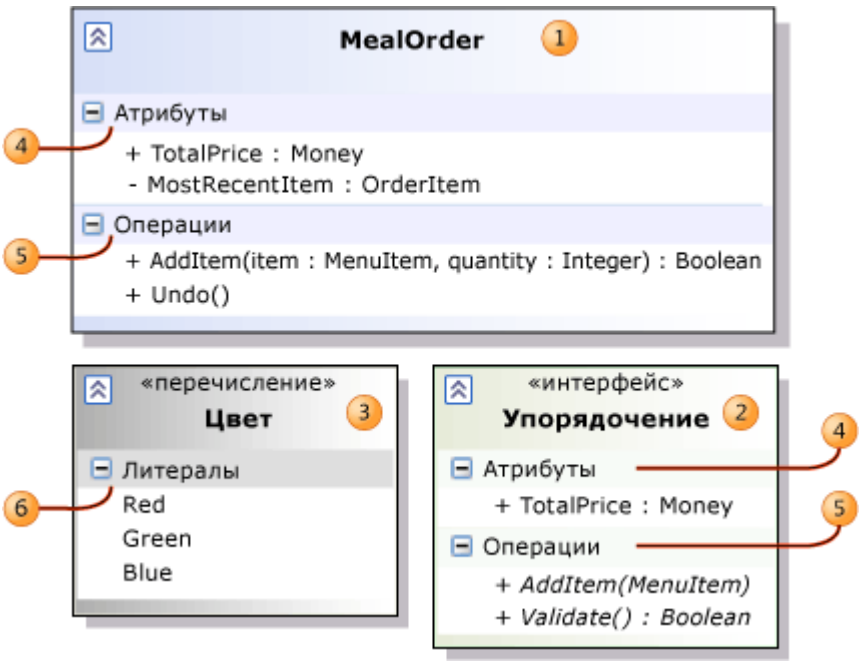
Можно создать сигнатуру, например f(x:Boolean):Integer.

Чтобы быстро добавить несколько элементов, дважды нажмите ВВОД в конце каждого элемента. Чтобы переместить элементы вверх или вниз по списку, можно воспользоваться клавишами со стрелками.

3. Чтобы развернуть или свернуть тип, щелкните значок шеврона в верхней левой части типа. Также можно развернуть и свернуть разделы**Атрибуты** и **Операции** в классе или интерфейсе.
4. Чтобы создать ссылки ассоциаций, наследования или зависимости между типами, щелкните соответствующий инструмент связывания, тип источника, а затем выберите тип целевого объекта.
5. Чтобы создать типы в пакете, создайте пакет с использованием инструмента **Пакет**, затем создайте новые типы и пакеты внутри этого пакета. Чтобы скопировать типы и вставить их в пакет также можно использовать команду копирования.
6. Каждая схема — это представление на модели, которое совместно используется другими схемами того же проекта. Чтобы просмотреть представление всей модели в виде дерева, щелкните **Вид**, выберите **Другие окна** и щелкните **Проводник по моделям UML**.

Использование классов, интерфейсов и перечислений

Существует три стандартных вида классификаторов, которые доступны на панели элементов. В этом документе их называют *типы*.



- В большинстве случаев для представления данных или типов объекта можно использовать **Классы** (1).
- Используйте **Интерфейсы** (2) в контексте, где необходимо различать чистые интерфейсы и конкретные классы, имеющие внутренние реализации.Различать эти сущности полезно при работе со схемами, целью которых является описание реализации программы. При моделировании пассивных данных или определении концептов для описания пользовательских требований это менее эффективно.
- Используйте **Перечисление** (3), чтобы представить тип, имеющий ограниченное число значений литералов, например Stop и Go.
    - Добавление значений литералов в перечисление Дайте каждому отдельное имя.
    - При желании каждому значению литерала также можно присвоить численное значение. Щелкните литерал в перечислении правой кнопкой мыши, выберите **Свойства** и введите число в поле **Значение** в окне **Свойства**.

Дайте каждому типу уникальное имя.

Получение типов из других схем

На UML-схеме классов можно отображать типы из другой схемы.

Тип из другой схемы	Как получать типы из другой схемы
UML-схема классов	Можно отображать класс на нескольких UML-схемах классов. Создав класс на одной схеме, перетащите его из **Проводника по моделям UML** на другую схему. Такой подход эффективен, если необходимо на каждой схеме отобразить определенную группу отношений. Например, можно показать связи между элементами "Заказ еды" и "Меню" ресторана на одной схеме, а связи между элементами "Заказ еды" и "Оплата" — на другой.
Схема компонентов	Если определены компоненты на схеме компонентов, можно перетащить компонент из **Проводника по моделям UML** на схему классов. В этом случае компонент отобразится как класс.
UML-схема последовательностей	На схеме последовательностей из линий жизни можно создавать классы и интерфейсы, а затем перетаскивать класс из**Проводника по моделям**

	UML на UML-схему классов. Каждая линия жизни на схеме последовательностей представляет экземпляр объекта, компонента или субъекта. Чтобы создать класс из линии жизни, щелкните линию жизни правой кнопкой мыши и выберите **Создать класс** или **Создать интерфейс**.

Атрибуты и операции

Атрибут (4) — это именованное значение, которое может быть присвоено каждому экземпляру типа. Осуществление доступа к атрибуту не меняет состояние экземпляра.
Операция (5) — это метод или функция, которая может выполняться экземплярами типа. Она может возвращать значение. Если ее свойство **isQuery**имеет значение true, операция не может изменить состояние экземпляра.
Чтобы добавить атрибут или операцию в тип, щелкните тип правой кнопкой мыши, выберите **Добавить** и щелкните **Атрибут** или **Операция**.
Чтобы просмотреть свойства, щелкните атрибут или операцию правой кнопкой мыши, затем выберите **Свойства**. Свойства отображаются в окне**Свойства**.
Чтобы просмотреть свойства параметров операции, щелкните **[…]** в свойстве **Параметры**. Отобразится новое диалоговое окно свойств.

Типы атрибутов и операций

Можно определить следующие *типы* атрибутов, операций и параметров.

- **(нет)** — можно не задавать тип в сигнатуре, опустив предшествующее двоеточие (:).
- Стандартными типами-примитивами являются следующие: **Boolean**, **Integer** и **String**.
- Тип, определенный в модели.
- Параметризованное значение типа шаблонов, записанное как Template<Parameter>.

Также можно записать имя типа, который еще не был определен в модели. Имя отобразится в разделе **Незаданные типы** в проводнике по моделям UML.

Примечание

Если впоследствии в модели определяется класс или интерфейс этого имени, прежние атрибуты и операции все равно относятся к элементу в разделе "Незаданные типы". Если нужно изменить их так, чтобы они относились к новому классу, необходимо открыть каждый атрибут или операцию и сбросить тип, выбирая новый класс из раскрывающегося меню.

Несколько типов

Можно задать количество элементов любого атрибута, операции или типа параметров. Допустимы следующие значения.

Количество элементов	Атрибут, параметр или возвращаемое значение содержит следующее.
[1]	Одно значение заданного типа. Это значение по умолчанию.
[0..1]	**Null** или значение заданного типа.
[*]	Коллекция, в состав которой может входить неограниченное число экземпляров заданного типа.
[1..*]	Коллекция хотя бы одного экземпляра заданного типа.
[n..m]	Коллекция, в которую входит от n до m экземпляров заданного типа.

Если количество элементов превышает 1, можно задать следующие свойства.

- **IsOrdered** — если значение true, коллекция имеет определенный порядок.
- **IsUnique** — если значение true, в коллекции отсутствуют повторяющиеся значения.

Видимость

Видимость указывает, можно ли получить доступ к атрибуту или операции за пределами определения класса. Допустимы следующие значения.

Имя	Краткая форма	Значение
Открытый	+	Возможен доступ изо всех других типов.
Закрытый	-	Доступ открыт только для внутреннего определения этого типа.
Пакет	~	Возможен доступ только внутри пакета, который содержит данный тип, а также в любых пакетах, явно импортирующих его.
Защищенный	#	Доступ открыт только данному типу и всем типам, которые его наследуют.

Задание сигнатуры атрибута или операции

Сигнатура атрибута или операции — это коллекция свойств, включающая видимость, имя, параметры (для операций) и тип.

Сигнатуру можно создать непосредственно на схеме. Щелкните атрибут или операцию, чтобы выделить элемент, затем повторно щелкните его.

Создайте сигнатуру в следующей форме.

other

visibility attribute-name : Type

- или -

other

visibility operation-name (parameter1 : Type1, ...) : Type

Пример.

other

+ AddItem (item : MenuItem, quantity : Integer) : Boolean

Используйте краткую форму значения свойства visibility. Значение по умолчанию — + (открытый).

Каждый тип может представлять собой типы, определенные в модели, стандартные типы (такие как Integer или String) или имя нового типа, который еще не был определен.

Примечание

Если в списке параметров создается имя без типа, оно представляет собой имя параметра, а не типа. В этом примере MenuItem и Integer являются именами двух параметров с незаданным типами.

AddItem(MenuItem, Integer) /* parameter names, not types! */

Чтобы задать в сигнатуре количество элементов типа, запишите количество элементов в квадратных скобках после имени типа. Например, как показано ниже.

other

+ AddItems (items : MenuItem [1..*])

+ MenuContent : MenuItem [*]

Если атрибут или операция статична, имя атрибута или операции отображается в сигнатуре подчеркнутым. Если атрибут или операция абстрактна, имя отображается курсивом.

Однако свойства **Является статическим** и **Является абстрактным** можно задать только в окне **Свойства**.

Полная сигнатура

При редактировании сигнатуры атрибута или операции в конце строки и после каждого параметра могут отображаться дополнительные свойства. Они отображаются заключенными в фигурные скобки {…}. Эти свойства можно редактировать и добавлять. Пример.

other
+ AddItems (items: MenuItem [1..*] {unique, ordered})
+ GetItems (filter: String) : MenuItem [*] {ordered, query}
Список содержит следующие свойства.

В сигнатуре	Свойство	Значение
unique	**Является уникальным**	В коллекции нет повторяющихся значений. Применимо к типам с количеством элементов больше 1.
ordered	**Является упорядоченным**	Коллекция — это последовательность. Если значение false, не существует определенного первого элемента.Применимо к типам с количеством элементов больше 1.
query	**Является запросом**	Операция не меняет состояние экземпляра. Применимо только к операциям.
/	**Является производным**	Атрибут вычисляется из значений других атрибутов или ассоциаций. "/" отображается перед именем атрибута. Пример. **other** /TotalPrice: Integer

Как правило, полная сигнатура отображается на схеме, только когда она редактируется. По завершении редактирования дополнительные свойства скрываются. Если нужно все время отображать полную сигнатуру, щелкните тип правой кнопкой мыши и выберите **Отображать полную сигнатуру**.
Создание и использование ассоциаций

Используйте ассоциацию, чтобы представить любые виды отношений между двумя элементами, независимо от того, как эта связь реализуется в программе. Например, можно использовать ассоциацию, чтобы представить указатель в C#, отношение в базе данных или перекрестную ссылку одной части XML-файла на другую. Может представлять связь между объектами в реальном мире, например землей и солнцем. Ассоциация не показывает, как представлена ссылка, а только свидетельствует о наличии сведений.
Свойства ассоциации
После создания ассоциации необходимо задать ее свойства. Щелкните ассоциацию правой кнопкой мыши и выберите **Свойства**.
Помимо свойств ассоциации в целом каждая *роль*, т. е. каждое окончание ассоциации, обладает собственными свойствами. Чтобы просмотреть их, расширьте свойства **Первая роль** и **Вторая роль**.
Некоторые свойства каждой роли напрямую видны на схеме. К ним относится следующее.

- Имя роли. Отображается на соответствующем окончании ассоциации на схеме. Его можно увидеть на схеме или в окне **Свойства**.
- **Количество элементов**, значение по умолчанию — **1**. Это значение также отображается на схеме рядом с соответствующим окончанием ассоциации.
- **Агрегат**. Отображается в форме ромбовидной фигуры на одном окончании соединителя. Можно использовать его для указания, что экземпляры в обобщающей роли владеют экземплярами другой роли или содержат их.
- **Является перемещаемым**. Если имеет значение true только для одной роли, в направлении перехода отображается стрелка. С помощью этого свойства можно показать возможности перехода по ссылкам и связи в базе данных в программе.

Возможность перехода

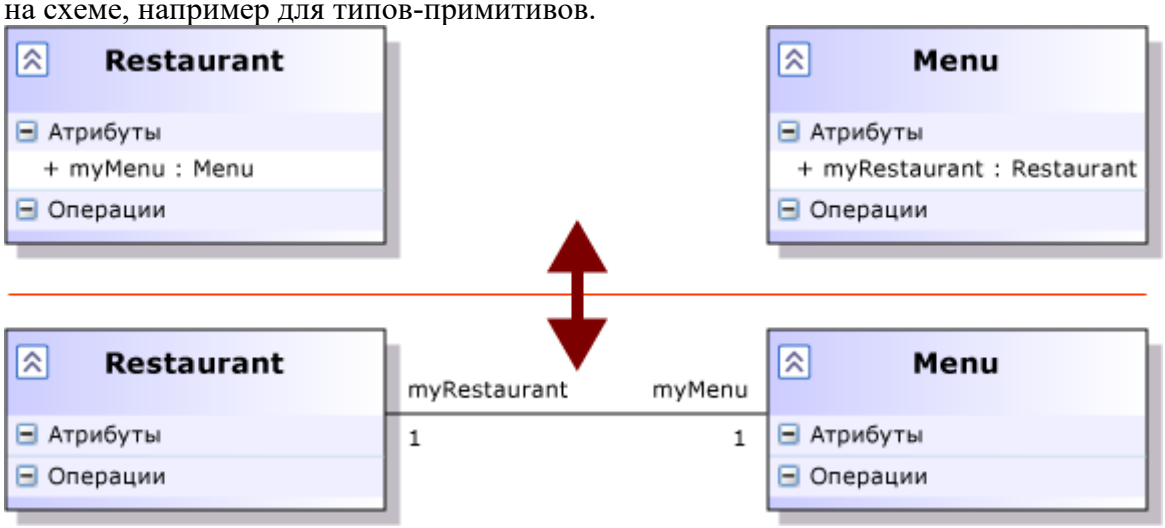
Когда изображается ассоциация, на одном конце у нее стрелка, обозначающая, что ассоциация дает возможность перехода в этом направлении. Это удобно, если схема классов представляет классы ПО, а ассоциации представляют указатели или ссылки. Но если схема классов представляет сущности и отношения или бизнес-концепции, возможность перехода показывать не обязательно. В таком случае можно изображать ассоциации без стрелок. Это можно сделать, задав для свойства **Является перемещаемым** на обоих концах ассоциации значение "true".

Атрибуты и ассоциации

Ассоциация — это графический способ представления атрибута. Например, вместо того чтобы создавать класс "Ресторан" с атрибутом типа "Меню", можно создать ассоциацию из элементов "Ресторан" и "Меню".

Каждое имя атрибута становится именем роли. Оно отображается на противоположном типу-владельцу окончании ассоциации. Например, обратите внимание на myMenu на этой иллюстрации.

Как правило, рекомендуется использовать атрибуты только для типов, которые не отображаются на схеме, например для типов-примитивов.



Наследование

Используйте инструмент **Наследование** для создания следующих отношений.

- Отношение *обобщения* между специализированным типом и общим типом.

- или -

- Отношение *реализации* между классом и реализуемым им интерфейсом.

Невозможно создавать циклы в отношениях наследования.

Обобщение

Обобщение означает, что специализирующий или производный тип наследует атрибуты, операции и ассоциации общего или базового типа.

Общий тип отображается на окончании отношения с наконечником стрелки.

Наследуемые операции и атрибуты, как правило, не отображаются в специализирующих типах. Однако можно добавить наследуемые операции в список операций специализирующего типа. Такой подход эффективен, если необходимо переопределить некоторые свойства операции в специализирующем типе, либо если необходимо указать, что переопределять свойства нужно с помощью реализующего кода.

Переопределение определения операции в специализирующем типе

1. Щелкните отношение обобщения.

Оно отображается подчеркнутым, рядом с ним отображается тег действия.

2. Щелкните тег действия и выберите **Переопределить операции**.

Появляется диалоговое окно **Переопределить операции**.

3. Выделите операции, которые нужно отобразить в специализирующем типе и нажмите кнопку **ОК**.

Выделенные операции теперь отображаются в специализирующем типе.

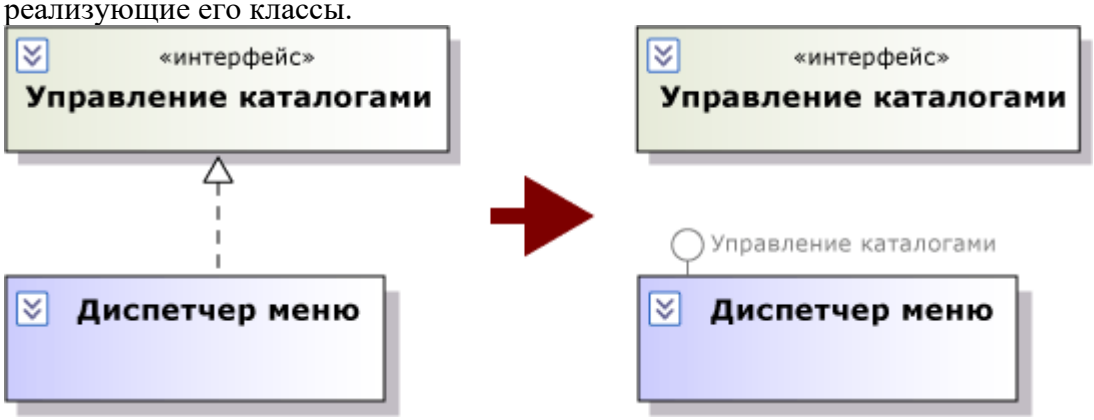
Реализация

Реализация означает, что класс реализует атрибуты и операции, заданные в интерфейсе. Интерфейс находится на окончании соединителя с наконечником стрелки.

При создании соединителя реализации операции интерфейса автоматически реплицируются в реализующем классе. При добавлении в интерфейс новых операций они реплицируются в реализующих классах интерфейса.

После создания отношения реализации можно преобразовать его в обозначение без описания операций. Щелкните отношение правой кнопкой мыши и выберите **Показывать без описания операций**.

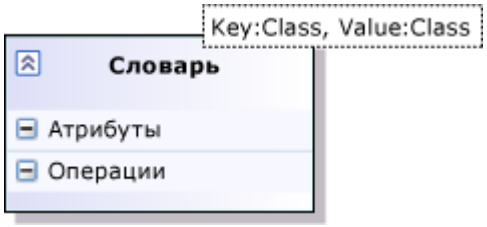
Так можно показать интерфейсы, реализуемые классом, не усложняя схемы классов многочисленными ссылками реализации. Также на отдельных схемах можно показать интерфейс и реализующие его классы.



Типы шаблонов

Можно определить общий тип или тип шаблона, параметры которого задаются другими типами и значениями.

Например, можно создать общий тип Dictionary, параметры которого задаются ключевыми типами и типами значений.



Создание типа шаблонов

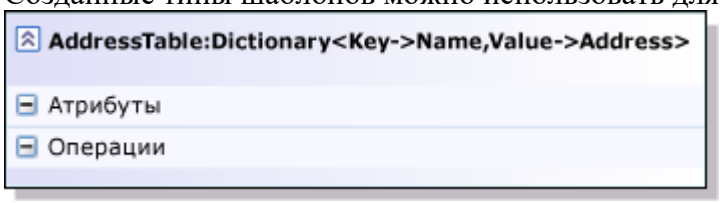
1. Создайте класс или интерфейс. Это ваш тип шаблонов. Присвойте ему соответствующее имя, например Dictionary.
2. Щелкните новый тип правой кнопкой мыши и выберите **Свойства**.
3. В окне **Свойства** щелкните **[…]** в поле **Параметры шаблона**.

Откроется диалоговое окно **Редактор коллекции параметров шаблонов**.

4. Нажмите кнопку **Добавить**.
5. В свойстве "Имя" задайте имя параметра для типа шаблонов, например Key.
6. Задайте значение в поле **Вид параметра**. **Class** — значение по умолчанию.
7. Если нужно, чтобы параметр принимал только производные классы определенного базового класса, задайте в поле **Ограниченное значение**необходимый базовый класс.
8. Добавьте необходимое количество параметров и нажмите кнопку ОК.
9. Добавьте атрибуты и операции в тип шаблонов так же, как при работе с другими классами.

В определении атрибутов и операций можно использовать параметры с видом **Класс**, **Интерфейс** или **Перечисление**. Например, используя классы параметров Key и Value, можно определить эту операцию в Dictionary.

Get(k : Key) : Value
Параметр с видом **Integer** можно использовать в качестве границы количества элементов. Например, максимально допустимое значение параметра Integer можно использовать для определения количества элементов атрибута в виде [0..max].
Созданные типы шаблонов можно использовать для определения привязок шаблонов.



Использование типа шаблонов

1. Создайте новый тип, например AddressTable.
2. Щелкните новый тип правой кнопкой мыши и выберите **Свойства**.
3. В свойстве **Привязка шаблона** выберите тип шаблона, например Dictionary, из раскрывающегося списка.
4. Разверните свойство **Привязка шаблона**.

Отображается строка для каждого параметра типа шаблонов.

5. Задайте подходящее значение для каждого параметра. Например, задайте для параметра Key класс Name.

Пакеты

На UML-схеме классов можно просматривать пакеты. Пакет — это контейнер для других элементов модели. Внутри пакета можно создать любой элемент.На схеме элементы внутри пакета перемещаются по схеме, если перемещается пакет.
Чтобы скрыть или отобразить содержимое пакета, можно использовать элемент управления "развернуть/свернуть".

Создание диаграммы деятельности

В Visual Studio Ultimate можно изображать схемы активности, описывающие бизнес-процесс или программный алгоритм как рабочий процесс, состоящий из ряда действий. Эти действия могут выполнять люди, программные компоненты или устройства.
Схемы активности можно использовать для различных целей.

- Для описания бизнес-процесса или рабочего процесса, в котором участвуют пользователи и система.
- Для описания шагов варианта использования.
- Для описания метода, функции или операции программного обеспечения.

Изображение схемы активности может помочь улучшить процесс. Если схема существующего процесса оказывается очень сложной, можно продумать пути упрощения процесса.
Изображая схему активности для описания бизнес-процесса или способа применения системы пользователями, можно создать также схему вариантов использования, чтобы показать различные точки зрения на информацию. На схеме вариантов использования действия изображаются как варианты использования. Присвойте вариантам использования имена, совпадающие с именами соответствующих действий. Представление вариантов использования позволяет выполнять следующее.

- С помощью отношения Includes показывать на одной схеме, как большие действия или варианты использования состоят из меньших.
- Явно привязывать каждое из действий или вариантов использования к пользователям или внешним системам, участвующим в его выполнении.
- Проводить границы вокруг действий или вариантов использования, поддерживаемых системой или основными ее компонентами.

Также можно использовать схему активности для подробного описания операции программы.
На схеме активности можно показать поток данных, передаваемых между действиями. Схема

активности, однако, не описывает структуру данных. **Основные этапы создания схем активности**

Создание схемы активности

1. В меню **Архитектура** выберите пункт **Создать схему**.
2. В разделе **Шаблоны** щелкните **Схема активности UML**.
3. Назовите схему.
4. В области **Добавить в проект моделирования** выберите существующий проект моделирования в решении или выберите пункт **Создать проект модели**.

Изображение элементов схемы активности

1. Перетаскивайте элементы из панели элементов на схему.

Начните с размещения на схеме основных действий, соедините их и закончите добавлением таких элементов, как начальные и конечные узлы.

Примечание

Нельзя перетаскивать на схему существующие элементы из обозревателя моделей UML.

2. Чтобы подключить элементы, выполните следующие действия.
    a. На панели элементов **Схема активности** щелкните пункт **Соединитель**.
    b. Щелкните на схеме исходный элемент.
    c. Щелкните целевой элемент.

Примечание

Чтобы использовать средство несколько раз, дважды щелкните данное средство.

Перемещение действия в другой пакет

- В **обозревателе моделей UML** перетащите действие в пакет.

- или -

- В **обозревателе моделей UML** щелкните действие правой кнопкой мыши и выберите команду **Вырезать**. Щелкните пакет правой кнопкой мыши и выберите команду **Вставить**.

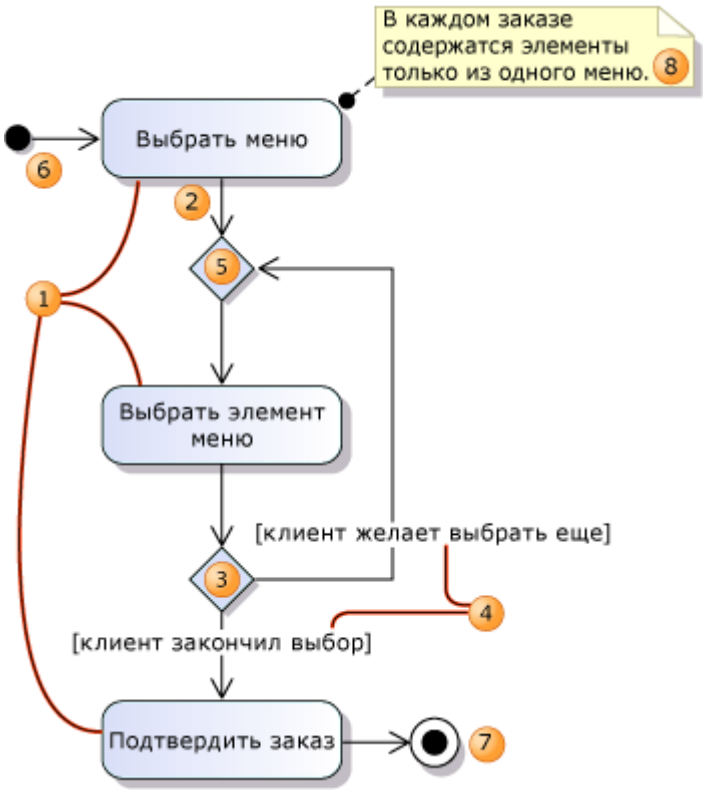
Примечание

Это действие отобразится в обозревателе моделей UML только при добавлении первого элемента на схему.

Описание потока управления

Схема активности описывает бизнес-процесс или программный алгоритм как последовательность действий. Стрелки-соединители показывают, как управление последовательно передается от одного действия к другому. Обычно действие может начаться только после завершения предыдущего действия.

На следующем рисунке приведен пример, как можно показывать последовательность действий с помощью действий, соединителей, ветвей и циклов.Более подробно элементы описаны в следующих разделах.



Схемы активности используют **Действия** и **Соединители**, чтобы описывать систему или приложение как последовательность действий, последовательно принимающих управление.

- Создайте **Действие** (1) для каждой основной задачи, выполняемой пользователем, системой или ими обоими.

Примечание

Попробуйте описать процесс или алгоритм всего несколькими действиями. С помощью **Действий вызова поведения** можно определять каждое действие подробнее на отдельных схемах.

- Убедитесь, что заголовок каждого действия четко показывает, зачем оно нужно.
- Свяжите действия в последовательность с помощью **Соединителей** (2).
- Каждое действие заканчивается до начала следующего действия в потоке управления. Если нужно описать перекрывающиеся действия, воспользуйтесь **Узлом ветвления**.

Хотя схема описывает последовательность действий, она не описывает то, как действия выполняются и как управление передается от одного действия к следующему. Если с помощью схемы изображается бизнес-процесс, управление может передаваться, например, при отправке сообщения электронной почты одним человеком другому. Если с помощью схемы иллюстрируется система программного обеспечения, управление может передаваться при нормальном потоке выполнения от одного оператора к другому.

Описание решений и циклов

- **Узел решения** (3) используется, чтобы указать точку, в которой результат решения определяет следующий шаг. Можно добавить любое количество исходящих путей.
- Если с помощью схемы активности определяется часть приложения, следует определить условия (4), чтобы было понятно, когда следует использовать каждый из путей. Щелкните соединитель правой кнопкой мыши, выберите **Свойства** и в окне **Свойства** введите значение в поле**Условие**.
- Не всегда необходимо определять условия. Например, если схема активности используется для описания бизнес-процесса или протокола взаимодействия, ветвь определяет диапазон параметров, доступных для пользователя или взаимодействующих компонентов.
- **Узел слияния** (5) используется для объединения двух или нескольких альтернативных потоков, ветвящихся в **Узле решения**.

Примечание

Для объединения альтернативных потоков следует использовать **Узел слияния**, а не соединять

потоки в действии. В приведенном примере будет неправильно сделать соединитель прямо от узла решения к **Выбор пункта меню**. Это связано с тем, что действие не запускается, пока потоки управления не поступили во все входящие соединители. Поэтому объединять в действии следует только параллельные потоки.

- Циклы можно описывать с помощью ветвей, как показано в примере.

Примечание

Попробуйте вкладывать циклы структурированно, как в коде программы. Если вы описываете существующий бизнес-процесс, это может показать некоторые возможности для его улучшения.

Запуск действия

Точки входа в действие можно указывать двумя способами.

- **Начальный узел**

Создайте один **Начальный узел** (6), чтобы указать первое действие действия.

Этот метод лучше всего подходит при описании вложенных действий или в случаях, когда не нужно явно указывать, что запускает действие. Например, действие "Заказ блюда" определенно начинается с того, что клиент проголодался.

- **Узел события получения**

Создайте **Узел события получения**, чтобы указать начало потока, реагирующего на определенное событие, например ввод данных пользователем. Не предоставляйте входящий в узел поток. Пропуск входящего потока означает, что поток запускается при каждом возникновении события.

Этот метод особенно полезен, если нужно описать ответ на определенное внешнее событие.

Завершение действия

Используйте **Конечный узел действия** (7), чтобы обозначить конец действия.

- При достижении потоком управления **Конечного узла действия** все параллельные и вложенные действия данного действия завершаются.
- Можно использовать несколько конечных узлов действия, чтобы уменьшить число лишних соединителей.

Прерывание действия

Чтобы описать, как можно прерывать обычный потока действий, например, если пользователь решает отменить процесс, можно создать узел события получения, прослушивающий это событие. Создайте поток управления из этого узла к конечному узлу действия (7).

Дорожки

Иногда полезно разделить части действия на области, соответствующие различным объектам или бизнес-ролям, выполняющим действия. Эти области оформляются в виде столбцов и называются *дорожками*.

- Используйте линии или прямоугольники в разделе панели элементов **Простые фигуры**, чтобы изображать дорожки или другие области.
- Чтобы подписать дорожку, создайте примечание и присвойте его свойству **Прозрачный** значение **True**.

Простые фигуры не являются частью UML-модели и не отображаются в обозревателе моделей UML.

Описание потока данных

Данные, передающиеся в или из действия, можно описать одним из двух следующих способов.

- С использованием **Узла объекта**. Это простейший способ описания информации, передаваемой между действиями. Узел объекта похож на переменную в программе. Он представляет элемент, хранящий одно или несколько значений, передаваемых из одного действия в другое.
- С использованием **Закрепления вывода** и **Закрепления ввода**. Этот метод позволяет отдельно описывать выходные данные одного действия и входные данные другого. Закрепления похожи на параметры в программе. Закрепления представляют порты, в которых объекты могут входить и выходить из действия.

Описание потока данных с помощью узлов объектов

Большинство потоков управления передает данные. Например, выходной поток действия "Клиент предоставляет сведения" передает ссылку на адрес поставки.

Если нужно описать эти данные на схеме, можно заменить соединитель узлом объекта и двумя соединителями, как показано на следующем рисунке.

Обратите внимание, что прямоугольники с закругленными углами, например Dispatch Goods (отправка товаров), представляют действия, в которых происходит обработка. Прямоугольники с прямыми углами, например Shipment Address (адрес поставки), представляют потоки объектов из одного действия в другое.

Присвойте узлу объекта имя, отражающее роль узла как передаточного звена или буфера для объектов, передаваемых между действиями.

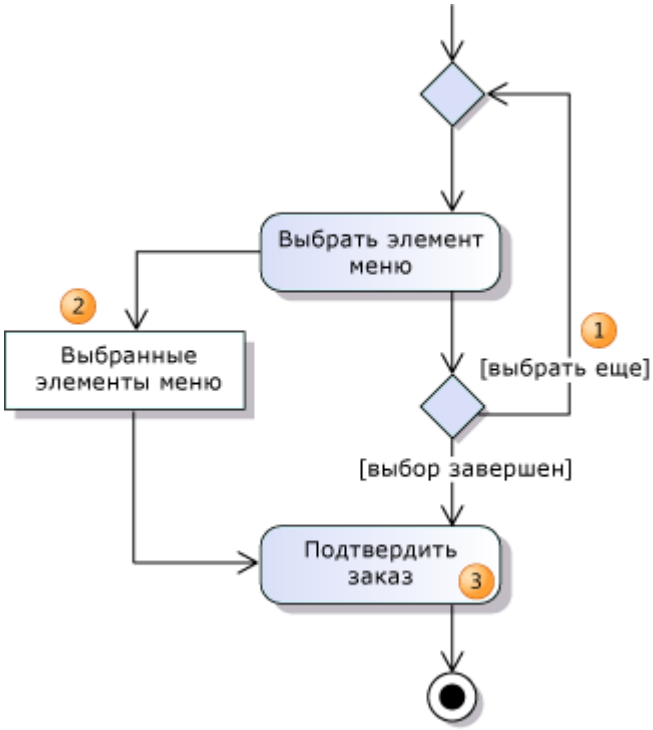
Тип узла объекта можно задать в окне свойств. Это может быть примитивный тип, например Integer, или класс, интерфейс или перечисление, определенные на схеме классов. Например, можно создать класс Shipment Address (адрес поставки) с атрибутами Street Address (улица), City (город) и так далее, связав его с другим классом под названием Customer (клиент).

Примечание

Если ввести имя типа, который еще не определен, в раздел **Неуказанные типы** в обозревателе моделей UML будет добавлен элемент. Если после этого определить тип с таким именем на схеме классов, необходимо сбросить тип узла объекта, чтобы он ссылался на новый тип.

Буферизация данных в узлах объектов

Узел объекта может действовать как буфер для нескольких объектов. На следующем рисунке поток управления показывает, что пользователь может пройти цикл [choose more] (выбрать еще) (1) много раз, а узел объекта Chosen Menu Items (выбранные элементы меню) (2) накапливает решения пользователя. Наконец, когда пользователь завершает выбор, управление передается действию Confirm Order (подтверждение заказа) (3), которое принимает полный список решений из буфера Chosen Menu Items.



Можно указать способ хранения элементов в буфере, установив следующие свойства узла объекта.

- Установите следующие значения для свойства **Упорядочение**.
  - **Unordered**, чтобы задать произвольный порядок или отсутствие порядка. (По умолчанию).
  - **Ordered**, чтобы задать порядок, соответствующий определенному ключу.
  - **Fifo**, чтобы задать порядок "первое на входе — первое на выходе".
  - **Lifo**, чтобы задать порядок "последнее на входе — первое на выходе".
- Установите свойство **Верхняя граница**, чтобы задать максимальное число объектов в буфере. Значение по умолчанию — *. Это означает отсутствие ограничения.

Описание потока данных с помощью закреплений ввода и вывода

С помощью **Закрепления вывода** и **Закрепления ввода** можно отдельно описать вывод из одного действия и ввод в другое.

Чтобы создать закрепление, нажмите кнопку **Закрепление ввода** или **Закрепление вывода** на панели элементов и выберите действие. Затем можно переместить закрепление по периметру действия и изменить его имя. Можно создавать закрепления ввода и вывода для любых действий, включая**Действия вызова поведения**, **Действия вызова операции**, **Действия отправки сигнала** и **Действия принятия события**.

Соединитель между двумя закреплениями представляет поток объектов, как и потоки из и в узел объекта.

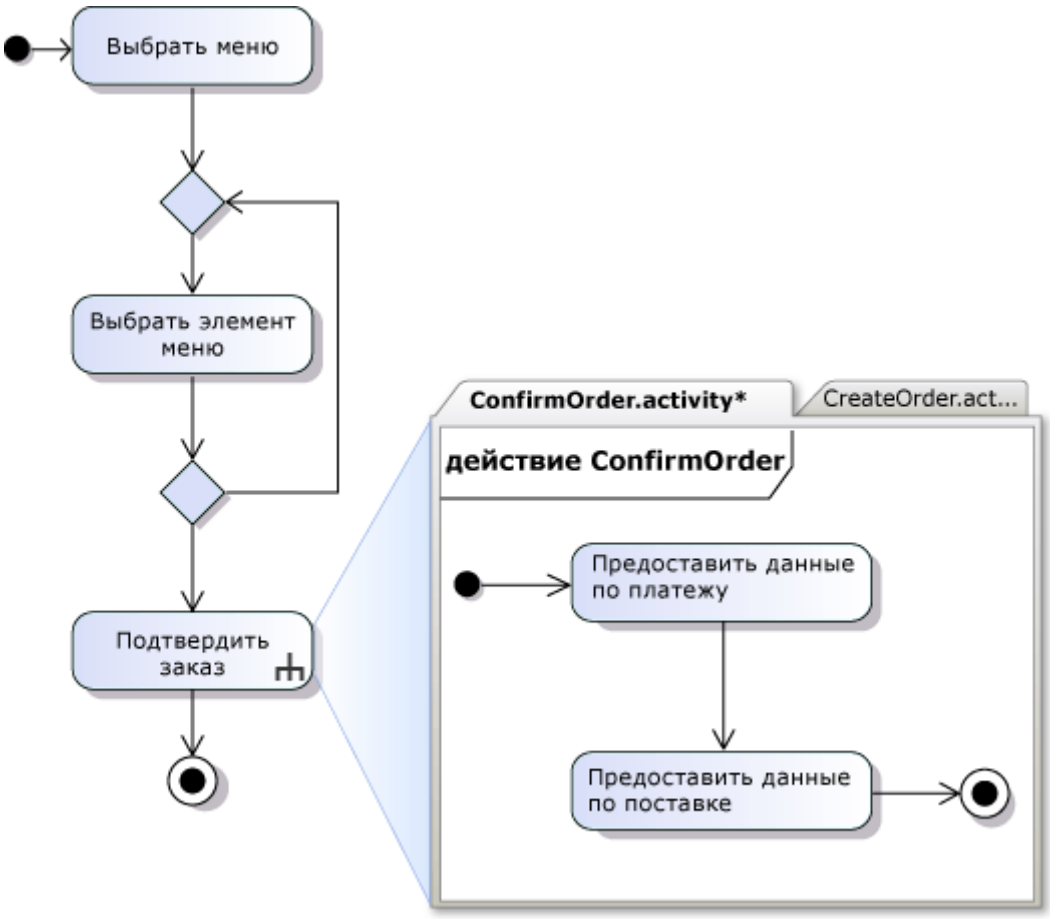
Присваивайте закреплениям имена, указывающие на роль объектов, которые они создают или

принимают, например имена параметров.
Можно задать тип передаваемых объектов в свойстве **Тип**. Это должен быть тип, созданный на UML-схеме классов.
Объекты, передающиеся между соединенными закреплениями, должны быть как-либо совместимы. Например, объекты, создаваемые закреплением вывода, могут принадлежать к производному типу типа закрепления ввода.
Также можно указать, что поток объектов включает преобразование, которое изменяет данные из типа закрепления вывода в тип закрепления ввода.Наиболее распространенные преобразования такого рода просто извлекают подходящую часть из большего типа. В примере на рисунке подразумевается преобразование, которое извлекает Shipping Address (адрес доставки) из Order Detail (сведений заказа).
Более подробное определение действий

Кроме имени, позволяющего описать получаемый действием результат, есть несколько способов описать действие подробнее.

- Написать более подробное описание в свойстве **Основная часть**. Например, можно написать фрагмент программного кода или псевдокода или полное описание получаемых результатов.
- Заменить действие действием вызова поведения и подробно описать его поведение на отдельной схеме активности.
- Установите свойства действия **Локальные постусловия** и **Локальные предусловия**, чтобы описать его результат подробнее

Описание вложенных действий с помощью действий вызова поведения
Можно подробно описать поведение действия с помощью отдельной схемы активности. Вызываемое поведение — это схема активности, представленная на основной схеме активности действием вызова поведения. Действие вызова поведения можно также использовать для описания поведения, используемого различными действиями совместно, чтобы не изображать одно вложенное действие несколько раз.
На следующем рисунке Diagram 1 (схема 1) показывает действие, содержащее действие вызова поведения, а Diagram 2 (схема 2) показывает схему вложенного действия, отображающую вызываемое поведение.



Описание вложенных действий с помощью действий вызова поведения

1. Чтобы создать схему вложенного действия, щелкните проект моделирования правой кнопкой мыши в **обозревателе решений** и последовательно выберите **Добавить** и **Новый элемент**.
2. В диалоговом окне **Добавление нового элемента** в разделе **Шаблоны** щелкните пункт **Схема активности** и в поле **Имя** введите имя **Действия вызова поведения**.
3. Подробно изобразите рабочий процесс вложенного действия. Это называется вызываемым поведением.
    - На схеме вызываемого вложенного действия **Начальный узел** указывает место, с которого начинается управление при вызове данного поведения. **Конечный узел действия** показывает, где управление возвращается к родительскому действию.
4. Установите для свойства **Поведение** действия **Действие вызова поведения** значение ссылки на схему вызываемого поведения.

Примечание

Схема вложенного действия должна содержать элементы; в противном случае она не будет доступна в раскрывающемся списке для свойства **Поведение**. Кроме того, значок трезубца не появится на фигуре **Действие вызова поведения**, если не установить его свойство **Поведение**.

5. Установите для свойства **Является синхронным** действия значение, указывающее, ожидает ли действие завершения вызванного действия.
    - Значение **Является синхронным**, равное false, означает, что поток может переходить к следующему действию до завершения вызванного действия. Не следует определять закрепления вывода или исходящие потоки данных из действия.

Описание потоков данных, входящих и исходящих из вложенных действий

Описание данных, передаваемых в и из вложенных действий, сходно с использованием параметров в программном обеспечении.

- Создайте закрепления ввода и вывода (1) для действия вызванного поведения для каждого фрагмента данных, передаваемого в или из действия. Назовите их.

- На схеме вложенного действия создайте **Узел параметра действия** (2) для каждого закрепления ввода и вывода вызывающего действия. Присвойте каждому узлу имя, совпадающее с именем соответствующего закрепления.

Примечание

Узел параметра действия похож на узел объекта. Чтобы проверить тип узла, щелкните правой кнопкой мыши узел и нажмите кнопку **Свойства**. Тип узла отображается в заголовке окна свойств.

- На схеме вложенного действия проведите соединители, показывающие поток объектов в или из каждого из узлов параметров действий.

Определение постусловий и предусловий

Можно использовать свойства **Локальные постусловия** и **Локальные предусловия**, чтобы подробно описать результаты действия. Эти свойства описывают результат действия, не описывая способ достижения этого результата.

Чтобы установить эти свойства, щелкните правой кнопкой мыши действие и нажмите кнопку **Свойства**. Введите значения свойств в окне свойств.

Локальные постусловия

Постусловие — это условие, которое должно выполняться, прежде чем действие может считаться завершенным. В примере действия Confirm Order (подтверждение заказа) постусловием может быть, например, следующее.

Клиент предоставил полные сведения в допустимом формате, которые необходимы для обработки данных кредитной карты пользователя.

Постусловие может выражать отношение между состояниями до и после выполнения действия. Пример.

Процентная ставка удвоилась.

Можно писать постусловия в более формальном стиле, ссылаясь на определенные атрибуты данных, с которыми работают действия. Пример.

InvoiceTotal == Sum(OrderItem.MenuItem.Price)

Локальные предусловия

Предусловие — это условие, которое должно быть удовлетворено в момент начала

действия. Например, у действия Confirm Order (подтверждение заказа) может быть следующее предусловие.
Пользователь выбрал хотя бы один элемент из меню.

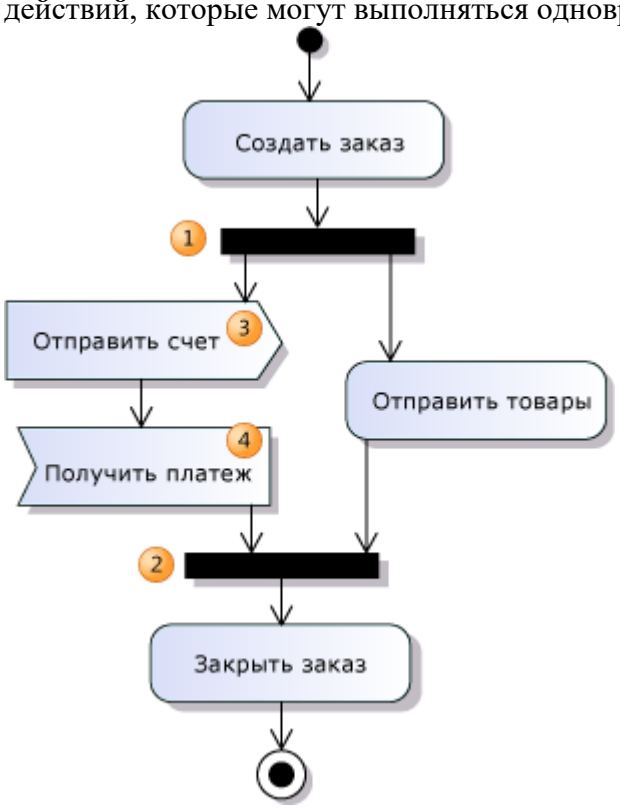
Описание вызовов к операциям

Как правило, действие описывает работу, выполняемую любым сочетанием людей, программ и компьютеров. Однако с помощью действия вызова операции можно описывать вызовы к определенным методам или функциям программы.

- Задайте имя действия вызова операции, указывающее, какая операция и для какого объекта или компонента вызывается.
- Добавьте к действию вызова операции закрепления ввода и вывода, чтобы описать параметры и возвращаемые значения.
- Можно установить значение свойства **Является синхронным** для действия, чтобы указать, ожидает ли действие завершения операции.
    - Значение **Является синхронным**, равное false, означает, что поток может переходить к следующему действию до завершения вызванной операции. Не следует определять закрепления вывода или исходящие потоки данных из действия.

Параллельные потоки

С помощью **Узлов ветвления** и **Соединительных узлов** можно описывать два или более потоков действий, которые могут выполняться одновременно.



Узел ветвления (1) разделяет поток управления на два или более потока. После завершения предыдущего действия могут запускаться все действия на выходе ветвления.
Соединительный узел (2) объединяет параллельные потоки. Действие после **Соединительного узла** не может начаться до завершения всех действий, входящих в **Соединительный узел**.

Описание сигналов и событий

Шаг в процессе, отправляющий сигнал, можно показать как действие отправки сигнала в действии. Шаг, ожидающий конкретного сигнала или события перед продолжением работы, можно показать как действие принятия события.
Например, можно показать шаг, отправляющий заказ, и другой шаг, который должен получить заказ перед обработкой этого заказа.

Отправка сигнала

Действия отправки сигнала (3) используются, чтобы указать, что какой-либо сигнал или сообщение отправлено другим действиям или процессам.Используйте имя действия, чтобы указать, какой тип сообщений оно отправляет.

- Управление немедленно передается следующему действую в потоке управления (при его наличии).
- Действие отправки сигнала нельзя использовать для описания ответа процесса на возвращаемую информацию. Для этого нужно использовать отдельное действие принятия события.
- Можно показать входящий в действие отправки сигнала поток данных, чтобы указать, какие данные могут отправляться в исходящих сообщениях.

Ожидание сигнала или события

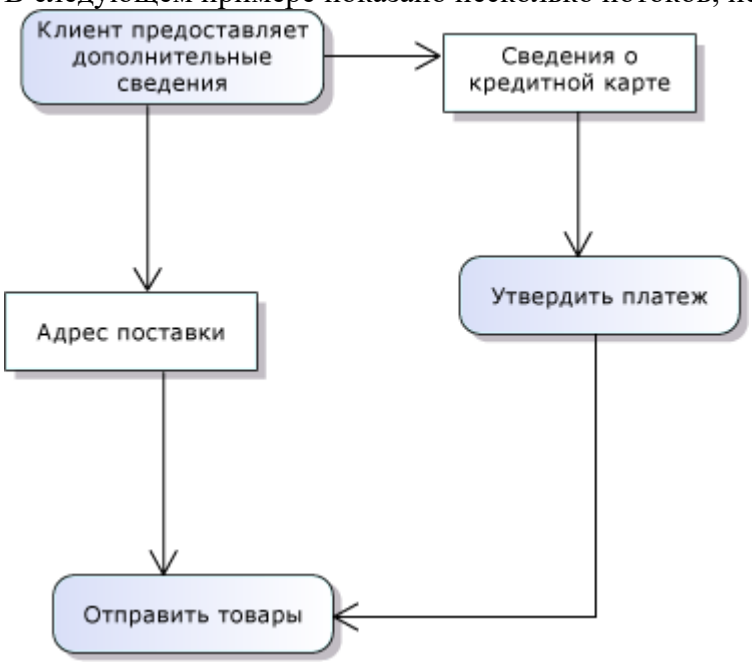
Действия принятия события (4) используются, чтобы указать, что данное действие ожидает некоторого внешнего события или входящего сообщения. Используйте имя действия, чтобы указать, какой тип сообщений оно ожидает.

- Чтобы показать, что действие ожидает внешнего события или сообщения в определенной точке потока, в подходящем месте действия изобразите действие принятия события с входящим потоком.
- Чтобы показать, что действие может отвечать на внешнее событие или сообщение в любое время, изобразите действие принятия события без входящего потока. При возникновении указанного внешнего события, в действии начнется новый поток, начинающийся с действия принятия события.
- Действия принятия события нельзя использовать для описания значений, возвращаемых отправителю сигнала. Для этого используется отдельное действие отправки сигнала.
- Можно показать исходящие из действия потоки данных, чтобы продемонстрировать, как действие обрабатывает данные, получаемые с сигналом.Если нужно показать несколько исходящих потоков, следует установить свойство **IsUnmarshall** действия принятия события, указывающее, что действие при анализе разделяет входящий сигнал на компоненты.

Описание нескольких потоков данных

Можно изобразить несколько потоков управления или потоков объектов, исходящих из действия, чтобы указать, что после завершения действия возникает несколько потоков. Это похоже на ветвление, за исключением того, что можно использовать одновременно и потоки управления, и потоки объектов.

В следующем примере показано несколько потоков, исходящих и входящих в действия.



После завершения действия "Customer provides details" (клиент предоставляет сведения) оно создает два объекта: "Shipment address" (адрес доставки) и "Credit card details" (сведения о кредитной карте). Эти два объекта затем обрабатываются различными действиями. Поскольку действие нуждается во всех входных данных до начала выполнения, последнее действие не начинается, пока не завершатся все ведущие к нему действия.

Потоки

С помощью схемы активности можно описывать конвейер или ряд действий, выполняющихся одновременно и постоянно передающих данные от действия к действию.

Суть следующего примера в том, что все действия могут создавать объекты и продолжать работу. Так как здесь нет потоков управления, действия могут начинаться сразу после получения первого объекта.

Следует обратить внимание, что соединители в этом примере являются потоками объектов, так как по меньшей мере один конец каждого из них находится в узле параметра действия, узле объекта или закреплении ввода или вывода.

1. В данном примере — три узла параметра действия, представляющие его вводы и выводы.
2. Узлы объектов и закрепления ввода и вывода могут представлять буферы. Можно задать свойство "верхняя граница" узла объектов, чтобы ограничить число объектов, одновременно находящихся в буфере.
3. С помощью узлов решений можно показать, что поток разделяется и отправляет различные объекты по различным ветвям. Можно использовать примечания или заголовки узлов для объяснения критериев разделения.
4. С помощью узлов ветвления можно показать, что создается две или более копии объектов, отправляемые в параллельную обработку.
5. С помощью соединительных узлов можно показать, что два потока обработки вновь сливаются.

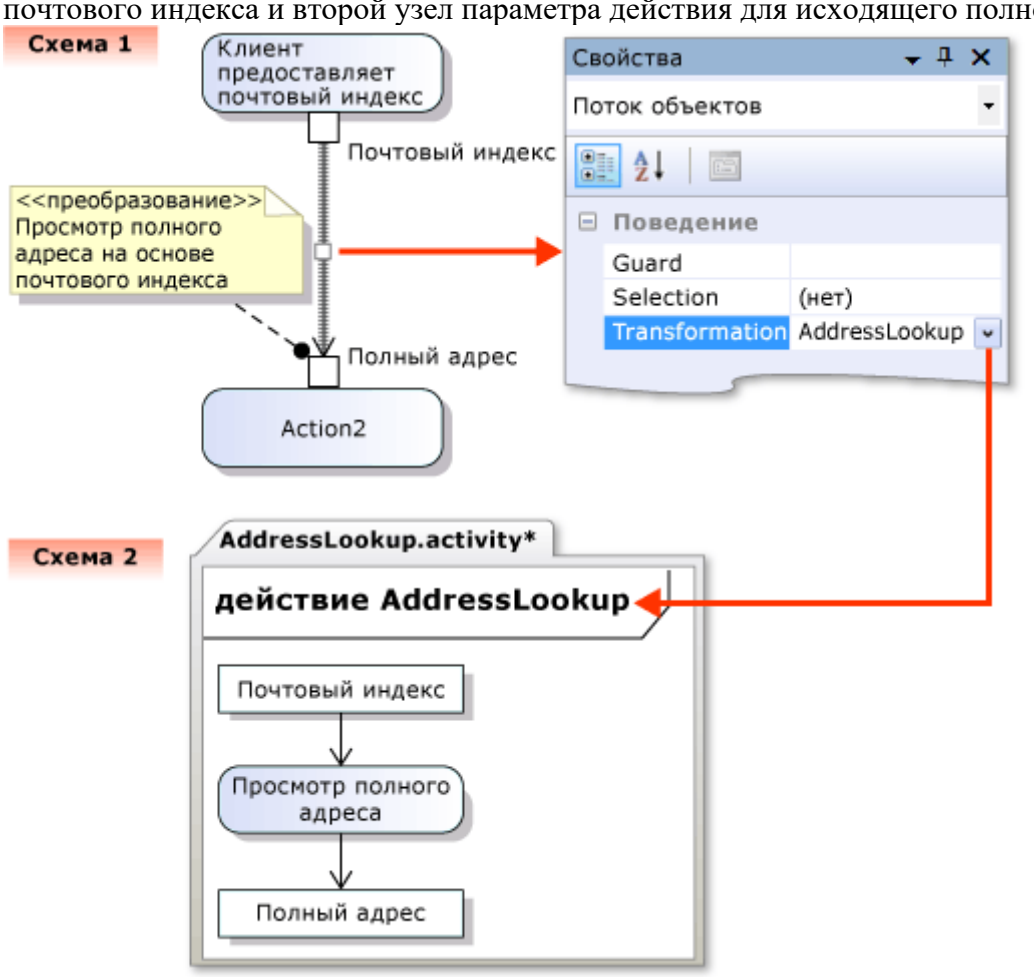
Выделение и преобразование

Можно указать, что объекты в потоке объектов преобразуются, выбираются или с ними

происходит и то, и другое. Поток объектов — это поток к или от закрепления или узла объекта.

- Преобразование описывает, как объекты, входящие в поток, преобразуются в другой тип.
- Выбор описывает, каким образом только часть объектов, входящих в поток, передается принимающему действию.

В примере показано преобразование. Первое действие на Diagram 1 (схеме 1) создает почтовый индекс на закреплении вывода. Он соединен с закреплением ввода второго действия. Однако второе действие ожидает полный адрес. Преобразование из одного типа в другой задано во втором действии, Address Lookup (поиск адресов). На него ссылается свойство "преобразование" потока объектов. Действие Address Lookup содержит один узел параметра действия для входящего почтового индекса и второй узел параметра действия для исходящего полного адреса.



Преобразование или выбор можно задать одним из двух способов.

- Присоединить примечание к закреплению ввода или вывода.
  - Чтобы отличить это описание от обычного примечания, начните его со слова <<преобразование>> или <<выбор>>.
- Описать преобразование или выбор подробнее на отдельной схеме активности.
  - При использовании этого способа также присоедините примечание, чтобы читатели поняли, что определено преобразование.

Описание преобразования или выбора на отдельной схеме активности

1. Создайте новую схему активности для описания потока преобразования или выбора.
   - В **обозревателе решений** щелкните проект правой кнопкой мыши, выберите команду **Добавить** и последовательно щелкните **Новый элемент** и **Схема активности**. Присвойте схеме имя, подходящее для потока преобразования или выбора. Нажмите кнопку **Добавить**.
2. На новой схеме выполните следующие действия.
   - Создайте два узла параметра действия, один для входящего потока, второй для вывода.

- Создайте действия, связанные потоками объектов. Это показывает, как работает преобразование или выбор.

3. На любой схеме, где нужно использовать преобразование или выбор, выполните следующие действия.
   - Создайте поток объектов, то есть соединитель в или из закрепления ввода или вывода, узел объектов или узел параметра действия.
   - Щелкните поток объектов правой кнопкой мыши и выберите пункт **Свойства**.
   - В свойстве **Преобразование** или **Выделение** выберите схему, на которой определяется поток преобразования или выбора.

Можно также определить выбор для узла объекта и отдельных закреплений ввода и вывода. Определите действие выбора, как в предыдущей процедуре, и установите свойство **Выделение** узла объекта или закрепления ввода или вывода.

Создание диаграммы компонентов

В Visual Studio Ultimate можно создать *схему компонентов*, чтобы показать структуру программной системы.

Компонент — это модульная единица, заменяемая в пределах среды. Его внутренние составляющие скрыты, но доступ к функциям компонента можно получить с помощью одного или нескольких четко определенных *предоставленных интерфейсов*. Компонент также может иметь *требуемые интерфейсы*. Требуемый интерфейс определяет, какие функции и службы он требует от других компонентов. Объединив предоставленные и требуемые интерфейсы нескольких компонентов, можно создать более крупный компонент. Можно сказать, что вся программная система, по сути, представляет собой компонент.

Создание схем компонентов имеет несколько преимуществ.

- Анализ основных элементов системы позволяет команде разработчиков понять принципы существующей системы и создать новую.
- Представление системы в качестве коллекции компонентов с четко определенными предоставленными и требуемыми интерфейсами позволяет более эффективно разделить компоненты. В свою очередь, это облегчает понимание конструкции системы и ее корректирование при изменении требований.

Можно использовать схему компонентов, чтобы представить конструкцию системы независимо от того, какой язык или платформа используется сейчас или будет использоваться в будущем.

Отношение к другим схемам

Можно использовать схему компонентов совместно с другими схемами.

Другая схема	Помогает обсуждать следующие аспекты конструкции и передавать сведения о них
UML-схема последовательностей	• Взаимодействия между компонентами системы • Взаимодействия между частями внутри компонента.
UML-схема классов	• Интерфейсы компонента и классы, формирующие его части. • Данные, отправляемые в параметрах по интерфейсам компонентов.
Схемы активности	• Внутренние обработки, выполняемые компонентом в ответ на входящие сообщения.
Схемы слоев	• Логические архитектурные уровни компонентов.

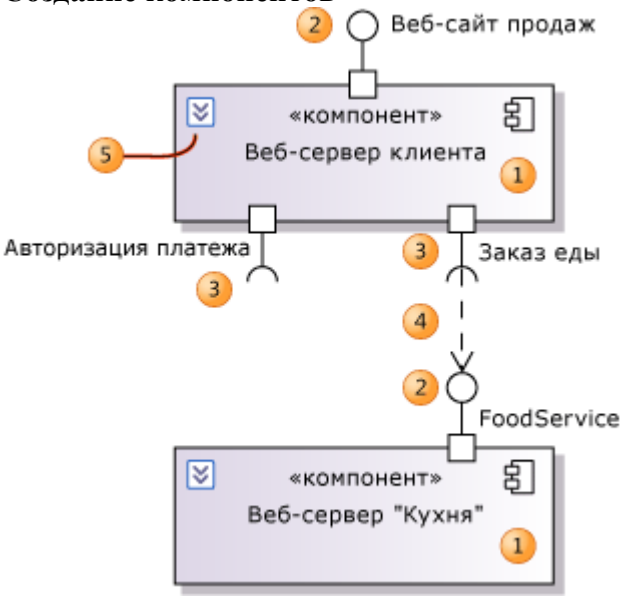
Основные этапы создания схем компонентов

Создание схемы компонентов

1. В меню **Архитектура** выберите пункт **Создать схему**.
2. В разделе **Шаблоны** щелкните **UML-схема компонентов**.
3. Назовите схему.
4. В области **Добавить в проект моделирования** выделите существующий проект моделирования в решении или выберите **Создать новый проект моделирования** и нажмите кнопку **ОК**.

Новая схема компонентов отображается на панели элементов **Схема компонентов** UML. Панель элементов содержит требуемые элементы и отношения.

Создание компонентов



Создайте *компонент* (1) для каждой крупной функциональной единицы в системе или приложении.

В качестве примеров можно привести приложение, аппаратное средство, веб-службу, сборку .NET, программный класс или группу классов или любой другой отделимый сегмент программы.

Создание компонентов

1. Щелкните **Компонент** на панели элементов, затем щелкните пустую область схемы.

- или -

Скопируйте и вставьте существующий компонент.

a. Найдите существующий компонент на схеме или в **Проводнике по моделям UML**.
b. Щелкните компонент правой кнопкой мыши и выберите **Копировать**.
c. Откройте схему, на которой необходимо отобразить скопированный компонент.
d. Щелкните правой кнопкой мыши пустую область схемы и выберите **Вставить**.

Копия компонента отображается с новым именем.

2. Щелкните имя компонента, чтобы изменить его.
3. Щелкните шеврон (5), если нужно отобразить только заголовок компонента.

Отображение портов компонента

Порт (2, 3) представляет группу сообщений или вызовов операций, входящих в компонент или выходящих из него. Группа описывается интерфейсом, который определяет тип порта. Порт может либо предоставлять, либо требовать интерфейс.

Порт с *предоставленным интерфейсом* (2) предоставляет операции, реализуемые компонентом, которые также могут использоваться другими компонентами.

В качестве примеров можно назвать пользовательский интерфейс, веб-службу, интерфейс .NET или коллекцию функций на любом языке программирования.

Порт с *требуемым интерфейсом* (3) представляет требование компонента к группе операций или служб, которые должны быть предоставлены другими компонентами или внешними системами. Например, веб-браузер требует веб-серверы, а надстройка приложения требует службы из

приложения.
Компонент может иметь неограниченное число портов.

Добавление портов в компонент

1. На панели элементов щелкните **Предоставленный интерфейс** или **Требуемый интерфейс**.
2. Щелкните компонент, который необходимо добавить в интерфейс.

На границе компонента появляется порт.
Новый интерфейс создается как тип порта. Этот интерфейс отображается в **Проводнике по моделям UML**.

3. Перетащите порт через границу компонента и разместите его, где нужно.
4. Перетащите метку порта, чтобы скорректировать его расположение.
5. Щелкните метку, чтобы изменить ее. Метка показывает имя интерфейса. Если изменить ее, изменится и имя интерфейса.

Ссылки между компонентами

Используйте зависимость (4), чтобы показать, что требования одного компонента можно удовлетворить операциями или службами, предоставленными другим компонентом.

Как показать, что предоставленный интерфейс может удовлетворить условия требуемого интерфейса

1. На панели элементов щелкните **Зависимость**.
2. Щелкните порт с требуемым интерфейсом одного компонента, затем щелкните порт с предоставленным интерфейсом другого компонента.

Следует избегать создания циклов зависимости, в которых каждый компонент группы зависит от всех остальных компонентов.

Добавление в компонент порта для существующего интерфейса

- В **Проводнике по моделям UML** найдите интерфейс и перетащите его в компонент.

– или –

- Скопируйте и вставьте ссылку на интерфейс из схемы.
    1. На схеме классов или схеме компонентов щелкните интерфейс правой кнопкой мыши и выберите **Копировать**.
    2. На схеме компонентов щелкните компонент правой кнопкой мыши и выберите **Вставить ссылку**.

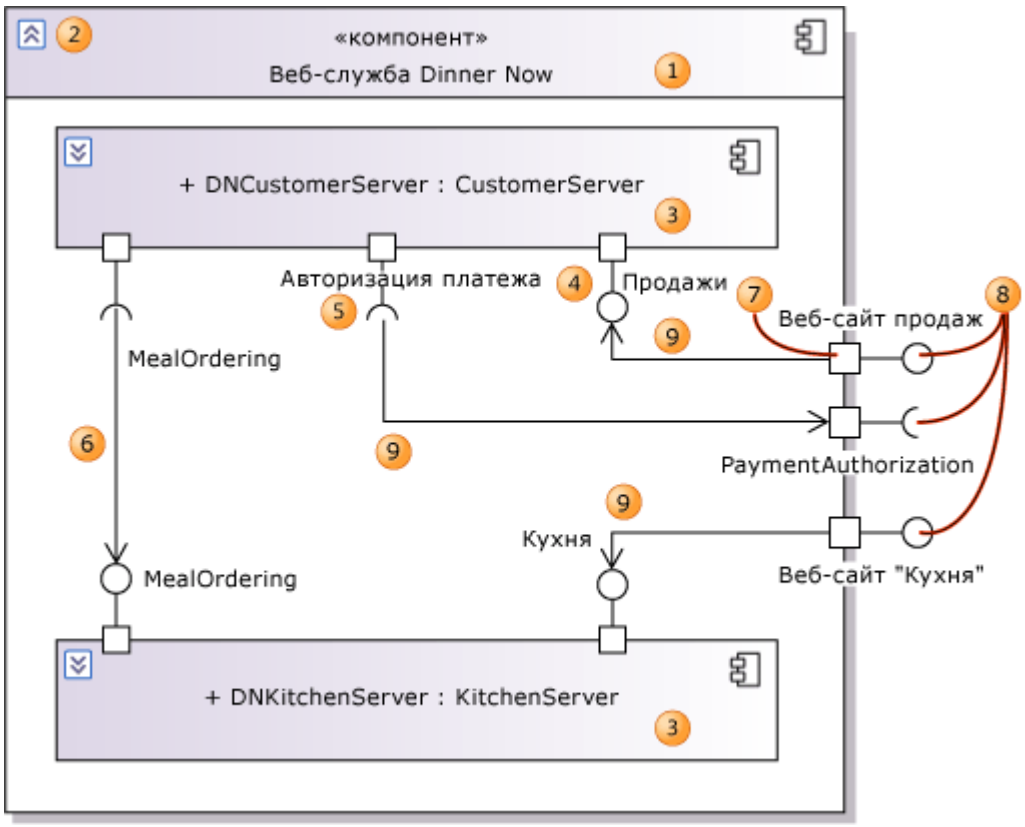
В компоненте появляется предоставленный интерфейс. Рядом появляется тег действия.

Примечание

Если вместо команды **Вставить ссылку** использовать команду **Вставить**, создается новый интерфейс с новым именем.

3. Если необходимо создать требуемый интерфейс, щелкните тег действия и выберите **Преобразовать в требуемый интерфейс**.

Отображение внутренних частей компонента



Можно разместить части (3) в компоненте (1), чтобы показать, что он состоит из более мелких компонентов, взаимодействующих друг с другом.

Схема на иллюстрации показывает, что каждый экземпляр веб-службы Dinner Now содержит один экземпляр сервера "Клиенты" и один экземпляр сервера "Кухня".

Часть — это свойство родительского компонента. Отношения между ними можно сравнить с тем, как атрибут принадлежит к обычному классу. Часть имеет собственный тип, который, как правило, также является компонентом. Метка части имеет ту же форму, что и обычный атрибут.

+ partName : TypeName

Внутри родительского компонента каждая часть показывает предоставляемые и требуемые интерфейсы, определенные для ее типа (4, 5). Операции или службы, требуемые одной частью, могут быть предоставлены другой. Можно использовать соединители **Сборки части**, чтобы показать, как части соединены друг с другом (6).

Также можно показать, что одна из частей родительского компонента фактически предоставляет или требует его интерфейс. Можно подключить порт родительского компонента к порту внутренней части, воспользовавшись отношением **Делегирование** (9). Оба порта должны относиться к одному виду (предоставленные или требуемые) и иметь совместимые типы интерфейса.

Новую часть можно создать либо с помощью нового типа, либо из существующего типа.

Добавление частей в компонент

1. Создайте часть для каждой крупной функциональной единицы, которая является частью родительского компонента.
    a. Щелкните **Компонент** на панели элементов, затем щелкните внутри родительского компонента (1).

Новая часть (3) появляется внутри родительского компонента.

В **Проводнике по моделям UML** создается новый компонент. Это тип новой части.

- или -

Перетащите существующий компонент из Проводника по моделям UML в родительский компонент.

Новая часть (3) появляется внутри родительского компонента. Ее тип — это компонент, перемещенный из Проводника по моделям UML.

- или -
На схеме или в Проводнике по моделям UML щелкните компонент правой кнопкой мыши и выберите **Копировать**.
Щелкните родительский компонент правой кнопкой мыши и выберите **Вставить ссылку**.
Новая часть (3) появляется внутри родительского компонента. Ее тип — это скопированный компонент.

b. Щелкните имя новой части, чтобы изменить его. Изменить ее тип невозможно.
c. В новую часть можно добавить предоставленные и требуемые интерфейсы (4, 5). Выберите **Предоставленный интерфейс** или **Требуемый интерфейс** и щелкните часть.

- или -
Перетащите существующий интерфейс из **Проводника по моделям UML** в часть.
Интерфейсы добавляются в тип части и отображаются в самой части. При необходимости корректируется размер родительского компонента.

2. Соедините части друг с другом.
   a. Используйте инструмент **Зависимость**, чтобы соединить порты разных частей (6).
3. Соедините части с портами родительского компонента.
   a. Создайте один или несколько портов (7) в родительском компоненте. Выберите **Требуемый интерфейс** или **Предоставленный интерфейс**на панели элементов и щелкните родительский компонент.
   b. Делегируйте (9) порт одной или нескольким частям. Последовательно щелкните инструмент **Делегирование**, порт в родительском компоненте и порт в части. Можно соединить порты, предоставляющие или требующие интерфейсы аналогичным способом.

Отображение частей части
После разложения компонента на части каждый тип части можно дополнительно разложить на внутренние части.
Удобнее разместить каждый слой разложения на отдельной схеме компонентов. Сначала нужно найти тип части. Например, на иллюстрации одна из частей называется DNCustomerServer, а ее тип представляет собой компонент, который называется CustomerServer. Этот тип можно найти в Проводнике по моделям UML и поместить его на другую схему. Затем можно создавать внутренние части типа.
Размещение типа части на схеме

1. Определите полное имя для типа части.

Щелкните часть правой кнопкой мыши и выберите пункт **Свойства**.
Имя типа отображается в поле **Тип** окна свойств.

2. Найдите тип части в **Проводнике по моделям UML**.

Щелкните **Вид**, выберите **Другие окна** и щелкните **Проводник по моделям UML**.
Разверните проект и (при необходимости) любой пакет, к которому принадлежит тип.
Тип отобразится в списке как **Компонент**.
При необходимости его имя можно изменить здесь.

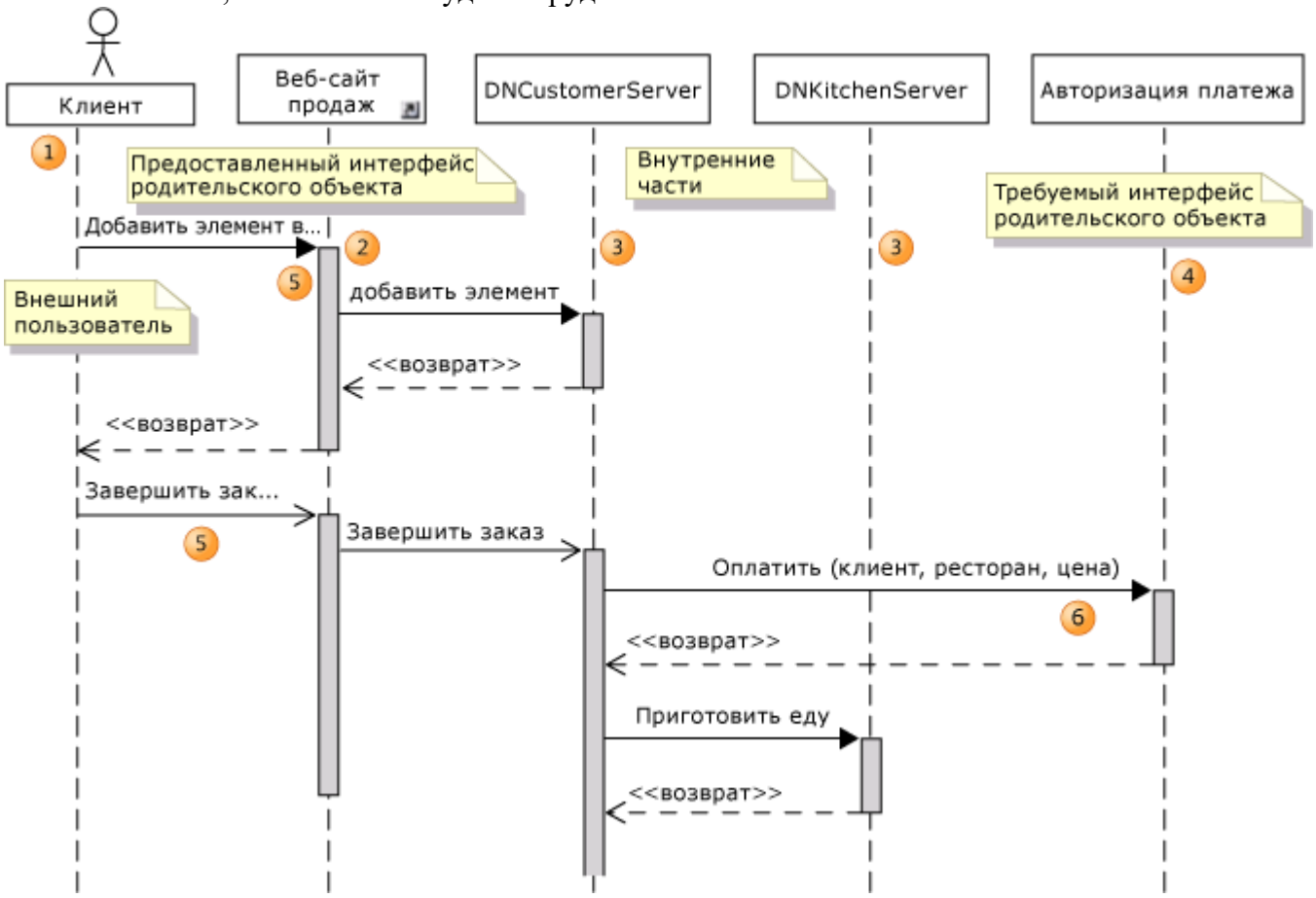
3. Откройте или создайте другую схему компонентов.
4. Перетащите тип из Проводника по моделям UML на схему.

Представление типа отображается как компонент на схеме.
Он имеет те же интерфейсы, которые были определены для части.
Теперь можно добавлять части в компонент.
Конструирование компонента

Описание взаимодействия частей
Можно создать схему последовательностей, чтобы показать, как части взаимодействуют друг с другом в ответ на сообщение, поступающее в родительский компонент.
Можно использовать эти схемы, чтобы проанализировать существующий компонент и создать

новый.
При конструировании компонента можно создавать схемы последовательностей, даже если еще не решено, какие части будут входить в его состав.Схемы последовательностей можно использовать, чтобы поэкспериментировать с различными частями, требуемыми интерфейсами и последовательностями сообщений. Создайте схемы последовательностей для наиболее часто встречающихся и наиболее важных входящих сообщений.После этого можно создавать части в компоненте, которые соответствуют выбранным линиям жизни.
Используйте схемы последовательностей, чтобы оценить, как функционирование системы распределяется между разными компонентами.

- Если слишком много функций делегировано одной части, вероятно, возникнут сложности с обновлением приложения. При равномерном распределении функций это маловероятно.
- Если функции, напротив, распределены между большим числом компонентов, между которыми существует много взаимодействий, производительность системы может снизиться, ее понимание будет затруднено.



Создание схемы последовательностей, показывающей взаимодействие частей

1. Создайте новую схему последовательностей.
2. Создайте линию жизни для внешнего компонента, пользователя, устройства или другого субъекта (1), отправляющего сообщения этому компоненту.

Свойству **Субъект** данной линии жизни можно задать значение true, чтобы указать, что это внешнее свойство рассматриваемого компонента. Над линией жизни отображается контурограмма.

3. Создайте линию жизни для предоставленного интерфейса (2) этого компонента, которому выбранный субъект отправляет сообщения.
4. Создайте линию жизни для каждой части (3) компонента.
5. Создайте линию жизни для каждого требуемого интерфейса (4) компонента.
6. Создайте сообщения от внешнего субъекта (5). Покажите, как сообщение передается частям и как они взаимодействуют в ответ на него.

7. При необходимости покажите сообщения, отправляемые требуемому интерфейсу (6). Не показывайте никаких подробностей в области выполнения сообщения.

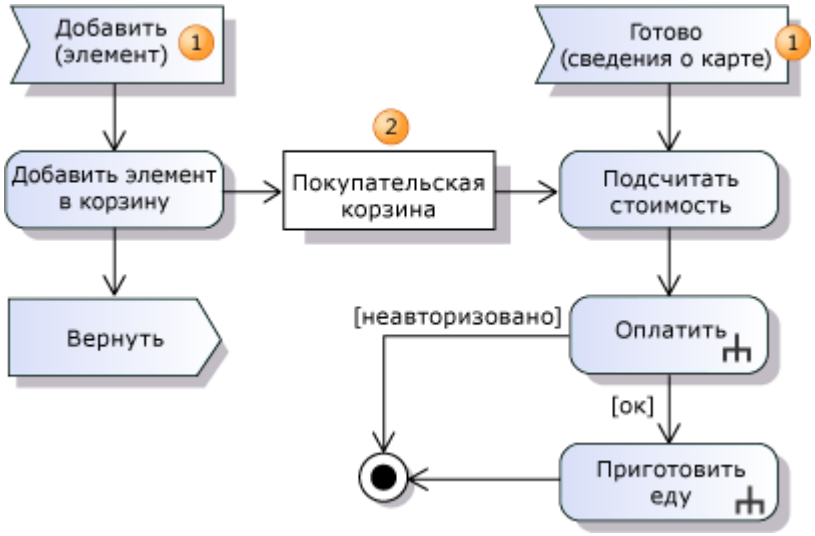
Компонент — это больше, чем его части?

В некоторых случаях компонент — это не больше, чем имя, присвоенное коллекции частей. Всю работу выполняют части, и во время выполнения нет ни кода, ни других артефактов, которые представляют компонент.

Это можно указать в модели, задав свойство **Является неявно создаваемым экземпляром** компонента. В этом случае все интерфейсы компонента должны находиться в портах и иметь делегирования во внутренние части.

Описание процесса внутри каждой части

Чтобы показать, как компонент обрабатывает каждое входящее сообщение, можно использовать схемы активности.



Используйте "Принять действие события" (1), чтобы показать, что входящее сообщение начинает новый поток.

Используйте узлы объекта и закрепления ввода или вывода, чтобы показать поток сведений и показать, где хранятся сведения. В этом примере узел объекта (2) показывает элементы, буферизованные между двумя потоками.

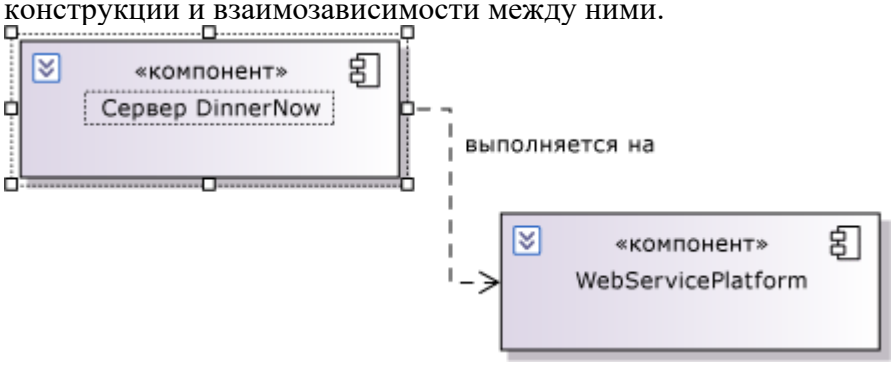
Определение данных и классов

UML-схему классов можно использовать, чтобы подробно описать содержимое следующих элементов:

- интерфейсов компонентов;
- данных, переданных в параметрах операций в интерфейсах;
- данных, сохраненных в компонентах, например как показано в потоках объектов на схемах активности;

общих зависимостей между компонентами.

Можно использовать схему компонентов только для того, чтобы показать основные части конструкции и взаимозависимости между ними.



Чтобы создать зависимость, воспользуйтесь инструментом **Зависимость**. Это означает, что

конструкция одного компонента зависит от конструкции другого.
Наиболее распространены следующие виды зависимостей.

- Один компонент вызывает код внутри другого.
- Один компонент создает экземпляр класса, определенного внутри другого класса.
- Один компонент использует сведения, созданные другим.

Можно использовать имя стрелки зависимости, чтобы обозначить конкретный вид использования. Чтобы задать имя, щелкните стрелку правой кнопкой мыши, затем щелкните **Свойства** и задайте значение в поле **Имя** в окне свойств.

Определение пакетов и пространств имен

В Visual Studio Ultimate *пакет* — это контейнер для определений UML-элементов, таких как классы, варианты использования и компоненты. Пакет также может содержать другие пакеты. В обозревателе моделей UML все определения внутри пакета вложены в этот пакет. Модель UML является своего рода пакетом и формирует корень дерева.
Пакеты позволяют разделить множество задач на несколько областей. Каждый пакет определяет пространство имен так, что имена, определенные в разных пакетах, не конфликтуют друг с другом.
Свойство "полное имя" каждого элемента представляет собой полное имя пакета, к которому этот элемент относится, за которым следует собственное имя элемента. Например, если пакет называется MyPackage, класс внутри пакета будет иметь полное имя MyPackage::MyClass. Поскольку каждый элемент содержится внутри модели, полное имя всегда начинается с имени модели.
Модель также определяет пространство имен так, что полное имя каждого элемента в модели начинается с имени модели.
Другие элементы модели также определяют пространства имен. Например, операция принадлежит пространству имен, определенному его родительским классом, тогда полное имя операции — MyModel ::MyPackage ::MyClass ::MyOperation. Точно так же действие относится к пространству имен, определенному его родительским действием.
Пакеты представляют собой контейнеры. Если переместить или удалить пакет, перемещаются или удаляются также классы, пакеты и другие объекты, определенные внутри него. То же самое можно сказать и о других элементах, определяющих пространства имен.

Создание и просмотр пакетов

Можно создать пакет на UML-схеме классов или в обозревателе моделей UML.

Создание пакета на UML-схеме классов

1. Откройте или создайте UML-схему классов.
2. Выберите средство **Пакет**.
3. Щелкните в любом месте схемы. Появится новая фигура пакета.

Чтобы вложить один пакет в другой, можно щелкнуть внутри существующего пакета.

4. Введите новое имя для пакета.

Создание пакета в обозревателе моделей UML

1. Откройте **Обозреватель моделей UML**. В меню **Архитектура** последовательно выберите пункты **Окна** и **Обозреватель моделей UML**.
2. Щелкните правой кнопкой мыши пакет или модель, к которой требуется добавить новый пакет.

Примечание

Можно вложить один пакет в другой.

3. В открывшемся контекстном меню выберите **Добавить**, а затем щелкните **Пакет**.

В модели откроется новый пакет.

4. Введите новое имя для пакета.

Если пакет создан в обозревателе моделей UML, можно отобразить его на UML-схеме классов. Пакет также можно отобразить на нескольких UML-схемах классов.

Отображение существующего пакета на UML-схеме классов

- Перетащите пакет из обозревателя моделей UML на схему классов.

Примечание

Это позволит создать представление пакета на этой схеме. В нем не обязательно будут отображаться все элементы, которые содержит пакет.Чтобы просмотреть все содержимое пакета, его нужно просматривать в обозревателе моделей UML.

Создание элементов модели внутри пакетов

Существует четыре способа разместить элементы модели внутри пакета.

- Добавьте новый элемент в пакет в обозревателе моделей UML.
- Добавьте классы и другие типы в пакеты на UML-схеме классов.
- Задайте свойство **Связанный пакет** схемы так, чтобы новые элементы, создаваемые на схеме, размещались внутри заданного пакета. Таким образом с пакетом можно связывать схемы классов, компонентов и вариантов использования.
- Переместите элементы в пакет или из него в обозревателе моделей UML.

В обозревателе моделей UML элемент в пакете отображается под пакетом, его полное имя начинается с полного имени пакета. Чтобы просмотреть полное имя элемента, щелкните элемент правой кнопкой мыши и выберите **Свойства**. Свойство **Полное имя** отображается в окне **Свойства**.

Создание элемента в пакете в обозревателе моделей UML

1. Откройте **Обозреватель моделей UML**. В меню **Вид** выберите пункт **Другие окна**, затем щелкните **Обозреватель моделей UML**.
2. Щелкните правой кнопкой мыши пакет или модель, в которую требуется добавить новый элемент.
3. Выберите команду **Добавить** и щелкните вид элемента, который необходимо добавить.

Новый элемент отображается под пакетом.

4. Введите имя для нового элемента.

Примечание

Новый элемент не отображается ни на одной схеме. Чтобы создать представление нового элемента, можно перетащить его из обозревателя моделей UML на схему. Схема должна представлять собой тип, который отображает эту разновидность элементов.

Создание элемента в пакете на UML-схеме классов

1. Откройте схему классов, на которой отображается пакет.
    - Создайте новый пакет, если вы еще не сделали это.
    - Чтобы существующий пакет отображался на схеме классов, можно перетащить пакет из **Обозревателя моделей UML** на схему классов.
2. Щелкните средство для класса, интерфейса, перечисления или пакета.
3. Щелкните пакет, в котором нужно разместить новый элемент.

Новый элемент отображается внутри пакета.

Создание всех элементов схемы в заданном пакете

1. Создайте пакет, если вы еще не сделали это.
2. Откройте схему компонентов, вариантов использования или UML-схему классов.
3. Откройте свойства схемы. Щелкните правой кнопкой мыши пустую область схемы и выберите **Свойства**.
4. В свойстве **Связанный пакет** выберите пакет, в котором нужно разместить содержимое схемы.
5. Создайте новые элементы на схеме. Они будут размещены внутри пакета.
    - **Полное имя** каждого элемента будет начинаться с полного имени пакета.
    - В **Обозревателе моделей UML** каждый элемент будет отображаться в пакете.

Перемещение элементов в пакет и из него

Элемент или элементы можно перемещать в пакет или из него.

При перемещении пакета перемещается все его содержимое.

Перемещение элемента в пакет или из него

- В обозревателе моделей UML перетащите элемент в дерево, корнем которого является пакет, или из него.

Полное имя элемента изменится, и в нем будет содержаться имя нового пакета или модели, которая им владеет.

- или -

- На схеме классов перетащите элемент в фигуру пакета.

Полное имя элемента изменится, и в нем будет содержаться имя нового пакета-владельца.

Примечание

Если перетащить элемент из пакета в пустую часть схемы, пакет-владелец не изменится. Это позволяет создать схему, на которой показаны элементы из нескольких пакетов, не показывая сами пакеты.

Вставка элементов в пакет

Элемент можно вставить в пакет. Если вставить группу связанных элементов в пакет, отношения между ними также переместятся в пакет.

Вставка элементов в пакет на UML-схеме классов

1. На UML-схеме классов выберите все элементы, которые необходимо скопировать. Щелкните один из них правой кнопкой мыши и выберите**Копировать**.
2. Щелкните пакет правой кнопкой мыши и выберите **Вставить**.

Примечание

Пакет может располагаться на другой схеме.

Импорт отношений между пакетами

Можно определить отношение импорта между пакетами, воспользовавшись средством **Импорт**. Импорт означает, что элементы, определенные в импортированном пакете, т. е. элементы на окончании отношения с наконечником стрелки, также эффективно определяются в импортирующем пакете. Любые элементы, видимость которых настроена на **Пакет**, будут также видны в импортирующем пакете.

Не рекомендуется создавать циклы в отношениях импорта.

Ссылки из одного пространства имен на другое

Если требуется создать ссылку на элемент одного пакета из другого, нужно использовать полное имя этого элемента.

Например, пусть пакет SalesCommon определяет тип CustomerAddress. В другом пакете, RestaurantSales, нужно определить тип MealOrder, который имеет атрибут типа "Адрес клиента". Имеются две возможности.

- Задайте тип атрибута с использованием полного имени SalesCommon::CustomerAddress. Это можно сделать, только если свойству **Видимость**атрибута типа CustomerAddress присвоено значение **Public**.
- Создайте отношение импорта из пакета RestaurantSales в пакет SalesCommon. В этом случае не обязательно использовать полное имяCustomerAddress.

Свойства пакетов

Каждый пакет имеет следующие свойства. Чтобы просмотреть свойства, щелкните пакет правой кнопкой мыши на схеме или в обозревателе моделей UML и выберите **Свойства**.

Свойство	Значение по умолчанию	Описание
Имя	(новое имя)	Имя пакета. Его можно изменить на схеме или в окне свойств.

Полное имя	*Контейнер*::*имя_пакета*	Полное имя, имеющее префикс в виде имени пакета или модели, в которой содержится пакет.
Профили	(пусто)	Список профилей, связанных с этим пакетом. Эти профили предоставляют стереотипы, которые можно применить к элементам внутри пакета.
Видимость	**Открытый**	Видимость пакета за пределами родительского пакета.
Рабочие элементы	(пусто)	Список связанных рабочих элементов.
Расположение определения	(имя)	Имя файла, где хранятся сведения о пакете. Файлы находятся внутри папки проекта **Определение модели**. Эти сведения могут оказаться полезными при работе с системой управления версиями.
Описание	(пусто)	Описание пакета.
Стереотипы	(пусто)	Стереотипы, которые применяются к пакету. Список доступных стереотипов определяется профилями, выбранными для этого пакета, и пакетами, которые его содержат.

Хранение пакетов

При создании нового пакета в папке проекта **ModelDefinition** создается новый файл **.uml**. Корневая модель, которая также представляет собой пакет, также хранится в файле **.uml**.
Кроме того, каждая схема хранится в двух файлах: один из них представляет фигуры схемы, а файл **.layout** регистрирует положения фигур.

Лабораторная работа 16. Демонстрационный пример. Моделирование требований пользователей

Microsoft Visual Studio Ultimate помогает понимать и обсуждать потребности пользователей, а также информировать о них других. Для этого можно составлять схемы о деятельности пользователей и о том, как система помогает им в достижении целей. Модель требований — это набор этих схем, каждая из которых иллюстрирует отдельный аспект потребностей пользователей.
Модель требований помогает:

- сфокусироваться на внешнем поведении системы отдельно от ее внутреннего строения;
- описать потребности пользователей и заинтересованных лиц более однозначно, чем с помощью языка;
- составить согласованный глоссарий терминов для пользователей, разработчиков и тест-инженеров;
- уменьшить число пропусков и несоответствий в требованиях;
- упростить реагирование на изменения требований;
- планировать последовательность разработки функций;
- использовать модели в качестве основы для системных тестов, устанавливая четкое отношение между тестами и требованиями. При изменении требований это отношение

помогает правильно обновлять тесты. Это позволяет обеспечить соответствие системы новым требованиям.

Максимально эффективное использование модели требований обеспечивается при использовании этой модели при выборе тем для обсуждения с пользователями или их представителями. Кроме того, необходимо пересматривать модель в начале каждой итерации. Не обязательно оформлять модель в подробностях до создания кода. Частично работающее приложение, даже если оно сильно упрощено, как правило, может стать хорошей основой для плодотворного обсуждения требований с пользователями. Такая модель позволяет эффективно свести воедино результаты обсуждений.

Примечание

В этих разделах под термином "система" подразумевается система или приложение в разработке. Это может быть крупная коллекция множества компонентов программного и аппаратного обеспечения, одно приложение или компонент программного обеспечения внутри более крупной системы.Во всех этих случаях модель требований описывает поведение, видимое вне системы (через пользовательский интерфейс или API).

Общие задачи

Можно создать несколько разных представлений требований пользователей. Каждое представление предоставляет определенный тип сведений. При создании этих представлений рекомендуется часто перемещаться от одного представления к другому. Начать можно с любого представления.

Схема или документ	Что описывает в модели требований
Схема вариантов использования	Кто использует систему, и какие действия они в ней совершают.
Концептуальная схема классов	Глоссарий типов, используемых для описания требований; типы видны в интерфейсе системы.
Схема деятельности	Рабочий процесс и обмен сведениями в процессе взаимодействия пользователей и системы или ее частей.
Схема последовательностей	Последовательность взаимодействий между пользователями и системой или ее частями. Альтернативное представление схемы деятельности.
Дополнительные документы или рабочие элементы	Критерии производительности, безопасности, полезности и надежности.
Дополнительные документы или рабочие элементы	Ограничения и правила, не относящиеся к конкретному варианту использования.

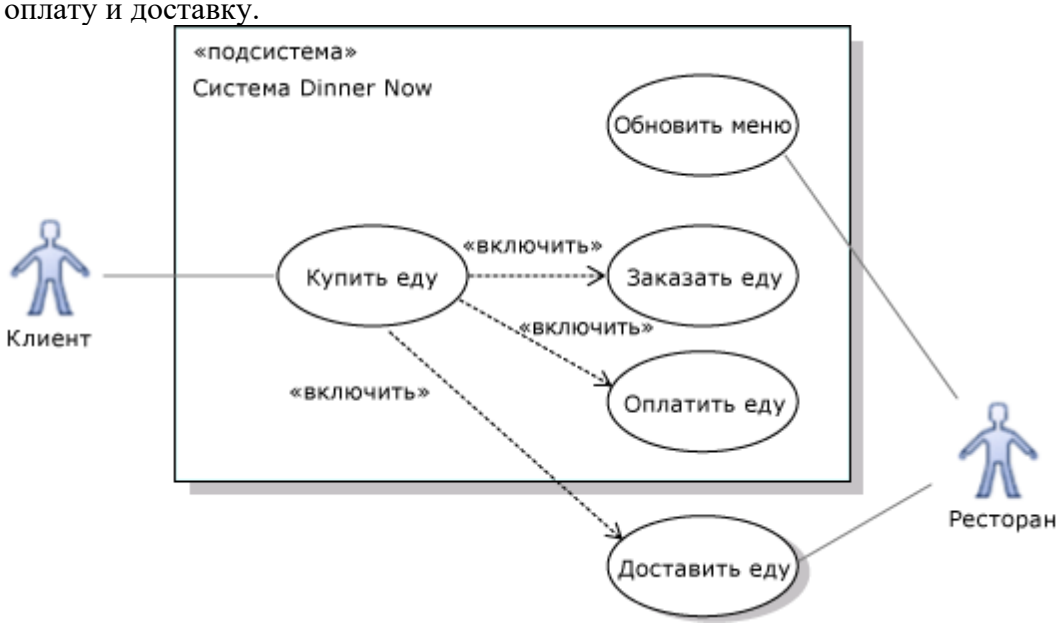
Обратите внимание, что большинство типов схем можно использовать и для других целей.

Описание использования системы

Схемы вариантов использования создаются, чтобы описать, кто и для чего использует систему. Вариант использования представляет цель пользователя системы и процедуру, выполняемую пользователем для достижения этой цели.

Например, система продажи еды через Интернет должна позволять клиентам выбирать элементы меню, а ресторанам-поставщикам — обновлять меню.Это можно объединить в схеме вариантов использования.

Также можно показать, что вариант использования состоит из более мелких вариантов, и показать эти варианты. Например, заказ еды — часть процесса покупки еды, который также включает оплату и доставку.



Кроме того, можно показать, какие варианты использования включены в разрабатываемую область системы. Например, изображенная на иллюстрации система не входит в состав варианта использования "Доставка еды". Это помогает задать контекст для разработки. (На схеме вариантов использования контейнеры подсистемы можно использовать для представления системы или ее компонентов).

Кроме того, это помогает команде разработчиков обсуждать, что будет включено в последующие выпуски. Например, можно обсудить, будет ли компонент "Оплата еды" в первоначальном выпуске системы функционировать непосредственно между рестораном и клиентом (а не обрабатываться в системе). В этом случае в начальном выпуске можно переместить компонент "Оплата еды" за пределы прямоугольника системы Dinner Now.

Схема вариантов использования предоставляет только сводку вариантов использования. Чтобы предоставить более подробные описания, можно связать варианты использования на схеме с отдельными документами и другими схемами.

Создание схемы вариантов использования помогает команде разработчиков:

- концентрироваться на том, как пользователи намерены работать с системой, не отвлекаясь на подробности реализации;
- обсуждать область действия системы или отдельных выпусков системы.

Определение терминов для описания требований

UML-схемы классов можно использовать для создания согласованного словаря бизнес-понятий, которые будут использоваться:

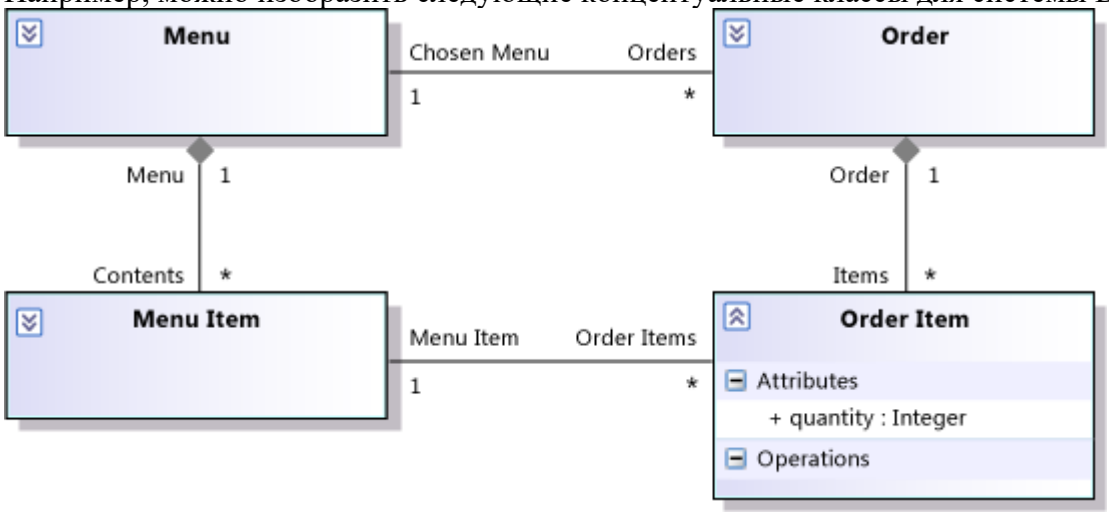
- пользователями для обсуждения сферы деятельности, в которой работает система;
- для описания потребностей пользователей, например в описании вариантов использования, бизнес-правилах и описании функциональности пользователей (user story);
- для описания типов сведений, обмен которыми осуществляется через API системы или пользовательский интерфейс;

- для описания системы или приемочных тестов.

Если понятия используются с этой целью, содержимое UML-схемы классов называется концептуальной схемой классов. (Она также называется *моделью домена* или *моделью классов анализа*).

В концептуальной схеме классов показаны только те классы, которые необходимо использовать в описаниях требований, и не показаны подробные сведения о внутреннем строении системы. На схеме не отображаются подробные сведения о внутреннем строении системы. Как правило, в концептуальных классах не отображаются операции и интерфейсы.

Например, можно изобразить следующие концептуальные классы для системы Dinner Now.



Концептуальная схема классов предоставляет словарь терминов, используемых в модели требований. Например, в подробном описании варианта использования "Заказ еды" можно написать следующее.

Клиент выбирает *меню* для формирования *заказа*, а затем создает в *заказепункты заказа*, выбирая из *менюпункты меню*.

Обратите внимание, что термины, используемые в этом описании, являются именами классов в модели. Схема устраняет неоднозначности в отношениях между этими классами. Например, на схеме хорошо видно, что каждый заказ связан только с одним меню.

Очень часто причиной неправильного понимания требований пользователей является неправильное понимание точных значений слов. Например, в большинстве ресторанов термины "меню" и "заказ" понимаются одинаково, а трактовка терминов "пункт заказа" и "пункт меню" гораздо менее однозначная. При обсуждении требований с заинтересованными лицами из сферы бизнеса необходимо раскрыть подобные различия. Схема классов — это полезное средство, с помощью которого можно прояснить термины и отношения между ними.

Концептуальная модель классов позволяет составлять базовый словарь для описания бизнес-логики системы. Однако классы в программном обеспечении, как правило, гораздо более сложные, чем концептуальная модель, так как в реализации необходимо учитывать производительность, распределение, гибкость и другие факторы. Часто в одной системе можно найти несколько разных реализаций концептуального класса.

Например, в разных частях системы и в разных интерфейсах, связывающих эти части, заказы могут быть представлены в форматах XML, SQL, HTML и C#.Связь между заказом и меню можно представить по-разному, например с помощью ссылок в коде C#, отношений в базе данных или ИД перекрестных ссылок в XML. Несмотря на эти различия, концептуальная модель предоставляет важные сведения, истинные во всех частях программного обеспечения.Схема классов в этом примере показывает, что в каждой реализации заказ связан только с одним меню.

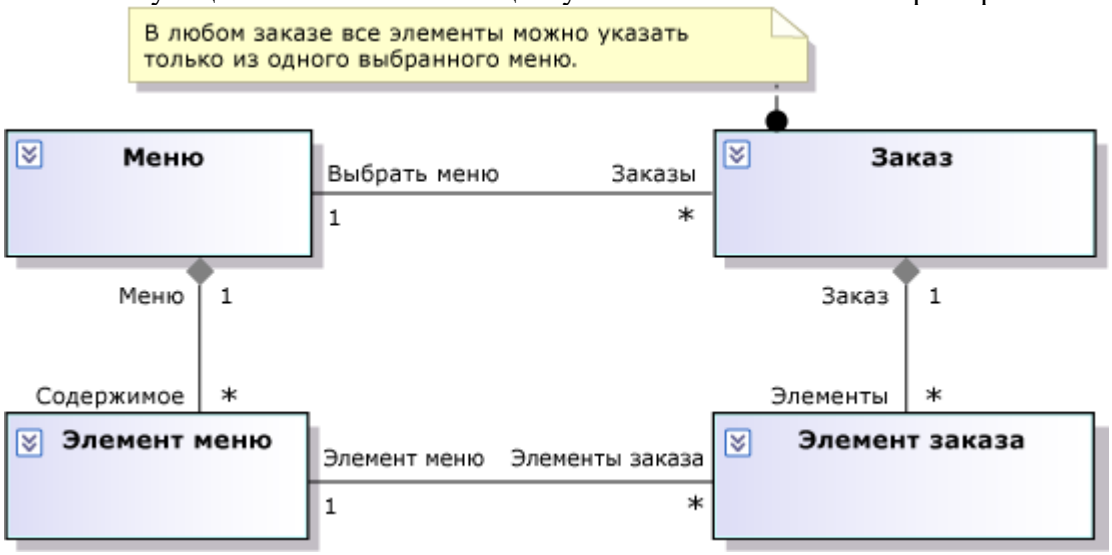
Создание схемы классов требований позволяет команде разработчиков:

- определять и стандартизировать основные термины, используемые для обсуждения потребностей пользователей;
- прояснять отношения между этими терминами.

В концептуальной схеме классов обычно не стоит изображать стрелки, указывающие на ассоциации, чтобы показать возможности перехода. На схеме не представляется реализация. Ассоциации представляют связи между объектами реального мира. Следующее расширение Visual Studio делает ненаправленные стрелки стрелками по умолчанию: Sample: UML Domain Modeling features.

Отображение бизнес-правил

Бизнес-правило — это требование, не связанное с определенным вариантом использования, которое необходимо соблюдать во всех частях системы.
Многие бизнес-правила представляют собой ограничения отношений между концептуальными классами. Эти *статическиебизнес-правила* можно записать в виде комментариев, связанных с соответствующими классами на концептуальной схеме классов. Пример.



Динамические бизнес-правила ограничивают допустимые последовательности событий. Например, схему последовательностей или схему деятельности можно использовать, чтобы показать, что прежде чем совершать другие операции в системе, пользователь должен выполнить вход. Например, многие динамические правила можно сформулировать более эффективно и в более общем виде, заменив их статическими правилами.Например, в класс в концептуальной модели классов можно добавить логический атрибут "Logged In". Тогда Logged In будет добавляться в качестве постусловия варианта использования "вход в систему" и предусловия большинства других вариантов использования. Благодаря этому подходу можно не определять все возможные комбинации последовательностей событий. Кроме того, этот подход является более гибким при необходимости добавить в модель новые варианты использования.
Обратите внимание, что в данном случае речь идет о выборе способа определения требований, который не зависит от способа реализации требований в программном коде.

Описание требований к качеству обслуживания

Существует несколько категорий требований к качеству обслуживания. Среди них можно выделить следующие.

- Производительность
- Безопасность
- Полезность
- Надежность
- Устойчивость

Некоторые из этих требований можно включить в описания конкретных вариантов использования. Другие требования не относятся к вариантам использования, поэтому рекомендуется записать их в отдельный документ. По возможности рекомендуется использовать словарь, определенный в модели требований. Обратите внимание, что в следующем примере

главные слова, используемые в требованиях, являются названиями субъектов, вариантов использования и классов из предыдущих иллюстраций.
Если ресторан удалит пункт меню, пока клиент заказывает еду, любые пункты заказа, связанные с этим пунктом меню, будут выделены красным.

Отображение рабочего процесса между пользователями и системой

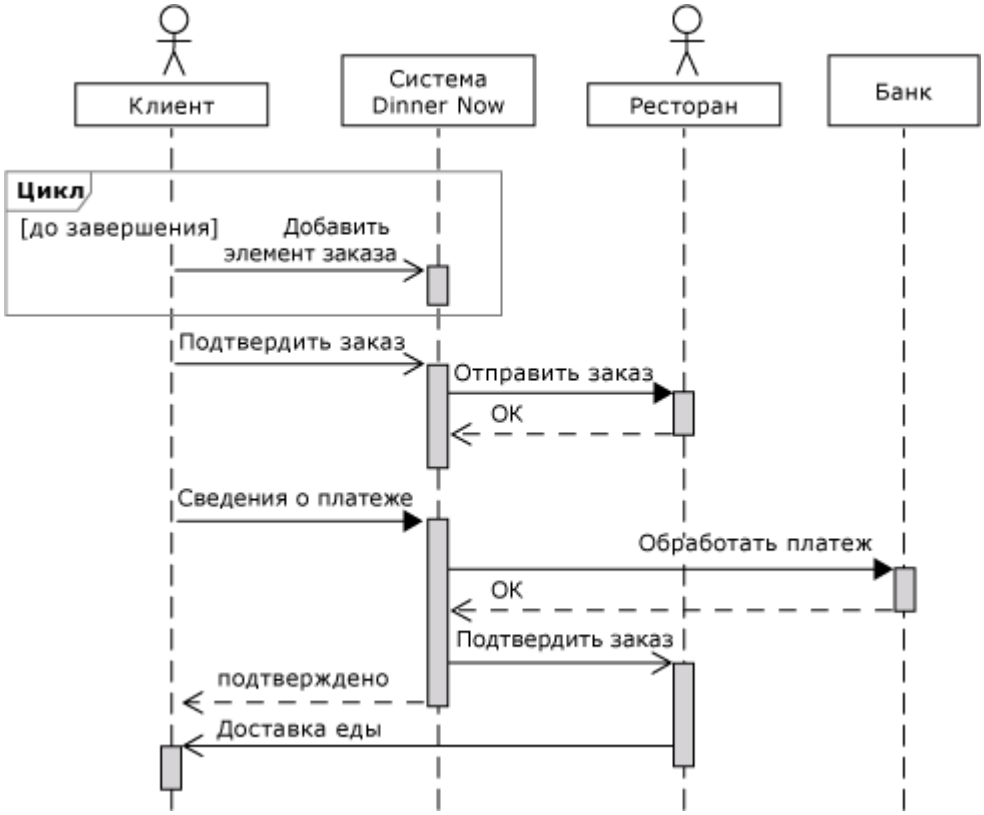
Для отображения рабочего процесса между разными вариантами использования можно использовать схему деятельности. Во многих случаях рекомендуется начинать создание модели требований с составления схемы деятельности, на которой должны быть отображены основные задачи, выполняемые пользователем — как в системе, так и за ее пределами.
Пример.

Можно составить схемы вариантов использования и схемы деятельности, чтобы показать разные представления одних сведений. Схема вариантов использования больше подходит для отображения вложенности отдельных действий в деятельность, однако на такой схеме не отображается рабочий процесс. Пример.

Обратите внимание, что схемы деятельности можно использовать для изображения алгоритмов, действующих в программном обеспечении, но при использовании схем для бизнес-процесса нужно отображать на них действия, видимые вне системы.

Отображение взаимодействий между пользователями и системой

Схемой последовательностей можно воспользоваться, чтобы показать обмен сообщениями между системой и внешними субъектами или между частями системы. Так можно создать представление шагов в варианте использования, которое ясно показывает последовательность взаимодействий. Схемы последовательностей особенно эффективны, если в варианте использования несколько взаимодействующих сторон либо система имеет API.
Пример.



Одним из преимуществ схем последовательностей является то, что они позволяют легко увидеть, какие сообщения поступают в разрабатываемую систему. Чтобы разработать систему, можно заменить один жизненный цикл системы отдельными жизненными циклами для каждого компонента, а затем показывать взаимодействия между ними в ответ на каждое входящее сообщение.

Использование модели для уменьшения числа несоответствий

Создание модели, как правило, позволяет значительно уменьшить число несоответствий и неоднозначностей в требованиях пользователей. Разные заинтересованные лица часто по-разному понимают производственную среду, в которой работает система. Следовательно, прежде всего, необходимо разрешить такие различия между клиентами.

Вы обнаружите, что многие вопросы о бизнес-среде возникают естественным образом при создании модели. Задавая эти вопросы пользователям, можно уменьшить число необходимых изменений на последующих этапах проекта. Ниже представлены некоторые конкретные вопросы, которые вначале можно задать себе, а потом обратиться с ними к заинтересованным лицам, если не удалось получить на них четкого ответа.

- Для каждого класса в модели требований необходимо задать вопрос "Какой вариант использования создает экземпляры данного класса?". Например, для службы заказа еды через Интернет можно спросить: «Какой вариант использования создает экземпляры класса "Меню ресторана"?» Это позволит начать обсуждение о том, как новый ресторан подписывается на работу со службой и предлагает свое меню.Аналогичные вопросы можно задавать о том, что создает или изменяет атрибуты и ассоциации.
- Для каждого варианта использования в модели требований попытайтесь описать результат (или постусловие) каждого варианта использования с помощью слов, предоставленных схемами классов. Во многих случаях рекомендуется показывать следствие варианта использования, составляя эскиз экземпляров классов до и после вхождения варианта использования. Например, если постусловие варианта использования гласит "пункт меню добавляется в заказ клиента", необходимо составить эскизы экземпляров для классов "пункт заказа" и "пункт меню". Покажите следствия варианта использования, например новую ссылку или новый объект, на новом рисунке или выделите их другим цветом. Часто

это позволяет начать обсуждение о том, какие сведения необходимы в модели. Хотя классы требований не связаны напрямую с реализацией, они описывают сведения, которые потребуется хранить и передавать в системе.
- Задайте вопрос об ограничениях атрибутов и ассоциаций. Особое внимание уделите ограничениям, распространяющимся на несколько атрибутов или ассоциаций.
- Задайте вопрос о допустимых и недопустимых последовательностях вариантов использования, проиллюстрировав их с помощью схем последовательностей или схем деятельности.

Проанализировав связи между представлениями, предоставляемыми разными схемами, можно быстро разобраться в основных терминах, с которыми работают пользователи, и помочь пользователям понять, что им требуется от системы. Также можно лучше понять, в отношении каких требований заинтересованные лица испытывают наибольшие сомнения. Можно запланировать разработку этих функций (по крайней мере, в упрощенной форме) на ранней стадии проекта, чтобы пользователи могли поэкспериментировать с ними.

Лабораторная работа 17. Проектирование готовых проектов. Задания для самостоятельной работы

Цель работы

Основываясь на знаниях, полученных в предыдущих работах разработать основные диаграммы UML для моделирования работы заданных инфокоммуникационных систем.

Общие сведения

Характеристика и назначение информационных систем

Информационная технология обработки данных предназначена для решения хорошо структурированных задач, по которым имеются необходимые входные данные и известны алгоритмы и другие стандартные процедуры их обработки. Эта технология применяется на уровне операционной (исполнительской) деятельности персонала невысокой квалификации в целях автоматизации некоторых рутинных постоянно повторяющихся операций управленческого труда. На уровне операционной деятельности решаются следующие задачи:
- обработка данных об операциях, производимых фирмой;
- создание периодических контрольных отчетов о состоянии дел в фирме;
- получение ответов на всевозможные текущие запросы и оформление их в виде бумажных документов или отчетов.

Пример: операция проверки на соответствие нормативу уровня запасов указанных товаров на складе. При уменьшении уровня запаса выдается заказ поставщику с указанием потребного количества товара и сроков поставки.
Существует несколько особенностей, связанных с обработкой данных, отличающих данную технологию от всех прочих:
1. каждой фирме предписано законом иметь и хранить данные о своей деятельности, которые можно использовать как средство обеспечения и поддержания контроля на фирме;
2. решение только хорошо структурированных задач, для которых можно разработать алгоритм;
3. выполнение стандартных процедур обработки. Существующие стандарты определяют типовые процедуры обработки данных и предписывают их соблюдение организациями всех видов;
4. выполнение основного объема работ в автоматическом режиме с минимальным участием человека;

5. использование детализированных данных. Записи о деятельности фирмы имеют детальный (подробный) характер, допускающий проведение ревизий. В процессе ревизии деятельность фирмы проверяется хронологически от начала периода к его концу и от конца к началу;
6. требование минимальной помощи в решении проблем со стороны специалистов других уровней.

Основные компоненты

Рис. Основные компоненты обработки данных.

Сбор данных. По мере того как фирма производит продукцию или услуги, каждое ее действие сопровождается соответствующими записями данных. Обычно действия фирмы, затрагивающие внешнее окружение, выделяются особо как операции, производимые фирмой.
Обработка данных. Для создания из поступающих данных информации, отражающей деятельность фирмы, используются следующие типовые операции:

- классификация одного или нескольких символов. Эти коды, выражающие определенные признаки объектов, используются для идентификации и группировки записей. *Пример*. При расчете заработной платы каждая запись включает в себя код (табельный номер) работника, код подразделения, в котором он работает, занимаемую должность и т.п.
- сортировка, с помощью которой упорядочивается последовательность записей;
- вычисления, включающие арифметические и логические операции. Эти операции, выполняемые над данными, дают возможность получать новые данные;
- укрупнение или агрегирование, служащее для уменьшения количества данных и реализуемое в форме расчетов итоговых или средних значений.

Хранение данных. Многие данные на уровне операционной деятельности необходимо сохранять для последующего использования. Для их хранения создаются базы данных.
Создание отчетов (документов). В информационной технологии обработки данных необходимо создавать документы для руководства и работников фирмы, а также для внешних партнеров. При этом документы могут создаваться как по запросу или в связи с проведенной фирмой операцией, так и периодически в конце каждого месяца, квартала или года.

Варианты заданий

Информационная система по отысканию рыночных ниш

При покупке товаров в некоторых фирмах информационная система регистрирует данные о покупателе, что позволяет:

- определять группы покупателей, их состав и запросы, а затем ориентироваться в своей стратегии на наиболее многочисленную группу;

- посылать потенциальным покупателям различные предложения, рекламу, напоминания;
- предоставлять постоянным покупателям товары и услуги в кредит, со скидкой, с отсрочкой платежей.

Информационные системы, ускоряющие потоки товаров

Предположим, фирма специализируется на поставках продуктов в больницу. Как известно, иметь большие запасы продуктов на складах фирмы очень невыгодно, а не иметь их невозможно. Для того чтобы найти оптимальное решение этой проблемы, фирма устанавливает терминалы в обслуживаемом учреждении и подключает их к информационной системе. Заказчик прямо с терминала вводит свои пожелания по предоставляемому ему каталогу. Эти данные поступают в информационную систему по учету заказов.
Менеджеры, делая выборки по поступившим заказам, принимают оперативные управленческие решения по доставке заказчику нужного товара за короткий промежуток времени.

Информационные системы по снижению издержек производства

Эти информационные системы, отслеживая все фазы производственного процесса, способствуют улучшению управления и контроля, более рациональному планированию и использованию персонала и, как следствие, снижению себестоимости производимой продукции и услуг.

Информационные системы автоматизации технологии ("менеджмент уступок")

Суть этой технологии состоит в том, что, если доход фирмы остается в рамках рентабельности, потребителю делаются разные скидки в зависимости от количества и длительности контрактов. В этом случае потребитель становится заинтересован во взаимодействии с фирмой, а фирма тем самым привлекает дополнительное число клиентов. Если же клиент не желает взаимодействовать с данной фирмой и переходит на обслуживание к другой, то его затраты могут возрасти из-за потери предоставляемых ему ранее скидок.

Система обработки метеоинформации

Фирма “NeMeo” желает заказать систему обработки метеоинформации, состоящую из двух частей. Первая предназначена для создания и редактирования карт местности. Вторая для нанесения на карты движения воздушных масс и циклонов. Процесс движения должен задаваться формулами. В целом система должна давать возможность благодаря анимации получить наглядное представление об изменении метеоусловий на несколько дней вперед.
Функционирование системы предполагается на локальном компьютере. Работа в системе должна включать в себя три части: редактирование карт; задание и редактирование движения воздушных масс и циклонов; демонстрация погодных условий.

Web-сервис

Необходимо реализовать на стороне сервера хранилище, в которое можно помещать алгоритмы в некотором стандартном виде, а потом исполнять их. Для простоты алгоритмы могут представлять собой математические формулы. В алгоритмах должны быть заявлены следующие данные:

- входные данные;
- выходные данные;
- код алгоритмов.

Доступ к алгоритмам должен быть ограничен на основе разделения прав по ролям.
Web-сервис должен быть рассчитан на небольшое число пользователей и работу в локальной сети.
К web-сервису должен быть реализован разделенный доступ пользователей.

Объекты системы: пользователь, роль, алгоритм, web-сервис.
Пользователи: логин, пароль, роль.
Пользователь может на web-сервисе осуществлять следующие действия: размещать алгоритмы; изымать на редактирование алгоритмы; удалять алгоритмы с web-сервиса; исполнять алгоритмы.
Алгоритм: название, код (математическое выражение), принадлежность пользователю, входные и выходные параметры.
Web-сервис предоставляет следующие возможности:

- хранить алгоритмы на сервере;
- предоставлять доступ к алгоритмам:
  - редактирование;
  - удаление;
- исполнение алгоритма на сервере.

Для хранения алгоритмов на сервере создается дерево каталогов и файлов. Для каждого пользователя создается корневой каталог. В этом каталоге могут храниться, как алгоритмы (файлы с кодом), так и другие каталоги. Разделения прав осуществляется на основе специального файла со списком пользователей, которым доступна эта папка. Права на папку наследуются. Также можно разрешить доступ сразу группе пользователей, принадлежащих определенной роли. Редактировать алгоритм может только пользователь, выложивший алгоритм. Исполнить алгоритм могут только те пользователи, которым доступна папка с алгоритмом. Исполнение производится через специальный интерфейс.

8. Вопросы к экзамену

1. Понятия и классификация ИС.
2. Понятия и структура проекта ИС.
3. Жизненный цикл ПО ИС. Стадии жизненного цикла ПО ИС.
4. Модели жизненного цикла ПО ИС.
5. Методы и средства проектирования ИС.
6. Стандарты проектирования .
7. Каноническое проектирование.
8. Стадии и этапы процесса проектирования ИС.
9. Цели и задачи предпроектной стадии создания ИС.
10. Техническое задание на создание ИС.
11. Состав работ на стадии технического и рабочего проектирования.
12. Состав работ на стадии ввода в действие ИС, эксплуатации и сопровождения.
13. Состав проектной документации на ИС.
14. Бизнес-модель. Модели деятельности организации "как есть" и "как должно быть".
15. Состав, содержание и принципы организации информационного обеспечения ИС.
16. Внемашинное информационное обеспечение.
17. Классификация информации. Состав и содержание операций проектирования классификаторов.
18. Понятия и основные требования к системе кодирования информации.
19. Внутримашинное информационное обеспечение.
20. Проектирование экранных форм электронных документов.
21. Понятие типового проекта, предпосылки типизации. Объекты типизации.
22. Методы типового проектирования. Технологии параметрически-ориентированного и модельно- ориентированного проектирования.
23. Типовое проектное решение (ТПР). Классы и структура ТПР.
24. Моделирование как методологическая основа современных методов разработки

информационных систем

25. Использование CASE-технологий. Функционально-ориентированный подход.
26. Использование CASE-технологий. Объектно-ориентированный подход.
27. Функциональная методика IDEF.
28. Принципы построения модели IDEF0. Диаграммы IDEF0.
29. CASE-средство BPWin.
30. Диаграммы потоков данных (Data Flow Diagramm
31. Метод описания процессов IDEF3
32. Моделирование данных. Диаграммы "сущность-связь". Метод IDEFI.
33. Основные принципы объектного проектирования ИС
34. Объектно-ориентированный анализ. Определение классов и объектов
35. Характерные черты языка моделирования UML
36. Общая структура языка UML
37. Диаграммы UML
38. Диаграммы прецедентов (Use Case diagram)
39. Диаграммы деятельности (Activity Diagram)
40. Диаграмма классов (Class diagram)
41. Диаграммы состояний (Statechart diagram)
42. Диаграммы отношений между объектами
43. Диаграммы последовательности действий (Sequence diagram)
44. Диаграммы взаимодействий (Collaboration diagram)
45. Диаграммы компонентов (Component diagram)
46. Диаграммы топологии (Deployment diagram)
47. Объектно-ориентированное CASE средство Rational Software Architect
48. Принципы разработки программных систем в Rational Software Architect
49. Технология быстрого проектирования ЭИС (RAD- технология).
50. Экстремальное программирование.

9. Методические указания студенту по выполнению курсовой работы

1 ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Цель и задачи выполнения курсовой работы

Целью курсового проектирования является:

– закрепление теоретических знаний, полученных по дисциплине «Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем»,

– приобретение практических навыков работы с литературой для получения новых знаний;

Задачами курсового проектирования являются:

– приобретение навыков самостоятельной работы с литературой

– формирование навыков самостоятельного решения задачи на ЭВМ

– приобретение навыков логичности, последовательности, системности в изложении материала при раскрытии темы

– формирование умения описывать и обобщать свой собственный и чужой опыт

1.2 Общие требования к курсовой работе

Курсовая работа выполняется в соответствии с задание на курсовое проектирование.

Курсовая работа выполняется в часы, отведенные на самостоятельную работу. Консультирование по возникающим в процессе работы вопросам проводится согласно расписанию консультаций.

Работа над курсовым проектом предполагает использование не только основной учебной литературы, но и дополнительной учебной и научной литературы с целью более полного и всестороннего раскрытия содержания вопросов задания.

Курсовая работа должен содержать пояснительную записку, графическую часть (алгоритм), код программы и демонстрационную рабочую программу на ЭВМ. Пояснительная записка должна быть представлена в виде машинописного текста с соблюдением стандартных требований к форматированию документов в текстовом редакторе Word. Объем пояснительной записки 15-20 листов машинописного текста. При необходимости, можно составить презентацию (5-8 слайдов).

Представляемая к рецензии работа должна содержать:

- титульный лист;
- задание на работу;
- содержание;
- основную часть в соответствии с утвержденным заданием на работу;
- заключение;
- список использованных источников и литературы;
- приложение - при необходимости;
- перечень условных обозначений, символов и терминов - при необходимости.

Изменение структуры работы устанавливается кафедрой.

Структура и содержание курсовых работ должны соответствовать выбранной теме.

При соблюдении указанных требований к изложению, автор работы должен показать:

– общую профессиональную компетентность и эрудированность в методах информационных технологий, приемами анализа сложных систем, практическое применение процедурного и объектного подхода для решения учебных и прикладных задач;

– достаточно глубокие теоретические и практические знания по избранной теме, умение на теоретической и практической основе получать новое знание, способность и умение выбирать методики и технологии применительно к информационным технологиям;

– умение подбирать, изучать и комплексно анализировать литературные источники (учебные пособия, научные статьи, научные рефераты, монографии и др.) и статистические материалы;

– умение проблемного изложения материала;

– умение сопоставлять и критически анализировать научные подходы и идеи, делать выводы, заключения, рекомендации;

– умение описывать и обобщать свой собственный и чужой опыт;

– умение излагать и «защищать» положения работы.

Общими требованиями к выполненной работе являются:

– логичность, последовательность, системность в изложении материала при раскрытии темы;

– четкость построения плана, полнота реализации исследовательских задач;

– целевая направленность приводимого теоретического и практического материала, его подчиненность исследовательскому замыслу и алгоритму решения поставленных задач;

– грамотность методологического и методического аппарата исследования программирования;

– конкретность целей и задач исследования, принципиальная возможность их достижения и решения в ходе работы;

– полнота изложения вопросов плана;

– привлечение широкого круга учебной и научной литературы и, в первую очередь, первоисточников; самостоятельность, убедительность аргументации и доказательность суждений, выводов и предложений, которые содержатся в соответствующих главах работы и заключении;

– грамотность постановки и описания экспериментальной работы;

– наличие приложений, содержащих исходный рабочий материал;

– грамотность написания, правильность и аккуратность оформления.

Требованиями к структурным элементам работы являются - обязательная логическая связь между главами (подразделами) и последовательность развития основной темы на протяжении всей работы.

Студент, самостоятельно выполнивший курсовую работу, будет в большей степени подготовлен к решению управленческих и прикладных проблем и ситуаций, возникающих в

процессе функционирования организации. Кроме того, курсовая работа подготавливает студента к выполнению дипломной работы, в которой принятие решений связано с успешной деятельностью экономической системы организации в условиях формирования рыночных отношений.

1.3 Выполнение курсовой работы

Курсовая работа – это комплексная задача, при решении которой студент использует все знания, полученные при теоретической части изучения дисциплины Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем

Работа должна иметь профессиональную теоретическую и практическую направленность.

Теоретическая направленность заключается в использовании в работе теоретических положений предмета, методологии современной науки управления в рыночных условиях. Практическая направленность выражается в практическом использовании принципов и методов предмета. Профессиональные требования предполагают наличие связи теории с практикой.

Выполнение курсовой работы представляет собой творческий процесс, состоящий из следующих основных этапов:

1. Уяснить задание на курсовое проектирование и составить план выполнения курсового проекта.
2. Выполнить поиск и подбор необходимой литературы по теоретическому вопросу проекта.
3. Выполнить теоретическую часть проекта, используя учебную и научную литературу.
4. Решить вычислительную задачу на ЭВМ, проанализировать результаты и оформить пояснительную записку.
5. Рецензирование, защита курсовой работы.

В процессе выполнения курсового проекта осуществляется текущий контроль его выполнения со стороны преподавателя.

Выполненный курсовая работа представляется на проверку для допуска к защите. Допущенный к защите курсовая работа подлежит защите на кафедре. Защита курсового проекта должна сопровождаться демонстрацией выполнения программ на ЭВМ. По результатам защиты выставляется оценка, которая складывается из оценок по выполнению курсового проекта, по оформлению пояснительной записки и графической части в соответствии с требованиями и по защите курсового проекта.

.

2. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЗАДАНИЯ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

2.1 Организация выполнения работы

Успешное выполнение курсовой работы, а в последующем и дипломной в целом существенно зависит от обеспеченности студента исходным материалом. Поэтому предварительно студент должен не только собрать и изучить необходимые исходные данные в исследуемой организации, но и проработать специальную литературу, законодательные акты и другие источники. Руководитель работы периодически консультирует студента, а он обязан отчитываться о выполненной работе перед своим руководителем.

Руководитель оказывает студенту методическую помощь, направляя его работу, развивая инициативу, всемерно содействуя развитию его творческой самостоятельности. Обо всех отклонениях в сроках и качестве выполнения работы студент обязан своевременно ставить в известность своего руководителя.

Основанием для разработки курсовых работ является задание, подписанное руководителем.

Заданием определяется перечень вопросов, подлежащих разработке в курсовых работах, также их объем.

Задание на выполнение работы брошюруется в пояснительной записке после титульного листа и включается в нумерацию работы.

Задания на курсовую работу должны быть подписаны ведущим преподавателем.

Задания должны иметь также даты выдачи задания и контрольного срока выполнения работы.

Задания на курсовое проектирование содержит две части:

теоретическую;

практическую.

Теоретическая часть задания предполагает выполнения анализа по одному из теоретических вопросов дисциплины «Проектирование, внедрение, сопровождение, настройка и эксплуатация информационных систем». Особое внимание при выполнении теоретического задания следует обратить на сравнительный анализ различных альтернативных вариантов или различных подходов к данной проблематике, построению систем и устройств.

2.2. Порядок выполнения теоретической части задания:

- изучить состояние вопроса на современном уровне, используя различные источники информации;
- определить наличие альтернативных вариантов решения данного вопроса;
- выполнить сравнительный анализ возможных вариантов;
- оформить пояснительную записку по теоретическому вопросу.

Практическая часть задания предполагает решение на ЭВМ трех задач: вычислительной, экономической и функциональной. При решении задачи на ЭВМ необходимо строго соблюдать последовательность выполнения всех этапов решения задач на ЭВМ.

2.3. Порядок выполнения практической части задания по каждой из решаемых задач:

- выполнить постановку задачи и ее формализацию в форме, удобной для дальнейшего решения задачи на ЭВМ;
- разработать алгоритм решения задачи на ЭВМ, представив его в словесной форме на естественном языке;
- написать программу на алгоритмическом языке согласно разработанному алгоритму;
- выбрать исходные данные и подготовить контрольный пример решения задачи на ЭВМ согласно варианту задания;
- ввести текст программы в ЭВМ, выполнить ее отладку и решение;
- проанализировать полученные результаты, убедиться в правильности выполнения контрольного примера;
- оформить пояснительную записку по данной решаемой задаче.

Пояснительная записка должна содержать также результаты решения задачи, тексты разработанных программ.

2.4. Структура пояснительной записки

Пояснительная записка должна оформляться в строгом соответствии с требования ГОСТ к текстовым документам. Как правило, пояснительная записка содержит введение, разделы и подразделы, приложения, список литературы и заключение.

Во введении дается краткая характеристика вопросов курсового проекта и возможных способов решения поставленных задач.

В теоретической части работы необходимо всесторонне раскрыть основные положения темы, обосновать их, попытаться раскрыть перспективы развития темы, решения проблем исследования систем управления и так далее.

Поэтому здесь описываются теоретические положения, раскрывающие сущность исследуемой темы. Дается характеристика изученности - разработанности темы на практике и в существующей научной литературе, на которую необходимы конкретные ссылки. Рассматривается освещение и анализ различных точек зрения на рассматриваемую тему.

В практической части работы необходимо поэтапно описать решение задачи на ЭВМ.

В заключении анализируются полученные результаты, и приводится краткое обоснование используемых при выполнении курсового проекта методов и приемов.

3 ОФОРМЛЕНИЕ КУРСОВЫХ РАБОТ И ИХ ЗАЩИТА

3.1 Требования к оформлению пояснительной записки

3.1.1 Оформление текста

Текст выполняется на листах формата А4 (210 х 297 мм) по ГОСТ 2.301**.**

Текст выполняют с применением печатающих и графических устройств вывода ЭВМ (ГОСТ 2.004).

На компьютере текст должен быть оформлен в текстовом редакторе Word for Windows версии не ниже 6.0.

Тип шрифта: Times New Roman Cyr. Шрифт основного текста: обычный, размер 14 пт. Шрифт заголовков разделов: обычный, размер 14 пт. Шрифт заголовков подразделов: обычный, размер 14 пт.

Межсимвольный интервал: обычный. Межстрочный интервал: полуторный.

Формулы должны быть оформлены в редакторе формул Equation Editor и вставлены в документ как объект.

Размеры шрифта для формул:

-обычный – 14 пт;

-крупный индекс – 10 пт;

-мелкий индекс – 8 пт;

-крупный символ – 20 пт;

-мелкий символ – 14 пт.

Иллюстрации должны быть вставлены в текст:

-либо командами ВСТАВКА-РИСУНОК, которые позволяют вставить рисунки из коллекции, из других программ и файлов, со сканера, созданные кнопками на панели рисования, автофигуры, объекты Word Art, диаграммы (все иллюстрации, вставляемые как рисунок, должны быть преобразованы в формат графических файлов, поддерживаемых Word);

-либо командами ВСТАВКА-ОБЪЕКТ, при этом необходимо, чтобы объект, в котором создана вставляемая иллюстрация, поддерживался редактором Word стандартной конфигурации.

Расстояние от верхней или нижней строки текста пояснительной записки до верхнего или нижнего поля листа должно быть не менее 10 мм. Абзацы в тексте начинают отступом, равным 1,25 пт.

3.1.2 Основная часть

Текст основной части разделяют на разделы, подразделы

Разделы должны иметь порядковые номера в пределах всего текста, обозначенные арабскими цифрами **без точки**.

Подразделы должны иметь нумерацию в пределах каждого раздела, номера подразделов состоят из номера раздела и номера подраздела, разделенных точкой. **В конце номера подраздела точка не ставится**.

Подраздел допускается разбивать на пункты, нумерация которых выполняется аналогично.

Пример 1.2.3 – обозначает раздел 1, подраздел 2, пункт 3.

Внутри пунктов или подпунктов могут быть приведены перечисления. Перед каждой позицией перечисления следует ставить дефис или, при необходимости ссылки на одно из перечислений, строчную букву, после которой ставится скобка. Для дальнейшей детализации перечислений необходимо использовать арабские цифры, после которых ставится скобка, а запись производится с абзацного отступа.

Пример

а) _____;

б) _____;

1) _____;

2) _____;

в) _____.

Наименования разделов и подразделов должны быть краткими. Наименование разделов и подразделов записывают с абзацного отступа с **первой прописной буквы без точки в конце, не подчеркивая. Переносы слов в заголовках не допускаются.**

Расстояние между заголовками и текстом должно быть равно 15 мм. Расстояние между заголовками раздела и подраздела - 8 мм. Расстояние между последней строкой текста и последующим заголовком подраздела - 15 мм.

Страницы курсовой работы следует нумеровать арабскими цифрами (1,2,3…), соблюдая сквозную нумерацию по всему тексту работы. Номер страницы проставляют в **центре верхней** части листа без точки. Титульный лист включается в общую нумерацию страниц. Номер страницы на титульном листе, обратной стороне титульного листа и задании не проставляется. Нумерация начинается с введения цифрой 5.

Иллюстрации и таблицы, расположенные на отдельных листах, включают в общую нумерацию страниц работы.

Опечатки, описки, графические неточности, обнаруженные в процессе выполнения, допускается исправлять подчисткой или закрашиванием белым корректором текста и нанесением на том же месте исправленного текста машинописным способом или черными чернилами. Помарки, следы не полностью удаленного прежнего текста не допускаются

3.1.3 Оформление иллюстраций

Иллюстрации (рисунки, чертежи, схемы, диаграммы) выполняют на листах дипломной работы или на листах чертежной бумаги формата А4 (210х297) ГОСТ 2.301 на печатающих и графических устройствах вывода ЭВМ.

Иллюстрации располагают после первой ссылки на них.

Допускается помещать иллюстрации вдоль длинной стороны текста с поворотом документа по часовой стрелке для чтения.

Все иллюстрации нумеруют арабскими цифрами сквозной нумерацией. Если один рисунок в тексте, то следует указать «Рисунок 1».

Допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой.

При ссылках на иллюстрации следует писать «... в соответствии с рисунком 2» при сквозной нумерации и «... в соответствии с рисунком 1.2» при нумерации в пределах раздела.

Рисунок 1.1

Иллюстрации могут иметь наименование и пояснительные данные (подрисуночный текст). Слово «Рисунок» и наименование помещают после пояснительных данных и располагают следующим образом:

Пример:

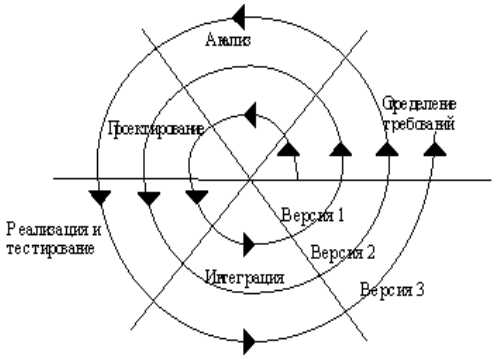


Рисунок 1 - Спиральная модель жизненного цикла информационной системы

3.1.4 Построение таблиц

Цифровой материал оформляют в виде таблиц согласно ГОСТ 2.105.

Таблицы следует нумеровать арабскими цифрами сквозной нумерацией. Если в тексте одна таблица, она должна быть обозначена Таблица 1. Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой.

Пример – Таблица 1.1

Слово «Таблица» и наименование помещают над таблицей следующим образом:

Таблица 1 – Показатели работы транзистора в разных режимах

Пример оформления таблицы:

Таблица 1 — Классификация рынка информационных систем

Локальные системы	Малые интегрированные системы	Средние интегрированные системы
1	2	3
БЭСТ Инотек Инфософт Супер-Менеджер	Concorde XAL Exact NS-2000 Platinum PRO/MIS Scala SunSystems БЭСТ-ПРО	Microsoft-Business Solutions - Navision, Axapta D Edwards (Robertson & Blums)

Продолжение таблицы 1

1	2	3
Супер-Менеджер Турбо-Бухгалтер Инфо-Бухгалтер	1С-Предприятие БОСС-Корпорация Галактика	MFG-Pro (QAD/BMS) SyteLine (СОКАП/SYMIX)

На все таблицы должны быть ссылки в тексте. При ссылке пишут слово «Таблица» с указанием ее номера.

Таблица может иметь заголовки и подзаголовки. Заголовки граф и строк таблицы следует писать с прописной буквы, а подзаголовки – со строчной буквы, если они составляют одно предложение с заголовком.

Графы таблицы допускается нумеровать для облегчения ссылок в тексте, при делении таблицы на части, а также при переносе части таблицы на следующую страницу.

Графу «Номер по порядку» в таблицу включать не допускается. При необходимости нумерации показателей, параметров или других данных порядковые номера следует указывать в первой графе (боковике) таблицы непосредственно перед их наименованием.

Если таблица не размещается на одном листе, допускается делить ее на части. Слово «Таблица...» указывают один раз слева над первой частью таблицы, над другими частями пишут слова «Продолжение таблицы...» с указанием номера таблицы.

Если все показатели, приведенные в графах таблицы, выражены в одной и той же единице физической величины, то ее обозначение необходимо помещать над таблицей справа, а при делении таблицы на части - над каждой ее частью.

Повторяющийся в графе таблицы текст, состоящий из одного слова, допускается заменять кавычками, если строки в таблице не разделены линиями. Если повторяющийся текст состоит из двух и более слов, то при первом повторении его заменяют словами «То же», а далее кавычками.

3.1.5 Список использованных источников литературы

В списке использованных при написании дипломной работы источников, должно быть не менее 25-30 источников, таких как Законы, ГОСТы, учебники, книги, статьи из журналов, ресурсы Интернет. Сведения об источниках следует располагать в порядке появления ссылок на источники в тексте дипломной работы, нумеровать арабскими цифрами и печатать с абзацного отступа.

Описание Web-страницы включает следующие обязательные элементы: Автор. Заглавие страницы. Указание типа документа. Электронный адрес [URL].

В конце текста приводится список использованных источников и литературы, нормативно-технической и другой документации, использованной при составлении пояснительной записки и вычерчивании графического материала.

Литература записывается и нумеруется в порядке ее упоминания в тексте. Оформление производится согласно ГОСТ 7.1. Ссылки на литературные источники приводятся в тексте и косых скобках в порядке их перечисления по списку источников, например /3/, /18/.

Пример:

Список использованных источников и литературы

1. Приказ от 26.12.94 № 170 Положение о бухгалтерском учете и отчетности в Российской Федерации, приказ Минфина РФ № 170 от 26.12.94.
2. Госс В.С., Семенюк Э.П., Урсул А.Д. Категории современной науки: Становление и развитие. – М.: Мысль, 2005. – 268 с.
3. Дик В.В. Информационные системы в экономике: Учебник / Под ред. проф. Дика В.В. – Москва.: Финансы и Статистика, 2006. –272 стр.: ил..
4. Москаленко А.Т. Методологические проблемы современной науки /Сост. А.Т. Москаленко. – М.: Политиздат, 2006. – 295 с.

3.1.6 Приложения

Каждое приложение следует начинать с нового листа с указанием наверху посередине слова «Приложение» и его обозначения. Приложение должно иметь заголовок, который записывают симметрично относительно текста с прописной буквы отдельной строкой.

Приложения обозначают прописными буквами русского алфавита, начиная с А, за исключением Ё, З, Й, О,Ч, Ь, Ы, Ъ.

Пример – Приложение В

Если в тексте работы одно приложение, то оно обозначается «Приложение А». Приложения располагают в порядке ссылок на них в тексте.

3.2. Порядок защиты курсовой работы

Выполненная работа проверяется руководителем, который дает допуск к защите или возвращает ее на доработку.

При положительном заключении работа допускается к защите. При наличии замечаний работа возвращается на доработку.

При подготовке к защите курсовой работы, студент обязан повторить теоретический материал дисциплины и подготовить доклад по материалу работы на 5-7 мин.

В докладе следует изложить сущность работы, методы решения задач, наиболее важные результаты и выводы по работе. После доклада студенту могут быть заданы вопросы, как по содержанию работы, так и по курсу в целом.

Оценка уровня выполнения курсовой работы, а также степени усвоения теоретического материала производится комиссией, состоящей не менее чем из двух человек с участием преподавателя – руководителя курсовой работы. Оценка за курсовую работу выставляется дифференцированно. Основными факторами, влияющими на оценку работы, являются:

- самостоятельность выполнения работы;
- использование в ней результатов своих исследований;
- уровень теоретической подготовки студента;
- грамотность обоснования предлагаемых мероприятий;
- практическая ценность принятых решений;
- качество оформления всех материалов курсовой работы;
- подготовленность доклада и правильность ответов на вопросы при защите курсовой работы

10. Примерные темы курсовых работ

1. Проектирование системы автоматизации документооборота учебного заведения на базе ИС 1С: Предприятие
2. Проектирование Web-интерфейса и средств формирования отчетности
3. Проектирование автоматизированного рабочего места руководителя (менеджера) подразделения организации.
4. Проектирование автоматизированной информационной системы по учету обеспеченности материалами процесса производства предприятия.
5. Проектирование АРМ оператора контрольно-испытательной станции
6. Проектирование АРМ сотрудника кредитного отдела банка
7. Проектирование АРМ экономиста по прогнозу закупок на предприятии оптовой торговли
8. Проектирование информационной системы учета бартерных операций
9. Проектирование информационной системы учета договоров и контроля за их исполнением
10. Проектирование информационной системы учета закупок товаров
11. Проектирование информационной системы учета запасов предприятия
12. Проектирование АС учета и оптимизации транспортных расходов на предприятии
13. Проектирование информационной системы учета материальных ресурсов предприятия
14. Проектирование информационной системы учета обмена валют
15. Проектирование информационной системы учета риэлтерских операций
16. Проектирование информационной системы учета сдельной оплаты труда
17. Проектирование информационной системы учета ценных бумаг
18. Проектирование БД учета осуждённых к наказаниям, не связанным с лишением свободы, для уголовно-исполнительных инспекций
19. Проектирование информационного и программного обеспечения информационной системы тестирования знаний
20. Проектирование информационной системы автоматизации учета и контроля публикации научных трудов.
21. Проектирование информационной системы «Организация учебного процесса в образовательном учреждении».
22. Проектирование информационной системы автоматизации деятельности кафедры.
23. Проектирование информационной подсистемы учета сведений о выполнении научно-исследовательских работ
24. Проектирование информационной системы автоматизации деятельности кафедры. Подсистема учета информации о конференциях и олимпиадах
25. Проектирование информационной системы учета офисной техники в среде 1С Предприятие
26. Проектирование ИС автотранспортного предприятия
27. Проектирование ИС ведения реестра акционеров в банке
28. Проектирование ИС поддержки биржевых торгов
29. Проектирование модели для организации работы фирмы с применением методологии SADT
30. Проектирование модели электронной подшивки (книги) на основе анализа отдельных страниц
31. Проектирование информационной подсистемы автоматизации складского учета
32. Проектирование информационной подсистемы автоматизации учета платежей по договорам
33. Проектирование информационной подсистемы регистрации командировочных удостоверений в информационной системе.
34. Проектирование информационной подсистемы учета амортизации основных средств
35. Проектирование информационной подсистемы учета внутреннего перемещения материалов
36. Проектирование информационной подсистемы учета операций по импорту товаров
37. Проектирование информационной подсистемы учета реализации товаров в оптовой торговле
38. Проектирование информационной подсистемы электронного каталога электронной библиотеки

39. Проектирование информационной системы автоматизации подготовки комплекта документации для заключения договоров на выполнение НИОКР
40. Проектирование информационной системы автоматизированного учета вычислительной техники подразделения
41. Проектирование информационной системы автоматизированной генерации Help-файлов на основе текстовых документов формата Microsoft Word
42. Проектирование информационной системы оценки качества кодирования видеоизображения
43. Проектирование информационной системы планирования начислений заработной платы для малого предприятия
44. Проектирование информационной системы учета и контроля деловой переписки
45. Проектирование программы автоматизации работы предприятия по торговле металлопластиковым профилем на основе системы «1С Предприятие»
46. Проектирование система цифровой обработки аудиосигналов в реальном масштабе времени
47. Проектирование информационной системы автоматизации кассовых операций торгового предприятия
48. Проектирование информационной системы автоматизации учета выбытия денежных средств с расчетного счета организации
49. Проектирование информационной системы автоматизации учета повременно-премиальной оплаты труда в организации
50. Проектирование информационной системы автоматизации учета поступления и выбытия малоценных и быстроизнашивающихся предметов в коммерческой организации
51. Проектирование информационной системы автоматизации учета поступления и выбытия, основных средств на предприятии
52. Проектирование информационной системы автоматизации учета поступления и реализации товаров в розничной торговле
53. Проектирование информационной системы автоматизации учета расчетов за проживание в общежитии
54. Проектирование информационной системы автоматизации учета реализации и затрат на доставку мебели
55. Проектирование информационной системы учета и обслуживания заявок в среде 1С Предприятие
56. Проектирование СУБД ведения кафедральной библиотеки
57. Проектирование элемента автоматизированной информационной системы для обработки информации о технологическом процессе изготовления приборов измерения времени.
58. Проектирование элемента автоматизированной информационной системы «Учебная работа кафедры» с использованием WEB-технологий
59. Проектирование элемента автоматизированной информационно-управляющей системы «Абитуриент»
60. Проектирование элемента информационной системы внутризаводского планирования производственной деятельности предприятия
61. Проектирование элемента информационной системы для анализа результатов тестирования
62. Проектирование элемента информационной системы обучения и контроля знаний
63. Проектирование элемента информационной системы оценки профессиональных компетенций и психологических характеристик принимаемого на работу сотрудника
64. Проектирование элемента информационной системы по учету амбулаторных талонов и историй болезни для муниципальных медицинских учреждений
65. Проектирование элемента информационной системы разработки учебных планов
66. Проектирование элемента информационной системы создания и ведения технологических маршрутов изделий
67. Проектирование элемента информационной системы сравнения учебных планов
68. Проектирование элемента информационной системы электронных библиотек и больших архивов

69. Проектирование элемента системы автоматизации ведения БД и торговых операций для компьютерного магазина

70. Проектирование элемента системы автоматизации оформления полисов обязательного страхования автогражданской ответственности

11. Критерии оценки знаний студентов при проведении промежуточной аттестации

Оценка студенту за устный ответ на вопрос, письменную работу и выполнение работ при текущем контроле на занятиях, экзаменах выставляется:

- «отлично», если студент показал глубокие знания и понимание программного материала по поставленному вопросу, умело увязывает его с практикой, грамотно и логично строит ответ, быстро принимает оптимальные решения;
- «хорошо», если студент твердо знает программный материал, грамотно его излагает, не допускает существенных неточностей в ответе на вопрос, правильно применяет полученные знания при решении практических вопросов;
- «удовлетворительно», если студент имеет знания только основного материала по поставленному вопросу, но не усвоил его деталей, требует в отдельных случаях наводящих вопросов для принятия правильного решения, допускает отдельные неточности;
- «неудовлетворительно», если студент допускает грубые ошибки в ответе на поставленный вопрос, не может применять полученные знания для решения задач или выполнения практического задания.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

ПРОЕКТИРОВАНИЕ, ВНЕДРЕНИЕ, СОПРОВОЖДЕНИЕ, НАСТРОЙКА И ЭКСПЛУАТАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Подписано в печать 30.10.2016. Формат 60х84 1/16

Бумага офсетная. Печать лазерная. Усл.-печ. л. 2,1.

Тираж 100 экз. Заказ 732

Отпечатано в издательстве НТИ,

357108, г. Невинномысск, Бульвар Мира, 17