Документ подписан квалифицированной электронной подписью
Сертификат: [illegible]
Владелец: "АНО ВО ""РОССИЙСКИЙ НОВЫЙ УНИВЕРСИТЕТ"
Действителен: с 28.05.2026 по 28.08.2027

Елецкий филиал Автономной некоммерческой организации высшего образования «Российский новый университет»

Отделение среднего профессионального образования

РАБОЧАЯ ПРОГРАММА

учебной дисциплины

МДК.02.02 АВТОМАТИЗАЦИЯ ПРОЦЕССОВ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

для специальностей среднего профессионального образования

09.02.12 Техническая эксплуатация и сопровождение информационных систем

на базе основного общего образования

Елец
2026 год

Одобрена
предметной (цикловой)
комиссией общеобразовательных
дисциплин

Разработана на основе Федерального Государственного Образовательного Стандарта Среднего Профессионального Образования/ ФГОС СПО/ по специальности 09.02.12 Техническая эксплуатация и сопровождение информационных систем, утвержденного приказом Министерства просвещения РФ от 10.03.2025 г. N 184

Протокол №____5____
от «28»___01___2026 г.
Председатель предметной
(цикловой) комиссии
_____/ С.В. Толоконников

Начальник отделения СПО
_____/ О.В. Рыжкова

Составитель (автор):_____/пр. С.В. Толоконников

Рецензент: Суздальская Ева Анатольевна, заместитель начальника отдела эксплуатации средств вычислительной техники, программного обеспечения и телекоммуникационного оборудования управления информационных технологий – Отделение Социального фонда Российской Федерации по Липецкой области, г. Елец

1.НАИМЕНОВАНИЕ И ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Программа учебной дисциплины Автоматизация процессов тестирования программного обеспечения предназначена для изучения тестирования программного обеспечения в профессиональных образовательных организациях, реализующих образовательную программу среднего общего образования в пределах освоения образовательной программы СПО на базе основного общего образования.

Программа разработана на основе Федерального Государственного Образовательного Стандарта Среднего Профессионального Образования/ ФГОС СПО/ по специальности 09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ (приказ Минобрнауки России от 10.03.2025 г. № 184).

Учебная дисциплина Автоматизация процессов тестирования программного обеспечения изучается в профессиональном цикле учебного плана ОП на базе основного общего образования с получением среднего общего образования. В учебных планах учебная дисциплина Автоматизация процессов тестирования программного обеспечения входит в состав профессиональных учебных дисциплин ФГОС среднего общего образования.

Целью учебной дисциплины является формирование у студентов знаний и навыков, необходимых для эффективного применения программных средств в процессе тестирования программного обеспечения.

Изучение дисциплины направлено на подготовку обучающихся к осуществлению профессиональной деятельности, выполнению обобщённых трудовых функций, трудовых действий, определённых профессиональным стандартом, соответствующим ОП.

Развитие у обучающихся навыков командной работы, межличностной коммуникации, принятия решений, лидерских качеств обеспечивается чтением лекций, проведением занятий с применением активных методов обучения, содержание которых разработано на основе результатов научных исследований, проводимых Университетом, в том числе с учетом региональных особенностей профессиональной деятельности выпускников и потребностей работодателей.

2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫЕ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

В результате освоения дисциплины обучающийся по программе спо по специальности должен:

- Осуществлять подготовку тестовых данных в соответствии с заданием на тестирование программного обеспечения; (ПК 2.1.)

Знать:

– основную терминологию по тестированию ПО;

– язык, на котором написана техническая документация тестируемого ПО на уровне, достаточном для чтения технической документации;

– основные термины и сокращения, используемые в технической документации и принятые в организации;

– процедуры обеспечения безопасности при выполнении тестирования ПО;

– область применения инструментальных средств для выполнения тестирования ПО;

– особенности основных операционных систем;

– требования по обеспечению безопасности аппаратных и программных средств автоматизированных систем, используемых при выполнении тестовых процедур, включая вопросы антивирусной защиты.

Уметь:

– устанавливать корректную последовательность операций при выполнении тестирования ПО;

– выявлять недостающую информацию для выполнения тестирования ПО в заданном объеме;

– устанавливать операционные системы;

– выполнять базовую настройку операционных систем;

– подготавливать необходимые средства и ресурсы для выполнения задания по тестированию ПО;

– составлять отчет о результатах подготовки к выполнению тестирования ПО.

- Выполнять тестирование программного обеспечения; (ПК 2.2.)

Знать:

– нормативно технические материалы по вопросам испытания и тестирования ПО;

– основные термины и сокращения, используемые в технической документации и принятые в организации;

– основы работы в операционной системе, в которой производится тестирование, на уровне, необходимом для тестирования ПО соответствующего типа

– основы теории алгоритмов и дискретной математики в объеме полученного профессионального образования;

– синтаксис языка программирования тестируемого ПО, особенности программирования на этом языке, стандартные библиотеки языка программирования.

Уметь:

– выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования;

– использовать системы контроля дефектов ПО;

– составлять отчет о выполнении тестирования ПО;

– работать в команде со специалистами по тестированию ПО и разработчиками.

- Тестировать эксплуатационную и техническую документацию на программное; (ПК 2.3.)

Знать:

– нормативно технические материалы по вопросам испытания и тестирования ПО;

– основные понятия о качестве ПО;

– виды технической документации;

– требования по обеспечению безопасности аппаратных и программных средств автоматизированных систем, используемых при выполнении тестовых процедур, включая вопросы антивирусной защиты;

– основы работы в операционной системе, в которой производится тестирование, на уровне, необходимом для тестирования разработанного ПО.

Уметь:

– читать техническую документацию на ПО в объеме, необходимом для выполнения задания;

– оформлять техническую документацию на ПО в рамках своей компетенции;

– составлять отчет о тестировании эксплуатационной и технической документации на ПО.

- Проводить регрессионные виды тестирования по разработанным тестовым случаям в соответствии с документацией на программное обеспечение и анализ результатов тестирования; (ПК 2.4.)

Знать:

– техники тестирования ПО, базирующиеся на интуиции и опыте инженера;

– техники тестирования ПО, базирующиеся на спецификации;

– техники тестирования ПО, ориентированные на код;

– тестирование ПО, ориентированное на дефекты;

– техники тестирования ПО, базирующиеся на условиях использования;

– тестирование ПО, базирующееся на надежности инженерного процесса;

– техники тестирования ПО, базирующиеся на природе приложения;

– стандарты оформления кода для используемых языков программирования;

– основные термины и сокращения, используемые в технической документации и принятые в организации;

– основы алгоритмизации и программирования;

– жизненный цикл программного продукта.

Уметь:

– составлять сценарии поведения пользователей ПО;

– выполнять интеграционное и модульное тестирование ПО;

– выполнять статическое тестирование ПО;

– использовать специальное ПО для автоматизированного тестирования ПО при необходимости;

– составлять отчет о проведении тестирования ПО по разработанным тестовым случаям;

– взаимодействовать с членами команды разработчиков ПО;

– использовать системы автоматизированного тестирования ПО.

- Выполнять восстановление тестов после сбоев, повлекших за собой нарушение работы системы, в том числе автоматизированных тестов; (ПК 2.5.)

Знать:

– архитектуру тестируемой системы;

– основы работы в операционной системе, в которой производится тестирование, на уровне, необходимом для тестирования разработанного ПО;

– техники тестирования ПО, базирующиеся на интуиции и опыте инженера;

– техники тестирования ПО, базирующиеся на спецификации;

– техники тестирования ПО, ориентированные на код;

– тестирование ПО, ориентированное на дефекты;

– техники тестирования ПО, базирующиеся на условиях использования;

– тестирование ПО, базирующееся на надежности инженерного процесса;

– техники тестирования ПО, базирующиеся на природе приложения;

– принципы регрессионного тестирования ПО;

– алгоритмы решения типовых задач, области и способы их применения;

– основные термины и сокращения, используемые в технической документации и принятые в организации.

Уметь:

– находить и использовать информацию, необходимую для восстановления тестов после сбоя;

– взаимодействовать с командой разработчиков при восстановлении системы после сбоя;

– применять языки программирования для написания программного кода;

– использовать системы автоматизированного тестирования ПО;

– составлять отчет о восстановлении работоспособности ПО.

- Выполнять проверку исправленных дефектов и оформление результатов тестирования; (ПК 2.6.)

Знать:

– жизненный цикл ПО, жизненный цикл дефекта ПО;

– принципы регрессионного тестирования ПО;

– техники тестирования ПО, базирующиеся на интуиции и опыте инженера;

– техники тестирования ПО, базирующиеся на спецификации;

– техники тестирования ПО, ориентированные на код;
– тестирование ПО, ориентированное на дефекты;
– техники тестирования ПО, базирующиеся на условиях использования;
– тестирование ПО, базирующееся на надежности инженерного процесса;
– техники тестирования ПО, базирующиеся на природе приложения;
– основные инструментальные средства организации работы в команде.

Уметь:
– взаимодействовать с членами команды разработчиков ПО;
– использовать инструменты командной работы над проектом ПО;
– вносить изменения в скрипты автоматизированных тестов при необходимости;
– использовать шаблоны тестов;
– применять тесты.

3. ОБЪЕМ ДИСЦИПЛИНЫ В ЗАЧЕТНЫХ ЕДИНИЦАХ С УКАЗАНИЕМ КОЛИЧЕСТВА АКАДЕМИЧЕСКИХ ЧАСОВ, ВЫДЕЛЕННЫХ НА КОНТАКТНУЮ РАБОТУ ОБУЧАЮЩИХСЯ С ПРЕПОДАВАТЕЛЕМ (ПО ВИДАМ УЧЕБНЫХ ЗАНЯТИЙ) И НА САМОСТОЯТЕЛЬНУЮ РАБОТУ ОБУЧАЮЩИХСЯ

Общая трудоемкость дисциплины составляет 1,5 зачетных единицы (54 час.).

Вид учебной работы	Объем часов
Максимальная учебная нагрузка (всего):	**54**
Обязательная аудиторная учебная нагрузка (всего)	**50**
в том числе:	
Лекционные занятия (Лек)	16
Практические занятия (Пр)	32
Контрольная работа (Контр)	0,3
Текущие консультации (ТКонс)	1,7
Самостоятельная работа обучающегося (всего)	**4**
Внеаудиторная самостоятельная работа	

Распределение учебного времени по темам и видам учебных занятий очная форма обучения

№	Наименование разделов, тем учебных занятий	Всего часов	Контактная работа с преподавателем							Сам. раб	Формируемые результаты обучения
			Всего	Лек	Пр	З	Конс	КоР	ТКонс		
Автоматизация процессов тестирования программного обеспечения											
1	Модульное тестирование	13	12	4	8					1	ПК 2.1, ПК 2.2, ПК 2.3, ПК 2.4, ПК 2.5, ПК 2.6.
2	Тестирование документации	9	8	4	4					1	
3	Основы интеграционного и системного тестирования	17	16	4	12					1	
4	Надежность, и безопасность ПО в процессе эксплуатации	13	12	4	8					1	
5	Промежуточная аттестация (контрольная работа)	2	2					0,3	1,7		

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ, СТРУКТУРИРОВАННОЕ ПО ТЕМАМ (РАЗДЕЛАМ)

Тема 1. Модульное тестирование

Основные средства модульного тестирования в актуальных операционных системах. Синтаксис языков программирования для проектирования модульных тестов Содержание отчета о тестировании. Типовые формы отчетов Системы контроля дефектов. Логирование. Сбор статистики. Область применения модульных автотестов. Средства создания и сопровождения библиотеки тестов. Средства развертывания и интеграции автотестов.

Тема 2. Тестирование документации

Виды технической документации. Актуальные стандарты оформления технической документации. Инструменты и методы тестирования документации на ПО. Программы и методики приемочного тестирования. Альфа- и бета тестирование, сбор результатов.

Тема 3. Основы интеграционного и системного тестирования

Основные инструменты интеграционного тестирования. Особенности организации системного тестирования. Средства автоматизации тестирования пользовательского интерфейса (тестирование UI). Инструменты тестирования интерфейсов (API-тестирования). Встроенные инструменты разработчика для тестирования в браузерах. REST и SOAP: структура запроса, методы, коды ответов. Типовые SQL-запросы для тестирования подключения баз данных. Понятие заглушки. Имитаторы (Mock). Настройка тестового окружения. Актуальные фреймворки для тестирования кода. Настройка автоматического сбора и хранения логов. Регрессионное тестирование в жизненном цикле ПО.

Тема 4. Надежность, и безопасность ПО в процессе эксплуатации

Методы тестирования безопасности. Инструменты выявления уязвимостей в исходном коде: статический анализ и имитаторы вредоносных атак. Уровни безопасности и защиты данных. Методы тестирования производительности. Методы нагрузочного тестирования. Принципы и методы обеспечения безопасности тестирования в процессе эксплуатации Планирование процесса системного тестирования, оценка рисков и затрат. Типовые причины сбоя системы в процессе тестирования. Стохастическое, стрессовое и другие виды тестирования, их область применения, достоинства и недостатки.

Тема 5. Промежуточная аттестация (контрольная работа)

Решение конкретной теоретической или практической задачи для установления степени усвоения изучаемого материала студентом.

См. приложение 1 «Контрольная работа»

Планы практических занятий

Тема 1. Модульное тестирование.

Время - 8 час.

Основные вопросы: Основные средства модульного тестирования в актуальных операционных системах. Синтаксис языков программирования для проектирования модульных тестов Содержание отчета о тестировании. Типовые формы отчетов Системы контроля дефектов. Логирование. Сбор статистики. Область применения модульных автотестов. Средства создания и сопровождения библиотеки тестов. Средства развертывания и интеграции автотестов. Разработка модульных автотестов для настольных, приложений. Оформление и размещение отчетов о тестировании в соответствии с заданием (создание библиотеки тестов). Запуск автотестов и сбор статистик. Ооформление отчета по результатам анализа статистики. Сборка и запуск тестов из консоли.

См. Приложение 2 «Практические работы».

Тема 2. Тестирование документации.

Время - 4 час.

Основные вопросы: Виды технической документации. Актуальные стандарты оформления технической документации. Инструменты и методы тестирования документации на ПО. Программы и методики приемочного тестирования. Альфа- и бета тестирование, сбор результатов. Выполнение приемочного тестирования и оформление отчета о его результатах. Выполнение тестирования пользовательской документации и оформление отчета о его результатах.

См. Приложение 2 «Практические работы».

Тема 3. Основы интеграционного и системного тестирования.

Время - 12 час.

Основные вопросы: Основные инструменты интеграционного тестирования. Особенности организации системного тестирования. Средства автоматизации тестирования пользовательского интерфейса (тестирование UI). Инструменты тестирования интерфейсов (API-тестирования). Встроенные инструменты разработчика для тестирования в браузерах. REST и SOAP: структура запроса, методы, коды ответов. Типовые SQL-запросы для тестирования подключения баз данных. Понятие заглушки. Имитаторы (Mock). Настройка тестового окружения. Актуальные фреймворки для тестирования кода. Настройка автоматического сбора и хранения логов. Регрессионное тестирование в жизненном цикле ПО. Разработка и запуск тестов для тестирования подключения баз данных. Оформление отчета о тестировании. Разработка автотеста с заглушками и имитаторами (моками). Настройка выбранной системы логирования с учетом ротации файлов. Анализ логов и подготовка отчета о результатах мониторига.

См. Приложение 2 «Практические работы».

Тема 4. Надежность, и безопасность ПО в процессе эксплуатации.

Время - 8 час.

Основные вопросы: Методы тестирования безопасности. Инструменты выявления уязвимостей в исходном коде: статический анализ и имитаторы вредоносных атак. Уровни безопасности и защиты данных. Методы тестирования производительности. Методы нагрузочного тестирования. Принципы и методы обеспечения безопасности тестирования в процессе эксплуатации Планирование процесса системного тестирования, оценка рисков и затрат. Типовые причины сбоя системы в процессе тестирования. Стохастическое, стрессовое и другие виды тестирования, их область применения, достоинства и недостатки. Выявление типовых уязвимостей кода веб-страниц в ручном режиме. Тестирование производительности приложения с помощью заданного инструментария. Тестирование авторизации, в том числе двухфакторной авторизации. Разработка сценария, плана тестирования, тестовых пакетов и задания для тестирования безопасности приложения. Комплексное тестирование безопасности приложения и подготовка отчета о результатах тестирования.

См. Приложение 2 «Практические работы».

5. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

5.1.1. Методические указания:

Самостоятельная работа обучающихся, это работа, выполняемая во внеаудиторное (аудиторное) время при методическом руководстве преподавателя с целью овладения необходимыми компетенциями по дисциплине, дополнительными знаниями, умениями и навыками, опытом творческой, исследовательской деятельности, развития самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного уровня.

Видами заданий для самостоятельной работы являются:

1. Заполнение глоссария
2. Поиск и анализ источников и литературы
3. Подготовка сообщений, презентаций, докладов, рефератов
4. Выполнение практических работ
5. Решение задач

Для выполнения заданий по самостоятельной работе рекомендуется использовать материалы учебников и интернет ресурсов:

Основная литература:

1. Щербак, А. В. Поддержка и тестирование программных модулей : учебник для среднего профессионального образования / А. В. Щербак. — Москва : Издательство Юрайт, 2026. — 145 с. — (Профессиональное образование). — ISBN 978-5-534-19290-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/590253

2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для среднего профессионального образования / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2026. — 312 с. — (Профессиональное образование). — ISBN 978-5-534-13221-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/588246

Дополнительная литература:

1. Щербак, А. В. Тестирование программного обеспечения : учебник для вузов / А. В. Щербак. — Москва : Издательство Юрайт, 2026. — 145 с. — (Высшее образование). — ISBN 978-5-534-19291-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/590250

2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2026. — 312 с. — (Высшее

образование). — ISBN 978-5-9916-9043-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/584673

Интернет-ресурсы:

https://urait.ru Образовательная платформа Юрайт. Для вузов и ссузов

https://acmp.ru/ Школа программиста

https://proglib.io/p/awesome-algorithms/ Изучаем алгоритмы: полезные книги, веб-сайты, онлайн-курсы и видеоматериалы

https://gb.ru/blog/gde-popraktikovatsya-v-programmirovanii-30-resursov/ Задачи, упражнения и соревнования для начинающих и опытных кодеров

http://www.gost.ru Официальный сайт Федерального агентства по техническому регулированию и метрологии

http://www.vniis.ru/ Всероссийский научно-исследовательский институт сертификации (ВНИИС)

http://mon.gov.ru/ (Сайт Министерства образования и науки РФ) /

www.edu.ru (Российский образовательный портал).

www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).

Российская государственная публичная библиотека http://elibrary.rsl.ru/

http://www.rambler.ru – Русская поисковая система.

http://www.yandex.ru – Русская поисковая система.

http://biblioteka.net.ru – Библиотека компьютерных учебников.

Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –http://www.edu-all.ru/

Рекомендации по составлению и заполнению глоссария:

Использование глоссария способствует прояснению, запоминанию и уточнению ключевых понятий дисциплины. Глоссарий охватывает все узкоспециализированные термины, встречающиеся в тексте. Глоссарий должен содержать не менее 30 терминов, они должны быть перечислены в алфавитном порядке, соблюдена нумерация. Глоссарий должен быть оформлен по принципу реферативной работы, в обязательном порядке присутствует титульный лист и нумерация страниц. Объем работы должен составлять 10-15 страниц. Тщательно проработанный глоссарий помогает избежать разночтений и улучшить в целом качество всей документации. В глоссарии включаются самые частотные термины и фразы, а также все ключевые термины с толкованием их смысла. Глоссарии могут содержать отдельные слова, фразы, аббревиатуры, слоганы и даже целые предложения.

Рекомендации по поиску и анализу источников и литературы

Найдите дополнительные источники литературы и источники, максимально соответствующие теме (заданию) и рекомендованные преподавателем. Ознакомьтесь с аннотацией источника, ее содержанием по оглавлению. Найдите необходимый материал и просмотрите его содержание. Определите форму записи (конспект, тезисы, цитаты, выписки, факты), составьте план и обсудите с преподавателем.

Выпишите незнакомые слова и найдите их определение в словаре или спросите у преподавателя. Выполните краткий конспект. В конце своих записей сделайте резюме к тексту.

Рекомендации по подготовке сообщений, презентаций, докладов, рефератов

Внимательно прочитайте тему доклада, составьте предполагаемый план раскрытия темы, обсудите его с преподавателем. При работе с источниками информации тщательно отбирайте примеры, факты, следите за логикой их обоснования. Помните, что объем вашего выступления не должен выходить за предложенные преподавателем нормы. Выберите из подготовленного текста наиболее существенные моменты будущего выступления и согласуйте с преподавателем. При необходимости сопроводите свое выступление иллюстрациями или подготовьте презентацию. Презентация должна быть краткой, доступной и целостной. Главное в презентации – содержание. Придерживайтесь единого стилевого оформления. Не рекомендуется использовать в стилевом оформлении презентации более трех цветов или трех типов шрифта. Оформление слайда не должно отвлекать от его содержательной части. Не все слайды презентации должны быть выдержаны в одном стиле.

На одном слайде не должно быть слишком много информационных блоков, обычно до трёх. Ключевые слова в информационном блоке необходимо выделить. Наиболее важную информацию – поместить в центр слайда. Приводите факты, цифры, графики - это хорошая поддержка для вашего выступления. Голый текст никого не заинтересует (если вы, конечно, не несете слушателям свет новой истины). Применяйте высококонтрастные цвета, крупные шрифты и четкие иллюстрации. Фотографии, рисунки и другие иллюстрации старайтесь размещать на отдельных слайдах. То же относится большим диаграммам, таблицам, схемам и графикам. Не забывайте указывать источники информации.

Приступая к подготовке реферата, доклада, на первом этапе, нужно собрать информацию. Можно использовать книги, статьи, научные публикации и интернет-ресурсы. Во время исследования нужно делать заметки, выделяя ключевые моменты, факты и примеры. Доклад необходимо структурировать. Обычно он включает три раздела: Введение - представить тему и объяснить её актуальность. Основная часть - разделить на несколько логически связанных частей. Каждая часть должна содержать ключевые идеи и факты. Заключение - подвести итоги, сделать выводы, дать рекомендации или предложить направления для дальнейшего изучения темы.

Подготовить визуальные материалы. Если доклад предполагает наличие презентации, нужно создать слайды, которые помогут визуализировать информацию.

Рекомендации по выполнению практических работ

Ознакомьтесь с темой, целью и заданием практической работы. Повторите теоретический материал, необходимый для выполнения работы.

Пройдите инструктаж по технике безопасности, если работа связана с оборудованием или лабораторными условиями. Запишите дату, тему и цель работы.

Следуйте предложенному алгоритму действий, если он есть. Работайте аккуратно, используйте линейки, карандаши, разборчивый почерк. Таблицы и схемы оформляйте чётко и по правилам.

По итогам работы сформулируйте выводы, ответьте на контрольные вопросы.

Проверьте, чтобы все расчёты и ответы были выполнены полностью и правильно.

Если работа письменная, составьте план, продумайте логику изложения, аргументируйте свои решения.

При возникновении трудностей обращайтесь к преподавателю или дополнительной литературе.

Соблюдайте требования к оформлению: аккуратность, читаемость, соответствие стандартам.

Рекомендации по самостоятельному изучению:
Начните с базовых понятий каждой категории
Решайте практические задачи регулярно
Используйте несколько источников для лучшего понимания
Создавайте конспекты и схемы
Применяйте изученное на практике
Участвуйте в онлайн-форумах и обсуждениях
Проверяйте свои знания через тесты и контрольные работы

Дополнительные советы:
Регулярность - занимайтесь минимум 30 минут ежедневно
Системность - следуйте логической последовательности изучения
Практика - решайте как можно больше задач
Обратная связь - проверяйте решения и исправляйте ошибки
Мотивация - ставьте конкретные цели и отслеживайте прогресс

Задания по самостоятельной работе должны быть выполнены до начала промежуточной аттестации.

Критерии оценки защиты мультимедийной презентации:

Оценка 5 ставится, если выполнены все требования к созданию мультимедийной презентации: обозначена тема, выполнен краткий анализ различных источников по данной теме, сформулированы выводы, выдержан объѐм, соблюдены требования к оформлению: фон, шрифт, объем текста, рисунки, анимация, выдержан единый стиль оформления

Оценка 4 – основные требования созданию мультимедийной презентации и ее защите выполнены, но при этом допущены недочѐты: имеются неточности в изложении материала; отсутствует логическая последовательность; не выдержан объѐм презентации; имеются упущения в оформлении: не полностью соблюден единый стиль оформления.

Оценка 3 – имеются существенные отступления от требований к созданию мультимедийной презентации: не полностью раскрыта тема, имеются существенные недочеты в оформлении.

Оценка 2 – тема презентации не раскрыта, не соблюден единый стиль оформления слайдов.

Критерии оценки выступления с сообщением:

Оценка 5 - выполнены все требования к написанию и защите сообщения: обозначена проблема и обоснована еѐ актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объѐм, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.

Оценка 4 – основные требования к сообщению и их защите выполнены, но при этом допущены недочѐты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объѐм; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы.

Оценка 3 – имеются существенные отступления от требований: тема освещена лишь частично; допущены фактические ошибки в содержании или при ответе на дополнительные вопросы; во время защиты отсутствует вывод.

Оценка 2 – тема не раскрыта, обнаруживается существенное непонимание проблемы.

Критерии оценки устных ответов:

Оценка 5 – обучающийся полно излагает изученный материал, дает правильное определение языковых понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только по учебнику, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.

Оценка 4 – обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «5», но допускает 1-2 ошибки, которые сам же исправляет, и 1-2 недочета в последовательности и языковом оформлении излагаемого.

Оценка 3 – обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий и формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.

Оценка 2 – обучающийся обнаруживает незнание большей части соответствующего раздела изучаемого материала, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал. Оценка «2» отмечает такие недостатки в подготовке обучающегося, которые являются серьезным препятствием к успешному овладению последующим материалом.

5.1.2. Задания для повторения и углубления приобретаемых знаний.

№	Задания
1	Типы связей между функциями
2	Характеристика современных Case-средств
3	Методология ARIS
4	Программный продукт ARIS Express
5	Применение имитационного моделирования

1.2. Задания, направленные на формирование профессиональных умений.

№	Задания
6	С помощью конструктора графического интерфейса пользователя IDE NetBeans создать графический интерфейс пользователя для приложения с именем MyApplication , по плану: - создать GUI Container (контейнер GUI), добавить компоненты, выронить их положение, изменить размер, осуществить привязку компонентов, продемонстрировать управление поведением при автоматическом изменении размера и редактировании свойства компонентов
7	Составьте физическую диаграмму в соответствии с описанием деятельности компании дистрибьютора «МЕД». Компания дистрибьютор «МЕД» закупает медицинские препараты отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Планирование закупок компания осуществляет на основании статистики продаж, которую предоставляют сеть аптек и дистрибьюторы. Компания осуществляет доставку медикаментов как собственным транспортом, так и с помощью услуг сторонних организаций.
8	Создайте и настройте Persistent Volumes для хранения данных
9	Создайте шаблоны для развёртывания инфраструктуры
10	Настройте систему мониторинга приложения

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

6.1. Средства оценивания в ходе текущего контроля:

Перечень знаний, осваиваемых в рамках учебной дисциплины

Результаты обучения	Критерии оценки	Методы оценки
– основную терминологию по тестированию ПО; – язык, на котором написана техническая документация тестируемого ПО на уровне, достаточном для чтения технической документации; – основные термины и сокращения, используемые в технической документации и принятые в организации; – процедуры обеспечения безопасности при выполнении тестирования ПО; – область применения инструментальных средств для выполнения тестирования ПО; – особенности основных операционных систем; – требования по обеспечению безопасности аппаратных и программных средств автоматизированных систем, используемых при выполнении тестовых процедур, включая вопросы антивирусной защиты. – устанавливать корректную последовательность	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; - контрольная работа

операций при выполнении тестирования ПО; – выявлять недостающую информацию для выполнения тестирования ПО в заданном объеме; – устанавливать операционные системы; – выполнять базовую настройку операционных систем; – подготавливать необходимые средства и ресурсы для выполнения задания по тестированию ПО; – составлять отчет о результатах подготовки к выполнению тестирования ПО.		
– нормативно технические материалы по вопросам испытания и тестирования ПО; – основные термины и сокращения, используемые в технической документации и принятые в организации; – основы работы в операционной системе, в которой производится тестирование, на уровне, необходимом для тестирования ПО соответствующего типа – основы теории алгоритмов и дискретной математики в объеме полученного профессионального образования; – синтаксис языка программирования	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; - контрольная работа

тестируемого ПО, особенности программирования на этом языке, стандартные библиотеки языка программирования. – выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования; – использовать системы контроля дефектов ПО; – составлять отчет о выполнении тестирования ПО; – работать в команде со специалистами по тестированию ПО и разработчиками.		
– нормативно технические материалы по вопросам испытания и тестирования ПО; – основные понятия о качестве ПО; – виды технической документации; – требования по обеспечению безопасности аппаратных и программных средств автоматизированных систем, используемых при выполнении тестовых процедур, включая вопросы антивирусной защиты; – основы работы в операционной системе, в которой производится	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы;

тестирование, на уровне, необходимом для тестирования разработанного ПО. – читать техническую документацию на ПО в объеме, необходимом для выполнения задания; – оформлять техническую документацию на ПО в рамках своей компетенции; – составлять отчет о тестировании эксплуатационной и технической документации на ПО.		- контрольная работа
– техники тестирования ПО, базирующиеся на интуиции и опыте инженера; – техники тестирования ПО, базирующиеся на спецификации; – техники тестирования ПО, ориентированные на код; – тестирование ПО, ориентированное на дефекты; – техники тестирования ПО, базирующиеся на условиях использования; – тестирование ПО, базирующееся на надежности инженерного процесса; – техники тестирования ПО, базирующиеся на природе приложения;	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; - контрольная работа

– стандарты оформления кода для используемых языков программирования; – основные термины и сокращения, используемые в технической документации и принятые в организации; – основы алгоритмизации и программирования; – жизненный цикл программного продукта. – составлять сценарии поведения пользователей ПО; – выполнять интеграционное и модульное тестирование ПО; – выполнять статическое тестирование ПО; – использовать специальное ПО для автоматизированного тестирования ПО при необходимости; – составлять отчет о проведении тестирования ПО по разработанным тестовым случаям; – взаимодействовать с членами команды разработчиков ПО; – использовать системы автоматизированного тестирования ПО.		
– архитектуру тестируемой системы; – основы работы в операционной системе, в	- знание и понимание материала; - грамотное использование терминологии в ходе	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины;

которой производится тестирование, на уровне, необходимом для тестирования разработанного ПО; – техники тестирования ПО, базирующиеся на интуиции и опыте инженера; – техники тестирования ПО, базирующиеся на спецификации; – техники тестирования ПО, ориентированные на код; – тестирование ПО, ориентированное на дефекты; – техники тестирования ПО, базирующиеся на условиях использования; – тестирование ПО, базирующееся на надежности инженерного процесса; – техники тестирования ПО, базирующиеся на природе приложения; – принципы регрессионного тестирования ПО; – алгоритмы решения типовых задач, области и способы их применения; – основные термины и сокращения, используемые в технической документации и принятые в организации. – находить и	устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; - контрольная работа

использовать информацию, необходимую для восстановления тестов после сбоя; – взаимодействовать с командой разработчиков при восстановлении системы после сбоя; – применять языки программирования для написания программного кода; – использовать системы автоматизированного тестирования ПО; – составлять отчет о восстановлении работоспособности ПО.		
– жизненный цикл ПО, жизненный цикл дефекта ПО; – принципы регрессионного тестирования ПО; – техники тестирования ПО, базирующиеся на интуиции и опыте инженера; – техники тестирования ПО, базирующиеся на спецификации; – техники тестирования ПО, ориентированные на код; – тестирование ПО, ориентированное на дефекты; – техники тестирования ПО, базирующиеся на условиях использования;	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; - контрольная работа

– тестирование ПО, базирующееся на надежности инженерного процесса; – техники тестирования ПО, базирующиеся на природе приложения; – основные инструментальные средства организации работы в команде. – взаимодействовать с членами команды разработчиков ПО; – использовать инструменты командной работы над проектом ПО; – вносить изменения в скрипты автоматизированных тестов при необходимости; – использовать шаблоны тестов; – применять тесты.		

6.2. ФОС для текущего контроля:

См. Приложение 3 «ФОС для текущего контроля»

6.3 ФОС для промежуточной аттестации:

Задания для оценки знаний.

№	Задания
1	Вопросы к контрольной работе 1-12
2	1. Необходимые инструментальные средства разработки программ 2. Часто используемые инструментальные средства разработки программ 3. Специализированные инструментальные средства разработки программ 4. Интегрированные среды разработки 5. Средства разработки программного обеспечения 6. Определение «разработка программ» 7. Три этапа разработки программ 8. Средства проектирования приложений 9. Средства реализации программного кода

	10.Средства тестирования программ 11. Классы инструментальных средств разработки программ 12. Четыре категории инструментальных программ, применяемые при проектировании экспертных систем.
3	Вопросы к контрольной работе 13-25
4	13. Оболочки экспертных систем 14. Языки программирования высокого уровня 15. Среда программирования, поддерживающая несколько парадигм 16.Дополнительные модули 17. Языки инженерии знаний 18. Язык логического программирования Prolog 19. Язык функционального программирования Lisp 20.Средства автоматизации разработки экспертных систем 21.Общее программное обеспечение 22.Специальное программное обеспечение 23. Инструментальная система технологии программирования 24. Четыре класса компьютерной поддержки инструментальных систем технологий программирования 25. Комплексность
5	Вопросы к контрольной работе 26-38
6	26. Ориентированность на коллективную разработку 27.Технологическая определенность 28.Интегрированность 29.Компоненты инструментальных систем технологий программирования 30.База данных разработки 31.Инструментарий 32.Интерфейсы 33.Общая архитектура инструментальных систем технологий программирования 34. Инструментальная система поддержки проекта 35.Языково-зависимая инструментальная система 36.Пользовательский интерфейс 37.Схема организации взаимодействия компьютера и пользователя 38.Процедурно-ориентированный подход к разработке интерфейсов
7	Вопросы к контрольной работе 39-50
8	39.Объектно-ориентированный подход к разработке интерфейсов 40.Типы интерфейсов 41.Интерфейс-меню 42.Интерфейсы со свободной навигацией 43.Критерии оценки интерфейса пользователем 44. Факторы появления Case-технологий 45. Что такое Case-технология? 46. Компоненты интегрированного Case-средства 47.Классификация по категориям Case-средств 48.Классификация по типам Case-средств

	49.Вспомогательные типы Case-средств 50.Объектно-ориентированный подход к разработке ИС
9	Вопросы к контрольной работе 51-60
10	51.Качества организации для успешного внедрения Case-средств 52.Факторы появления Case-технологий 53.Структурный системный анализ 54.Диаграммы «сущность-связь» 55.Язык графического описания UML 56.Диаграмма компонентов 57.Диаграмма композитной структуры 58.Диаграмма развертывания 59. Диаграмма объектов 60.Преимущества UML

Задания для оценки умений.
См. Приложение 4 «Задания для самостоятельной работы»

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

Основная литература:

1. Щербак, А. В. Поддержка и тестирование программных модулей : учебник для среднего профессионального образования / А. В. Щербак. — Москва : Издательство Юрайт, 2026. — 145 с. — (Профессиональное образование). — ISBN 978-5-534-19290-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/590253

2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для среднего профессионального образования / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2026. — 312 с. — (Профессиональное образование). — ISBN 978-5-534-13221-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/588246

Дополнительная литература:

1. Щербак, А. В. Тестирование программного обеспечения : учебник для вузов / А. В. Щербак. — Москва : Издательство Юрайт, 2026. — 145 с. — (Высшее образование). — ISBN 978-5-534-19291-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/590250

2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2026. — 312 с. — (Высшее

образование). — ISBN 978-5-9916-9043-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/584673

Интернет-ресурсы:

https://urait.ru Образовательная платформа Юрайт. Для вузов и ссузов

https://acmp.ru/ Школа программиста

https://proglib.io/p/awesome-algorithms/ Изучаем алгоритмы: полезные книги, веб-сайты, онлайн-курсы и видеоматериалы

https://gb.ru/blog/gde-popraktikovatsya-v-programmirovanii-30-resursov/ Задачи, упражнения и соревнования для начинающих и опытных кодеров

http://www.gost.ru Официальный сайт Федерального агентства по техническому регулированию и метрологии

http://www.vniis.ru/ Всероссийский научно-исследовательский институт сертификации (ВНИИС)

http://mon.gov.ru/ (Сайт Министерства образования и науки РФ) /

www.edu.ru (Российский образовательный портал).

www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).

Российская государственная публичная библиотека http://elibrary.rsl.ru/

http://www.rambler.ru – Русская поисковая система.

http://www.yandex.ru – Русская поисковая система.

http://biblioteka.net.ru – Библиотека компьютерных учебников.

Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –http://www.edu-all.ru/

8. ПЕРЕЧЕНЬ КОМПЛЕКТОВ ЛИЦЕНЗИОННОГО И СВОБОДНО РАСПРОСТРАНЯЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ИСПОЛЬЗУЕМОГО ПРИ ИЗУЧЕНИИ УЧЕБНОЙ ДИСЦИПЛИНЫ

При изучении учебной дисциплины (в том числе в интерактивной форме) предполагается применение современных информационных технологий. Комплект программного обеспечения для их использования включает в себя: свободно распространяемое программное обеспечение OpenOffice; веб-браузер (Google Chrome); электронные библиотечные системы IPR Smart и ЮРАЙТ; систему размещения в сети «Интернет» и проверки на наличие заимствований курсовых, научных и выпускных квалификационных работ «Антиплагиат.ВУЗ».

Для доступа к учебному плану и результатам освоения дисциплины, формирования Портфолио обучающегося используется Личный кабинет студента (он-лайн доступ через сеть Интернет https://portal.rosnou.ru). Для обеспечения доступа обучающихся во внеучебное время к электронным образовательным ресурсам учебной дисциплины ЮРАЙТ и IPR Smart, а также для студентов, обучающихся с применением дистанционных образовательных технологий, используется портал через сеть Интернет https://e-edu.rosnou.ru, https://id.rosnou.ru.

9. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-

ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

https://urait.ru Образовательная платформа Юрайт. Для вузов и ссузов

https://acmp.ru/ Школа программиста

https://proglib.io/p/awesome-algorithms/ Изучаем алгоритмы: полезные книги, веб-сайты, онлайн-курсы и видеоматериалы

https://gb.ru/blog/gde-popraktikovatsya-v-programmirovanii-30-resursov/ Задачи, упражнения и соревнования для начинающих и опытных кодеров

http://www.gost.ru Официальный сайт Федерального агентства по техническому регулированию и метрологии

http://www.vniis.ru/ Всероссийский научно-исследовательский институт сертификации (ВНИИС)

http://mon.gov.ru/ (Сайт Министерства образования и науки РФ) /

www.edu.ru (Российский образовательный портал).

www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).

Российская государственная публичная библиотека http://elibrary.rsl.ru/

http://www.rambler.ru – Русская поисковая система.

http://www.yandex.ru – Русская поисковая система.

http://biblioteka.net.ru – Библиотека компьютерных учебников.

Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –http://www.edu-all.ru/

10. ОБУЧЕНИЕ ИНВАЛИДОВ И ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

Изучение учебной дисциплины обучающимися инвалидами и лицами с ограниченными возможностями здоровья осуществляется в соответствии с Приказом Министерства просвещения РФ от 31 марта 2025 г. № 253 «Об утверждении Порядка обеспечения условий доступности для инвалидов объектов и предоставляемых услуг в сфере общего, среднего профессионального образования и соответствующего дополнительного профессионального образования, профессионального обучения, дополнительного образования детей и взрослых, организации отдыха и оздоровления детей, а также оказания им при этом необходимой помощи», Положением АНО ВО «РосНОУ» Об организации обучения студентов – инвалидов и лиц с ограниченными возможностями здоровья.

Лица с ограниченными возможностями здоровья и инвалиды обеспечиваются электронными образовательными ресурсами, адаптированными к состоянию их здоровья.

Предоставление специальных технических средств обучения коллективного и индивидуального пользования, подбор и разработка учебных материалов для обучающихся с ограниченными возможностями здоровья производится преподавателями с учетом индивидуальных психофизиологических особенностей обучающихся и специфики приема-передачи учебной информации на основании просьбы, выраженной в письменной форме.

С обучающимися по индивидуальному плану или индивидуальному графику проводятся индивидуальные занятия и консультации.

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Учебная аудитория для проведения лекционных занятий (поточная), оснащенная оборудованием: специализированная мебель; маркерная доска (переносная), проектор (портативный); экран (переносной); ноутбук для преподавателя (с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду вуза); колонки; микрофон, наглядные пособия (плакаты).

Учебная аудитория для проведения практических занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, воспитательной работы (**Лаборатория информационных технологий, программирования и баз данных**), оснащенная оборудованием: специализированная мебель; магнитно-маркерная доска; проектор; экран; ПК для преподавателя (с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду вуза), персональные компьютеры в сборке с пакетом специализированных программ; наглядные пособия (плакаты).

Помещение для самостоятельной работы обучающихся, оснащенное оборудованием: персональные компьютеры в сборке с пакетом специализированных программ, с выходом в интернет и обеспечением доступа в электронную информационно-образовательную среду.

Учебная аудитория для занятий обучающихся из числа инвалидов и лиц с ОВЗ. Специализированная мебель (стол для инвалидов-колясочников СИ-1), проектор; экран; мнемосхема тактильная; персональные компьютеры в сборке со специализированным пакетом программ, клавиатура с подсветкой с возможностью работы слабовидящим; комплект для маркировки Брайлем (клавиатура для компьютера); индукционная система переносная ИЦР-2.

Приложение 1

«Контрольная работа»

«Инструментальные средства разработки программного обеспечения»

1. Программное средство, предназначенное для поддержки разработки других программ, называется -…

а) аппаратным инструментом б) программным инструментом

в) программной средой г) инструментарий технологии программирования

2. Анализаторы обеспечивают…

а) конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

в) статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

г) выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

3. Преобразователи обеспечивают…

а) конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

в) статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

г) выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

4. Сколько классов инструментальных средств выделяют в инструментальной среде разработки и сопровождения программ?

а) 2 б) 4 в) 3 г) 5

5. Среда программирования предназначена для…

а) конструирования тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

в) поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

г) поддержки процессов программирования (кодирования), тестирования и отладки программ

6. Инструментальные среды программирования бывают

а) языково-ориентированные среды и среды общего назначения

б) объектно-ориентированные и языково-ориентированные среды

в) среды общего назначения и прикладные среды

г) среды общего назначения, прикладные среды, логические и математические среды

7. Для поддержки разработки программного продукта на каком-либо одном языке программирования используют…

а) среду программирования общего назначения

б) языково-ориентированную среду программирования

в) интерпретирующую среду программирования

г) прикладную среду программирования

8. Синтаксически-управляемая инструментальная среда программирования базируется на знании

а) семантики языка программирования

б) синтаксиса языка программирования

в) синтаксиса и семантики языка программирования

г) основных управляющих структур языка программирования

9. На рисунке представлена классификация

а) инструментальной системы технологии программирования

б) инструментальной среды разработки и сопровождения программ

в) рабочего места компьютерной технологии

г) языков программирования

10. Инструментальная система технологии программирования – это…

а) программное средство, предназначенное для поддержки разработки других программ

б) устройство компьютера, специально предназначенное для поддержки разработки программного средства

в) интегрированная совокупность программных и аппаратных инструментов, поддерживающая все процессы разработки и сопровождения больших программных продуктов

г) логически связанная совокупность программных и аппаратных инструментов, поддерживающих разработку ПП

11. Устройство компьютера, специально предназначенное для поддержки разработки программного средства, называется -…

а) аппаратным инструментом б) программным инструментом

в) программной средой г) инструментарий технологии программирования

12. Редакторы обеспечивают…

а) конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

в) статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

г) выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

13. Инструменты, поддерживающие процесс выполнения программ, обеспечивают…

а) конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматический привод документов к другой форме представления или перевод документа одного вида к документу другого вида

в) возможность выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

г) статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

14. Инструментальная система технологии программирования предназначена для…

а) поддержки всех процессов разработки и сопровождения в течение всего жизненного цикла ПС и ориентирована на коллективную разработку больших программных систем с длительным жизненным циклом

б) автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

в) поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

г) поддержки процессов программирования (кодирования), тестирования и отладки

программ

15. Рабочее место компьютерной технологии предназначено для…

а) конструирования тех или иных программных продуктов и документов на различных этапах жизненного цикла

б) автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

в) поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

г) поддержки процессов программирования (кодирования), тестирования и отладки программ

16. Инструментальные среды программирования содержат

а) редактор, анализатор и компилятор

б) редактор, интерпретатор и компилятор

в) интерпретатор, компилятор, преобразователь

г) редактор и интерпретатор

17. Для поддержки разработки программного продукта на разных языках программирования (например, текстовый редактор, редактор связей или интерпретатор языка целевого компьютера) используют…

а) среду программирования общего назначения

б) языково-ориентированную среду программирования

в) интерпретирующую среду программирования

г) прикладную среду программирования

18. На рисунке представлена классификация

а) инструментальной системы технологии программирования

б) инструментальной среды программирования

в) рабочего места компьютерной технологии

г) языков программирования

19. При использовании компьютерных технологий для разработки ПП жизненный цикл ПП представлен следующей цепочкой:

а) прототипирование – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение

б) прототипирование – разработка спецификаций – автоматизированный контроль спецификаций – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение

в) разработка спецификаций – автоматизированный контроль спецификаций – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение

г) прототипирование – разработка спецификаций – кодогенерация – аттестация, применение, сопровождение

20. Основными чертами инструментальной системы технологии программирования являются…

а) массовость, дискретность, результативность, определенность, понятность

б) комплексность, ориентированность на коллективную разработку, технологическая определенность, интегрированность

в) актуальность, непротиворечивость, полнота

г) комплексность, актуальность, интегрированность, массовость, понятность

20. Современные крупные проекты информационных систем характеризуются следующими особенностями:

а) сложность описания, требующая тщательного моделирования и анализа данных и процессов

б) наличие совокупности тесно взаимодействующих компонентов

в) наличие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений

г) невозможность интеграции существующих и вновь разрабатываемых приложений;

21. Под CASE-средства понимаются программные средства, поддерживающие…

а) процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО

б) процессы тиражирования программного продукта

в) процессы создания и эксплуатации программного продукта

г) процессы компилирования и интерпретации программных продуктов

22. Репозитарий Case – средства – это…

а) специализированная база данных проекта, предназначенная для отображения состояния проектируемой системы в каждый момент времени

б) компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

в) компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

г) компонент, позволяющий получать информацию о проектах в виде отчетов

д) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

е) набор системных утилит по обслуживанию репозитария

23. Графический редактор Case – средства – это…

а) компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

б) компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

в) компонент, позволяющий получать информацию о проектах в виде отчетов

г) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

24. Верификатор Case – средства – это…

а) компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

б) компонент, позволяющий получать информацию о проектах в виде отчетов

в) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

г) набор системных утилит по обслуживанию репозитария

25. Документатор проекта Case – средства – это…

а) компонент, позволяющий получать информацию о проектах в виде отчетов

б) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

в) набор системных утилит по обслуживанию репозитария

г) компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

26. Сервис Case – средства – это…

а) компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

б) компонент, позволяющий получать информацию о проектах в виде отчетов

в) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

г) набор системных утилит по обслуживанию репозитария

27. Администратор проекта Case – средства – это…

а) компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

б) компонент, позволяющий получать информацию о проектах в виде отчетов

в) компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

г) набор системных утилит по обслуживанию репозитария

28. Какие методологии проектирования используют Case – средства?

а) структурного и модульного проектирования

б) структурного и объектно-ориентированного проектирования

в) объектно-ориентированного и нисходящего проектирования

г) нисходящего и восходящего проектирования

29. Структурное проектирование системы основано на…

а) объектно-ориентированной декомпозиции

б) алгоритмической декомпозиции

в) модульной декомпозиции

г) функциональной декомпозиции

30. Объектно-ориентированное проектирование системы основано на…

а) объектно-ориентированной декомпозиции

б) алгоритмической декомпозиции

в) модульной декомпозиции

г) функциональной декомпозиции

31. Case – средства представляют собой…

а) набор инструментальных средств для проектирования программного продукта

б) набор программных средств для сопровождения программного продукта

в) набор программных и инструментальных средств, поддерживающие все процессы жизненного цикла программного продукта

г) набор аппаратных средств, поддерживающих все процессы жизненного цикла программного продукта

32. Компания-разработчик приобрела новое Case – средство. Сразу ли компания получит ожидаемый результат от применения новой технологии?

а) да б) нет

33. Сколько классов Case – средств выделяют?

а) 5 б) 3 в) 7 г) 2

34. Case – средства анализа и проектирования, предназначенные для

а) моделирования данных и генерации схем баз данных

б) построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы

в) обеспечения комплексной поддержки требований к создаваемой системе

г) поддержки всего жизненного цикла программного продукта

35. Case – средства управления требованиями предназначены для

а) моделирования данных и генерации схем баз данных

б) построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы

в) обеспечения комплексной поддержки требований к создаваемой системе

г) поддержки всего жизненного цикла программного продукта

36. Case – средства проектирования баз данных предназначены для

а) моделирования данных и генерации схем баз данных

б) построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы

в) обеспечения комплексной поддержки требований к создаваемой системе

г) поддержки всего жизненного цикла программного продукта

37. На каких стандартах базируется технология освоения и внедрения Case – средств?

а) ГОСТ 2107-99

б) IEEE (IEEE Std 1348-1995 и IEEE Std 1209-1992)

в) AES

г) ISO

38. Из каких этапов состоит процесс освоения и внедрения Case – средств?

а) определение потребностей в CASE-средствах, оценка и выбор CASE-средств, практическое внедрение CASE-средств

б) определение потребностей в CASE-средствах, оценка и выбор CASE-средств, выполнение пилотного проекта, практическое внедрение CASE-средств

в) определение потребностей в CASE-средствах, проектирования CASE-средств, практическое применение CASE-средств

г) проектирование CASE-средств, оценка и внедрение CASE-средств, практическое применение CASE-средств

39. Критериями для выбора CASE-средств могут являться

а) открытая архитектура, поддержка полного жизненного цикла ИС с обеспечением эволюционности ее развития, обеспечение целостности проекта, независимость от программно-аппаратной платформы и СУБД

б) модифицируемость, простота, эффективность, учет человеческого фактора, многоплатформенность

в) закрытая архитектура, поддержка полного жизненного цикла ИС с обеспечением эволюционности ее развития, простота, эффективность

г) максимальная зависимость от программных и аппаратных средств системы и характеристик самой системы, жесткая привязка к конкретным информационным процессам, прочность внутренней связи отдельных компонентов системы

40. Комплексность компьютерной поддержки разработки ПП с использованием инструментальной системы технологии программирования означает

а) что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана

б) что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов

в) что все инструменты объединены единым пользовательским интерфейсом

г) что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

41. Ориентированность инструментальной системы технологии программирования на коллективную разработку означает

а) что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана

б) что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов

в) что все инструменты объединены единым пользовательским интерфейсом

г) что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

42. Технологическая определенность инструментальной системы технологии программирования означает

а) что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана

б) что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов

в) что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования

г) что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

43. Интегрированность инструментальной системы технологии программирования по данным означает

а) что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана

б) что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов

в) что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования

г) что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

44. Интегрированность инструментальной системы технологии программирования по пользовательскому интерфейсу означает

а) что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана

б) что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов

в) что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования

г) что все инструменты объединены единым пользовательским интерфейсом.

Вопросы к контрольной работе

1. Необходимые инструментальные средства разработки программ

2. Часто используемые инструментальные средства разработки программ

3. Специализированные инструментальные средства разработки программ

4. Интегрированные среды разработки

5. Средства разработки программного обеспечения

6. Определение «разработка программ»

7. Три этапа разработки программ

8. Средства проектирования приложений

9. Средства реализации программного кода

10.Средства тестирования программ

11. Классы инструментальных средств разработки программ

12. Четыре категории инструментальных программ, применяемые при проектировании экспертных систем.

13. Оболочки экспертных систем

14. Языки программирования высокого уровня

15. Среда программирования, поддерживающая несколько парадигм

16.Дополнительные модули

17. Языки инженерии знаний

18. Язык логического программирования Prolog

19. Язык функционального программирования Lisp 20.Средства автоматизации разработки экспертных систем

21.Общее программное обеспечение 22.Специальное

программное обеспечение

23. Инструментальная система технологии программирования

24. Четыре класса компьютерной поддержки инструментальных систем технологий программирования

25. Комплексность

26. Ориентированность на коллективную разработку

27.Технологическая определенность 28.Интегрированность

29.Компоненты инструментальных систем технологий программирования

30.База данных разработки

31.Инструментарий

32.Интерфейсы

33.Общая архитектура инструментальных систем технологий программирования

34. Инструментальная система поддержки проекта

35.Языково-зависимая инструментальная система

36.Пользовательский интерфейс

37.Схема организации взаимодействия компьютера и пользователя

38.Процедурно-ориентированный подход к разработке интерфейсов

39.Объектно-ориентированный подход к разработке интерфейсов

40.Типы интерфейсов

41.Интерфейс-меню

42.Интерфейсы со свободной навигацией 43.Критерии оценки интерфейса пользователем

44. Факторы появления Case-технологий

45. Что такое Case-технология?

46. Компоненты интегрированного Case-средства

47.Классификация по категориям Case-средств

48.Классификация по типам Case-средств

49.Вспомогательные типы Case-средств

50.Объектно-ориентированный подход к разработке ИС

51.Качества организации для успешного внедрения Case-средств

52.Факторы появления Case-технологий

53.Структурный системный анализ

54.Диаграммы «сущность-связь»

55.Язык графического описания UML

56.Диаграмма компонентов

57.Диаграмма композитной структуры

58.Диаграмма развертывания

59. Диаграмма объектов

60.Преимущества UML

Приложение 2

«Практические работы»

Разработать техническое задание. Примеры:

1. Спроектировать пользовательский интерфейс, определить цели и исходных требований к программе, провести анализ пользователей и создание сценариев поведения пользователей. – Определить предметную область и сферу применения программного продукта. – Определить целевую аудиторию. – Построить описательную модель пользователя (профиль). Выделить группы пользователей. – Сформировать множество сценариев поведения пользователей на основании составленной модели. – Выделить функциональные блоки приложения и схему навигации между ними (структуру диалога).

2. Разработать программный код для задач: – Конвертер валют. Пользователь вводит сумму, выбирает две валюты и получает эквивалент во второй валюте. – Средняя температура. Пользователь вводит данные о температуре за некоторый период времени. Программа находит среднее значение. – Пользователь указывает цвет и радиус круга. Программа показывает прямоугольник, в котором круг данного размера и цвета движется горизонтально, меняя направление при касании границы.

3. С помощью конструктора графического интерфейса пользователя IDE NetBeans создать графический интерфейс пользователя для приложения с именем MyApplication , по плану: – создать GUI Container (контейнер GUI), добавить компоненты, выронить их положение, изменить размер, осуществить привязку компонентов, продемонстрировать управление поведением при автоматическом изменении размера и редактировании свойства компонентов.

4. Составьте физическую диаграмму в соответствии с описанием деятельности компании дистрибьютора «МЕД» . Компания дистрибьютор "МЕД" закупает медицинские препараты отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Планирование закупок компания осуществляет на основании статистики продаж, которую предоставляют сеть аптек и дистрибьюторы. Компания осуществляет доставку медикаментов как собственным транспортом, так и с помощью услуг сторонних организаций.

Ответить на вопросы:

1. Необходимые инструментальные средства разработки программ
2. Часто используемые инструментальные средства разработки программ
3. Специализированные инструментальные средства разработки программ
4. Интегрированные среды разработки
5. Средства разработки программного обеспечения
6. Определение «разработка программ»
7. Три этапа разработки программ
8. Средства проектирования приложений
9. Средства реализации программного кода
10. Средства тестирования программ
11. Классы инструментальных средств разработки программ
12. Четыре категории инструментальных программ, применяемые при проектировании экспертных систем
13. Оболочки экспертных систем
14. Языки программирования высокого уровня
15. Среда программирования, поддерживающая несколько парадигм
16. Дополнительные модули

17. Языки инженерии знаний
18. Язык логического программирования Prolog
19. Язык функционального программирования Lisp
20. Средства автоматизации разработки экспертных систем
21. Общее программное обеспечение
22. Специальное программное обеспечение
23. Инструментальная система технологии программирования
24. Четыре класса компьютерной поддержки инструментальных систем технологий программирования
25. Комплексность
26. Ориентированность на коллективную разработку
27. Технологическая определенность
28. Интегрированность
29. Компоненты инструментальных систем технологий программирования
30. База данных разработки
31. Инструментарий
32. Интерфейсы
33. Общая архитектура инструментальных систем технологий программирования
34. Инструментальная система поддержки проекта
35. Языково-зависимая инструментальная система
36. Пользовательский интерфейс
37. Схема организации взаимодействия компьютера и пользователя
38. Процедурно-ориентированный подход к разработке интерфейсов
39. Объектно-ориентированный подход к разработке интерфейсов
40. Типы интерфейсов
41. Интерфейс-меню
42. Интерфейсы со свободной навигацией
43. Критерии оценки интерфейса пользователем
44. Факторы появления Case-технологий
45. Что такое Case-технология?
46. Компоненты интегрированного Case-средства
47. Классификация по категориям Case-средств
48. Классификация по типам Case-средств
49. Вспомогательные типы Case-средств
50. Структурный подход к разработке ИС
51. Объектно-ориентированный подход к разработке ИС
52. Факторы, усложняющие определение возможного эффекта от использования Case-средств
53. Качества организации для успешного внедрения Case-средств
54. Проблемы использования Case-средств
55. Факторы появления Case-технологий
56. Структурный системный анализ
57. Диаграммы «сущность-связь»
58. Диаграммы классов
59. Язык графического описания UML
60. Диаграмма компонентов
61. Диаграмма композитной структуры
62. Диаграмма развѐртывания
63. Диаграмма объектов
64. Диаграмма пакетов
65. Диаграмма деятельности
66. Преимущества UML
67. IDEF
68. Диаграммы переходов состояний

69. Методология функционального моделирования ИС
70. Состав функциональной модели
71. Иерархия диаграмм
72. Типы связей между функциями
73. Характеристика современных Case-средств
74. Методология ARIS
75. Программный продукт ARIS Express
76. Основные элементы, используемые в нотации ARIS
77. Архитектура ARIS
78. Имитационное моделирование
79. Применение имитационного моделирования
80. Виды имитационного моделирования
81. Дискретнособытийное моделирование
82. Системная динамика
83. Области применения имитационного моделирования
84. Основные этапы компьютерного моделирования
85. Построение концептуальной модели системы
86. Постановка задачи машинного моделирования
87. Анализ задачи моделирования
88. Определение требований к исходной информации
89. Выдвижение гипотез и принятие предположений
90. Определение параметров и переменных
91. Установление основного содержания модели
92. Обоснование критериев оценки эффективности системы
93. Определение процедур аппроксимации
94. Описание концептуальной модели
95. Проверка достоверности модели
96. Составление технической документации

Критерии оценки эссе (рефератов, докладов, сообщений)

Оценка «отлично»: выполнены все требования к написанию и защите реферата: обозначена проблема и обоснована ее актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объем, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.

Оценка «хорошо»: основные требования к реферату и его защите выполнены, но при этом допущены недочеты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объем реферата; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы.

Оценка «удовлетворительно»: имеются существенные отступления от требований к реферированию. В частности: тема освещена лишь частично; допущены фактические ошибки в содержании реферата или при ответе на дополнительные вопросы.

Оценка «неудовлетворительно»: тема освоена лишь частично; допущены грубые ошибки в содержании реферата или при ответе на дополнительные вопросы; во время защиты отсутствует вывод. Тема реферата не раскрыта, обнаруживается существенное непонимание проблемы.

Перечень тем рефератов:

1. Инструментальные средства поддержки процесса управления требованиями.

2. Инструментальные средства поддержки процесса разработки проекта.
3. Инструментальные средства реализации кода.
4. Инструментальные средства тестирования.
5. Инструментальные средства поддержки процесса управления конфигурациями.
6. Семейство стандартов моделирования IDEF. Структурные карты. Диаграммы переходов состояний.
7. Инструментальные средства поддержки методологий функционального моделирования.
8. Определение, характеристики и основные элементы процессного подхода. 9. Процессный подход к моделированию деятельности.
10.Общие сведения, функциональное назначение методологии ARIS. 11.Архитектура ARIS - пять типов представлений, отражающих основные аспекты деятельности организации.
12. Базовая модель ARIS - этапы описания бизнес-процессов.
13. Виды моделей методологии ARIS - основные принципы построения, структура, свойства, составляющие элементы.
14. Использование методологии ARIS в различных областях деятельности.

Темы эссе (рефератов, докладов, сообщений) по дисциплине «Инструментальные средства разработки программного обеспечения»

1. Анализ предметной области
2. Разработка UML диаграмм
3. Работа с инструментальными средствами, поддерживающими методологию объектно-ориентированного моделирования.
4. Работа с CASE – средствами проектирования программного обеспечения
5. Работа с CASE – средствами кодирования программного обеспечения
6. Работа с CASE – средствами тестирования программного обеспечения
7. Инструментальные средства поддержки процесса управления требованиями.
8. Инструментальные средства поддержки процесса разработки проекта.
9. Инструментальные средства реализации кода.
10.Инструментальные средства тестирования.
11. Инструментальные средства поддержки процесса управления конфигурациями.
12. Структурные карты.

Вопросы для коллоквиумов, собеседования по дисциплине «Инструментальные средства разработки программного обеспечения»

1. Инструментальные среды программирования.
2. Понятие компьютерной технологии разработки.
3. Визуальное проектирование.
4. Delphi. Основные характеристики продукта.
5. Структура среды программирования.
6. Стандартные компоненты.
7. Понятие и структура проекта.
8. Базовые конструкции среды Delphi.
9. Средства тестирования и отладки.
10. Основные свойства ООП в Delphi.
11. Методы в Delphi.
12. Основы работы с базами данных в Delphi.

Разработка программного обеспечения Инструментальные среды программирования.

1. Визуальное проектирование.
2. Delphi. Основные характеристики продукта.

3. Структура среды программирования.
4. Стандартные компоненты.
5. Принципы работы с объектами.
6. Использование библиотек, процедур и функций.
7. Состав интегрированной среды разработки VBA.
8. Типы данных в VBA.
9. Константы VBA.
10. Операции языка VBA.
11. Управляющие структуры для организации циклов в VBA.
12. Основные принципы работы в среде VBA.
13. Отличительные особенности сред программирования VBA и Delphi.
14. Инструментальные средства разработки ПО.
15. Классификация инструментальных средств участвующих в разработке приложения.

Перечень дискуссионных тем для круглого стола (дискуссии, полемики, диспута, дебатов) по дисциплине Инструментальные средства разработки программного обеспечения

1. Средства тестирования и отладки.
2. Основные свойства ООП в Delphi.
3. Методы в Delphi.
4. Использование библиотек, процедур и функций.
5. Состав интегрированной среды разработки VBA.
6. Объектно-ориентированный подход к моделированию деятельности.
7. Инструментальные средства, поддерживающие методологию объектноориентированного моделирования.
8. Метод имитационного моделирования.
9. Этапы моделирования, технологическая схема.
10.Построение концептуальных моделей сложных систем.
11.Базовые концепции структуризации формализации имитационных систем. 12.Языковые средства и системы моделирования.

Анализ предметной области. Разработка и оформление технического задания

Цель: Изучить, описать и проанализировать предметную область, в которой будет создаваться информационная база. Ознакомление с процедурой разработки технического задания на создание программного продукта с применением ГОСТ 34.602-89.

Форма отчета:

1. Провести анализ предметной области в соответствии с выданным заданием.
2. Составить техническое задание в соответствии с ГОСТ 34.602-89.
3. Защитить практическую работу.

Теоретические сведения Анализ предметной области

Выделяются следующие шаги работы над проектом (системой):

1. Описание предметной области, под которой понимается та часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи. Следовательно, разработчикам необходимо выделить основные объекты (компоненты), участвующие в функционировании системы, определить их наиболее существенные характеристики, взаимосвязи в рамках решаемой задачи, а также определить основные информационные потоки в системе. При этом отдельные компоненты

выбираются таким образом, чтобы при последующей разработке их было удобно представить в форме классов и объектов. В этом случае немаловажное значение приобретает и сам язык представления информации о концептуальной схеме предметной области.

Сложность предметной области определяет количество объектов и связей между ними, поэтому описание должно включать в себя базовые термины и определения, сопровождаться различными примерами, в нем могут приводиться различного рода классификации, поясняющие различные свойства описываемых объектов. Если в системе используются математические модели, то они также должны быть описаны с учетом специфики применения.

2. Обзор существующих систем-аналогов – неотъемлемая часть описания предметной области, которая позволяет разработчику определить основные концепции, необходимые для реализации в системе. Описание должно приводиться с указанием отличительных особенностей разработанных систем, с перечислением их достоинств и недостатков, в отчете обязательно приводятся экранные формы этих систем.

3. Результатом последнего этапа является диаграмма объектов предметной области и краткое описание их свойств и функций. При построении данной диаграммы нужно помнить о том, что в данном случае объект – это «конкретная материализация абстракции», а не экземпляр класса. Диаграмма объектов представляет статическую составляющую взаимодействующих между собой объектов, она должна включить в себя только те объекты предметной области, которые потом преобразуются в диаграмму классов. Связи между объектами показывают отношения между ними, при необходимости в диаграмме можно привести и атрибуты (свойства) объектов.

Диаграммы объектов не позволяют полностью описать объектную структуру системы, поэтому при их использовании нужно сосредоточиться на изображении интересующих вас наборов конкретных объектов.

Для сбора, хранения, поиска и выдачи информации о предметной области и ее объектов настоящее время в информационных системах широко используются базы данных.

Анализ предметной области начинается с выделения сущностей и определения их свойств или атрибутов.

Видимые сущности представляют собой объекты предметной области, которые может распознать человек.

Поддерживаемые сущности или абстрактные сущности разрабатываются для физической поддержки общей логической модели.

Пример анализа предметной области

Проектирование БД «Туристическая фирма»

Туристическая фирма является юридическим лицом, имеет собственный баланс, расчетный счет в банке, печать и бланк со своим наименованием, и другие реквизиты. Деятельность осуществляется самостоятельно на принципах полного хозяйственного расчета. Предприятие имеет лицензию на право ведения деятельности по оказанию услуг в сфере туризма и сертификат соответствия.

Фирма предоставляет туристическое обслуживание по России, странам СНГ, ближнему и дальнему зарубежью. Управление фирмой осуществляется ее владельцем.

Целевой аудиторией считаются люди среднего и пожилого возраста, дети и молодежь, возможен подбор туров по индивидуальным параметрам, составление ознакомительных экскурсий по странам с учетом пожеланий клиента.

Туристические путевки, возможно, заказать практически в любые страны и направления вплоть до самых отдаленных и экзотических, предоставляемые услуги ориентированы в основном на индивидуальные заявки.

Документы, регламентирующие деятельность туристического агентства:

– внешние документы: законодательные и нормативные акты, касающиеся выполняемой работы;

– внутренние документы: устав туристического агентства, приказы и распоряжения учредителя туристического агентства;

– положение о туристическом агентстве, должностные инструкции сотрудников тур агентства, правила внутреннего трудового распорядка.

В офисе работают директор и менеджер.

Директор турагентства исполняет следующие обязанности:

– руководит в соответствии с действующим законодательством хозяйственной и финансово- экономической деятельностью турагентства, неся всю полноту ответственности за последствия принимаемых решений, сохранность и эффективное использование имущества турагентства, а также финансово-хозяйственные результаты его деятельности;

– обеспечивает достижение запланированных финансово-экономических показателей турагентства;

– руководит деятельностью структурных подразделений турагентства;

– обеспечивает выполнение турагентством всех обязательства перед сторонними организациями, заказчиками услуг, покупателями турпродуктов, а также хозяйственных, трудовых договоров и бизнес-планов;

– организует деятельность по оказанию туристических услуг и реализации турпродуктов на основе научных форм управления и организации труда, изучения конъюнктуры рынка туристических услуг и турпродуктов в целях повышения уровня их качества;

– принимает меры по обеспечению турагентства квалифицированными кадрами, рациональному использованию и развитию их профессиональных знаний и опыта, созданию безопасных и благоприятных для жизни и здоровья условий труда;

– обеспечивает правильное применение принципа материальной заинтересованности и ответственности каждого работника за порученное ему дело и результаты работы всего коллектива, выплату заработной платы в установленные сроки;

– обеспечивает соблюдение законности в деятельности турагентства и осуществлении его хозяйственно-экономических связей, использование правовых средств для финансового управления и функционирования в рыночных условиях, укрепления договорной и финансовой дисциплины, регулирования социально-трудовых отношений;

– осуществляет контроль за рациональным использованием материальных, финансовых ресурсов, дает оценку результатам деятельности турагентства и качества оказываемых услуг;

– утверждает правила внутреннего трудового распорядка, график отпусков, должностные инструкции, производственные инструкции и иные организационно-правовые документы;

– принимает решения о приеме, перемещении и увольнении подчиненных работников;

– применяет меры поощрения к отличившимся работникам; налагает взыскания на нар

– решает в установленном порядке вопросы направления работников в служебные командировки.

Менеджер занимается следующими видами деятельности:

– изучает справочники по туризму, каталоги, иные источники туристской информации с целью формирования собственных информационных баз по туроператорам;

– изучает требования клиентов к туристским продуктам, осуществляет анализ маркетинговых исследований спроса на туристские услуги;

– устанавливает контакты с туроператорами с целью изучения программ туров, определения туров,

Исходя из анализа предметной области, нам потребуются следующие таблицы: Страны, Покупатели, Путешествия.

Далее выделяем структуру таблиц.

Структура таблицы БД «СТРАНЫ МИРА».

имя поля	тип	Описание
Код страны	числовой	Код страны
Страна	текстовый	Название страны
Столица	текстовый	Название столицы
Часть света	текстовый	Название части света
Население	числовой	Население (в тыс. человек)
Площадь	числовой	Площадь (в тыс. кв. км)

Структура таблицы БД «Путешествия».

имя поля	тип	описание
Код путевки	числовой	Код путевки
Код страны	числовой	Код страны
Продолжительность	числовой	Кол-во дней
Дата заезда	дата	Дата выезда
Стоимость	денежный	Стоимость путевки

Структура таблицы БД «Покупатели».

имя поля	тип	описание
Код покупателя	числовой	Код покупателя
Код путевки	числовой	Код путевки
Фамилия ИО	текстовый	Фамилия ИО
Оплата	логический	Да – оплатил, нет – бронь
Цель	текстовый	Цель поездки
Количество	числовой	Кол-во путевок

Модели сущность-связь основаны на выделении в предметной области, для которой осуществляется проектирование базы данных, различных типов объектов, информацию о которых требуется хранить в базе данных.

Набор однотипных объектов предметной области образует сущность. Между сущностями могут быть установлены информационные связи (зависимости), которые также могут быть учтены при проектировании схемы базы данных. Совокупность сущностей и связи между ними составляют информационную модель данных предметной области (Entity-Relationship диаграмму).

В настоящее время существует несколько приемов выделения сущностей и связей – нотации Чена, Мартина, Баркера, IDEF1X и т.д.

Основные определения

Сущность (Entity) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность должна обладать

уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Атрибут (Attribute) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе.

Связь (Relationship) – поименованная ассоциация между сущностями, значимая для рассматриваемой предметной области.

Нотация Чена

В нотации Чена различают зависимые и независимые сущности.

Сущность называется ***независимой***, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями.

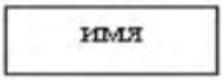
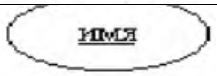
Сущность называется ***зависимой*** от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности.

Сущности и связи могут иметь атрибуты. Для каждой сущности находится атрибут (или набор атрибутов), значение которого однозначно определяет экземпляр сущности. Этот атрибут является ключом сущности.

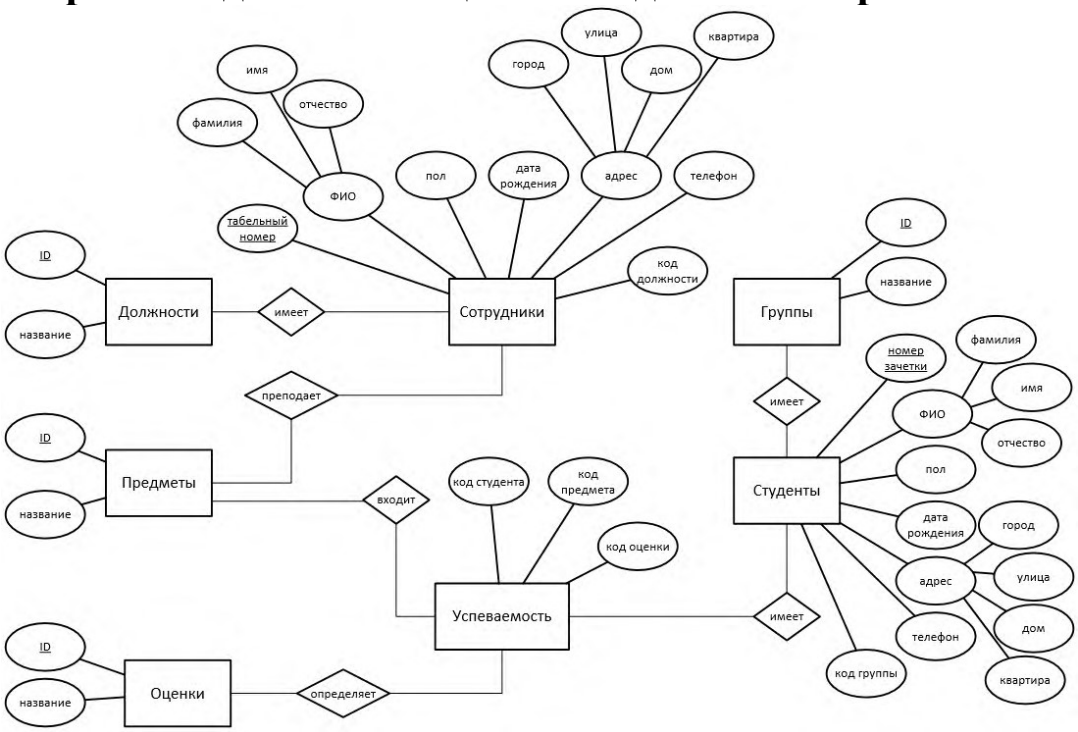
Связь также может иметь ключевой атрибут. В ряде случаев для удобства организации связей в состав атрибутов сущности вводится искусственный ключ (обычно число). Ключевой атрибут (набор атрибутов) на диаграмме отмечается двумя линиями снизу, внешние ключи отмечаются одной линией. В таблице представлены основные элементы, используемые для формирования ER- диаграммы в нотации Чена.

Таблица 1 – Элементы ER-диаграммы в нотации Чена

Элемент диаграммы	Описание
	Сущность
	Связь
	Атрибут
	Первичный ключ
	Внешний ключ

Связь имеет кардинальное число, определяющее, какое количество экземпляров одной сущности имеет информационную связь с экземплярами другой сущности.

Пример ER-модели в нотации Чена для «Университет».



Понятие информационной модели. Уровни информационной модели

Методология IDEF1X – язык для семантического моделирования данных, основанный на

концепции «сущность-связь».

Различают два уровня информационной модели: **логический** и **физический**.

Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.

Различают 3 подуровня логического уровня модели данных, отличающиеся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity-Relationship Diagram (ERD);
- модель данных, основанная на ключах (Key Based Model (KB);
- полная атрибутивная модель (Fully Attributed Model (FA).

Физическая модель отражает физические свойства проектируемой базы данных (типы данных, размер полей, индексы). Параметры физической информационной модели зависят от выбранной системы управления базами данных (СУБД).

Основные элементы информационной модели логического уровня

Сущности и атрибуты

Сущность – это множество **реальных** или **абстрактных объектов** (людей, предметов, документов и т.п.), **обладающих общими атрибутами или характеристиками**. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. *Именование сущности* осуществляется с помощью *существительного в единственном числе*. При этом имя сущности должно отражать **тип** или **класс** объекта, а не его **конкретный экземпляр** (например, **Студент**, а не **Петров**).

Студент

Любая сущность характеризуется набором атрибутов (**свойств**).

Атрибут сущности – характеристика сущности, то есть свойство реального объекта. Например, на рисуке атрибутами сущности **«Студент»** являются **«ID студента»**, **«Фамилия»**, **«Имя»**,

«Отчество», **«Дата поступления»** и **«Номер билета»**.

В свою очередь, *атрибуты сущности* делятся на 2 вида: *собственные* и *наследуемые. Собственные* атрибуты являются уникальными в рамках модели. *Наследуемые* атрибуты передаются от сущности-родителя при установке связи с другими сущностями.

Первичный ключ (Primary Key, PK). Каждая сущность должна обладать *атрибутом* или *комбинацией атрибутов*, чьи значения *однозначно определяют* каждый *экземпляр сущности*. Эти атрибуты образуют *первичный ключ* сущности.

Внешний ключ (Foreign Key, FK). Если между двумя сущностями *имеется специфическое отношение* связи или *категоризации*, то *атрибуты*, входящие в *первичный ключ родительской* или *общей сущности*, *наследуются* в качестве *атрибутов сущностью-*

потомком или *категориальной сущностью* соответственно. Эти атрибуты и называются внешними ключами. Наследуемый атрибут может использоваться в качестве части или целого первичного ключа, альтернативного ключа или не ключевого атрибута.

Отношения в IDEF1X-модели

При построении информационной модели различают следующие типы отношений между сущностями: *идентифицирующее*, *не идентифицирующее*, *не специфическое (многие-ко-многим) и отношения категоризации*.

Мощность отношения служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Нормализация данных

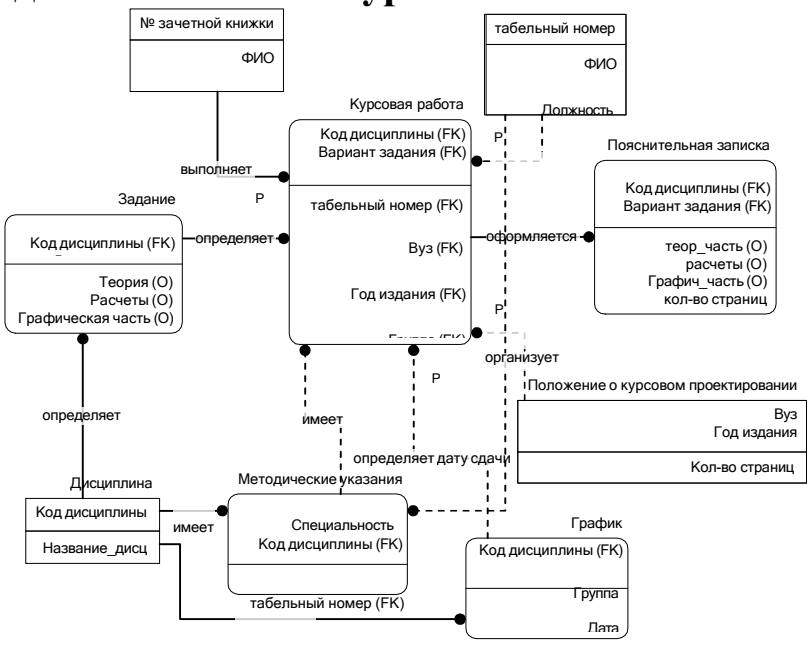
Нормализация – это процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Процесс нормализации сводится к последовательному приведению структур данных к нормальным формам – формализованным требованиям к организации данных.

Первая нормальная форма (1НФ). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра.

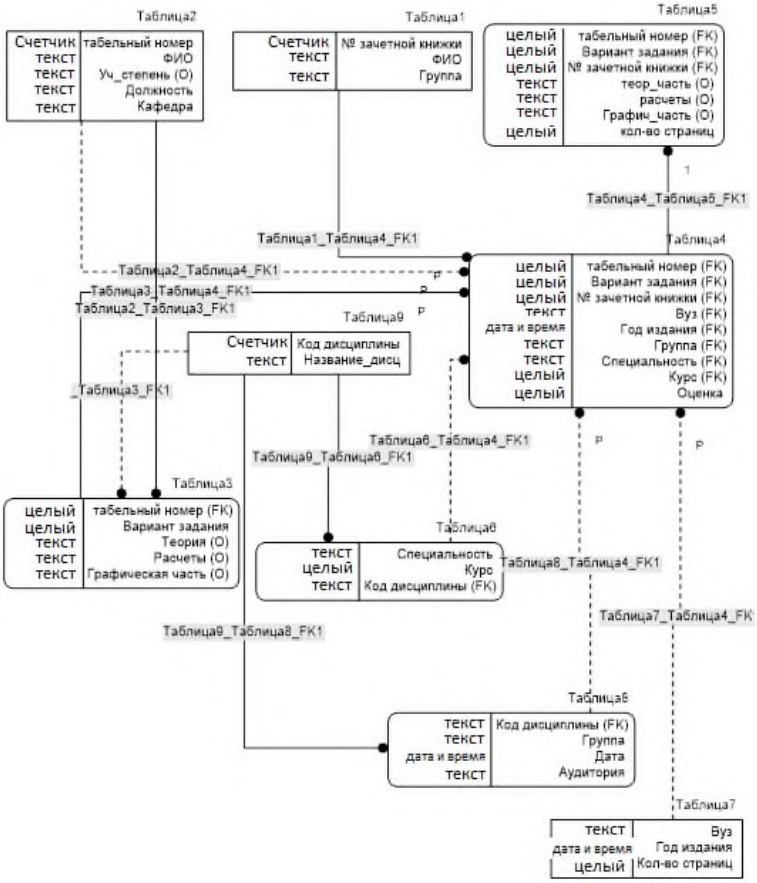
Вторая нормальная форма (2НФ). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый не ключевой атрибут полностью зависит от первичного ключа (не может быть зависимости от части ключа).

Третья нормальная форма (3 НФ). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой не ключевой атрибут не зависит от другого не ключевого атрибута (не должно быть зависимости между не ключевыми атрибутами).

Пример модели логического уровня



Пример модели физического уровня



Разработка и оформление технического задания

Для обеспечения потребностей по разработке программного обеспечения (ПО) ИС необходимо взаимосвязанное совершенствование технических решений, технологий проектирования и программирования, инструментальных средств, а также обучения специалистов.

Стандарты для процессов, инструментальных средств и данных, которые отражают лучшую практику и защищают от неблагоприятных последствий, играют определенную роль в обеспечении указанных потребностей, в частности, поддерживая доверие потребителя к продуктам, услугам и технологиям разработки ПО.

Использование стандартов при разработке ПО ИС позволит обеспечить:

- повышение надежности;
- повышение эффективности применения и снижение затрат в сфере сопровождения программных средств (ПС);
- увеличение коэффициента повторного использования ПС общего назначения;
- повышение объективности оценок качества ПС;
- создание условий для конкуренции разработчиков на внутреннем рынке ПС;
- обеспечение конкурентоспособности отечественных ПС на мировом рынке.

Стандарты комплекса ГОСТ 34 на создание и развитие автоматизированных систем (АС) – обобщенные, но воспринимаемые как весьма жесткие по структуре жизненного цикла (ЖЦ) и проектной документации. Наиболее популярными можно считать ГОСТ 34.601-90 **(**Автоматизированные системы. Стадии создания) и ГОСТ 34.602-89 (Техническое задание на создание АС). Введение единой, достаточно качественно определенной терминологии, наличие достаточно разумной классификации работ, документов, видов обеспечения и др. способствует более полной и качественной стыковке разных систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС.

В последние годы в стране идет интенсивная работа по гармонизации государственных стандартов Российской Федерации с международными стандартами ISO в области создания нормативной базы управления жизненным циклом программных средств и информационных

систем. В основе стандартизации - ГОСТ Р ИСО/МЭК 12207-99 "Информационная технология. Процессы жизненного цикла программных средств", который является аутентичным переводом международного стандарта ISO/IEC 12207: 1995-08-01.

Техническое задание (ТЗ) на АС - утвержденный в установленном порядке документ, определяющий цели, требования и основные исходные данные необходимые для разработки АС и содержащий предварительную оценку экономической эффективности.

ТЗ содержит основные технические требования, предъявляемые к изделию и исходные данные для разработки; в ТЗ указываются назначение объекта, область его применения, стадии разработки документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов, полученных в ходе предпроектного обследования.

Как инструмент коммуникации в связке общения заказчик-исполнитель, техническое задание позволяет:

А) Обеим сторонам:

- представить готовый продукт;
- выполнить проверку готового продукта по пунктам (приёмочное тестирование – проведение испытаний);
- уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением испытаний).

Б) Заказчику:

- осознать, что именно ему нужно, опираясь на существующие на данный момент технические возможности и свои ресурсы;
- требовать от исполнителя соответствия продукта всем условиям, оговорённым в ТЗ. В) Исполнителю:
- правильно понять суть задачи, показать заказчику «технический облик» будущего программного изделия или АС;
- спланировать выполнение проекта и работать по намеченному плану;
- отказаться от выполнения работ, не указанных в ТЗ.

Структура технического задания

Титульный лист

Содержание

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Полное наименование системы и ее условное обозначение

1.2 Номер договора

1.3 Наименования организации-заказчика и организации-исполнителя

1.4 Перечень документов, на основании которых создается система

1.5 Плановые сроки начала и окончания работы по созданию системы

1.6 Порядок оформления и предъявления заказчику результатов работ по созданию системы

1.7 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

1.8 Определения, обозначения и сокращения

2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

2.1 Назначение системы

2.2 Цели создания системы

3 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

3.1 Краткие сведения об объекты автоматизации или ссылки на документы, содержащие такую информацию

3.2 Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды

4 ТРЕБОВАНИЯ К СИСТЕМЕ

4.1 Требования к системе в целом

4.2 Требования к функциям, выполняемым системой
4.3 Требования к видам обеспечения
4.3.1 Требования к математическому обеспечению системы
4.3.2 Требования информационному обеспечению системы
4.3.3 Требования к лингвистическому обеспечению системы
4.3.4 Требования к методическому обеспечению системы
4.3.5 Требования организационному обеспечению системы
4.3.6 Требования к правовому обеспечению системы
4.3.7 Требования к программному обеспечению системы
4.3.8 Требования к техническому обеспечению
4.3.9 Требования к эргономическому обеспечению
5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ
6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ
7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ
8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ
9 ИСТОЧНИКИ РАЗРАБОТКИ

Шаблон ТЗ задания приведен в приложении Б.

Задание практической работы

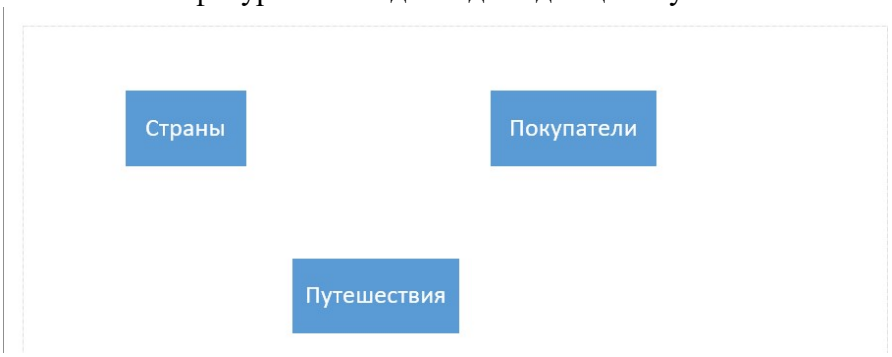
Изучить правила проведения анализа предметной области и составления ТЗ для «Туристической фирмы»

Методика выполнения

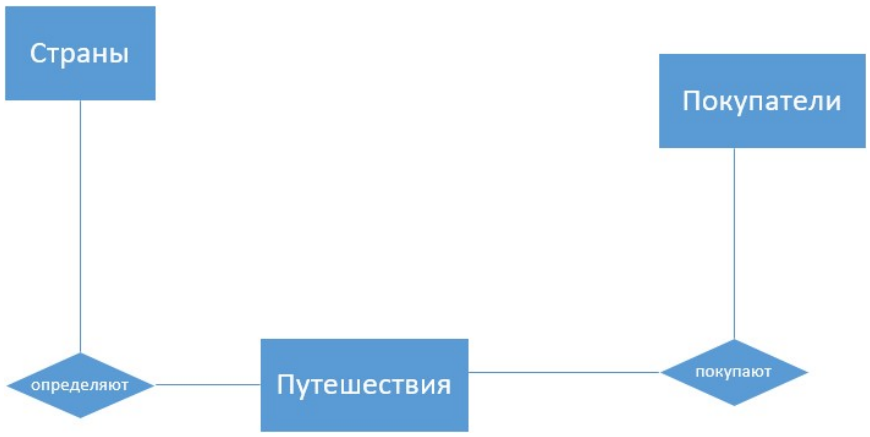
В качестве примера возьмем ***БД Туристическая фирма.***

Исходя из анализа предметной области, нам потребуются следующие таблицы: Страны, Покупатели, Путешествия.

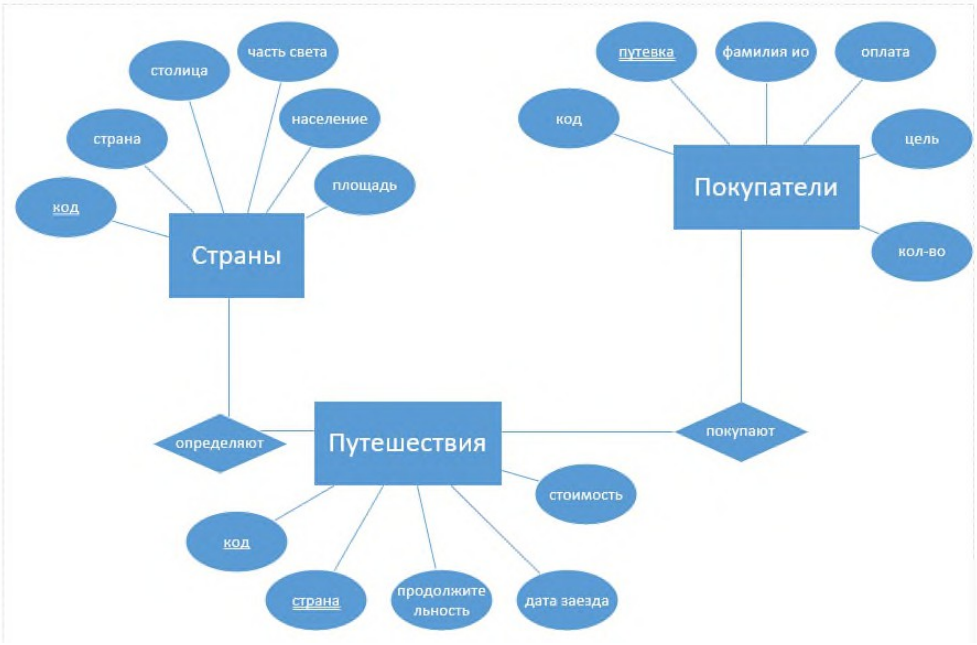
1. Запустите MS Visio.
2. В категориях шаблонов выберите *Базы данных* и в ней элемент *Нотация Чена*. Нажмите кнопку *Создать*.
3. В соответствии с анализом предметной области необходимо добавить на рабочий лист три фигуры Сущность.
4. Для добавления текста фигуре необходимо дважды щелкнуть по ней левой кнопкой мыши.



5. Далее необходимо добавить связи между сущностями. Для этого необходимо использовать фигуру ромб.



6. На данном этапе получена общая диаграмма Чена. Для ее детализации необходимо добавить атрибуты ко всем сущностям. Атрибуты изображаются в виде овала

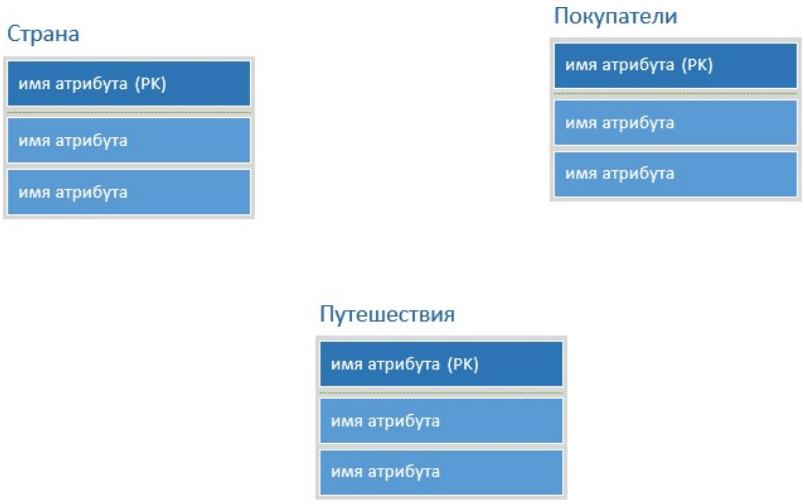


7. Запустите MS Visio.

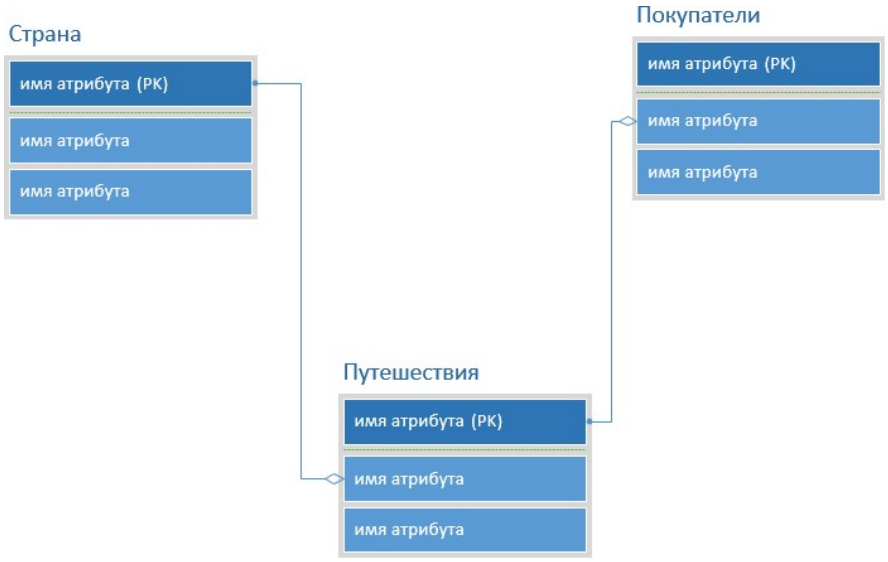
8. В категориях шаблонов выберите *Базы данных* и в ней элемент *Нотация IDEF1X*. Нажмите кнопку *Создать*.

9. В соответствии с анализом предметной области необходимо добавить на рабочий лист три фигуры Сущность.

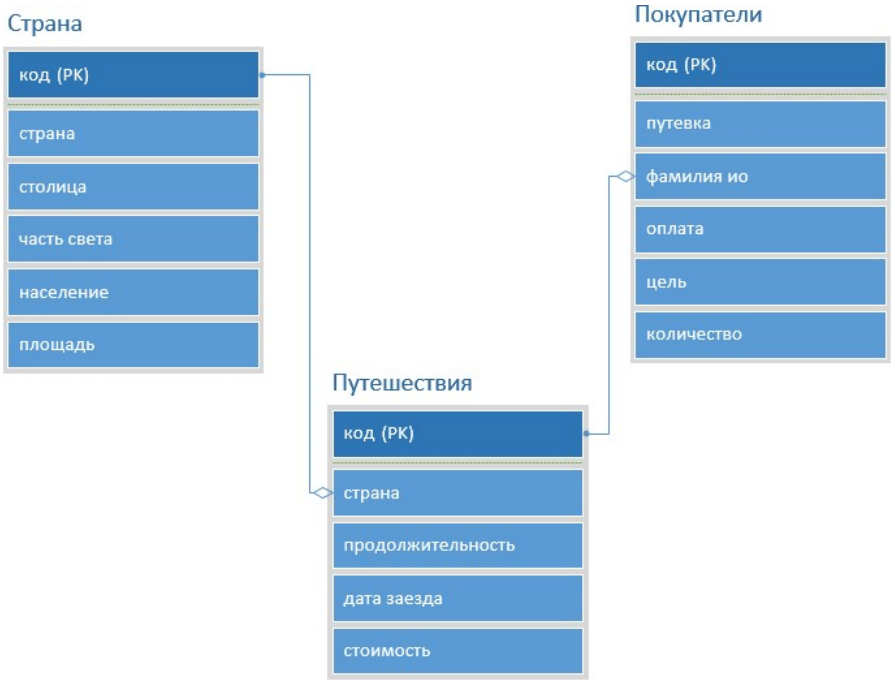
10. Для добавления текста фигуре необходимо дважды щелкнуть по ней левой кнопкой мыши.



11. Далее необходимо добавить связи между сущностями.

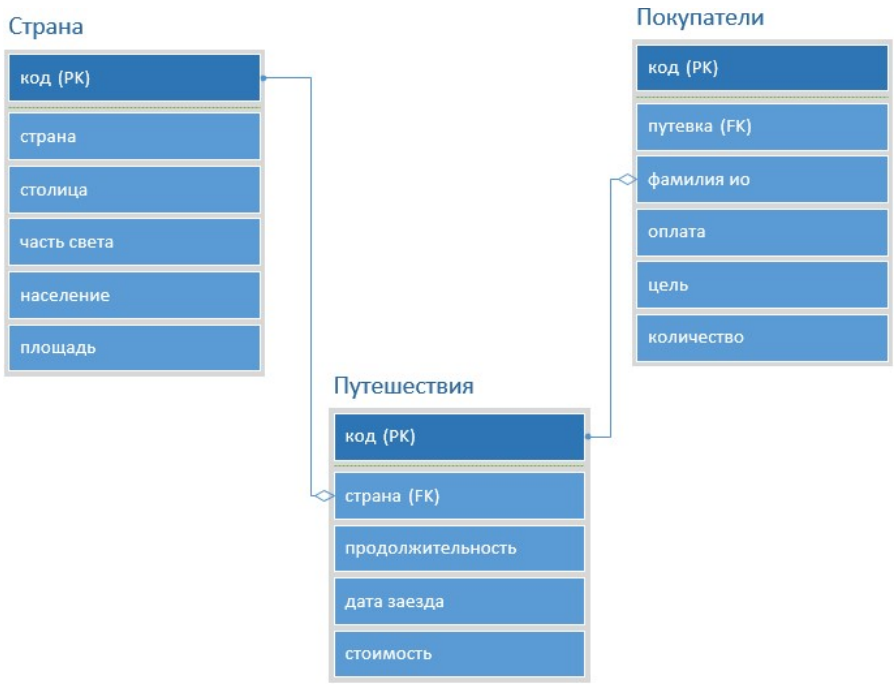


12. Добавить имена атрибутам (и сами атрибуты при необходимости).



13. Указать атрибуты, которые являются внешними ключами

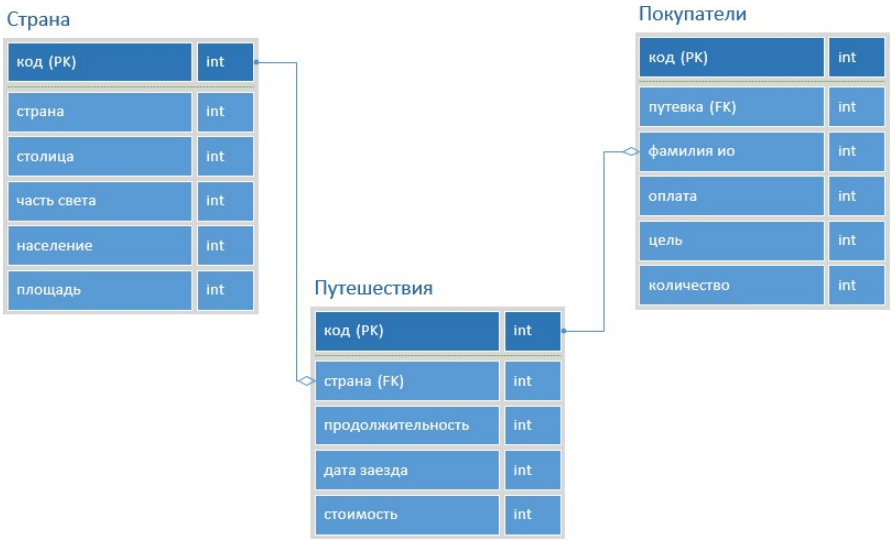
Для этого нужно выделить атрибут. Щелкнуть по нему правой кнопкой мыши и в контекстном меню выбрать пункт «Задать внешний ключ».



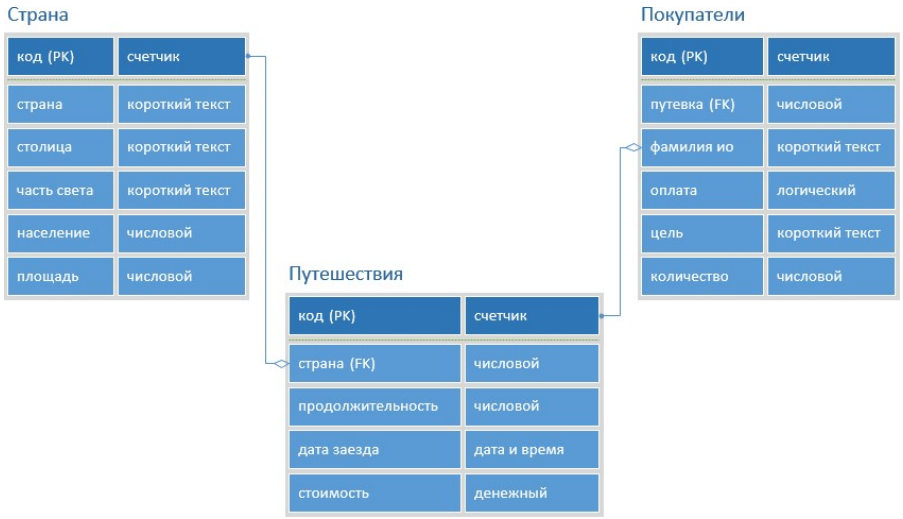
14. Создание физической модели

Для этого необходимо дублировать созданную модель на новый лист. Для этого щелкнуть правой кнопкой мыши по текущему листу и в контекстном меню выбрать пункт Дублировать.

15. Затем на новом листе по очереди выбрать каждую сущность и в ее контекстном меню выбрать пункт Показать типы атрибутов.



16. Затем необходимо указать соответствующие типы данных, в зависимости от СУБД в которой будет выполняться реализация системы. Например, для MS Access модель примет вид:



Чтобы изменить размер ячейки, в которой указывается тип данных нужно выделить сущность и используя желтые точки изменить размер.

Задание самостоятельной работы

В соответствии с индивидуальным вариантом, провести анализ предметной области, построить ER-модель в нотации Чена и IDEF1X и на основании ГОСТ 34.602-89, разработать документ Техническое задание.

Перечень индивидуальных вариантов приведен в приложении А.

Контрольные вопросы

1. Что такое предметная область?
2. Этапы анализа предметной области.
3. Что такое ER-модель?
4. В каких нотациях можно построить ER-модель?
5. Элементы диаграммы Чена.
6. Элементы диаграммы в нотации IDEF1X.
7. Для чего предназначена диаграмма «сущность-связь»?
8. Что представляет собой нормализация?
9. В чем разница между логическим уровнем модели данных и физическим?

10. Назначение, содержание и степень адаптивности стандарта ГОСТ 34.601-90?
11. Стадии и этапы создания АС в соответствии с ГОСТ 34.601-90?
12. Назначение, содержание стандарта ГОСТ 34.602-89?
13. Назначение программного документа Техническое задание на создание АС. Порядок разработки, согласования и утверждения документа?
14. Состав и содержание документа ТЗ?

Изучение работы в системе контроля версий

1. Отработать навыки использования хранилища на локальной машине.

1.1. Настроить Git, указав имя и e-mail разработчика для подписи commit-ов.

Указание. Диалог настроек вызывается пунктом TorotiseGit → Settings контекстного меню любого каталога, нужная вкладка называется «Git». Задавать следует глобальные настройки (для всех хранилищ), установив переключатель «Global».

1.2. Создать хранилище для учебного проекта.

1.3. Совершить несколько commit-ов.

1.3.1. Скопировать sdt.h в каталог хранилища и создать файл main.cpp со включением sdt.h и пустой функцией main().

1.3.2. Добавить в программу ввод двух целых чисел с приглашением.

Указание № 1. После выполнения каждого подпункта необходимо убеждаться, что программа работает, и совершать commit изменений.

Указание № 2. Следите за тем, какие файлы отмечены в списке для commita изменений в них,

— кроме main.cpp и, иногда, sdt.h, больше никаких других не нужно.

1.4. Предотвратить автоматическое добавление в хранилище файлов, не нуждающихся в контроле версий, — *.o и *.exe.

Правило об игнорировании следует помещать в файл .gitignore в корневом каталоге хранилища («.gitignore in repository root»). Этот файл также попадает под контроль версий, поэтому после создания правил требуется совершить commit изменений в файле .gitignore.

1.5. Добавить в программу вывод суммы введенных чисел и совершить commit.

1.6. Просмотреть историю (журнал) хранилища.

1.7. Просмотреть разность (diff) между пунктами истории 1.3.2 и 1.5.

2. Освоить передачу истории хранилища по сети.

2.1. Организовать общее хранилище на удаленном сервере.

2.1.1. Зарегистрироваться на GitHub.

2.1.2. Создать пустое удаленное хранилище с любым наименованием.

Указание. Вопреки инструкции на GitHub, добавлять в хранилище файл README.md не нужно, удаленное хранилище должно быть пустым.

2.1.3. Разрешить пользователям-преподавателям совершать commit-ы.

Требуется на странице хранилища выбрать «Settings» (справа), далее «Collaborators», где ввести имена пользователей, которым будет предоставлен полный доступ к хранилищу (эти имена можно узнать у лаборантов).

2.1.4. Настроить локальное хранилище для синхронизации с удаленным.

Необходимо в контекстном меню каталога локального хранилища выбрать TortoiseGit → Settings, где перейти к пункту Remote . Достаточно ввести условное имя удаленного хранилища «origin» и его адрес, который отображается на webстранице удаленного хранилища в разделе «Quick setup» (вариант HTTP).

От загрузки сведений о ветвлениях в удаленном хранилище отказаться.

2.2. Передать локальное хранилище на удаленный сервер (push).

Замечание. Здесь и далее при взаимодействии с удаленным сервером потребуется вводить имя пользователя и пароль, с которыми выполнялась регистрация на GitHub.

2.3. Перейти к странице хранилища на GitHub (обновить её) и ознакомиться с возможностями просмотра содержимого через web-интерфейс.

2.4. Загрузить копию удаленного хранилища на локальную машину (clone).

Замечание. Целью является имитация совместной работы с удаленным хранилищем. Для этого на одной машине организуются 2 локальных хранилища: созданное в пункте 1.1 (RepoA) и загруженное с удаленного сервера (RepoB).

Указание. Диалог «Git clone» следует вызывать из контекстного меню каталога вне локального хранилища. В качестве URL потребуется указать адрес удаленного хранилища, а в качестве Directory — имя каталога для нового локального хранилища.

2.5. Сымитировать параллельную работу над проектом.

2.5.1. В локальном хранилище RepoB добавить в программу печать разности введенных чисел, сделать commit и передать изменения на сервер.

2.5.2. В локальном хранилище RepoA добавить над функцией main() комментарий о том, что программа является учебной, сделать commit, но не отправлять изменений на сервер.

2.6. На странице хранилища на GitHub перейти в раздел Commits и ознакомиться с возможностью просмотра истории изменений через web-интерфейс.

2.7. В локальном хранилище RepoA выполнить загрузку с сервера новейшихветвлений и изменений (fetch) и просмотреть журнал хранилища.

Указание. По умолчанию показывается только текущая активная ветвь (по умолчанию — master). Просмотреть все commit-ы во всех ветвях, в том числе в загруженной из удаленного хранилища ветви origin/master, нужно включить флажок «All branches» слева снизу окна журнала.

2.8. Совместить изменения в локальном хранилище с загруженными.

2.8.1. Использовать действие pull для загрузки изменений с удаленного сервера и автоматического совмещения их с имеющимися локально.

Просмотреть журнал изменений (или обновить кнопкой Refresh).

Примечание. Фактически, при обновлении производится слияние ветвей masteru origin/master

— то есть, двух версий истории, существовавших удаленно и локально. При этом история стала нелинейной и появился лишний commit слияния. Иногда такое усложнение имеет смысл, но в данном случае было бы желательно сохранить историю линейной и просто перенести локальные наработки вслед за новейшими. Добьемся желаемого.

2.8.2. Отменить неудобный результат действия pull.

Указание. В журнале изменений в контекстном меню commit-а, где был добавлен комментарий (то есть, последнего перед слиянием), выбрать «Reset master to this…» и указать тип отмены

«Hard».

Указание. Журнал изменений не всегда обновляется автоматически, используйте кнопку Refresh, если изменения не появились сразу.

2.8.3. Выполнить перенос (rebase) локальных изменений на основу новейшего загруженного состояния проекта.

Указание. В журнале изменений в контекстном меню пункта, на котором находится конец ветви origin/master (прямоугольник в TortoiseGit), следует выбрать пункт «Rebase "master" onto this…» и далее нажать кнопку «Start rebase».

2.9. Передать итоговое состояние локального хранилища RepoA на удаленный сервер, используя команду push.

2.10. Действуя аналогично п. п. 2.7 и 2.8.3, синхронизировать с удаленным локальное хранилище RepoB (в нем не хватает commit-а с комментарием).

Замечание. На данном этапе во всех трех хранилищах (локальных RepoA и RepoBи удаленном на GitHub) должна быть одинаковая линейная история из пяти — шести commit-ов.

3. Изучить действия, связанные с ветвлениями и разрешением конфликтов.

Замечание. Все действия выполняются в одном локальном хранилище, например, в RepoA.

3.1. Добавить в программу печать произведения чисел и совершите commit. На данном этапе программа может быть такой:

```
1  #include "sdt.h"
2
3  // This program is just an example one under VCS.
4  int main()
5  {
6      int a, b;
7      cout << "Enter A and B: ";
8      cin  >> a >> b;
9      cout << "A + B = " << a + b << '\n'
10          << "A - B = " << a - b << '\n'
11          << "A * B = " << a * b << '\n';
12 }
```

3.2. Создать новую ветвь (branch) под названием division. из пункта истории, в котором был добавлен комментарий над main().

3.3. В новой ветви повторить пункт 3.1, заменив умножение делением.

Указание. Переключиться на ветвь можно, выбрав в контекстном меню commit-а, которым эта ветвь оканчивается, пункт «Switch/checkout to this». При создании ветви можно сразу установить флажок «Switch to new branch». Переключаться можно только при чистом (clean) хранилище, то есть, без изменений в рабочей копии.

3.4. Переключиться обратно на ветвь master.

3.5. Выполнить слияние ветви division в ветвь master так, чтобы в последней оказался код для печати и произведения, и частного.

3.5.1. В журнале изменений в контекстном меню пункта-окончания ветви division выбрать пункт «Merge into "master"…» и начать слияние, не меняя настроек.

Действие завершится ошибкой из-за конфликта (conflict): в файле main.cpp строка 10 изменена в обоих commit-ах одинаково, а строка 11 — по-разному, и СКВ не может автоматически выбрать
«правильный» вариант. Требуется вручную указать, какие строки должны войти в итоговую версию файла.

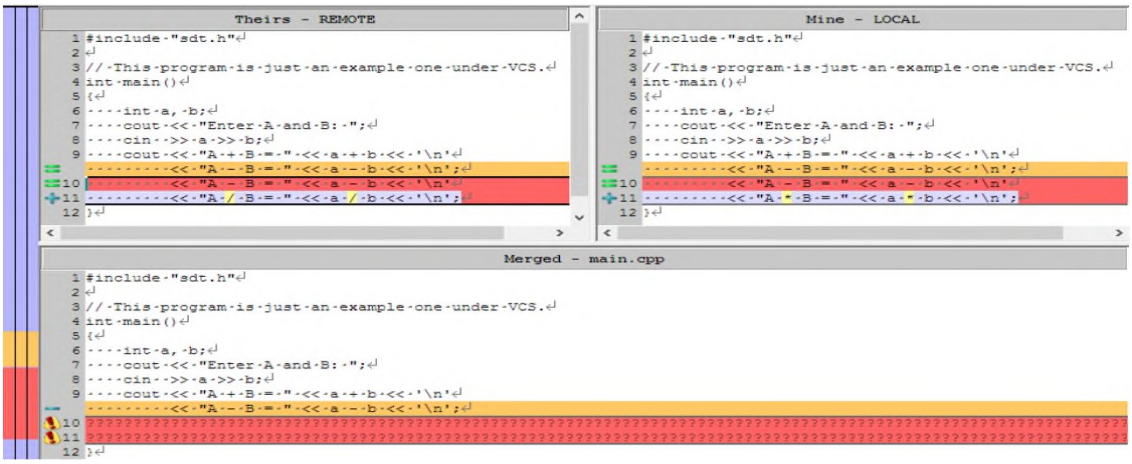
3.5.2. Приступить к разрешению конфликта.

Указание. Следует нажать кнопку Resolve (снизу), а затем выбрав пункт Edit conflicts из контекстного меню main.cpp.

Примечание. Редактор конфликтов похож на программу для просмотра разностей между файлами: слева показывается файл в ветви, откуда делается слияние (division), справа — ветви, куда делается слияние (master), снизу — результат слияния. Знаками равенства в соседних верхних полях отмечаются не только строки, оставшиеся неизменными, но и

строки с одинаковыми изменениями: здесь, в строке 10 убрана точка с запятой в конце в обеих ветвях.

В нижнем поле каждый восклицательный знак обозначает отдельный конфликт. Примерный вид окна редактирования конфликтов представлен на рисунке ниже



3.5.3. Разрешить конфликты:

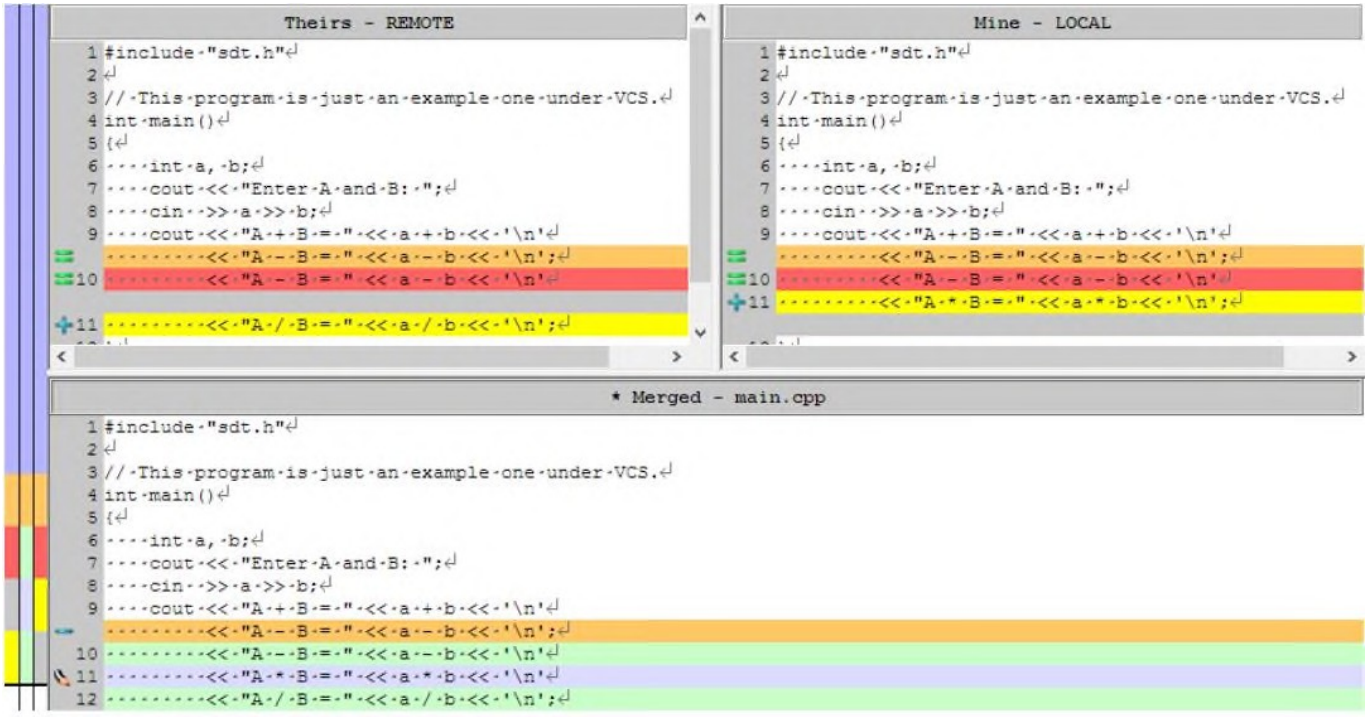
— в качестве строки 10 предпочесть строку 10 любой ветви;

— на место строки 11 вставить 2 строки: строку с печатью произведения и строку с печатью частного;

— лишнюю точку с запятой на строке 11 поля-результата удалить.

Указание. Переносить строки из той или иной версии в итоговую можно, выбирая конфликтные блоки в левом или правом верхнем поле (простым щелчком левой кнопкой мыши) и пользуясь контекстным меню. Например, «Use this text block» переносит выбранный блок в поле- результат; пункт «Use text block from 'mine' before 'theirs'» переносит в результат 2 строки:

сначала из правого блока, потом из левого (одну под другой). На рисунке ниже показан возможный верный результат



3.5.4. Завершить процедуру разрешения конфликтов.

Указание. Следует нажать на кнопку «Mark as resolved», чтобы отметить файл как избавленный от конфликтов, и закрыть программу для их разрешения.

3.5.5. Завершить слияние ветви division в ветвь master, написав осмысленный комментарий к слиянию и совершив commit.

3.5.6. Убедиться, что программа компилируется и верно работает. Если это не так, исправить все ошибки и добиться правильной работы. Совершить commit.

Замечание-указание. Ситуация, когда после слияния программа все-таки оказывается не вполне корректной, случается на практике довольно часто. В этом случае commit, созданный при слиянии, оказывается логически неправильным, он не имеет ценности без последующего исправления. В Git имеется возможность изменить (amend) уже совершенный commit, пока он не передан на сервер. Это делается при следующем commit-исправлении: следует установить флажок

«Amend Last Commit» в диалоге commit — нового commit не появится, а вместо этого изменения будут приписаны предыдущему пункту истории. Можно воспользоваться данной возможностью при выполнении пункта

3.5.7. Передать все изменения всех ветвей в удаленное хранилище.

Указание. По умолчанию передаются только изменения текущей ветви, для передачи изменений всех ветвей следует отметить флажок «Push all branches» диалога push.

Замечание. В данном задании отрабатывается навык слияния ветвей, существующих только в локальном хранилище и вступивших в конфликт с ведома единственного автора. Постоянно возникают и ситуации, когда одну и ту же ветвь, но в локальном и удаленном хранилище независимо изменяют разные авторы. В этом случае действия push и rebase приведут к конфликтам. Их разрешение выполняется совершенно аналогично.

Задание самостоятельной работы

В соответствии с индивидуальным вариантом, определить и описать особенности реализации системы (подходы, технологии и т. п.). На основе описанных особенностей системы обосновать выбор вида архитектуры, наиболее подходящей для реализации данной системы. Произвести визуальное проектирование архитектуры системы (рекомендуется использовать программу Microsoft Visio). Добавить текстовое описание к архитектуре, поясняющее ее структуру и связи.

Перечень индивидуальных вариантов приведен в приложении А.

В соответствии с индивидуальным вариантом разработать программу на любом известном языке программирования. Добавить ее в систему контроля версий и отследить изменения.

Перечень индивидуальных вариантов приведен в приложении В.

Контрольные вопросы

1. Что подразумевается под созданием архитектуры приложения?
2. Каково основное назначение архитектуры приложения?
3. Перечислите основные принципы разработки архитектуры программного обеспечения.
4. Перечислите основные типовые архитектурные стили.
5. Перечислите основные архетипы приложений.
6. Перечислите основные элементы, которые можно использовать для построения архитектуры программного обеспечения.
7. Что такое системы контроля версий (СКВ) и для решения каких задач они

8. предназначаются?
9. Объясните следующие понятия СКВ и их отношения: хранилище, commit, история,
10. рабочая копия.
11. Что представляют собой и чем отличаются централизованные и децентрализованные
12. СКВ? Приведите примеры СКВ каждого вида.
13. Опишите действия с СКВ при единоличной работе с хранилищем.
14. Опишите порядок работы с общим хранилищем в централизованной СКВ.
15. Что такое и зачем может быть нужна разность (diff)?
16. Что такое и зачем может быть нужно слияние (merge)?
17. Что такое конфликты (conflict) и каков процесс их разрешения (resolve)?
18. Поясните процесс синхронизации с общим хранилищем («обновления»)
19. в децентрализованной СКВ.
20. Что такое и зачем могут быть нужны ветви (branches)?
21. Объясните смысл действия rebase в СКВ Git.

Как и зачем можно игнорировать некоторые файлы при commit?

Построение функциональных диаграмм IDEF0 и диаграмм потоков данных DFD

Цель: получение навыков создания и редактирования функциональных моделей в нотации IDEF0. Изучение основных характеристик и основ работы с DFD-моделями в графическом редакторе.

Форма отчета:

Функциональная диаграмма IDEF0 и диаграмма потоков данных DFD с описанием процесса построения диаграмм.

Краткие теоретические сведения Функциональная диаграмма IDEF0

1. Основные сведения по методологии IDEF0

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Цель моделирования. Модель не может быть построена без четко сформулированной цели.

Пример цели: «Описать функциональность предприятия с целью написания спецификаций ИС».

Точка зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Цель и точка зрения документируются.

Основные элементы IDEF0-модели

В основе методологии IDEF0 лежат 4 основных понятия:

– функциональный блок;
– интерфейсная дуга (стрелка);
– декомпозиция;
– глоссарий.

1. Функциональный блок

Функциональные блоки обозначают поименованные процессы, функции или задачи,

а) б)

которые происходят в течение определенного времени и имеют распознаваемые результаты. Графически функциональные блоки изображаются в виде прямоугольников. Все блоки должны быть названы и определены. Имя функционального блока должно быть выражено сочетанием отглагольного существительного, обозначающего процесс, или глаголом:

Определение функционального блока заносится в глоссарий или словарь работ (Activity Dictionary).

Все функциональные блоки модели нумеруются. Номер состоит из префикса и числа. Может использоваться префикс любой длины, но обычно используется префикс А. Контекстная (корневая) работа (функциональный блок) имеет номер А0.

2. Интерфейсная дуга (стрелка – Arrow)

Взаимодействие функциональных блоков с внешним миром и между собой описывается в виде интерфейсных дуг (стрелок). Стрелки представляют собой некую информацию и обозначаются существительными (например, «Заготовка», «Изделие») или именуемыми сочетаниями (например,
«Готовое изделие»). Все стрелки должны быть определены.

В IDEF0 различают типы стрелок.

2 Создание контекстной диаграммы

1. На панели инструментов выберите пиктограмму функции ( ) и мышью укажите месторасположение на рабочем пространстве.

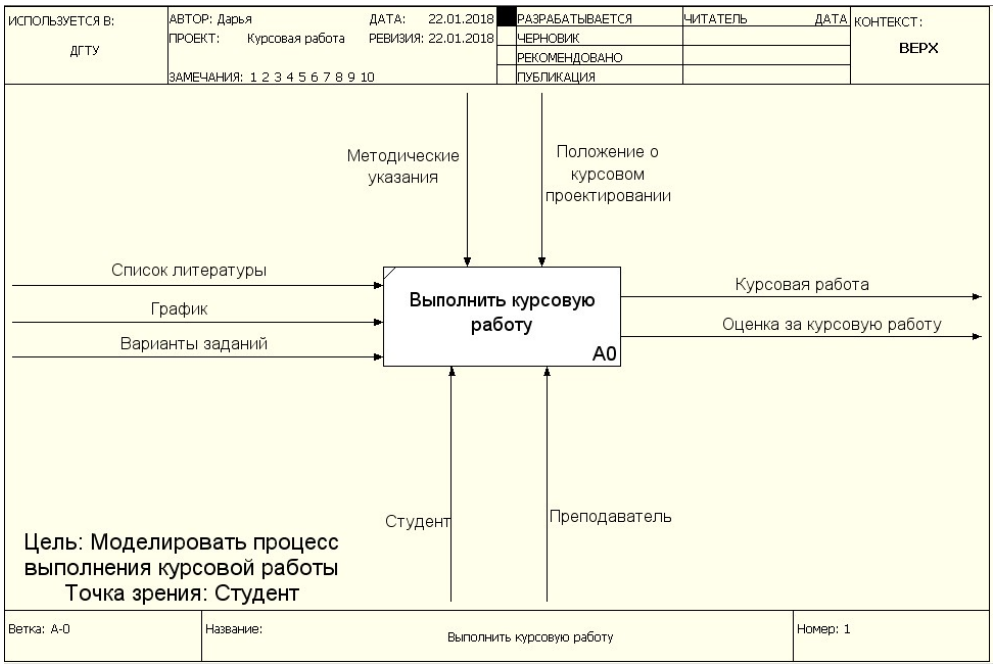
1. Дайте данному функциональному блоку имя «Выполнить курсовую работу». Для этого дважды щелкните внутри блока.

2. Используя пиктограмму панели инструментов , создайте стрелки на контекстной диаграмме согласно Таблица 1.

Таблица 2 – Контекстная диаграмма

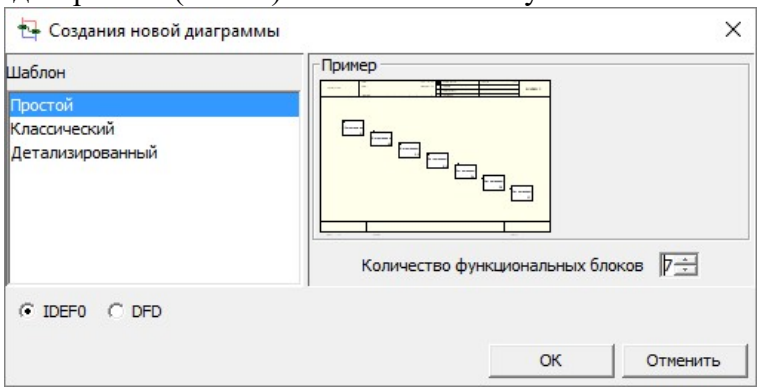
Имя стрелки (Arrow Name)	**Определение стрелки (Arrow Definition)**	**Тип стрелки (Arrow Type)**
График	График консультаций и сроки сдачи	Input
Список литературы	Источники информации для выполнения курсовой работы	Input
Варианты заданий	Список заданий на курсовую работу, подлежащий распределению между студентами	Input
Методические указания	Документ, содержащий указания по выполнению курсовой работы, описывающий содержание ее частей и основные требования	Control
Положение о курсовом проектировании	Документ, отражающий организационные требования по выполнению и сдаче курсовой работы	Control
Курсовая работа	Документ, являющийся основанием для получения оценки	Output
Оценка за курсовую работу	Результат выполнения курсовой работы	Output
Студент	Тот, кто выполняет курсовую работу	Mechanism

3. В результате должна получиться контекстная диаграмма.

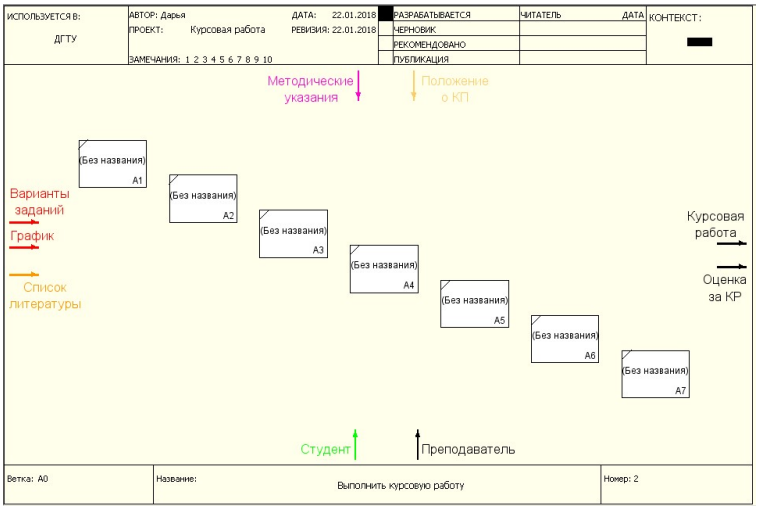


3 Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне «Создание новой диаграммы» установите количество функциональных блоков 7, укажите тип диаграммы (IDEF0) и нажмите кнопку ОК.



2. Автоматически будет создана диаграмма первого уровня декомпозиции с перенесенными в нее потоками родительской диаграммы.



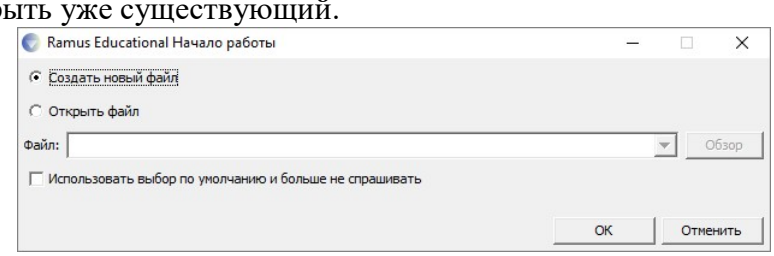
3. Выделите первую работу (функциональный блок), затем двойным щелчком мыши или, выбрав в контекстном меню пункт «Редактировать активный элемент», откройте окно свойств и внесите имя работы. Повторите операцию для оставшихся работ.

4. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом (через вход, управление, механизм или выход) с соответствующим функциональным блоком. В результате должна получиться детализирующая диаграмма.

Диаграмма потоков данных DFD

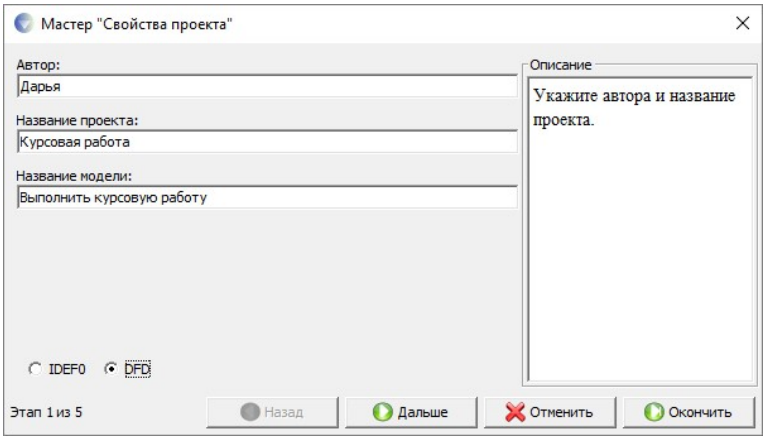
1 Начало работы

1. Запустите программу Ramus Educational. В появившемся окне предлагается создать новый проект или открыть уже существующий.



2. После нажатия на кнопку «ОК» осуществляется запуск мастера проекта.

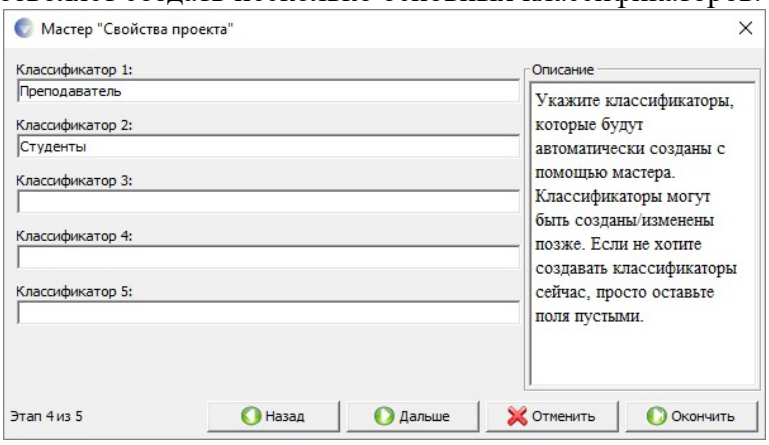
– На первом шаге в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD) – в данном случае – DFD.



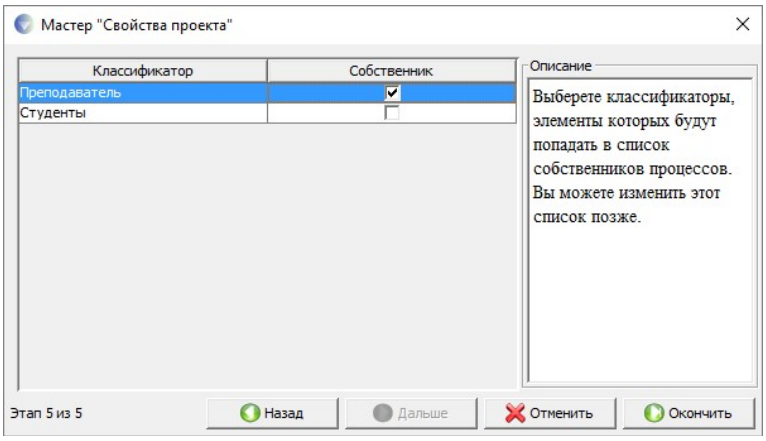
– На втором шаге вводится название организации, использующей данный проект.

– На третьем – дается краткое описание будущего проекта.

– Четвертый шаг позволяет создать несколько основных классификаторов.

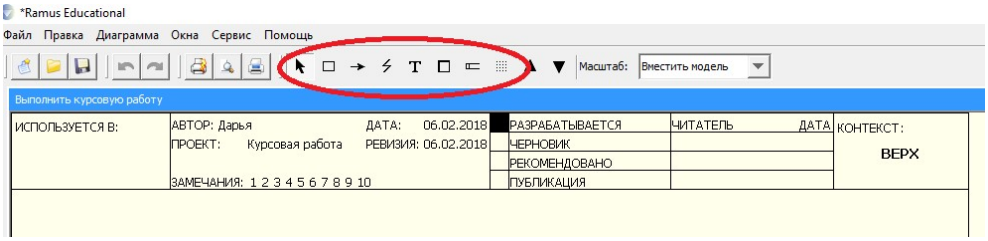


– На пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов.



При необходимости можно завершить работу мастера, нажав кнопку «Окончить».

После завершения работы мастера, откроется рабочее пространство «Диаграммы», в котором можно приступить к построению графической модели. На панели инструментов, в верхней части окна рабочего пространства программы, содержатся элементы диаграммы потоков данных в нотации Gane-Sarson.

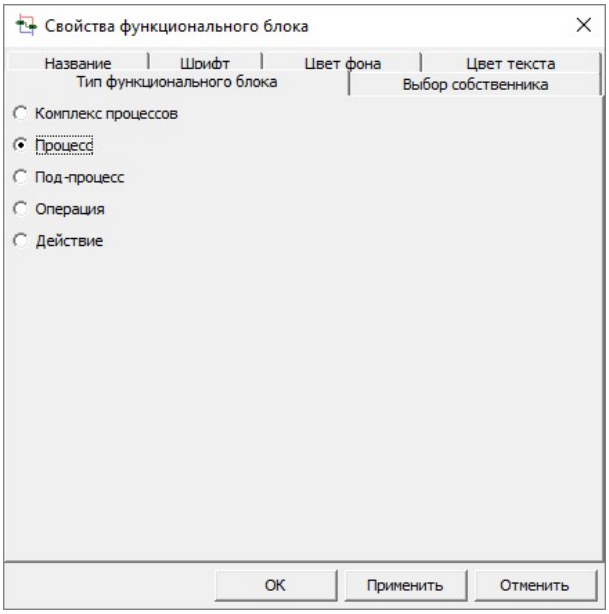


3. Сохраните созданную модель, выбрав опцию меню «Файл» – «Сохранить как».

2 Создание контекстной диаграммы

2. На панели инструментов выберите инструмент создания процесса ( ) и мышью укажите месторасположение на рабочем пространстве нового процесса.

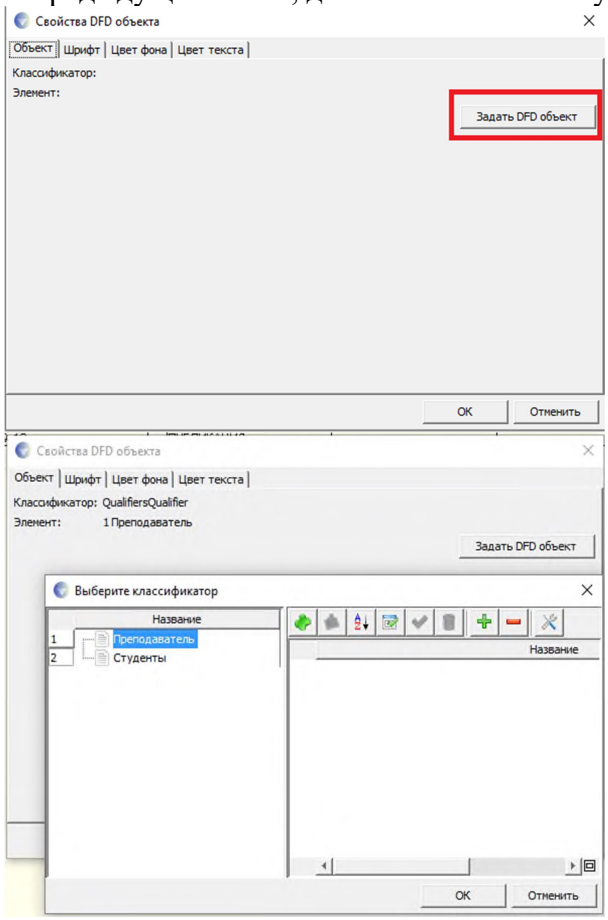
3. Выделив процесс, выберите в контекстном меню опцию **«Редактировать активный элемент»**. В появившемся диалоговом окне на вкладке **«Название»** присвойте процессу имя «*Выполнить курсовую работу*»; на вкладке «**Тип функционального блока**» укажите тип элемента – «Процесс».



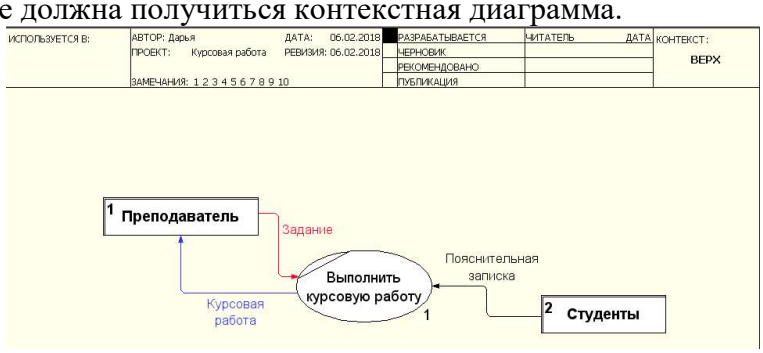
4. На панели инструментов выберите инструмент создания внешней сущности и мышью укажите произвольное ее месторасположение в области построения.

5. В контекстном меню созданной внешней сущности выберите опцию **«Редактировать активный элемент»**, на вкладке **«Объект»** нажмите **«Задать DFD объект»**, после чего, в появившемся окне выделите классификатор *«Преподаватель»* и нажмите «ОК».

6. Повторяя действия предыдущего шага, добавьте внешнюю сущность *«Студенты»*.



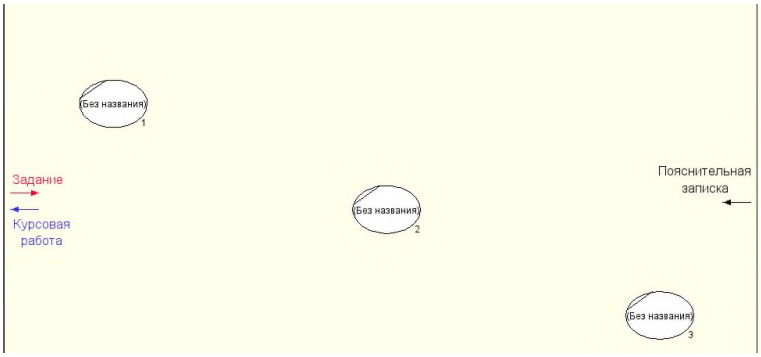
7. Выбра на панели инструментов элемент →, создайте стрелки на контекстной диаграмме. В результате должна получиться контекстная диаграмма.



3 Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне **«Создание новой диаграммы»** установите количество функциональных блоков 3, укажите тип диаграммы (DFD) и нажмите кнопку ОК.

2. Автоматически будет создана диаграмма первого уровня декомпозиции с перенесенными в нее потоками родительской диаграммы.



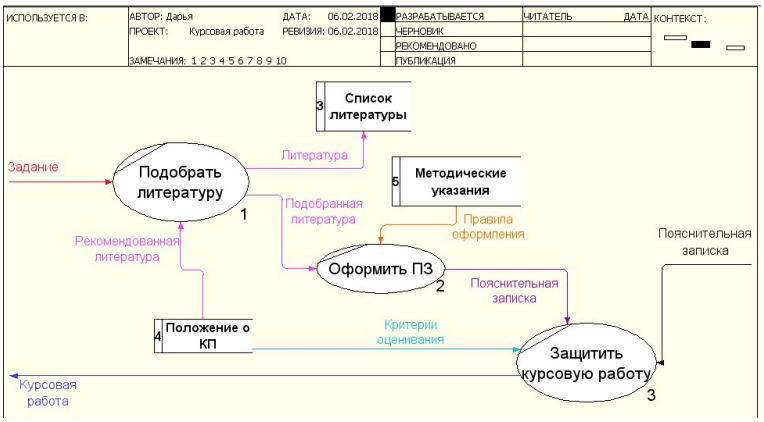
3. Двойным щелчком мыши на одном из процессов, или, выбрав в контекстном меню процесса пункт **«Редактировать активный элемент»**, откройте окно редактирования свойств и задайте процессу имя. Повторите операцию для оставшихся процессов.

4. Добавьте недостающие классификаторы для задания DFD объектов хранилищам данным. Для этого в меню выберите Окна -> Показать окно -> Классификаторы.

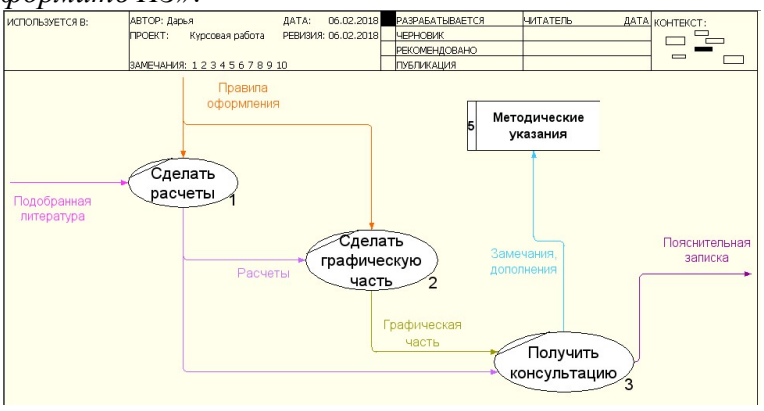
После этого нажмите на кнопку . Название классификатора можно ввести в созданную строку, дважды, медленно кликнув мышью по строке, или же нажав клавишу F2, предварительно выделив нужную строку мышью.

5. Добавьте хранилища данных, воспользовавшись кнопкой палитры инструментов.

6. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом с соответствующим процессом. В результате должна получиться детализирующая диаграмма.



7. На основе описанных выше действий постройте диаграмму декомпозиций второго уровня для процесса *«Оформить ПЗ»*.



Задание практической работы

По образцу построить диаграммы IDEF0 и DFD.

Задание самостоятельной работы

В соответствии с индивидуальным вариантом, построить диаграммы IDEF0 и DFD.

Перечень индивидуальных вариантов приведен в приложении А.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

Требования к построению модели

1. На контекстной диаграмме необходимо указать точку зрения и цель моделирования.
2. Количество блоков любой декомпозиции не менее 3-х и не более 9.
3. Количество декомпозиций – 3 уровня декомпозиции.

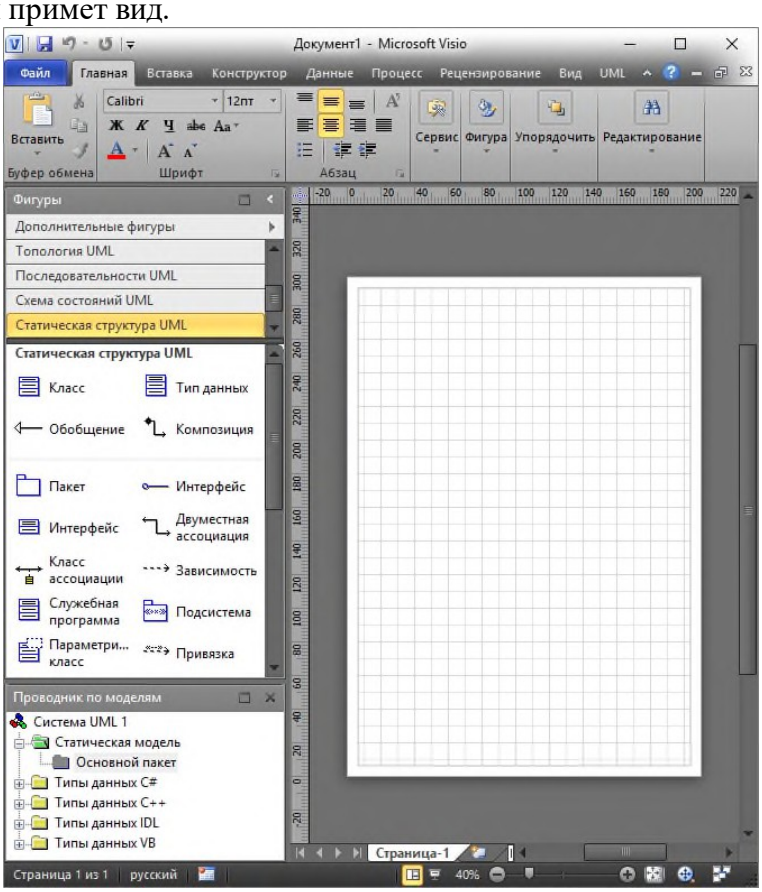
Контрольные вопросы

1. Каковы цели функционального моделирования?
2. Назовите основные компоненты функциональной модели.
3. Какие виды интерфейсных дуг различают в IDEF0?
4. Для чего нужна цель и точка зрения?
5. Что такое функциональный блок?
6. Какие виды диаграмм может содержать функциональная модель?
7. Каково назначение стандарта DFD?
8. В чем основные отличия стандартов IDEF0 и DFD?
9. Каким образом в MS Visio создается схема DFD? Какие для этого используются нотации?
10. Какова роль основных элементов в стандарте DFD?

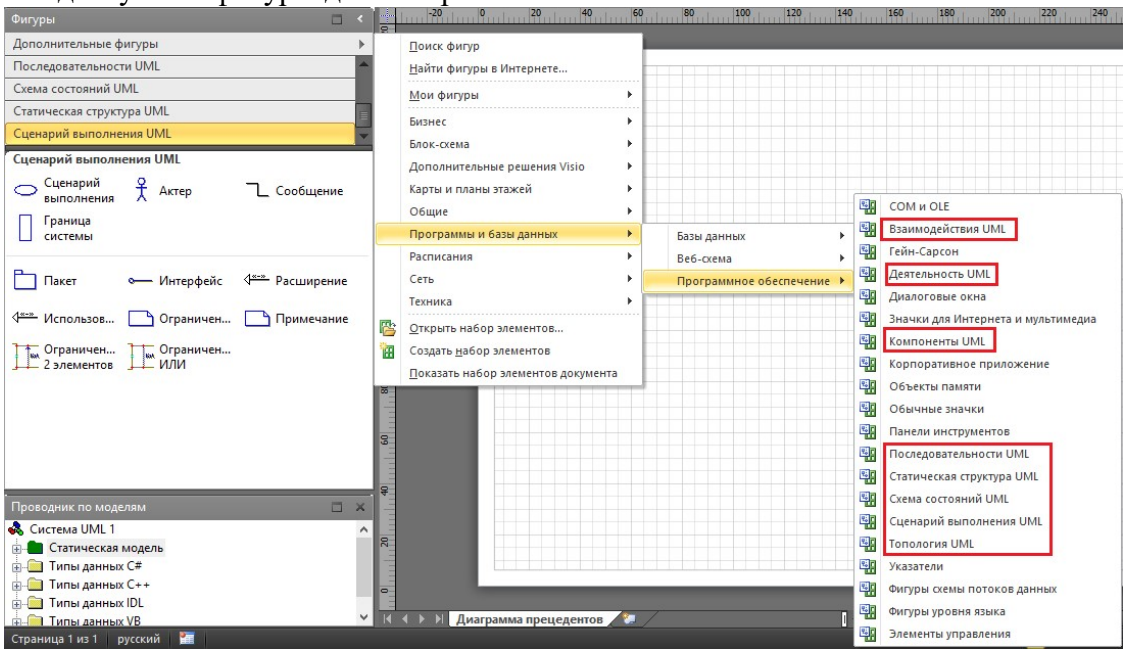
Построение диаграммы Вариантов использования

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите MS Visio.
2. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
3. Окно программы примет вид.



4. Далее необходимо открыть все фигуры, необходимые для построения UML-диаграмм. Для этого в левой части экрана необходимо нажать кнопку *Дополнительные фигуры.* В открывшемся вспомогательном меню выбрать *Программы и БД -> Программное обеспечение* и выбрать все доступные фигуры для построения UML.



5. После этого необходимо провести следующие этапы моделирования.

5.1. Выбор актеров.

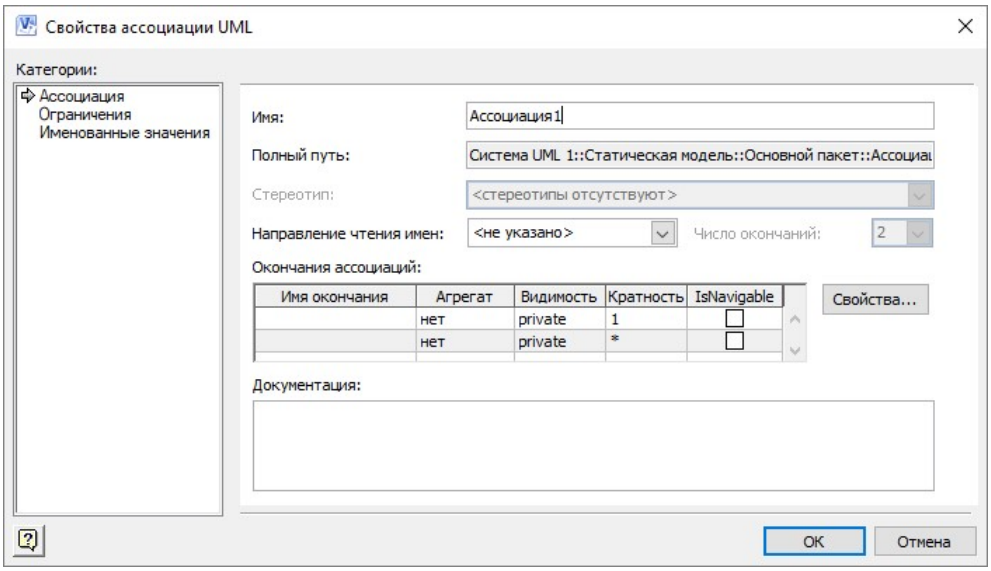
В качестве актеров данной системы могут выступать два субъекта, один из которых является продавцом, а другой – покупателем. Каждый из этих актеров взаимодействует с рассматриваемой системой продажи товаров по каталогу и является ее пользователем, т. е. они оба обращаются к соответствующему сервису «Оформить заказ на покупку товара». Как следует из существа выдвигаемых к системе требований, этот сервис выступает в качестве варианта использования разрабатываемой диаграммы, первоначальная структура которой может включать в себя только двух указанных актеров и единственный вариант использования (рис. 14).

– В группе фигур *Сценарий выполнения UML* выбрать блок *Граница системы* и добавить его на лист.

– Внутрь границы системы добавить блок *Сценарий выполнения* и добавить к нему название, дважды щелкнув внутри блока.

– Добавить два блока *Актер* – покупатель и продавец.

– С помощью блока *Сообщение* установите связь актеров и варианта использования. Двойным щелчком правой кнопки мыши по блоку Сообщение откройте окно *Свойств ассоциации UML,* проведите настройки.

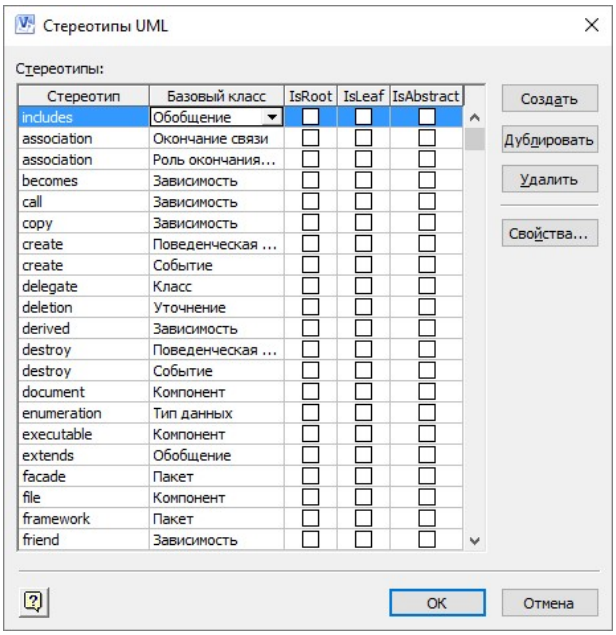


5.2. Выделение дополнительных вариантов использования.

Детализировать вариант использования «Оформить заказ на продажу товара» можно выделив следующие дополнительные варианты использования:

- обеспечить покупателя информацией – является отношением включения;
- согласовать условия оплаты – является отношением включения;
- заказать товар со склада – является отношением включения;
- запросить каталог товаров – является отношением расширения.

Так как в MS Visio отсутствует отношение включения, его необходимо добавить самостоятельно. Для этого перейти на вкладку *UML ->* в группе *Модель* выбрать пункт *Стереотипы.* В открывшемся окне нажать кнопку *Создать* и настроить стереотип



Далее на новом листе необходимо добавить границу системы и все варианты использования. После чего соединить варианты использования с помощью блока *Расширение.*

Для того, чтобы изменить тип отношения дважды щелкните по стрелке и окне свойств задайте необходимые параметры.

Дополненная диаграмма вариантов использования примет вид.

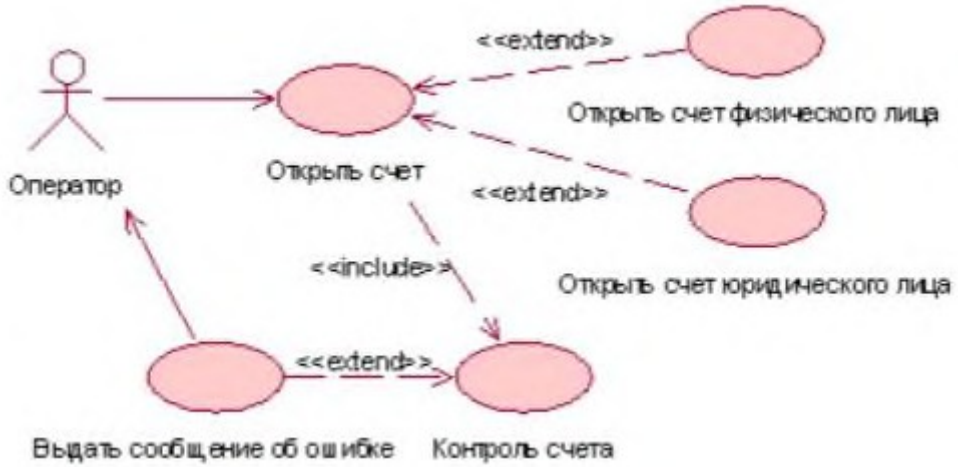
5.3. Написание описательной спецификации для каждого варианта использования.

Спецификация для варианта использования «Оформить заказ на покупку компьютера» приведена в таблице.

Раздел	**Описание**

Краткое описание	Покупатель желает оформить заказ на покупку компьютера, который он выбрал в каталоге товаров. При условии, что клиент зарегистрирован и выбранный компьютер есть в наличии оформляется заказ. Если клиент не зарегистрирован, то предлагается ему пройти регистрацию, и после этого заказать выбранный компьютер. Если компьютера нет в наличии, то предлагается заказать товар со склада в течении заданного срока поставки.
Субъекты	Продавец, Покупатель
Предусловия	В каталоге товаров имеются компьютеры, которые можно заказать. У покупателей есть доступ к системе для регистрации. Продавцы умеют пользоваться рассматриваемой системой продажи. У покупателя есть бонусы.
Основной поток	Зарегистрированный покупатель имеет возможность заказать любой компьютер из каталога товаров. В случае наличия выбранного компьютера оформляется заказ с присвоением ему уникального номера. После этого покупателю предлагается выбрать способ оплаты и способ получения компьютера. В случае отсутствия компьютера в наличии предлагается оформить заказ со склада и ожидания его поставки в рамках указанного срока или выбрать другой компьютер.
Альтернативный поток	Покупатель не зарегистрирован. В этом случае, прежде чем оформить заказ на компьютер, ему предлагается пройти регистрацию. Попытка заказать товар, который отсутствует на складе Начисление бонусов
Постусловия	Заказ оформлен и определен срок поставки компьютера и место его получения

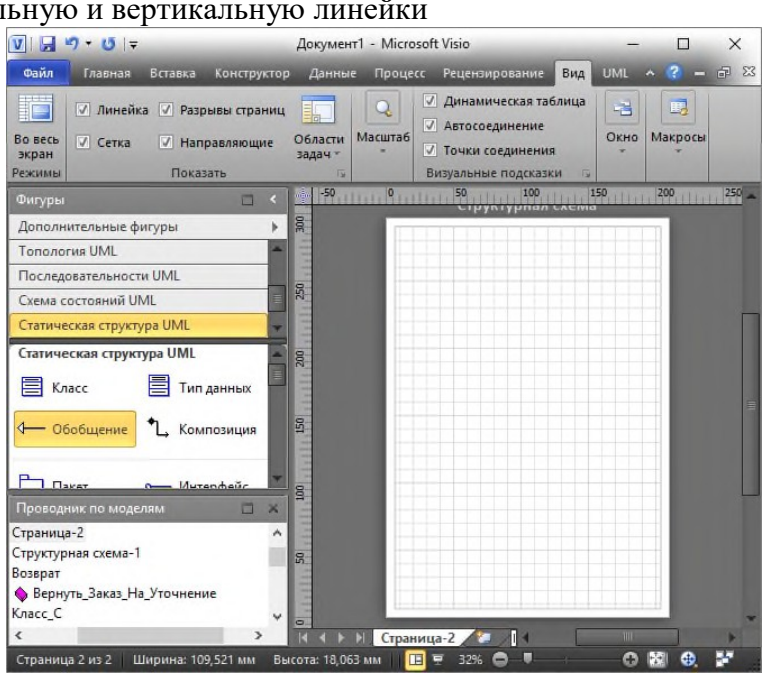
На рисунках приведены примеры диаграмм вариантов использования для различных систем.



Построение диаграммы Классов

1. Запустите MS Visio.
2. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
3. Ознакомьтесь с элементами графического интерфейса и найдите обязательные панели

инструментов **Фигуры**, содержащие категории **Деятельность UML, Взаимодействия UML, Компоненты UML, Топология UML, Последовательности UML, Схема состояний UML, Статическая структура UML, Сценарий выполнения UML, Проводник по моделям**, содержащий иерархическую структуру объектов Системы UML1, Рабочую область, ярлык Страница_1, горизонтальную и вертикальную линейки



4. Установите следующие параметры страницы: **Ориентация** – Альбомная, **Автоподбор размера** – выключен, **Имя страницы** – *Диаграмма классов для системы продажи товаров по каталогу.*

5. Перейдите в категорию **Статическая структура UML**, ознакомьтесь с содержимым этой категории и найдите элементы: Класс, Пакет, Подсистема, Интерфейс, Метакласс, Двусторонняя ассоциация, Обобщение, Композиция, Примечание, Ограничение и др.

6. Создайте поэтапно статическую структуру классов UML, с помощью которой может быть сформирована некоторая функциональная часть системы, например, *Система продажи товаров по каталогу*. Для чего:

– Выберите структурные элементы (идентифицируйте классы), участвующие в организации продаж, например, *Продавец, Товар, Заказ, Заказ_Оплата, Клиент, Корпоративный_Клиент, Частный_Клиент* и создайте предварительный вариант совокупности классов с указанием имен

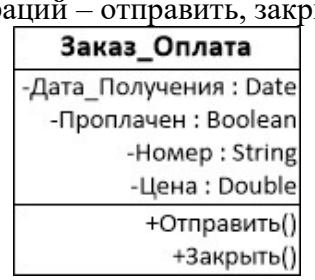
– Установите для каждого класса атрибуты в соответствии с перечнем и содержательным описанием бизнес-процессов:

например, для класса *Продавец* в качестве атрибутов могут выступать данные: *фамилия, имя, отчество, телефон*. В данном случае все атрибуты видимы, принадлежат основному пакету *Продавец* .

для класса *Товар* в качестве атрибутов могут выступать данные: тип, марка, артикул.

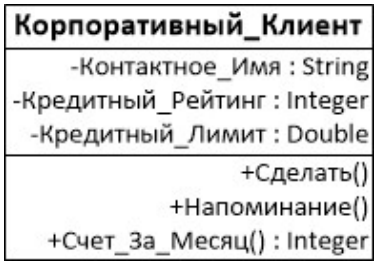
для класса *Заказ* в качестве атрибутов могут выступать данные: количество, цена, статус, а в качестве операций – сформировать заказ.

для класса *Заказ_Оплата* в качестве атрибутов могут выступать данные: дата получения, проплачен, номер, цена, а в качестве операций – отправить, закрыть.



для класса *Клиент* в качестве атрибутов могут выступать данные: имя, адрес, а в качестве операций – кредитный рейтинг.

для класса *Корпоративный_Клиент* в качестве атрибутов могут выступать данные: контактное имя, кредитный рейтинг, кредитный лимит, а в качестве операций – сделать, напоминание, счет за месяц.



для класса *Частный_Клиент* в качестве атрибутов могут выступать данные: номер кредитной карты.

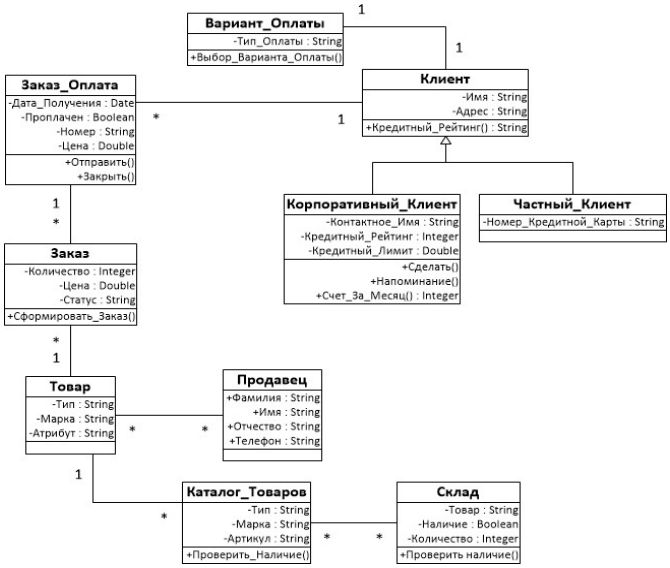
для класса *Вариант_Оплаты* в качестве атрибутов могут выступать данные: тип оплаты, а в качестве операций – выбор варианта оплаты.

для класса *Каталог_Товаров* в качестве атрибутов могут выступать данные: тип, марка, артикул, а в качестве операций – проверить наличие.

для класса *Склад* в качестве атрибутов могут выступать данные: товар, наличие, количество, а в качестве операций – Проверить наличие.

– Убедитесь, что все элементы наполнены адекватным содержанием и расположите все структурные элементы диаграммы наиболее оптимально на странице для установления отношений между ними.

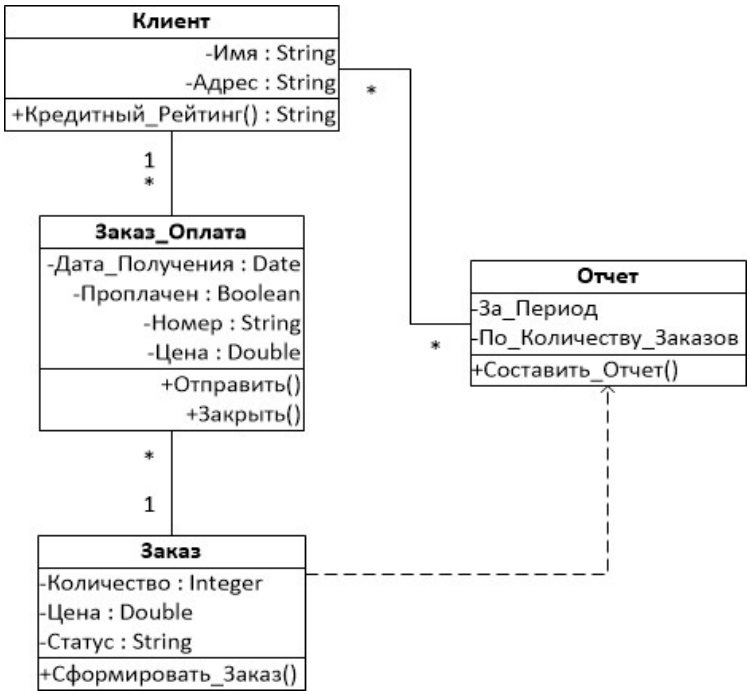
В качестве примера на рис. 26 показан набор классов, описывающих реализацию системы продаж товаров по каталогу. Акцент сделан на классе *Клиент*, с которым связан класс *Заказ_Оплата* посредством двусторонней ассоциации «один-ко-многим», *Вариант_Оплаты* – двусторонней ассоциацией «один-к-одному» и классы *Корпоративный_Клиент* и *Частный_Клиент* посредством отношения обобщения. Классы *Заказ_Оплата* и *Товар* связаны с классом *Заказ* посредством двусторонней ассоциации «один-ко-многим». Класс *Товар* связан с классом *Продавец* двусторонней ассоциацией «многие-ко-многим» и классом *Каталог_Товаров* двусторонней ассоциацией «один-ко-многим». Класс *Каталог_Товаров*



связан посредством двусторонней ассоциации «многие-ко-многим» с классом *Склад.*

7. Создайте новую страницу с именем *Диаграмма классов учета клиентов*, и установите следующие опции: **Ориентация** – Альбомная, **Автоподбор размера** – выключен.

8. Идентифицируйте классы учета клиентов, осуществляющих заказы и создайте диаграмму классов с указанием их имен, атрибутов, операций, например.



Задание практической работы

По образцу построить диаграммы вариантов использования и классов.

Задание самостоятельной работы

В соответствии с индивидуальным вариантом, построить диаграммы вариантов использования и классов.

Перечень индивидуальных вариантов приведен в приложении А.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

Контрольные вопросы

1. Для чего используется язык UML?
2. Назначение диаграммы вариантов использования?
3. Что такое «актер»?
4. Что такое «вариант использования»?
5. Что такое «интерфейс»?
6. Что такое «примечание»?
7. Перечислить виды отношений между актерами и вариантами использования, охарактеризовать каждое из них?
8. Каково назначение диаграммы классов?
9. Назовите основные элементы диаграммы классов.
10. Какие виды связей доступны в диаграмме классов?
11. Для чего используется каждый вид связи?
12. Как создать диаграмму классов в VISIO?

Построение диаграммы Состояний

Цель: изучение основ создания диаграмм состояний на языке UML, получение навыков построения диаграмм состояний, применение приобретенных навыков для построения объектно- ориентированных моделей определенной предметной области.

Форма отчета:

Диаграмма Состояний с описанием процесса построения диаграмм.

Краткие теоретические сведения

Диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Именно для описания реакции объекта на подобные внешние воздействия и используются диаграммы состояний.

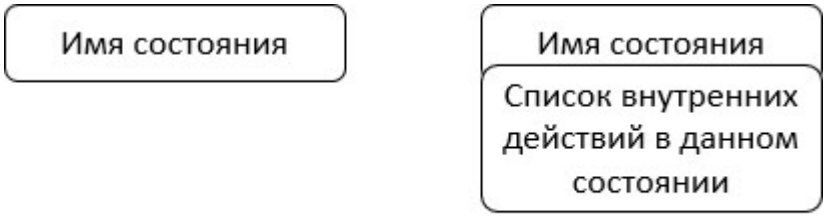
Диаграмма состояний описывает возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

Хотя диаграммы состояний чаще всего используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

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

Состояние

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



(а) (б)

Секция «Список внутренних действий» содержит перечень внутренних действий или деятельностей, которые выполняются в процессе нахождения моделируемого элемента в данном состоянии. Каждое из действий записывается в виде отдельной строки и имеет следующий формат:

<метка-действия '/' выражение-действия>

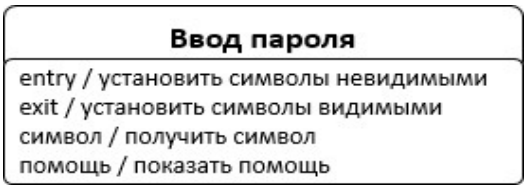
Метка действия указывает на обстоятельства или условия, при которых будет выполняться деятельность, определенная выражением действия:

– ***entry*** – эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент входа в данное состояние (входное действие);

– ***exit*** – эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент выхода из данного состояния (выходное действие);

– ***do*** – эта метка специфицирует выполняющуюся деятельность («doactivity»), которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия. В последнем случае при завершении события генерируется соответствующий результат;

– ***include*** – эта метка используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата.



Пример: Аутентификация входа

Начальное и конечное состояния

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояния). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме состояний графической области, от которой начинается процесс изменения состояний.

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени.

Переход

Простой переход (simpletransition) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий. В этом случае говорят, что переход срабатывает, Или происходит срабатывание перехода. До срабатывания

перехода объект находится в предыдущем от него состоянии, называемым исходным состоянием, или в источнике (не путать с начальным состоянием – это разные понятия), а после его срабатывания объект находится в последующем от него состоянии (целевом состоянии).

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

На диаграмме состояний ***переход изображается*** сплошной линией со стрелкой, которая направлена в целевое состояние. Каждый переход может помечен строкой текста, которая имеет

следующий общий формат:

<сигнатура события>'['<сторожевое условие>']' <выражение действия>

Событие

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

События играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. Имя события идентифицирует каждый отдельный переход на диаграмме состояний и может содержать строку текста, начинающуюся со строчной буквы.

Сторожевое условие

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

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

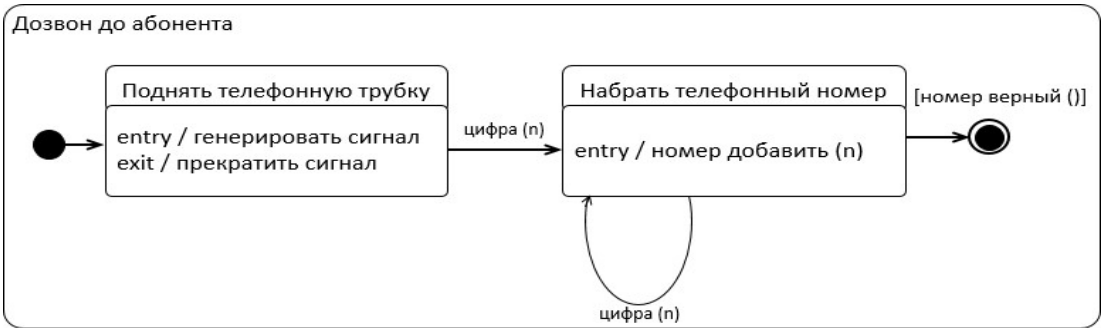
В общем случае из одного состояния может быть несколько переходов с одним и тем же событием-триггером. При этом никакие два сторожевых условия не должны одновременно принимать значение «истина». Каждое из сторожевых условий необходимо вычислять всякий раз при наступлении соответствующего события.

Пример

Составное состояние и подсостояние

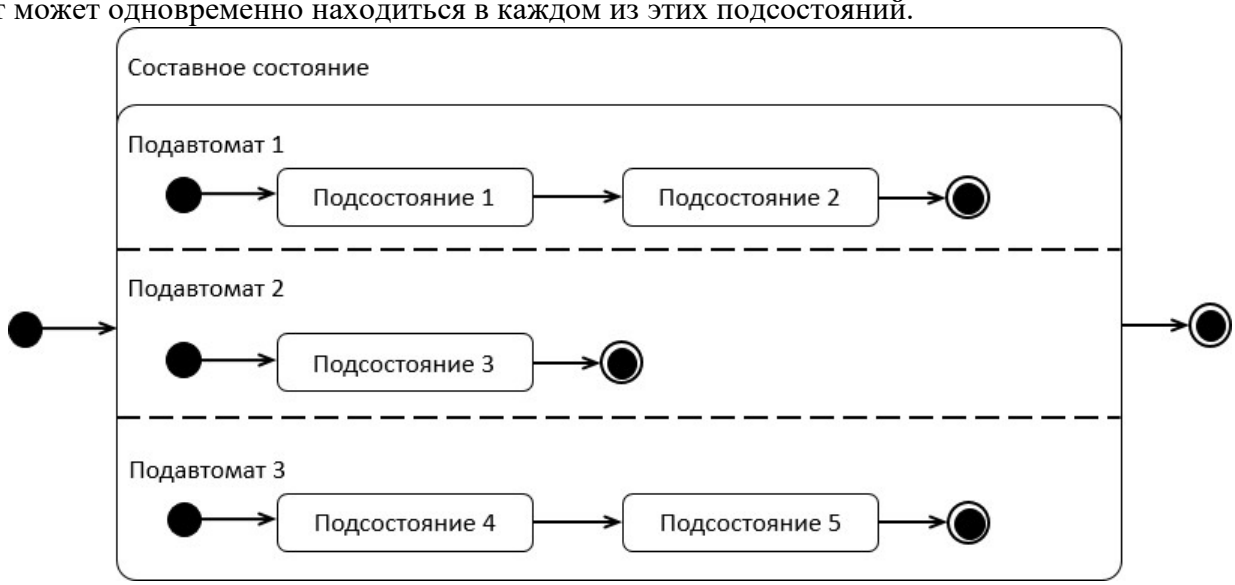
Составное состояние (compositestate) – такое сложное состояние, которое состоит из других вложенных в него состояний. Последние будут выступать по отношению к первому как подсостояния (substate).

Последовательные подсостояния (sequential substates) используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостояний. Поведение объекта в этом случае представляет собой последовательную смену подсостояний, начиная от начального и



заканчивая конечным подсостояниями. Хотя объект продолжает находиться в составном состоянии, введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты его внутреннего поведения.

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

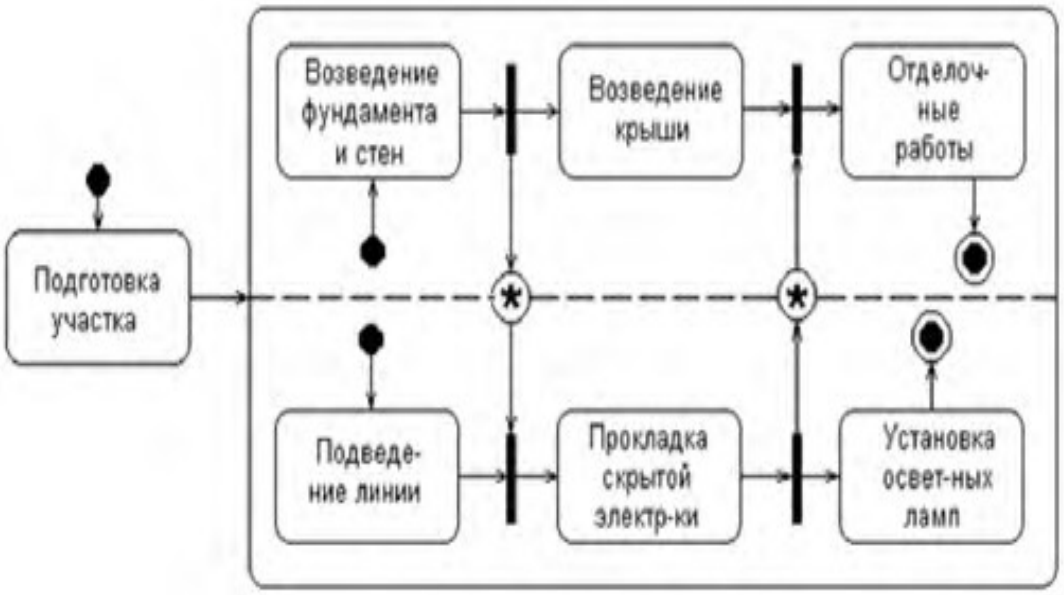


Синхронизирующие состояния

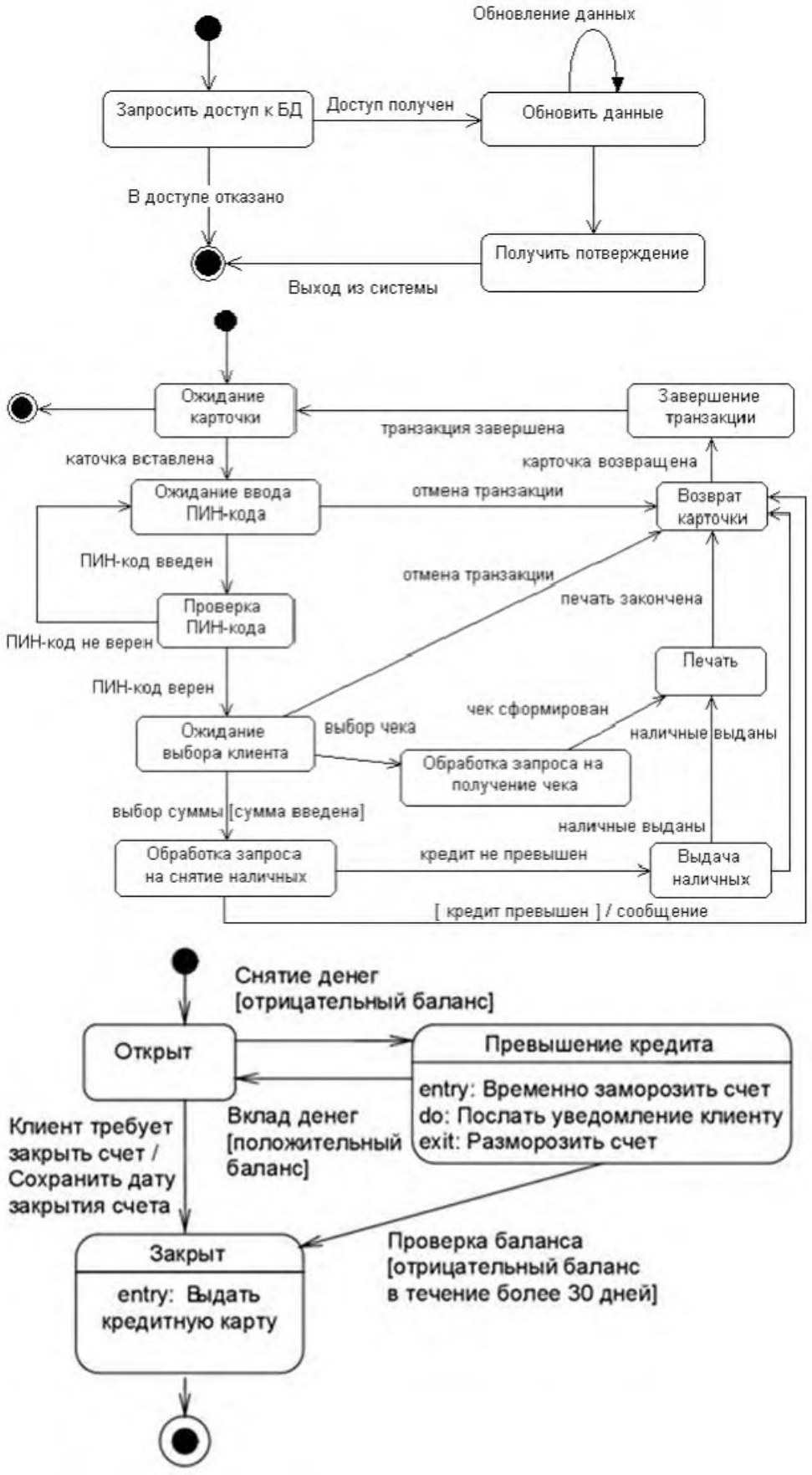
Поведение параллельных подавтоматов независимо друг от друга, что позволяет реализовать многозадачность в программных системах. Однако в отдельных случаях может возникнуть необходимость учесть в модели синхронизацию наступления отдельных событий. Для этой цели в языке UML имеется специальное псевдосостояние, которое называется синхронизирующим состоянием.

Синхронизирующее состояние (synch state) обозначается небольшой окружностью, внутри которой помещен символ звездочки "*". Оно используется совместно с переходом-соединением или переходом-ветвлением для того, чтобы явно указать события в других подавтоматах, оказывающие непосредственное влияние на поведение данного подавтомата.

Для иллюстрации использования синхронизирующих состояний рассмотрим упрощенную ситуацию с моделированием процесса постройки дома. Предположим, что постройка дома включает в себя строительные работы (возведение фундамента и стен, возведение крыши и отделочные работы) и работы по электрификации дома (подведение электрической линии, прокладка скрытой электропроводки и установка осветительных ламп). Очевидно, два этих комплекса работ могут выполняться параллельно, однако между ними есть некоторая взаимосвязь. В частности, прокладка скрытой электропроводки может начаться лишь после того, как будет завершено возведение фундамента и стен. А отделочные работы следует начать лишь после того, как будет закончена прокладка скрытой электропроводки. В противном случае отделочные работы придется проводить повторно. Рассмотренные особенности синхронизации этих параллельных процессов учтены на соответствующей диаграмме состояний с помощью двух синхронизирующих состояний.



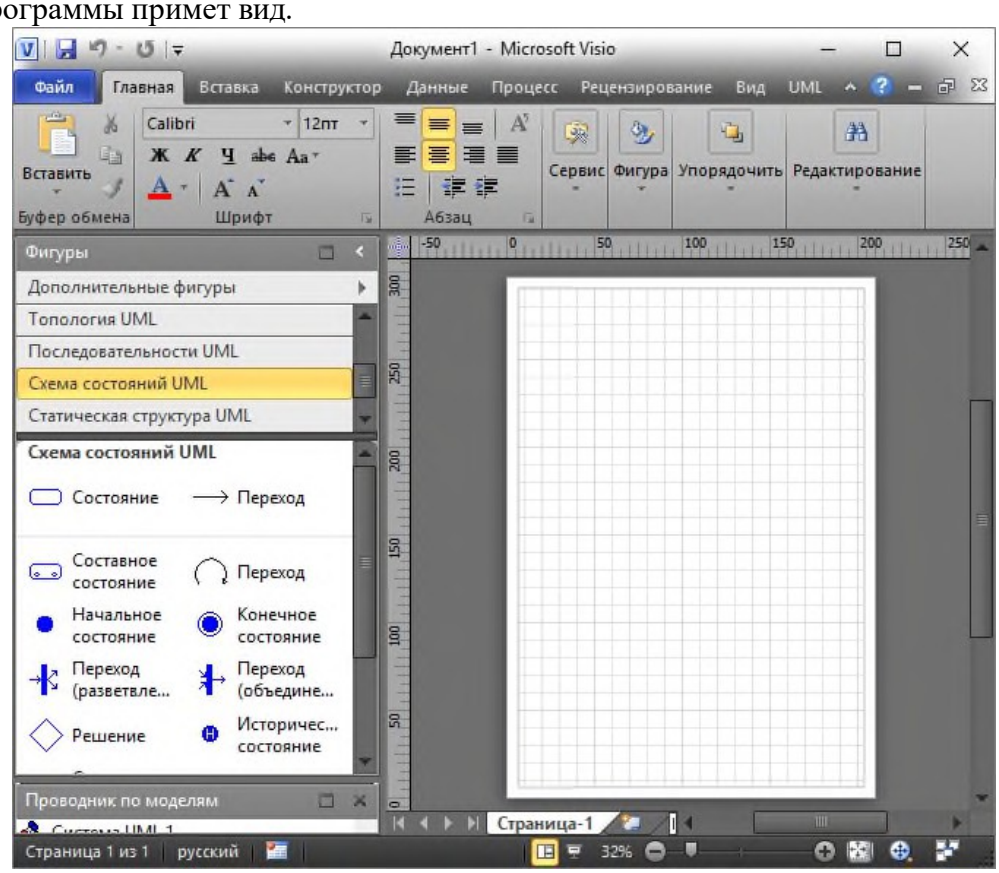
Примеры диаграмм состояний.



Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите MS Visio.
2. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
3. Окно программы примет вид.



Клиент оформляет заказ. Класс Заказ имеет атрибут Статус. Проследим динамику движения заказов в системе с помощью ***диаграммы состояний, составленной для класса Заказ.***

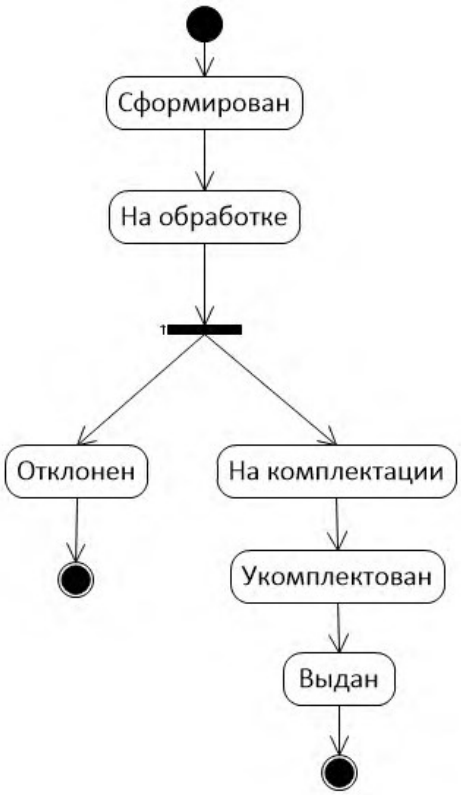
Данные о сформированном заказе поступают продавцу, который проверяет наличие товаров из заказа, проверяет оплату заказа, комплектует его и делает отметку о готовности. После оплаты заказа он выдается клиенту. Продавец делает отметку о том, что заказ выдан.

Если после проверки кредитного рейтинга клиента, он окажется отрицательным, то заказ будет отклонен.

Построим диаграмму состояний для класса Заказ. Для этого, в файле с диаграммой классов, созданной в практическом занятии 8, необходимо проделать следующие действия:

1. Щелкнуть правой кнопкой мыши по *классу Заказ*.
2. В контекстном меню выбрать пункт *Схемы*.
3. Т.к. в настоящее время уже созданных схем нет, нажать кнопку *Создать* и выбрать *Схема состояний*.
4. Переименовать созданный лист в *Схема состояний-Заказ*.
5. Построить диаграмму состояний для класса Заказ.

После формирования заказа он должен быть оплачен. Обработка заказа подразумевает проверку наличия товара и проверку оплаты. Переход в одно из состояний На комплектации, Укомплектован, Выдан означает смену Статуса заказа.



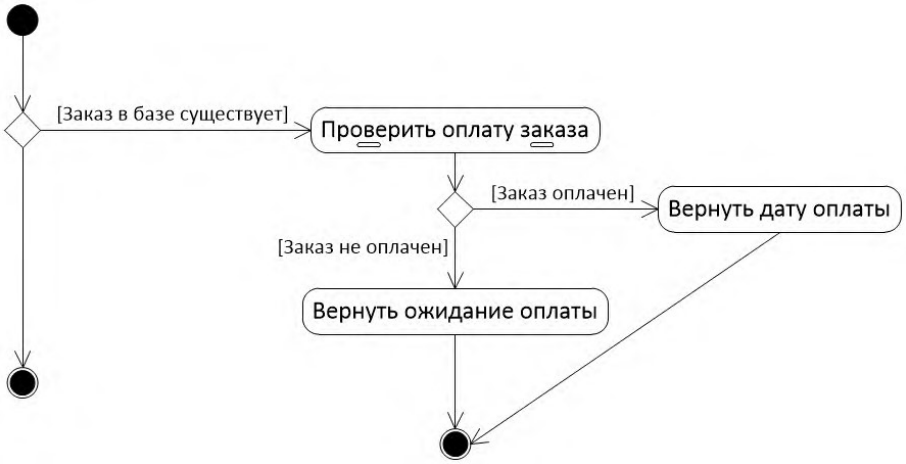
Далее опишем с помощью ***диаграммы состояний процесс оплаты заказа клиентом***, которому соответствует класс ЗаказОплата.

Построим диаграмму состояний для проверки оплаты заказа.

Чтобы проверить оплату заказа, необходимо определить, существует ли сам заказ. Результатом проверки оплаты заказа является вывод либо сообщения о произведенной оплате с параметрами (дата оплаты), либо сообщения об ожидании оплаты.

Событием, предшествующим проверке оплаты заказа, является занесение информации о заказе в базу данных заказов.

Чтобы построить диаграмму состояний для класса ЗаказОплаты, необходимо проделать действия, описанные в пунктах 1-4 построения диаграммы состояний для класса Заказ. Полученная диаграмма должна иметь вид, изображенный на рис. 15.



На этой диаграмме есть составное состояние ***«Проверить оплату заказа»***, т.к. оно включает в себя *проверку кредитного рейтинга клиента* и *проверку выбора варианта оплаты клиентом*.

Оплату заказа может произвести только клиент с положительным кредитным рейтингом,

поэтому необходимым условием проверки оплаты заказа является проверка кредитоспособности клиента. Если клиент имеет отрицательный кредитный рейтинг, то заказ отклоняется, и на этом дальнейшие события не имеют смысла.

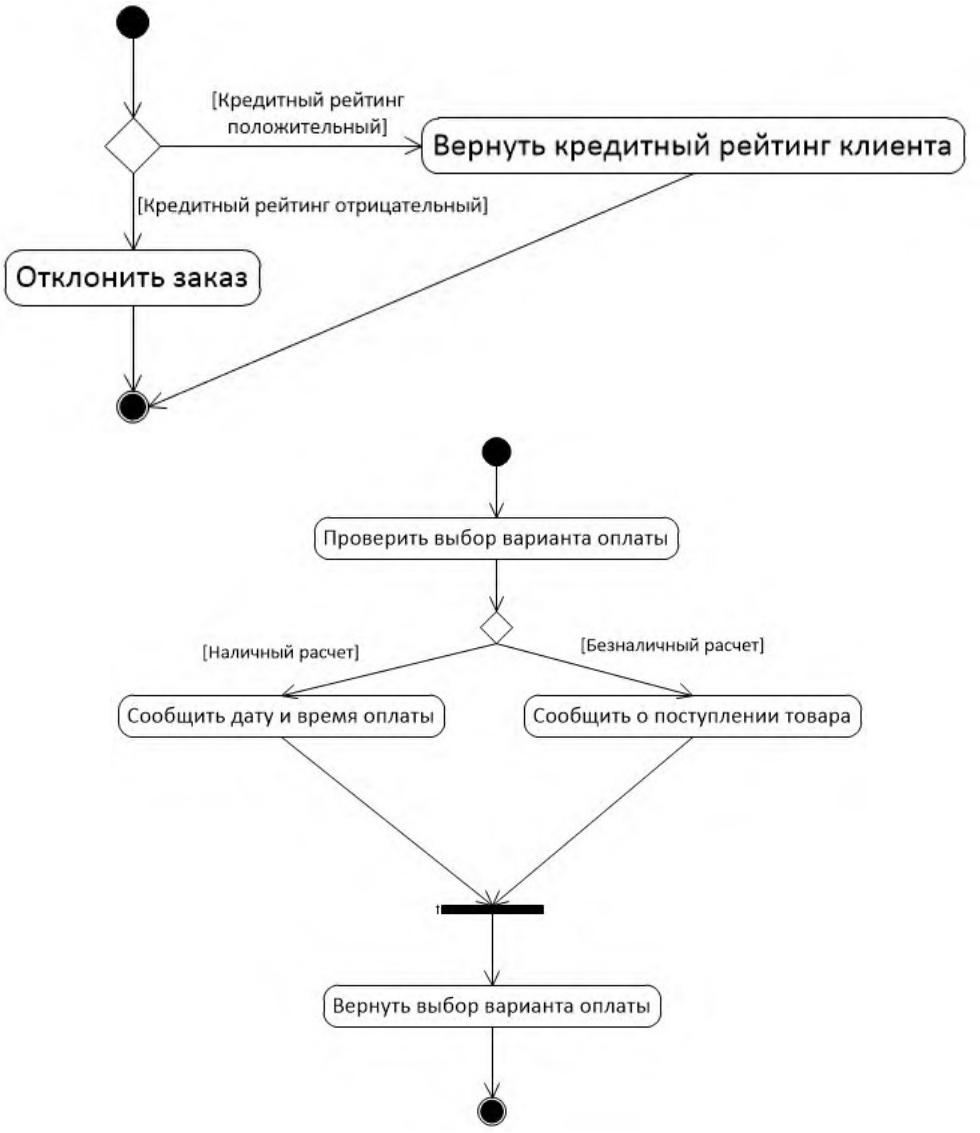
Если кредитный рейтинг клиента положительный, то необходимо проверить, выбрал ли клиент вариант оплаты. Событие, которое переводит систему в состояние ожидания выбора варианта оплаты клиентом, является получение сообщения о кредитоспособности клиента.

Оплата может быть произведена наличными средствами в магазине или с помощью безналичного расчета. В первом случае необходимо договориться с клиентом о дате и времени его прибытия в магазин. Во втором случае необходимо сообщить клиенту о наличии/поступлении товара. Событие, которое переводит систему в состояние ожидания оплаты, является выбор клиентом варианта оплаты.

Соответствующие диаграммы состояний имеют вид

Для создания диаграмм состояний, которые входят в состав составного состояния, нужно:

1. Щелкнуть правой кнопкой мыши по Составному состоянию и выбрать пункт Схема.
2. Либо, в Проводнике по моделям выделить название составного состояния и создать новую страницу с помощью кнопки .



Задание практической работы

По образцу построить диаграмму состояний.

Приложение 3

ФОС для текущего контроля

Задания для оценки знаний

1. Основы инструментальных средств
    - Что такое инструментальные средства разработки ПО?
    - Перечислите основные категории инструментальных средств
    - В чём отличие интегрированных сред разработки от отдельных инструментов?
    - Как происходит выбор инструментальных средств для проекта?
2. Интегрированные среды разработки (IDE)
    - Опишите основные компоненты IDE
    - Какие функции предоставляет современный IDE?
    - Как происходит настройка рабочей среды?
    - Сравните популярные IDE для различных языков программирования
3. Системы контроля версий
    - Что такое система контроля версий?
    - Опишите принципы работы Git
    - Какие существуют альтернативные системы контроля версий?
    - Как происходит ветвление и слияние кода?
4. Средства отладки
    - Какие существуют типы отладчиков?
    - Как происходит пошаговая отладка программы?
    - Какие методы используются для поиска ошибок?
    - Как работают точки останова и условные точки останова?
5. Компиляторы и интерпретаторы
    - В чём разница между компилятором и интерпретатором?
    - Как происходит процесс компиляции?
    - Какие оптимизации выполняет компилятор?
    - Как работают JIT-компиляторы?

6. Средства профилирования
    - Что такое профилирование кода?
    - Какие метрики измеряют профилировщики?
    - Как происходит анализ производительности?
    - Какие существуют инструменты профилирования?
7. Средства автоматизации сборки
    - Что такое система сборки ПО?
    - Опишите основные инструменты автоматизации сборки
    - Как работает система непрерывной интеграции?
    - Какие существуют подходы к управлению зависимостями?
8. Средства тестирования
    - Какие инструменты используются для автоматизированного тестирования?
    - Как работают тестовые фреймворки?
    - Какие существуют средства нагрузочного тестирования?
    - Как происходит автоматизация тестирования?
9. Средства документирования
    - Какие инструменты используются для создания документации?
    - Как работает генерация документации из кода?
    - Какие форматы документации существуют?
    - Как происходит управление технической документацией?
10. Средства управления проектами
    - Какие инструменты используются для планирования проектов?
    - Как работает трекинг задач?
    - Какие существуют средства визуализации прогресса?
    - Как происходит управление ресурсами проекта?
11. Средства анализа кода
    - Какие существуют статические анализаторы кода?
    - Как работают инструменты поиска уязвимостей?
    - Какие метрики качества кода существуют?

- Как происходит анализ покрытия кода тестами?

12. Средства контейнеризации
    - Что такое контейнеризация приложений?
    - Как работает Docker?
    - Какие существуют инструменты оркестрации контейнеров?
    - Как происходит управление контейнеризированными приложениями?
13. Средства облачной разработки
    - Какие существуют облачные IDE?
    - Как работает serverless-разработка?
    - Какие инструменты используются для облачной разработки?
    - Как происходит управление облачными ресурсами?
14. Средства безопасности
    - Какие инструменты используются для анализа безопасности?
    - Как работают средства защиты кода?
    - Какие существуют инструменты шифрования?
    - Как происходит управление доступом к коду?
15. Современные тенденции
    - Какие новые инструменты появляются в разработке ПО?
    - Как развивается AI в разработке ПО?
    - Какие существуют инструменты для low-code/no-code разработки?
    - Как происходит выбор современных инструментальных средств?

Задания для оценки умений

Практические задания по дисциплине «Инструментальные средства разработки программного обеспечения»

1. **Работа с системами контроля версий**
    - Настройте локальный репозиторий Git
    - Создайте несколько веток разработки
    - Выполните слияние веток с разрешением конфликтов
    - Создайте pull request и проведите code review

- Настройте удалённый репозиторий на GitHub/GitLab

2. **Настройка IDE**
    - Установите и настройте популярную IDE (IntelliJ IDEA, Visual Studio Code)
    - Настройте систему автодополнения кода
    - Конфигурируйте инструменты отладки
    - Установите необходимые плагины и расширения
    - Настройте форматирование кода
3. **Отладка программного кода**
    - Используйте отладчик для поиска ошибок в программе
    - Установите точки останова и условные точки останова
    - Проанализируйте стек вызовов
    - Исследуйте значения переменных в процессе выполнения
    - Создайте дамп памяти
4. **Автоматизация сборки**
    - Создайте проект с использованием системы сборки (Maven, Gradle)
    - Настройте зависимости проекта
    - Автоматизируйте процесс компиляции
    - Добавьте этапы тестирования в процесс сборки
    - Создайте исполняемый jar-файл
5. **Профилирование производительности**
    - Используйте профилировщик для анализа производительности
    - Определите узкие места в коде
    - Проанализируйте использование памяти
    - Измерьте время выполнения критических участков
    - Предложите оптимизации на основе результатов профилирования
6. **Тестирование с использованием фреймворков**
    - Настройте тестовый фреймворк (JUnit, TestNG)
    - Создайте набор юнит-тестов
    - Реализуйте интеграционные тесты

- Настройте автоматическую генерацию отчётов
- Внедрите тестовые данные

7. **Контейнеризация приложений**
    - Создайте Dockerfile для приложения
    - Соберите Docker-образ
    - Запустите контейнер
    - Настройте взаимодействие между контейнерами
    - Создайте docker-compose файл
8. **Анализ качества кода**
    - Используйте статический анализатор кода
    - Проанализируйте метрики качества
    - Найдите потенциальные уязвимости
    - Исправьте найденные проблемы
    - Настройте автоматическую проверку кода
9. **Непрерывная интеграция**
    - Настройте CI/CD pipeline
    - Добавьте этапы тестирования
    - Реализуйте автоматическое развёртывание
    - Настройте мониторинг
    - Внедрите систему уведомлений
10. **Работа с облачными инструментами**
    - Создайте проект в облачной IDE
    - Настройте serverless-функции
    - Используйте облачные базы данных
    - Реализуйте масштабирование
    - Настройте мониторинг в облаке
11. **Безопасность приложений**
    - Проведите анализ безопасности кода
    - Реализуйте шифрование данных

- Настройте аутентификацию и авторизацию
- Защитите от распространённых уязвимостей
- Документируйте меры безопасности

12. **Документация программного обеспечения**
    - Создайте техническую документацию
    - Используйте генераторы документации
    - Настройте автоматическую генерацию API-документации
    - Создайте пользовательскую документацию
    - Организуйте хранение документации
13. **Управление проектами**
    - Настройте систему управления задачами
    - Создайте план разработки
    - Отслеживайте прогресс проекта
    - Управляйте ресурсами
    - Анализируйте эффективность работы команды
14. **Оптимизация производительности**
    - Проанализируйте использование ресурсов
    - Оптимизируйте алгоритмы
    - Улучшите работу с памятью
    - Ускорьте время отклика
    - Документируйте проведённые оптимизации
15. **Интеграция инструментов разработки**
    - Настройте взаимодействие между различными инструментами
    - Создайте единый workflow
    - Интегрируйте систему контроля версий с IDE
    - Свяжите инструменты тестирования с системой сборки
    - Организуйте автоматическую проверку кода при коммите

.

Приложение 4

Задания для самостоятельной работы

Теоретические вопросы

1. Что такое автоматизация тестирования ПО и каковы её основные цели?
2. Какие преимущества и недостатки автоматизации тестирования по сравнению с ручным тестированием?
3. Какие критерии отбора тестов для автоматизации вы знаете?
4. Что такое тестовый фреймворк и какие виды фреймворков существуют?
5. В чём разница между data-driven, keyword-driven и behavior-driven подходами?
6. Какие инструменты автоматизации тестирования вы знаете для веб-, мобильных и десктопных приложений?
7. Что такое CI/CD и как автоматизация тестирования интегрируется в эти процессы?
8. Какие языки программирования чаще всего используются для написания автотестов?
9. Что такое API-тестирование и как его автоматизировать?
10. Как организовать генерацию тестовых данных для автоматизации?
11. Что такое Page Object Model и зачем она нужна?
12. Как организовать логирование и отчётность в автотестах?
13. Какие метрики используются для оценки эффективности автоматизации?
14. Как поддерживать и развивать набор автотестов в долгосрочной перспективе?
15. Что такое регрессионное тестирование и как его автоматизировать?
16. Как организовать параллельное выполнение автотестов?
17. Что такое mock-объекты и когда их используют в автоматизации?
18. Как работать с локаторами элементов в веб-автоматизации?
19. Какие существуют стратегии работы с динамическими элементами интерфейса?
20. Как организовать тестирование безопасности с помощью автоматизации?
21. Что такое нагрузочное тестирование и как его автоматизировать?
22. Как использовать Docker для автоматизации тестирования?
23. Как организовать тестирование микросервисной архитектуры?
24. Что такое BDD и какие инструменты поддерживают этот подход?
25. Как интегрировать автотесты с системами управления дефектами (Jira, YouTrack)?
26. Как организовать работу с версиями автотестов (Git)?
27. Как проводить code review для автотестов?
28. Что такое smoke, sanity, regression, acceptance тесты и как их автоматизировать?
29. Как организовать тестирование кроссбраузерности и кроссплатформенности?

30. Как использовать облачные сервисы для запуска автотестов (BrowserStack, SauceLabs)?
31. Как организовать тестирование мобильных приложений (Appium, Espresso)?
32. Что такое flaky-тесты и как с ними бороться?
33. Как организовать мониторинг стабильности автотестов?
34. Как использовать Allure/ExtentReports для построения отчётов?
35. Как организовать тестирование REST API (Postman, REST Assured)?
36. Как автоматизировать тестирование баз данных?
37. Как организовать тестирование с использованием микросервисных stub-ов?
38. Как использовать Selenide/Selenoid для масштабирования автоматизации?
39. Как обучать новых сотрудников автоматизации тестирования?
40. Какие перспективы развития автоматизации тестирования вы видите на ближайшие 5 лет?

Практические задания

1. Написать простой автотест для проверки авторизации на сайте с использованием Selenium WebDriver.
2. Реализовать data-driven автотест, который проверяет форму регистрации с разными наборами данных.
3. Создать keyword-driven фреймворк для автоматизации UI-тестов.
4. Автоматизировать тестирование API с помощью Postman или REST Assured.
5. Настроить запуск автотестов в Jenkins/GitLab CI.
6. Реализовать Page Object Model для одного из разделов веб-приложения.
7. Настроить параллельное выполнение автотестов с помощью Selenide/Selenoid.
8. Автоматизировать создание тестовых данных (например, генерация пользователей через API).
9. Настроить интеграцию автотестов с Jira (создание багов при падении тестов).
10. Реализовать BDD-автотест с использованием Cucumber/Behave.
11. Автоматизировать тестирование мобильного приложения на Android/iOS.
12. Настроить отчёты Allure/ExtentReports для вашего проекта.
13. Реализовать smoke-suite автотестов для вашего приложения.
14. Автоматизировать проверку кроссбраузерности (например, через BrowserStack).
15. Настроить мониторинг стабильности автотестов (например, с помощью Grafana).

16. Реализовать тест на проверку безопасности (например, XSS/SQL-инъекции).

17. Автоматизировать нагрузочное тестирование с помощью JMeter/Gatling.

18. Реализовать автотест с использованием mock-объектов.

19. Настроить code review для автотестов в Git.

20. Автоматизировать тестирование базы данных (проверка целостности данных после операций).

21. Реализовать тест на проверку локализации интерфейса.

22. Автоматизировать тестирование микросервисной архитектуры (например, с использованием WireMock).

23. Настроить запуск автотестов в Docker-контейнерах.

24. Реализовать автотест для проверки работы с файлами (загрузка/скачивание).

25. Автоматизировать тестирование drag-and-drop операций.

26. Реализовать тест на проверку работы с cookies/localStorage.

27. Настроить автоматическую отправку отчётов по почте/Slack после прогона тестов.

28. Реализовать тест на проверку адаптивности интерфейса.

29. Автоматизировать тестирование WebSocket-соединений.

30. Реализовать тест на проверку работы с CAPTCHA (через mock/stub).

31. Настроить работу с переменными окружения для разных стендов (dev, test, prod).

32. Автоматизировать тестирование интеграции с внешними сервисами (например, платежные системы).

33. Реализовать тест на проверку работы с уведомлениями (email, push).

34. Настроить версионирование автотестов в Git и организовать branching strategy.

35. Автоматизировать тестирование drag-and-drop операций.

36. Реализовать тест на проверку работы с cookies/localStorage.

37. Настроить автоматическую отправку отчётов по почте/Slack после прогона тестов.

38. Реализовать тест на проверку адаптивности интерфейса.

39. Автоматизировать тестирование WebSocket-соединений.

40. Реализовать тест на проверку работы с CAPTCHA (через mock/stub).