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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине
«Системная инженерия»
для студентов направления подготовки «Информационные системы и технологии»
Направленность (профиль)
«Искусственный интеллект, математическое моделирование и суперкомпьютерные технологии в разработке информационных систем»

Ставрополь
2026

Содержание

ВВЕДЕНИЕ

1. Цели и задачи освоения дисциплины:

1.1.Познакомить обучающихся с теоретические знаниями о комплексе технических, организационных и управленческих вопросов создания сложных систем.
1.2. Выработать у обучающихся целостное представление о задачах, проблемах, подходах и применяемых инструментальных средствах в области системной инженерии.
1.3. Выработать у обучающихся практические навыки по разработке моделей процессов системной инженерии и жизненного цикла систем.

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
УК-2 Способен управлять проектом на всех этапах его жизненного цикла	УК-2.1: Разрабатывает концепцию проекта в рамках научной постановки проблемы: формулирует цели, задачи, обосновывает актуальность, значимость, ожидаемые результаты и возможные сферы их применения	Правильно формулирует цели и задачи проекта. актуальность, значимость, правильно формулирует ожидаемые результаты. Правильно формулирует концепцию проекта с учетом практик системной инженерии. Грамотно применяет подходы системной инженерии к управлению жизненным циклом проекта. Грамотно применяет методы управления проектом. Грамотно применяет анализ результатов и возможные сферы их применения проекта Выбирает адекватные методы управления жизненным циклом проекта. Выбирает адекватные методы постановки задач проекта Выбирает адекватные методы применения практик системной инженерии
	УК-2.2: Планирует необходимые ресурсы	Правильно использует необходимые ресурсы Грамотно выбирает необходимые ресурсы Выбирает адекватные методы планирования необходимых ресурсов
	УК-2.3: Разрабатывает план реализации проекта с использованием инструментов планирования и осуществляет мониторинг хода его реализации	Правильно использует инструменты планирования Грамотно разрабатывает план реализации проекта с использованием инструментов планирования Выбирает адекватные методы разработки плана реализации проекта с использованием инструментов

		планирования и осуществляет мониторинг хода его реализации
УК-3 Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели	УК-3.1: Вырабатывает стратегию сотрудничества; организует отбор членов команды для достижения поставленной цели	Грамотно выбирает стратегию сотрудничества Правильно применяет стратегию сотрудничества Выбирает адекватные методы отбора членов команды для достижения поставленной цели
	УК-3.2: Планирует командную работу, распределяет поручения и делегирует полномочия членам команды	Выбирает адекватные методы планирования командной работы Грамотно планирует командную работу Правильно распределяет поручения и делегирует полномочия членам команды
	УК-3.3: Организует дискуссии по заданной теме и обсуждение результатов работы команды с привлечением оппонентов	Выбирает адекватные методы организации дискуссий Выбирает адекватные методы взаимодействия с оппонентами Грамотно организует дискуссии по заданной теме с привлечением оппонентов
ОПК-5 Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем	ОПК-5.1: Анализирует, выбирает и использует программное и аппаратное обеспечение информационных и автоматизированных систем	Грамотно анализирует и выбирает программное обеспечение информационных и автоматизированных систем Логически обосновывает выбор программного и аппаратного обеспечения информационных и автоматизированных систем Правильно использует программное и аппаратное обеспечение информационных и автоматизированных систем
	ОПК-5.2: Модернизирует программное обеспечение информационных и автоматизированных систем	Грамотно анализирует необходимость модернизации программного обеспечения информационных и автоматизированных систем Выбирает адекватные методы модернизации программного обеспечения информационных и автоматизированных систем Логически обоснованно проводит модернизацию программного обеспечения информационных и автоматизированных систем
ОПК-6 Способен использовать методы и средства системной инженерии в области получения,	ОПК-6.1: Анализирует, выбирает методы и средства системной инженерии в области получения, передачи, хранения, переработки и	Грамотно анализирует методы и средства системной инженерии в области получения, передачи, хранения, переработки и представления информации посредством информационных технологий

передачи, хранения, переработки и представления информации посредством информационных технологий	представления информации посредством информационных технологий	Выбирает адекватные информационные технологии для реализации практик системной инженерии Логически обоснованно выбирает методы и средства системной инженерии в области получения, передачи, хранения, переработки и представления информации посредством информационных технологий
	ОПК-6.2: Применяет и развивает методы и средства системной инженерии в профессиональной деятельности	Выбирает адекватные методы системной инженерии для применения в профессиональной деятельности Грамотно применяет практики системной инженерии в профессиональной деятельности Логически обоснованно применяет и развивает методы и средства системной инженерии в профессиональной деятельности
ОПК-8 Способен осуществлять эффективное управление разработкой программных средств и проектов	ОПК-8.1: Осуществляет управление работами по выявлению и анализу требований к программным средствам и проектам	Грамотно анализирует требования к программным средствам и проектам Правильно управляет работами по выявлению и анализу требований к программным средствам и проектам Выбирает адекватные методы управления работами по выявлению и анализу требований к программным средствам и проектам
	ОПК-8.2: Проводит мониторинг и управляет работами проекта в ИТ области	Проводит мониторинг процесса выполнения проекта в ИТ области Выбирает адекватные методы управления работами проекта в ИТ области Логически обоснованно управляет работами проекта в ИТ области
ОПКД-6 Способен использовать методы научных исследований и математического моделирования в области проектирования и управления системами искусственного интеллекта, в том числе универсального искусственного интеллекта	ОПКД-6.1: Применяет логические методы и приемы научного исследования, методологические принципы современной науки, направления, концепции, источники знания и приемы работы с ними, основные особенности научного метода познания, программно-целевые методы решения научных проблем в профессиональной деятельности	Грамотно анализирует логические методы и приемы научного исследования, методологические принципы современной науки, направления, концепции, источники знания и приемы работы с ними основные особенности научного метода познания, программно-целевые методы решения научных проблем в профессиональной деятельности Выбирает адекватные методы анализа логических методов и приемов научного исследования, применяемых в системной инженерии Логически обосновывает необходимость использования методов научных исследований и математического моделирования в области системной

		инженерии
	ОПКД-6.2. Осуществляет методологическое обоснование научного исследования, создание и применение библиотек искусственного интеллекта	Осуществляет методологическое обоснование научного исследования, создание и применение библиотек искусственного интеллекта для решения задач системной инженерии Грамотно применяет библиотеки искусственного интеллекта для решения задач системной инженерии Логически обосновывает применение библиотек искусственного интеллекта для решения задач системной инженерии

РАЗДЕЛ 1. ОСНОВНЫЕ ПОНЯТИЯ СИСТЕМНОЙ ИНЖЕНЕРИИ

Лабораторная работа № 1

ЗНАКОМСТВО С ВОЗМОЖНОСТЯМИ ARGOUML

Цель: изучить возможности программы ArgoUML и ее установку.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

В последнее десятилетие прошлого тысячелетия, объектно-ориентированная (ОО) технология, наконец, завершила превращение из лабораторного инструмента 1960-х в основную парадигму программного обеспечения. Путь развития ОО технологии был длинным и трудным в первую очередь потому, что для ее применения был необходим значительный сдвиг в мышлении программистов, дизайнеров, разработчиков и других лиц, участвующих в жизненном цикле разработки программного обеспечения. Основной предпосылкой, лежащей в основе ОО технологии, является моделирование. ОО технология разработана и реализована по существу, как моделирование реального мира с помощью программного обеспечения артефактов. Эта посылка дает такие мощные результаты благодаря своей простоте. При анализе, разработке и внедрении ОО систем могут быть использованы идеи, изложенные на одном и том же языке. Это позволяет применять моделирование при разработке и испытаниях вместо того, чтобы в первую очередь на самом деле строить систему. Эта особенность в сочетании с возможностью проектировать системы на очень высоком уровне благодаря использованию опыта практиков позволила осуществить разработку и успешную реализацию более сложных систем, чем это было возможно ранее.

Принятие унифицированного языка моделирования (UML) в качестве стандартного языка для описания объектно-ориентированного подхода продолжило продвижение ОО технологии в мейнстрим.

Программа ArgoUML была задумана как инструмент описания окружающей среды для использования при анализе и проектировании

объектно-ориентированных программных систем. В этом смысле программа ArgoUML похожа на многие коммерческие CASE-средства для моделирования программных систем. Но ArgoUML имеет ряд очень важных отличий многих из этих инструментов:

- ArgoUML опирается на исследования в области когнитивной психологии и включает в себя ряд функций, которые способствуют повышению производительности за счет поддержки когнитивных (познавательных) потребностей разработчиков объектно-ориентированного программного обеспечения.
- ArgoUML поддерживает открытые стандарты - UML, XMI, SVG, OCL и другие. Поэтому ArgoUML продолжает развиваться и имеет преимущество перед коммерческими CASE-средствами.
- ArgoUML является 100% приложением Java. Это позволяет ArgoUML работать на всех платформах, для которых доступна платформа Java2.
- ArgoUML является программным продуктом с открытым исходным кодом. Открытый код гарантирует, что разработчики и исследователи программного обеспечения нового поколения теперь будут иметь проверенную основу, которая позволит им управлять развитием и эволюцией CASE- технологий.

ArgoUML является предметно-ориентированной средой разработки, что обеспечивает когнитивную поддержку объектно-ориентированного проектирования. ArgoUML обеспечивает те же функции автоматизации, что и коммерческие CASE-средства, но она ориентирована на поддержку когнитивных (познавательных) потребностей разработчиков. Эти познавательные потребности описаны в тех когнитивных теориях:

1) рефлексия-в-действии (reflection-in-action);
2) оппортунистический дизайн (opportunistic design);
3) понимание и решение проблем (comprehension and problem solving).

Первый выпуск ArgoUML стал доступен в 1998 году. С настоящее время программа ArgoUML стала популярной в образовательной и коммерческой сферах. Узнать больше о проекте Арго можно, используя ссылки http://argouml.tigris.org и http://argouml.tigris.org/servlets/ProjectMailingListList

Методика и порядок выполнения работы

Изучить теоретическое обоснование. Дополнительно можно изучить материалы о программе ArgoUML на сайте открытой энциклопедии Википедии. Для установки программы ArgoUML на домашнем компьютере можно бесплатно в Internet скачать ArgoUML для Windows, без регистрации и отправки СМС. Для установки можно использовать файл ArgoUML-0.34-setup.exe, который предоставляется вместе с электронной версией методических указаний к лабораторным работам.

Программа ArgoUML работает на операционных системах:

- Windows 7 x64

- Windows 7
- Windows Vista x64
- Windows Vista
- Windows 2003
- Windows XP x64
- Windows XP
- Windows 2000
- Mac OS X .

Задание 1. Установить программу ArgoUML на том компьютере, который вы будете использовать для выполнения лабораторных работ. Если программа уже установлена, то убедитесь, что ее версия v0.34 (рисунок 1.1).

При загрузке выводится одно из шести окон:

• Splash. (установлено по умолчанию). Выводит логотип ArgoUML и номер текущей версии (рисунок 8.1).

• Version. Выводит информацию о текущей версии, различных пакетах ArgoUML , некоторые сведения об операционной системе и среде разработки.

• Credits. Содержит информацию о создателях ArgoUML, включая детали о разработчиках отдельных модулей.

• Contact Info. Выводит контактную информацию об ArgoUML, адрес web-сайта.

• Report bugs. Выводит сообщения об ошибках.

После загрузки программы картинка (Splash), представленная на рисунке 1.1, исчезнет. Вместо нее появится интерфейс пользователя, описанный в лабораторной работе 2.

Рисунок 1.1 – Вид экрана во время загрузки ArgoUML

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из описания основных этапов установки программы.

Вопросы для защиты работы

1. Что такое «лицензионное соглашение»?
2. Какие виды лицензионных соглашений вы знаете?
3. Какие условия установки ArgoUML используются?

Лабораторная работа № 2
ИЗУЧЕНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ ARGOUML

Цель: изучить интерфейс программы ArgoUML и ее основные команды.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть

После загрузки программы ArgoUML на экране монитора появляется главное окно, показанное на рисунке 2.1. В верхней части экрана находится *строка меню.* Если при установке в качестве рабочего языка был выбран русский, то наименования команд в строке меню будет таким же, как на рисунке. Меню содержит интуитивно понятные команды, например, в меню Файл можно сохранить проект или открыть другой проект.

По умолчанию главное окно программы разделено на 4 части или панели. Размеры панелей можно изменять, перетаскивая влево, вправо, вверх и вниз границу между ними. В нижней части окна находится строка *состояния.* По часовой стрелке от верхнего левого угла находятся навигационная панель, панель редактирования, область сведений (панель деталей) и «To-Do» области.

1. Верхний левый угол. Навигационная панель показывает древовидную структуру всех классов, интерфейсов, типов данных и объектов. Эта структура может быть адаптирована к потребностям пользователя путем фильтрации объектов, которые можно перетаскивать на диаграммы.

2. В правом верхнем углу находится панель редактирования, которая показывает текущую диаграмму. Объекты на диаграмму можно перетаскивать из навигационной панели, или создавать с помощью команд меню, при наведении курсора на выбранный объект.

3. Внизу слева на панели «To do» выводится список всех выполненных пользователем действий для рассматриваемой модели.

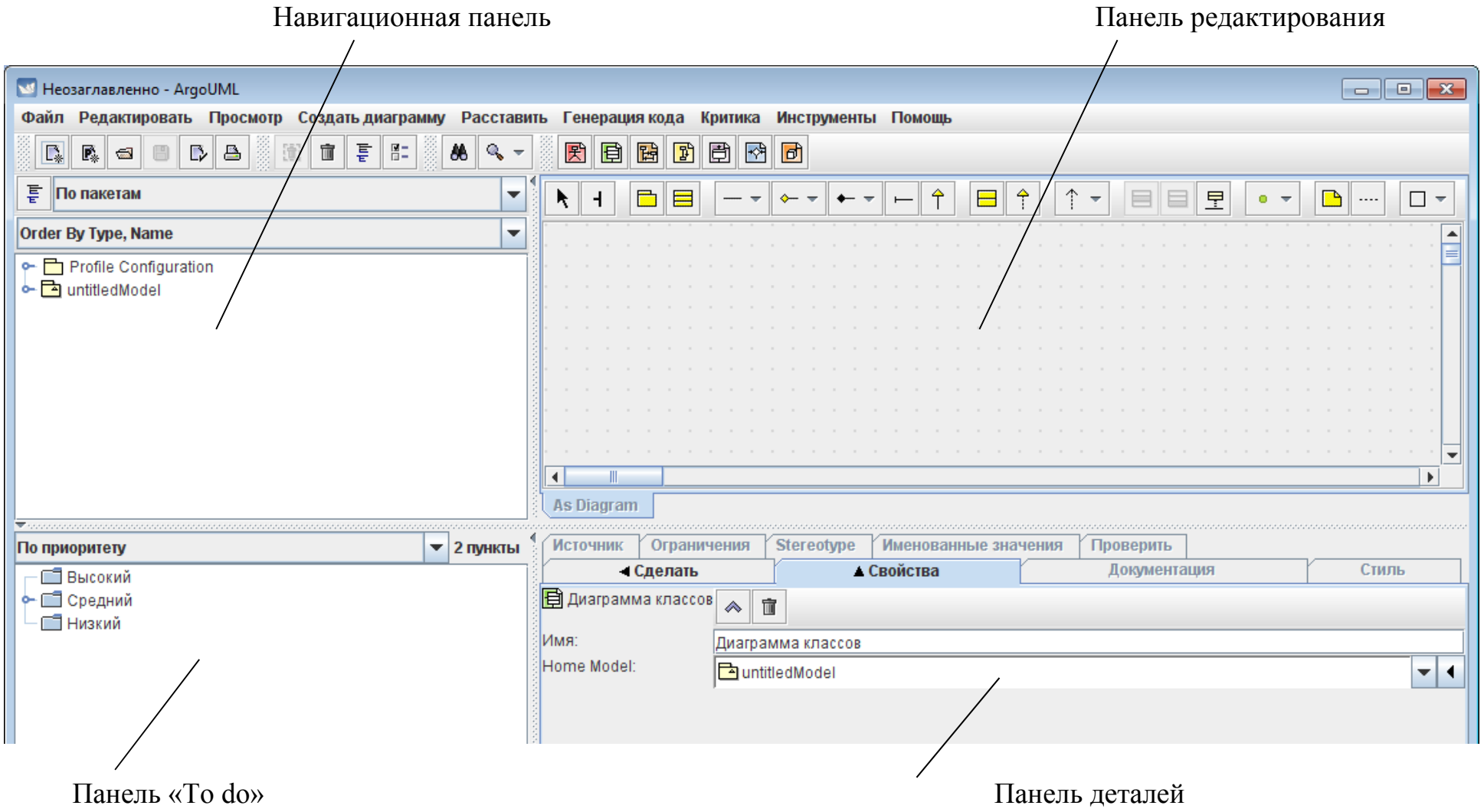


Рисунок 2.1 – Интерфейс пользователя ArgoUML

4. Внизу справа на панели деталей содержится подробная информация о выбранном объекте (актер, связь, прецедент).

Команды меню «Файл» представлены в таблице 2.1.

Таблица 2.1 - Команды меню «Файл»

Команда	Описание
	Создать новый проект
	Открыть проект
	Сохранить проект
	Установка параметров страницы: размера, ориентации, других опций
	Стандартное диалоговое окно для выбора параметров печати для текущей диаграммы

Команды меню «Редактировать» (The Edit Menu) представлены в таблице 2.2. Это меню позволяет пользователю выбирать элементы и размещать их на текущей панели редактирования, удалять элементы и управлять пользовательскими установками.

Таблица 2.2 - Команды меню «Редактировать»

Команда	Описание
Select All (Ctrl-A)	Выбор всех элементов модели на текущую панель.
 Navigate Back (Вернуться назад)	Перемещение к предыдущему элементу. Если его нет, то команда не активна.
 Navigate Forward (Продвинуться вперед)	Отмена команды . Перемещение к следующему элементу. Если его нет, то команда не активна.
Invert Selection	Инвертирует текущее выделение на текущей панели. Все выбранные и не выбранные элементы на текущей диаграмме меняются местами
	Удаление выбранного элемента из текущей диаграммы. В модели элемент остается.
 (Ctrl-Delete)	Полное удаление выбранного элемента из модели. Если удаляемый элемент присутствует на других диаграммах, то появляется диалоговое окно (рисунок 2.2).
	Конфигурация (Сконфигурировать перспективы)
 Settings... (Установки)	Открывает диалоговое окно для выбора параметров установки (рисунок 2.3).

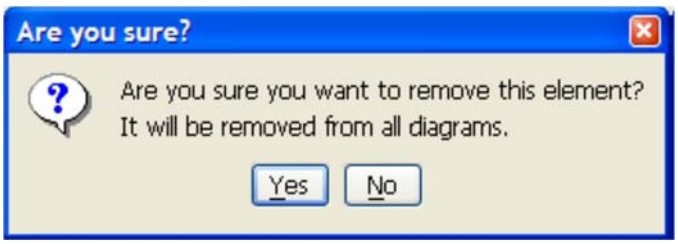


Рисунок 2.2 – Диалоговое окно для согласования полного удаления элемента из всех диаграмм

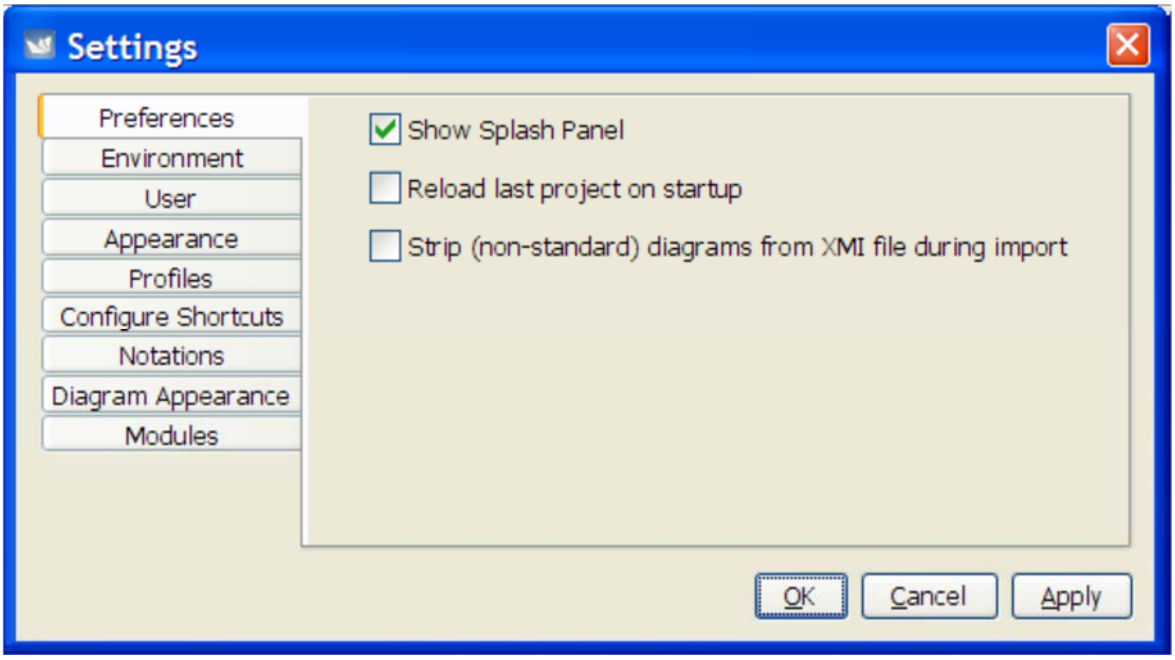


Рисунок 2.3 – Диалоговое окно для выбора параметров установки

Команды меню «Просмотр» (The View Menu) используется для просмотра различных диаграмм. Они представлены в таблице 2.3.

Таблица 2.3 - Команды меню «Просмотр»

Команда	Описание
Goto Diagram... (Перейти к диаграмме)	Открывает немодальное диалоговое окно, в котором перечисляются все диаграммы текущего проекта
Find... (Найти…)	Открывает немодальное диалоговое окно для поиска
Zoom (Масштаб)	Изменяет масштаб. Подменю содержит команды: • `Zoom Out`. Shortcut (Ctrl-Minus). Уменьшает размер и увеличивает обзор. • `Zoom Reset`. Возвращает размер к установленному по умолчанию (т.е. 100%). • `Zoom In`. Shortcut (Ctrl-Plus). Увеличивает масштаб.
Adjust Grid (Настройка	Позволяет регулировать параметры решетки на панели регулирования:

решетки)	• Линии 16: решетка из сплошных линий с размером клеток 16 пикселей. • Линии 8: решетка из сплошных линий с размером клеток 8 пикселей. • Точки 16: решетка из точек с размером клеток 16 пикселей (по умолчанию). • Точки 32: решетка из точек с размером клеток 32 пикселя. • Решетка отсутствует.
Adjust Snap (Регулировка оснастки)	Позволяет регулировать привязку к сетке на расстоянии в пикселях: • Snap 4 • Snap 8 (по умолчанию). • Snap 16 • Snap 32
Adjust PageBreaks	Настройка разрыва страниц
Toolbars	Позволяет скрывать или выводить на экран команды. По умолчанию все команды показаны.
XML Dump	Выводит контент текущего проекта в XML формате.

Меню «Создать диаграмму» (The Create Menu) используется для создания различных UML диаграмм. Список диаграмм выводится, если нажать на команду «Создать диаграмму». Они представлены в таблице 2.4.

Таблица 2.4 - Команды меню «Создать диаграмму»

Команда	Описание
	Команда для создания диаграммы прецедентов (вариантов использования)
	Команда для создания диаграммы классов
	Команда для создания диаграммы последовательностей
	Команда для создания диаграммы кооперации (collaboration)
	Команда для создания диаграммы деятельности (Activity Diagram)
	Команда для создания диаграммы размещения (Deployment Diagram)

Меню «Расставить» используется для управления размещением элементов на рабочей панели. Команды меню «Расставить» представлены в таблице 2.5.

Таблица 2.5 - Команды меню «Расставить»

Команда	Описание
Выровнять	Команда для выравнивания имеет подменю:

(Align)	• Выравнивание по верхней границе. • Выравнивание по нижнему краю • Выравнивание по правому краю • Выравнивание по левому краю • Выравнивание по центру. Все выбранные элементы размещаются симметрично относительно вертикальной линии, проходящей через центры элементов • Все выбранные элементы размещаются симметрично относительно горизонтальной линии, проходящей через центры элементов • Выравнивание по сетке. Все вершины выбранных элементов и их правые края выравниваются по границам сетки
Распределить (Distribute)	Команда для управления размещением элементов по горизонтали и вертикали имеет подменю: • Размещение элементов, при котором крайний элемент слева и крайний элемент справа не перемещаются, а находящиеся между ними элементы перемещаются таким образом, что расстояние между краями элементов вдоль горизонтали устанавливается одинаковым; • Размещение элементов, при котором крайний элемент слева и крайний элемент справа не перемещаются, а находящиеся между ними элементы перемещаются таким образом, что расстояние между центрами элементов вдоль горизонтали устанавливается одинаковым; • Размещение элементов, при котором крайний элемент сверху и крайний элемент снизу не перемещаются, а находящиеся между ними элементы перемещаются таким образом, что расстояние между краями элементов вдоль вертикали устанавливается одинаковым; • Размещение элементов, при котором крайний элемент сверху и крайний элемент снизу не перемещаются, а находящиеся между ними элементы перемещаются таким образом, что расстояние между центрами элементов вдоль вертикали устанавливается одинаковым.
Переложить (Reorder)	Команда для управления упорядочиванием элементов (на переднем или заднем плане) имеет подменю: •`Forward` , выбранный элемент перемещается вперед (на передний план) относительно соседнего; •`Backward`, выбранный элемент перемещается назад (на задний план) относительно соседнего; •`To Front`, выбранный элемент перемещается вперед (на передний план) по отношению ко всем элементам;

	•To Back, выбранный элемент перемещается назад (на задний план) по отношению ко всем элементам.
Size To Fit Contents	Команда устанавливает минимальный размер текстовых надписей таким образом, чтобы они поместились внутри элементов на диаграмме

Панель «To-do» расположена в нижней части главного окна слева (рисунок 9.1).

Щелчок справа от команды «Средний» выводит сообщение о том, что задания имеют средний приоритет (рисунок 2.4).

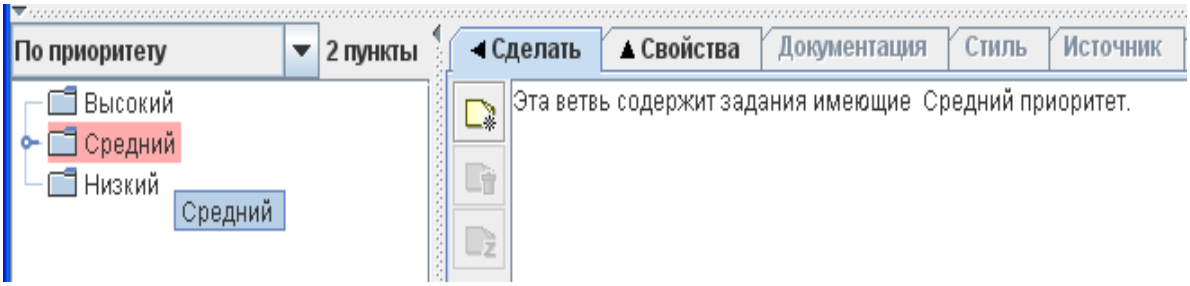


Рисунок 2.4 – Панель «To-do» (слева) и панель деталей (справа)

Щелчок слева от элемента «Средний» в окрестности ключа выводит два элемента: Add Elements to Package untitledModel (Добавить элементы в пакет неозаглавленной модели) и Revise Package Name untitledModel (Переименовать пакет неозаглавленной модели). Они показаны на рисунке 2.5.

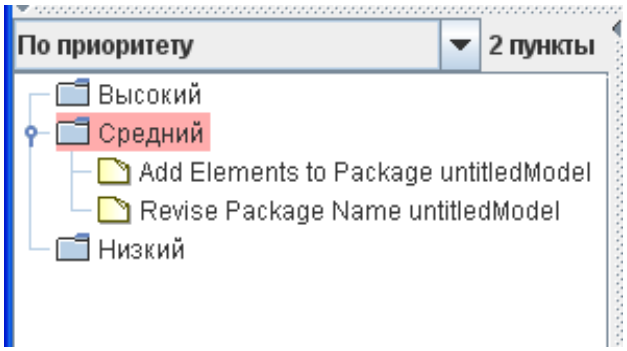


Рисунок 2.5 – Панель «To-do»

Чтобы дать имя новому еще неозаглавленному пакету необходимо щелкнуть на элементе Revise Package Name untitledModel на панели «To-do», а затем выбрать команду «Следующий» на панели деталей. В результате на панели деталей появится диалоговое окно (рисунок 2.6), в котором в области «Имя:» можно ввести название пакета. Чтобы сохранить введенное имя необходимо нажать клавишу «Закончить» на панели деталей.

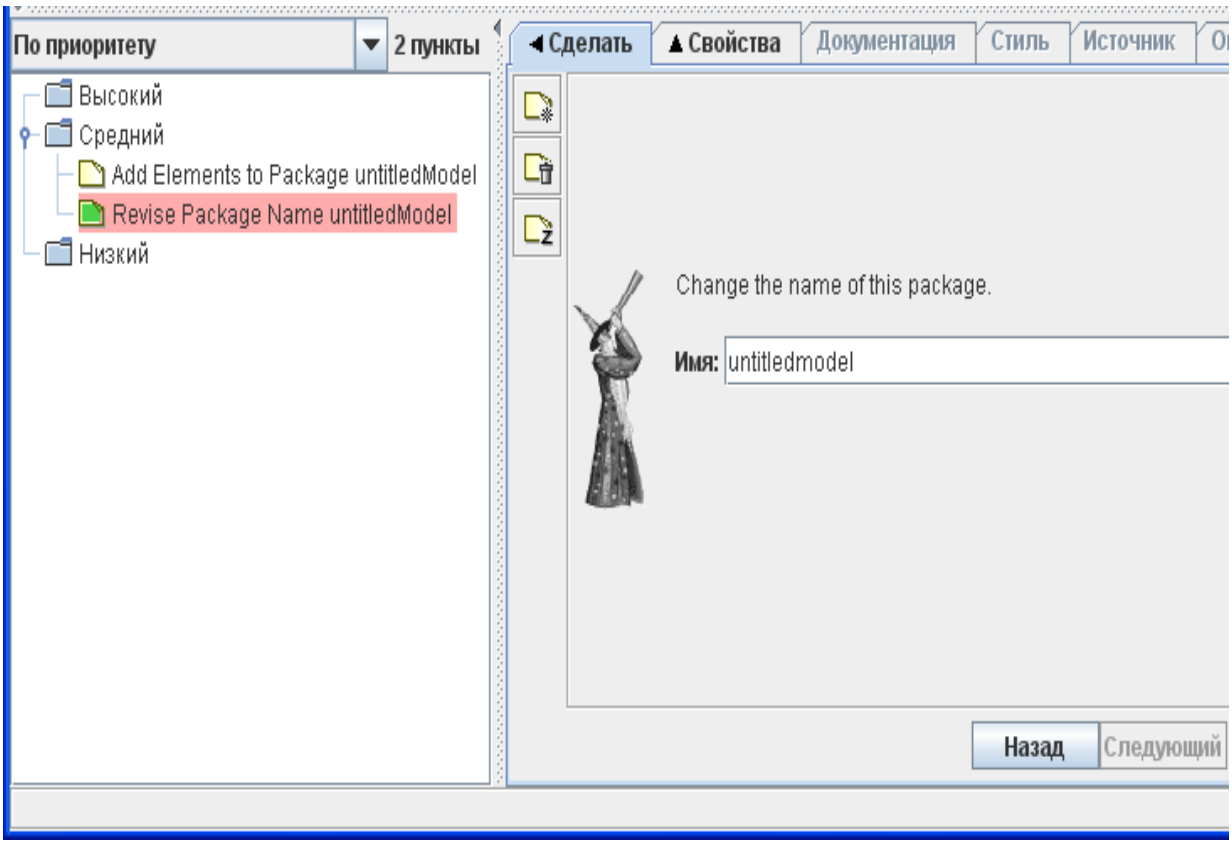


Рисунок 2.6 – Диалоговое окно, предназначенное для ввода имени нового пакета

Задание 1. Изучить интерфейс программы ArgoUML и ее основные команды.

Задание 2. Повторить принципы объектно-ориентированного представления программных систем.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из описания основных панелей программы ArgoUML.

Вопросы для защиты работы

1. Какие известные стандартные команды реализованы в ArgoUML?
2. Какие диаграммы UML можно построить с помощью ArgoUML?
3. Можно ли с помощью ArgoUML получить описание контента в виде гипертекста?

Лабораторная работа № 3
ДИАГРАММЫ ПРЕЦЕДЕНТОВ (ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ)

Цель: изучить порядок построения диаграммы прецедентов с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Термин «Use Cases» применительно к диаграммам UML переводится на русский язык как «Прецеденты» или «Варианты использования». В литературе на русском языке встречаются оба названия, соответствующие «Use Cases».

После загрузки программы ArgoUML выводится главное окно, имеющее вид, показанный на рисунке 2.1, где на панели редактирования размещены инструменты для построения диаграммы классов. Чтобы перейти к инструментам для создания диаграммы прецедентов необходимо среди команд меню «Создать диаграмму» (таблица 2.4) выбрать команду для создания диаграммы прецедентов (вариантов использования). При этом на панели редактирования появятся инструменты диаграммы прецедентов (вариантов использования), показанные на рисунке 3.1.

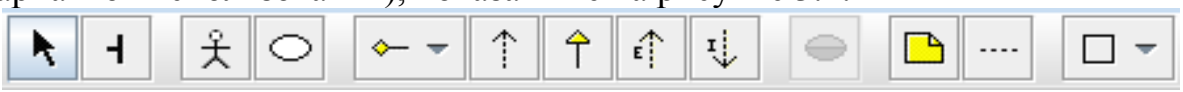


Рисунок 3.1 – Панель инструментов диаграммы прецедентов

По умолчанию однократное нажатие на кнопку с изображением инструмента приводит к его однократному использованию. Для многократного использования инструмента используется двойной щелчок или клавиша CTRL одновременно с однократным щелчком. Инструменты диаграммы прецедентов описаны в таблице 3.1.

Таблица 3.1 - Инструменты диаграммы прецедентов

Инструмент	Описание
 Select	Отменяет выбор инструмента для многократного использования
 Brooom	(Метла). Перемещает сразу несколько элементов на диаграмме. Используется, чтобы быстро выстроить элементы диаграммы вдоль выбранной линии. Для включения можно использовать клавишу ALT.
	Актер. После однократного нажатия на кнопку можно добавить одного актера. После двойного щелчка или использования CTRL одновременно с однократным щелчком можно добавлять элемент на диаграмму многократно.
	Прецедент (вариант использования).
	Точка расширения. Активна только при выборе прецедента. Можно отредактировать двойным щелчком и

	использованием клавиатуры.
	Инструмент для добавления комментария к элементу диаграммы.
	Инструмент для связывания комментария с элементом диаграммы.

Инструмент для рисования показан на рисунке 3.2. При нажатии на кнопку  появляется всплывающее меню, содержащее пиктограммы инструментов, описанные в таблице 3.2.

Рисунок 3.2 – Всплывающее меню для рисования

Таблица 3.2 - Инструменты для рисования диаграммы прецедентов

Инструмент	Описание
	Прямоугольник.
	Прямоугольник с закругленными краями.
	Овал.
	Линия.
	Текст.
	Многоугольник.
	Открытый сплайн. Последнюю точку отмечает двойной щелчок.
	Множество линий, проведенных через выбранные точки. Линии могут пересекаться.

Виды ассоциаций, поддерживаемые ArgoUML, показаны на рисунке 3.3. Выбрать вид ассоциации с указанием направления или без указания (простая ассоциация, агрегация или композиция) можно после нажатия на кнопку  , расположенную справа от символа ассоциации «–». Если затем нажать на одну из пиктограмм, показанных на рисунке 3.3 и не соответствующих символу «–», то выбранный символ появится рядом с кнопкой выбора вместо символа ассоциации «–». Тогда на диаграмме прецедентов будет отображаться тот вид ассоциации, который выбран. Линию следует проводить от зависимого элемента к актеру. Инструменты диаграммы прецедентов для выбора ассоциаций описаны в таблице 3.3.

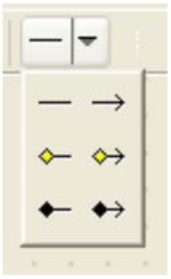


Рисунок 3.3 – Инструмент для выбора типа ассоциации ArgoUML

Таблица 3.3 - Инструменты для выбора ассоциаций диаграммы прецедентов

Инструмент	Описание
	Зависимость
	Обобщение. Проводится от ребенка к родителю.
	Расширение. Проводится от дополнительного элемента к основному.
	Включение

Для построения диаграммы прецедентов рассмотрим некоторые виды работы, выполняемой деканатом. Изобразим на диаграмме актера, щелкнув один раз на изображении актера на панели инструментов, а затем на панели редактирования. Получим изображение актера внутри прямоугольника – рисунок 3.4. Сразу, не перемещая указатель мыши за пределы прямоугольника, дадим актеру название: «деканат», набрав его на клавиатуре. Созданный на панели редактирования элемент (в данном случае актер) появляется и в древовидной структуре навигационной панели.

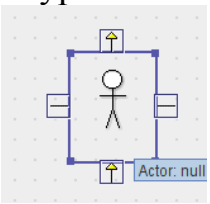


Рисунок 3.4 – Добавление актера на диаграмму прецедентов

Затем добавим прецеденты (варианты использования) и ассоциации. Эти элементы добавляются так же, как и актеры с помощью панели инструментов (рисунок 3.1). Полученная диаграмма прецедентов показана на рисунке 3.5. Присваивать имена диаграммам, элементам диаграмм, а также редактировать названия элементов можно в окне «Имя» на панели деталей. Добавлять элементы на панель редактирования можно и без использования панели инструментов диаграммы прецедентов, которая показана на рисунке 3.1. Если подходящие элементы (актеры, прецеденты и ассоциации) были ранее созданы для данной диаграммы или для других диаграмм текущего проекта, то они перечислены на навигационной панели. На текущую диаграмму их можно скопировать, перетаскивая мышью на панель редактирования с навигационной панели. Выделение элемента диаграммы на панели

редактирования сопровождается его выделением на навигационной панели и на панели деталей.

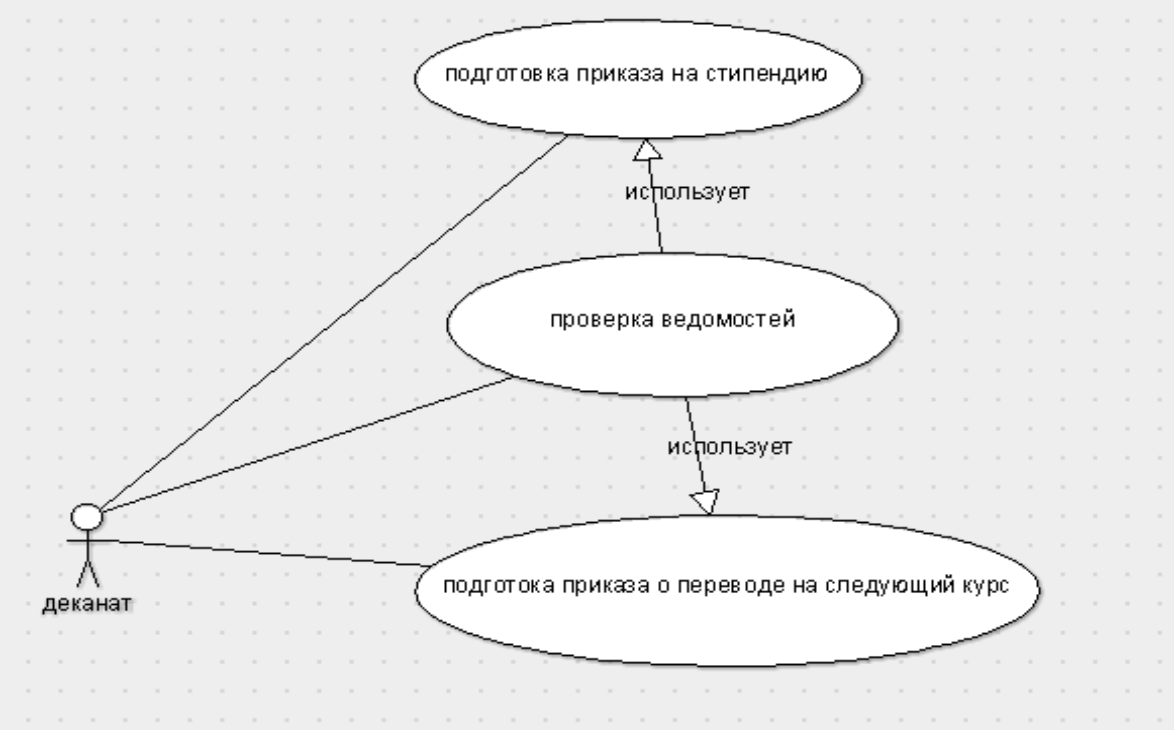


Рисунок 3.5 – Диаграмма прецедентов

На рисунке 3.6 на панели редактирования показана диаграмма прецедентов для студента, на которой выделен прецедент «выполнение задания». Имя этого элемента одновременно выделено на навигационной панели и отображено в окне «Имя» на панели деталей.

Прецедент «выполнение задания» связан связями «расширение» с прецедентами, которые определяют дополнительные варианты испол

Задание 1. Необходимо изучить предметную область, для варианта, согласованного с преподавателем. Выделите актеров (не менее трех) и прецеденты. Для них постройте диаграмму прецедентов.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать диаграмму прецедентов и описание использованных ассоциаций ArgoUML.

Вопросы для защиты работы:

1. Какие команды реализованы в панели инструментов диаграммы прецедентов ArgoUML?
2. Какие виды ассоциаций можно показать на диаграмме прецедентов с помощью ArgoUML?
3. Каким образом можно изменить имя элемента на диаграмме прецедентов?

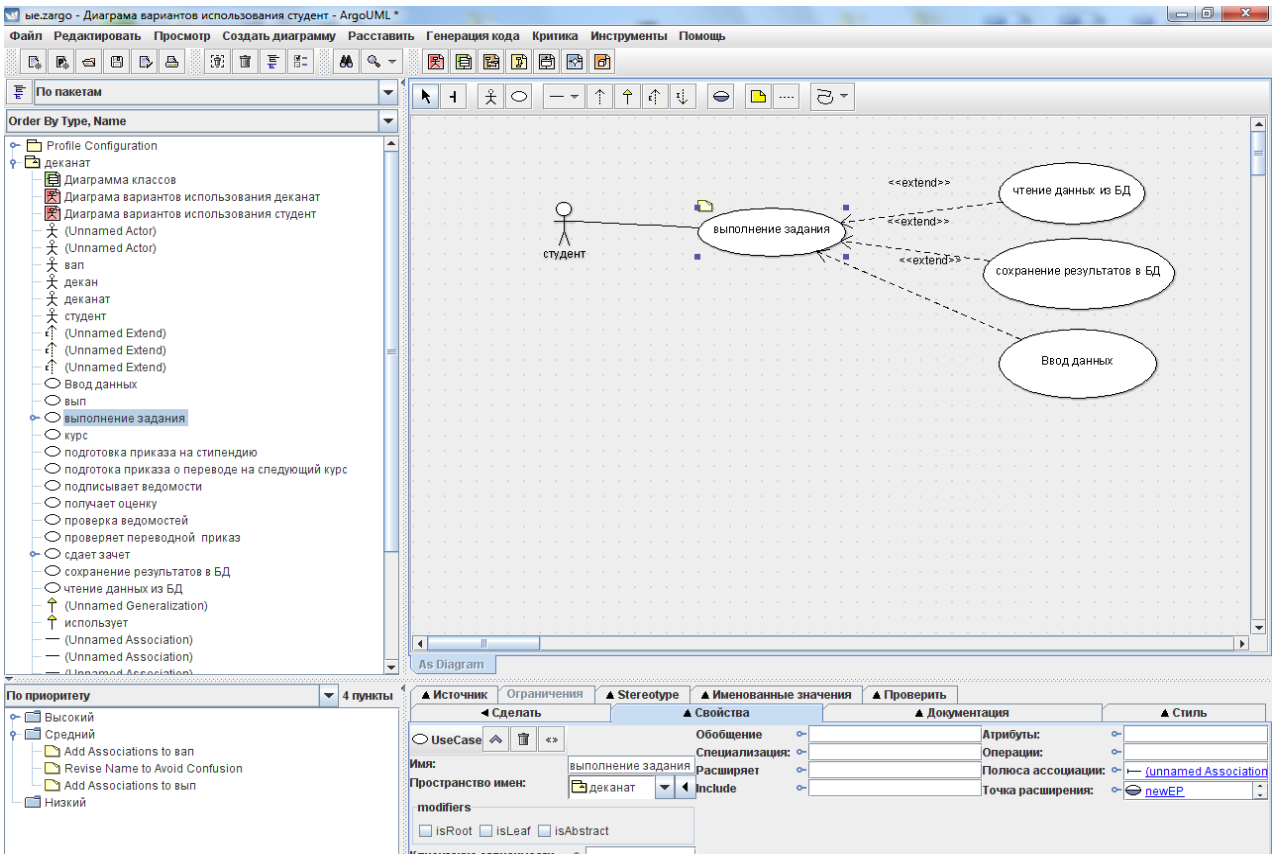


Рисунок 3.6 – Выделение прецедента «выполнение задания» на диаграмме прецедентов

РАЗДЕЛ 2. ПРАКТИКИ СИСТЕМНОЙ ИНЖЕНЕРИИ

Лабораторная работа № 4
ДИАГРАММЫ КЛАССОВ

Цель: изучить порядок построения диаграммы классов с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Инструменты для построения диаграммы классов (на панели редактирования) показаны на рисунке 4.1. Описание инструментов для построения диаграммы классов приведено в таблице 4.1.

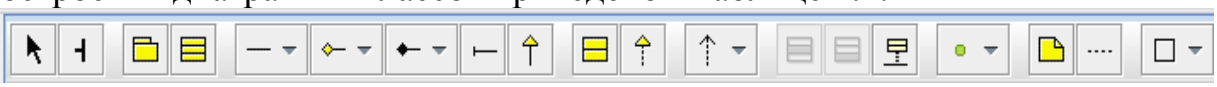


Рисунок 4.1 – Инструменты для построения диаграммы классов

Таблица 4.1 - Инструменты диаграммы классов

Инструмент	Описание
	Пакет
	Класс
	Ассоциация. Может быть двунаправленной.
	Агрегация. Может быть двунаправленной.
	Композиция. Может быть двунаправленной.
	Конец ассоциации. Для добавления другого конца к уже существующей ассоциации (с помощью левой кнопки проводится от центра ассоциации к классу или наоборот). Применяется для создания N-арных ассоциаций.
	Обобщение. Проводится от ребенка к родителю.
	Добавление интерфейса на диаграмму. Для удобства, когда указатель мыши находится над выбранным интерфейсом, на нем отображается ручка, за которую можно перетаскивать элемент
	Реализация. Добавляет реализацию между классом и интерфейсом с помощью перемещения левой кнопки мыши. Проводится от класса к интерфейсу.
	Зависимость. Добавляет зависимость между двумя элементами с помощью перемещения левой кнопки мыши. Проводится от зависимого элемента. Существует 2 типа зависимости: разрешение (по умолчанию) и использование.

	Атрибут. Элемент активен только, когда выбранный элемент – класс.
	Операция. По умолчанию получает имя newOperation, которое можно редактировать.
	Association Class. Ассоциированный класс.
 Datatype.	Тип данных. Для удобства, когда указатель мыши находится над выбранным элементом, появляются ручки наверху и в основании, по которым можно щелкнуть или тянуть, чтобы сформировать необходимый элемент. Есть 2 доступных элемента - Перечисление и Стереотип.

Рассмотрим пример, связанный с деятельностью деканата. Анализ предметной области приведет нас к выделению следующих классов:

1. Факультет.
2. Кафедра.
3. Курс.
4. Семестр.
5. Куратор.
6. Студент.
7. Секретарь.

Определение атрибутов

Следующий шаг состоит в определении атрибутов (свойств) каждого класса. Например, класс «Студент» может иметь свойства: полное имя, дата рождения и номер паспорта. Класс «Курс» может иметь свойства: Код, объем часов. Каждый атрибут будет иметь свой тип данных, который определяет, как данные будут храниться в нем, строковые (символьные) или числовые.

Определение функций (операций)

Функции (операции) представляют собой действия, которые может выполнить сущность, например: студент может зарегистрироваться на дополнительный курс или на ДПВ (дисциплину по выбору). Каждая функция может иметь несколько входных параметров и возвращать только один объект.

Определение отношений между классами (ассоциации)

Для любой предметной области необходимо проанализировать возможные отношения между сущностями. Для предметной области «Деканат» существуют следующие отношения между сущностями: в состав факультета входит много кафедр и за каждой кафедрой закреплено более одного курса. Эти отношения можно представить ассоциациями с помощью языка UML.

Ассоциация определяет связь между двумя классами, причем на каждом конце ассоциации определяется кратность. Например, тип отношения факультет – кафедра соответствует связи 1:M, что изображается на диаграмме как "1 .. *

Построение диаграммы классов

Чтобы добавить новый класс нужно нажать на пиктограмму  на панели инструментов, которая показана на рисунке 4.1. После этого указатель мыши покажет фигуру, имеющую форму крестика, которую можно перемещать по панели редактирования. После перемещения курсора (в форме крестика) в то место, где нужно разместить новый класс, необходимо щелкнуть левой кнопкой. После этого на панели редактирования появится фигура, изображающая класс (рисунок 4.2). Красной волнистой линией подчеркнуто место, предназначенное для ввода имени класса. Класс разделен на три части: верхняя часть будет содержать имя класса, средняя часть будет содержать атрибуты класса, нижняя часть – операции (или функции класса).

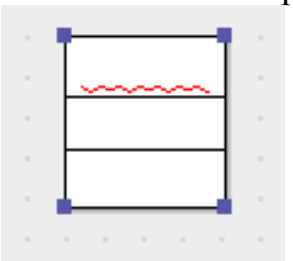


Рисунок 4.2 – Добавление нового класса на диаграмму

Щелкнув указателем на красной волнистой линии можно добавить на диаграмму имя класса. Имена атрибутов можно добавить с помощью в помощью всплывающего меню, показанного на рисунке 4.3. Тип атрибута можно выбрать из ниспадающего списка на панели деталей.

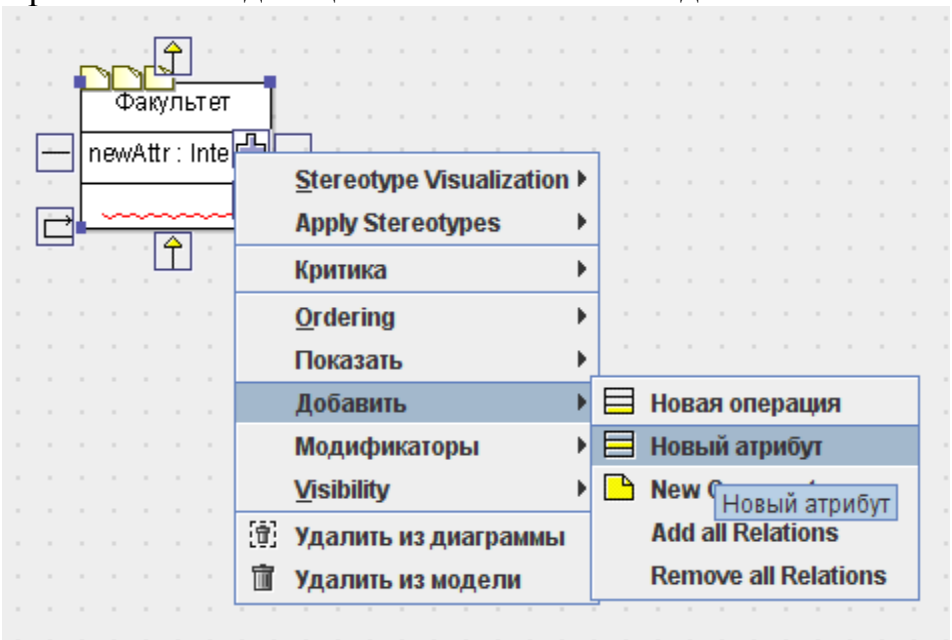


Рисунок 4.3 – Добавление атрибутов класса на диаграмму

Добавление новой операции показано на рисунке 4.4. Имя и тип параметра операции можно указать в круглых скобках.

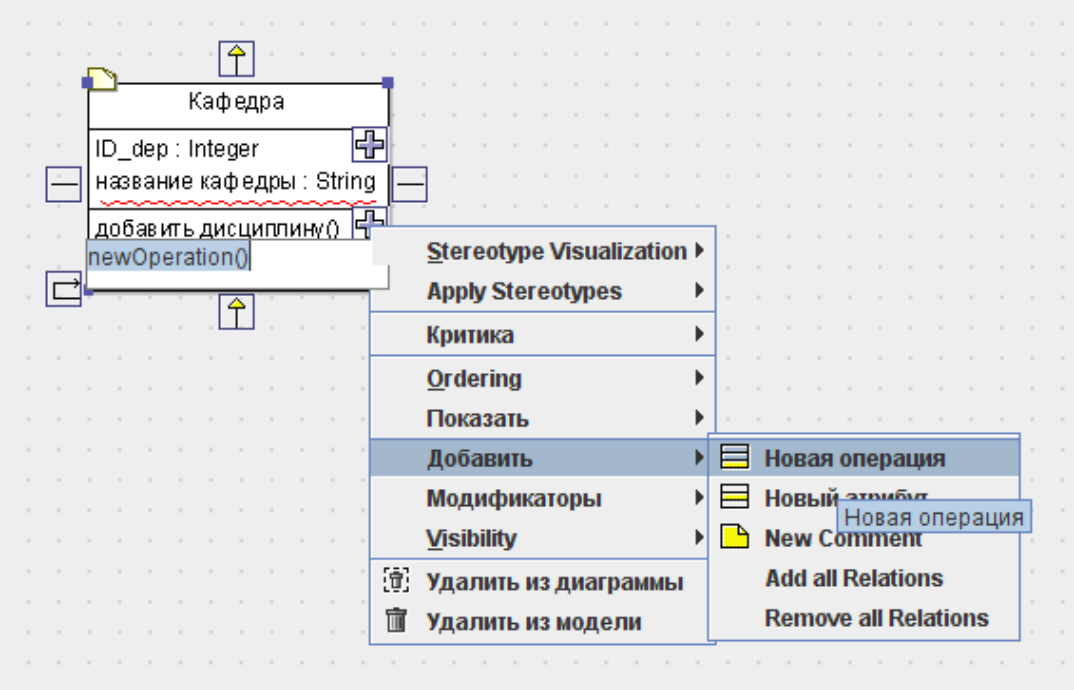


Рисунок 4.4 – Добавление новой операции на диаграмму классов

После создания двух классов можно добавить ассоциацию с команды меню и таких же действий, как и при добавлении ассоциаций на диаграмму прецедентов. Указание кратности (множественности) ассоциации показано на рисунке 4.5. Возможные виды кратности показаны на ниспадающем меню команды «Множественность».

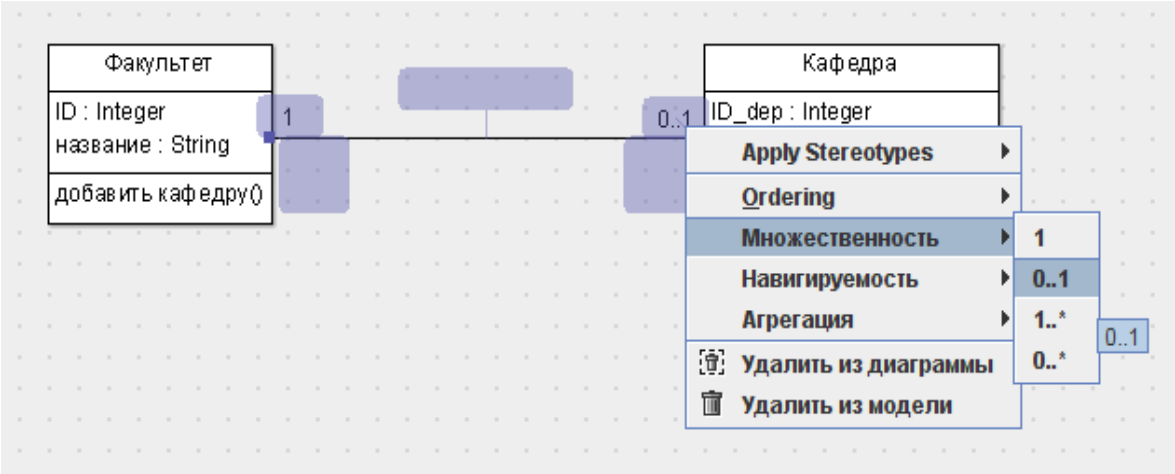


Рисунок 4.5 – Указание кратности ассоциации на диаграмме классов

Чтобы указать кратность ассоциации с другой стороны необходимо подвести курсор к выделенному на границе ассоциации цветному прямоугольнику, дважды щелкнуть левой кнопкой мыши и ввести с клавиатуры. Полученная диаграмма должна выглядеть как на рисунке 4.6.

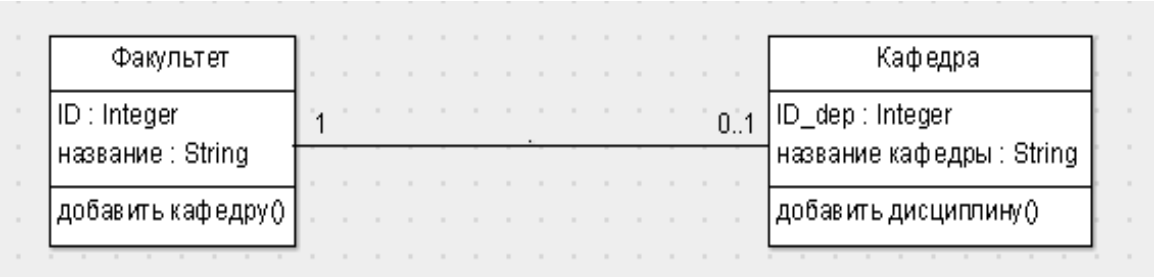


Рисунок 4.6 – Диаграмма классов

Задание 1. Необходимо изучить предметную область, соответствующую вашему варианту задания. Выделите классы (не менее трех), опишите их атрибуты и операции. Для них постройте диаграмму классов.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать диаграмму классов и описание использованных ассоциаций ArgoUML.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы классов ArgoUML?
2. Какие виды ассоциаций можно показать на диаграмме классов с помощью ArgoUML?
3. Каким образом можно изменить имя элемента на диаграмме классов?
4. Каким образом можно добавлять атрибуты и операции на диаграмму классов?

Лабораторная работа № 5
ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ

Цель: изучить порядок построения диаграммы последовательности с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

На диаграмме последовательности изображаются только те объекты, которые непосредственно участвуют во взаимодействии, обмениваясь сообщениями (вызывая методы) в определенной хронологической последовательности. Их используют для моделирования специфических, ограниченных по времени, ситуаций. Диаграммы последовательностей специально выделяют порядок и времена отсылки сообщений объектам. Ключевым моментом для диаграмм последовательности является динамика взаимодействия объектов во времени. В качестве объектов на диаграммах последовательности в ArgoUML можно использовать как актеров, так и экземпляры классов. Внешний вид диаграммы последовательности показан на рисунке 5.1.

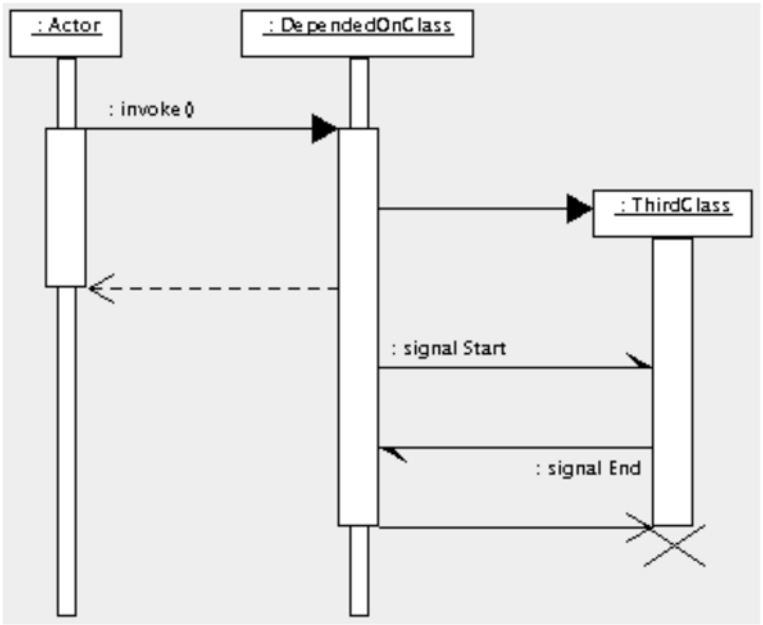


Рисунок 5.1 – Диаграмма последовательности

В UML на диаграмме последовательности крайним слева изображается объект, который является инициатором взаимодействия. Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности размещаются по порядку слева направо в соответствии с очередностью или степенью активности при взаимодействии объектов друг с другом. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни (вертикальная линия). Внутри прямоугольника записываются имя объекта [:имя класса]. Временная ось направлена сверху вниз. Масштаб на оси времени не указывается, так как диаграмма последовательности

моделирует лишь хронологическую упорядоченность взаимодействий типа «раньше-позже». Сообщения, посылаемые от одного объекта к другому, отображаются стрелками с указанием операции и параметров.

Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и может участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности от самой верхней ее части до самой нижней. Линия жизни изображается тонкой (или штриховой) линией, когда объект пассивен, и жирной линией, если объект активен.

Отдельные объекты, выполнившие свою работу в системе, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML используется специальный символ в форме латинской буквы «X». Ниже этого символа линия жизни не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий.

Не обязательно создавать все объекты диаграммы в начальный момент времени. Отдельные объекты могут создаваться по мере необходимости, экономя ресурсы системы и повышая ее производительность. В этом случае прямоугольник объекта изображается не в верхней части диаграммы последовательности, а в той части, которая соответствует моменту создания объекта. При этом прямоугольник объекта вертикально располагается в том месте диаграммы, которое по оси времени совпадает с моментом его создания в системе. Объект обязательно создается со своей линией жизни.

Панель инструментов для построения диаграммы последовательности показана на рисунке 5.2, а описание отдельных инструментов приведено в таблице 5.1.

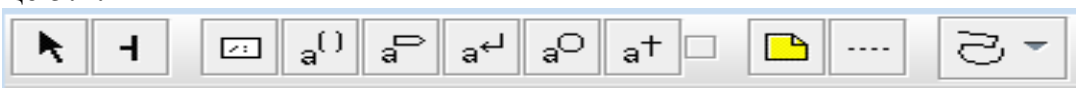


Рисунок 5.2 – Инструменты для построения диаграммы последовательности

Таблица 5.1 – Описание инструментов диаграммы последовательности

Инструмент	Описание
	Добавление нового объекта на диаграмму
	Двухсторонние сообщения о вызове процедур, выполнении операций
	Сообщение об отправке сообщения (одностороннее)
	Сообщение о возврате из вызова процедуры. Примером может служить простое сообщение о завершении некоторых вычислений без предоставления результата расчетов объекту-клиенту.

	Действие создания
	Действие уничтожения

В качестве примера для предметной области, связанной с работой деканата, построим диаграмму последовательности для варианта использования «Записать студента на изучение ДПВ» - рисунок 5.3.

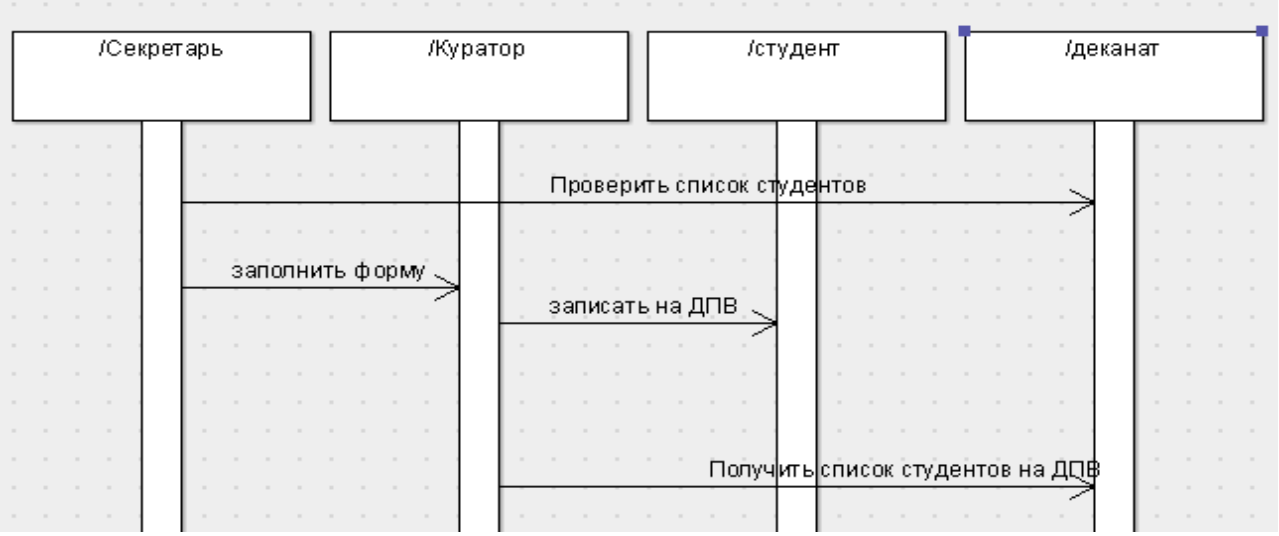


Рисунок 5.3 – Диаграмма последовательности

Задание 1. Необходимо изучить предметную область, соответствующую вашему варианту задания, которое представлено в таблице 1.1. Выделите классы (не менее трех), актеров. Опишите объекты и последовательность их взаимодействия с помощью диаграммы последовательности (для двух вариантов использования).

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать диаграммы последовательности для двух вариантов использования.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы последовательности ArgoUML?
2. Какие виды сообщений можно показать на диаграмме последовательности?
3. Каким образом отображается линия жизни объекта?

Лабораторная работа № 6
ДИАГРАММЫ СОТРУДНИЧЕСТВА (КООПЕРАЦИИ)

Цель: изучить порядок построения диаграммы сотрудничества с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграммы последовательности и кооперации (collaboration) отображают взаимодействие элементов моделируемой предметной области. Диаграммы последовательности представляют временные аспекты взаимодействия и хронологическую последовательность действий, а диаграммы кооперации представляют структурные аспекты взаимодействия. На диаграмме кооперации изображаются только те отношения между объектами, которые играют определенные роли во взаимодействии. Представление структуры системы как совокупности взаимодействующих объектов на диаграмме кооперации соответствует статическому описанию системы.

Классификатор (Classifier) – это механизм, описывающий структурные и поведенческие свойства. Классификаторами называются те элементы моделирования, которые могут иметь экземпляры. Структурные свойства классификаторов характеризуются с помощью атрибутов, а поведенческие свойства выражаются в форме операций. Все экземпляры одного классификатора обладают рядом общих свойств. К числу классификаторов относятся классы, интерфейсы, типы данных, сигналы, компоненты, узлы, прецеденты и подсистемы. Несмотря на различие классификаторов между собой в ArgoUML на диаграммах последовательности и кооперации классификаторы изображаются с помощью одной и той же пиктограммы – прямоугольника. Строка текста в прямоугольнике должна иметь следующий формат:

/ <Имя роли классификатора>: <Имя классификатора>

В лабораторной работе 5 классификатор описан как объект.

Кооперация может быть представлена на двух уровнях:

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

Диаграмма кооперации уровня спецификации показывает роли, которые играют участвующие во взаимодействии элементы. Элементами кооперации на этом уровне являются классы и ассоциации, которые обозначают отдельные роли классификаторов и ассоциации между участниками кооперации. Пример кооперации показан на рисунке 6.1.

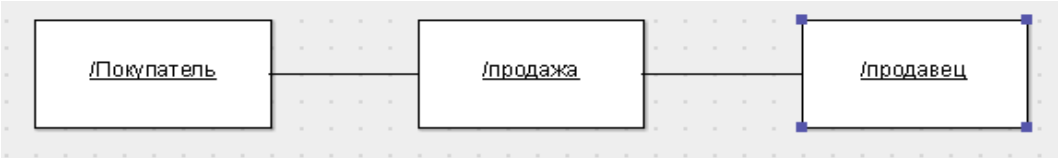


Рисунок 6.1 – Диаграмма кооперации

Панель инструментов для построения диаграммы кооперации показана на рисунке 6.2, а описание отдельных инструментов приведено в таблице 6.1.

Рисунок 6.2 – Инструменты для построения диаграммы кооперации

Таблица 6.1 – Описание инструментов диаграммы кооперации

Инструмент	Описание
	Роль классификатора
	Роль ассоциации
	Обобщение
	Зависимость
	Добавить сообщение

На рисунке 6.3 показано отношение обобщения между отдельными кооперациями уровня спецификации.

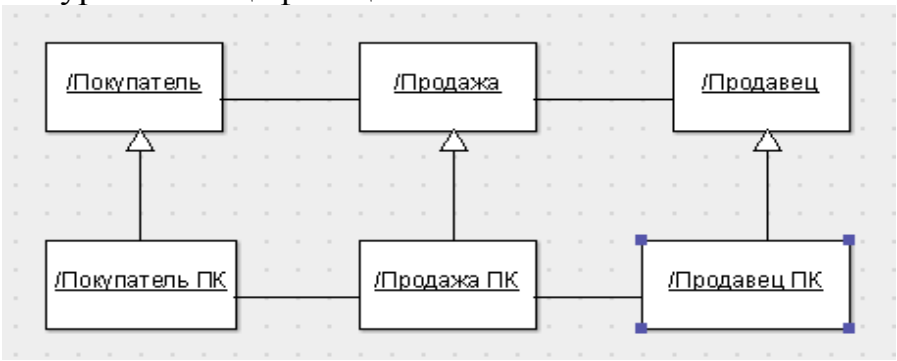


Рисунок 6.3 – Диаграмма кооперации с отношениями обобщения

Отличительной особенностью диаграммы кооперации является присутствие на ней сообщений. Сообщения уже рассматривались ранее при изучении диаграммы последовательности в лабораторной работе 5. Сообщения показаны на рисунках 5.1 и 5.3 направленными стрелками. При построении диаграммы кооперации они имеют некоторые дополнительные семантические особенности. Сообщение на диаграмме кооперации специфицирует коммуникацию между двумя объектами, один из которых передает другому некоторую информацию. При этом первый объект ожидает, что после получения сообщения вторым объектом последует выполнение некоторого действия. Таким образом, именно сообщение является причиной

или стимулом для начала выполнения операций, отправки сигналов, создания и уничтожения отдельных объектов. Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю. Сообщения в языке UML также специфицируют роли, которые играют объекты - отправитель и получатель сообщения. Сообщения на диаграмме кооперации изображаются помеченными стрелками рядом (выше или ниже) с соответствующей связью или ролью ассоциации. Направление стрелки указывает на получателя сообщения. Сообщениям на диаграмме кооперации в среде ArgoUML автоматически присваиваются номера. На рисунке 6.4 показана диаграмма кооперации с сообщением.

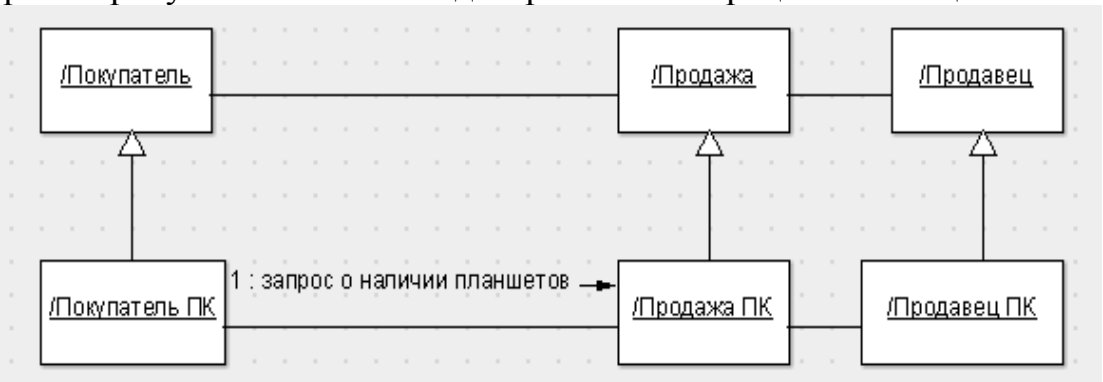


Рисунок 6.4 – Диаграмма кооперации

Задание 1. Построить диаграмму кооперации для той же предметной области, для которой в лабораторной работе 5 была построена диаграмма последовательности. Сравнить диаграммы последовательности и кооперации.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать диаграммы кооперации для двух примеров.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы кооперации ArgoUML?
2. Какие виды сообщений можно показать на диаграмме кооперации?
3. Что общего у диаграмм последовательности и кооперации?
4. Чем отличаются диаграммы последовательности и кооперации?

Лабораторная работа 7. ДИАГРАММЫ СОСТОЯНИЙ

Цель: изучить порядок построения диаграммы состояний с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграммы состояний отображают динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия, совершаемые пользователями или дрёугими системами, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то поведение модели является асинхронным.

Диаграммы состояний можно использовать для описания поведения не только экземпляров классов (объектов), но и для описания других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний представляет собой граф, описывающий поведение некоторого автомата - математической абстракции, используемой для моделирования детерминированного поведения технических объектов или объектов реального мира.

Понятие автомата использовано и в языке UML. Вершинами графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из одного состояния в другое состояние. Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели. Переход объекта из состояния в состояние рассматривается как мгновенный. Каждое последующее состояние всегда наступает позже предыдущего. Последовательность состояний автомата охватывает все этапы его жизненного цикла, от создания до уничтожения. Простейший пример диаграммы состояний для технического устройства (телефон, компьютер) показан на рисунке 7.1.

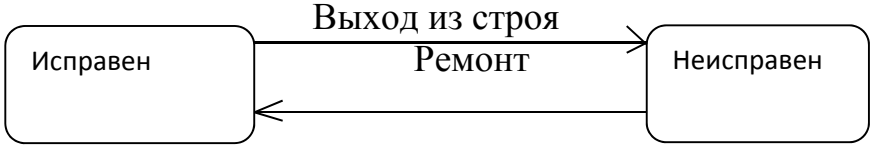


Рисунок 7.1 – Диаграмма состояний технического устройства

Понятие «автомат (state machine)» в языке UML основано на выполнении следующих обязательных условий:

1. Автомат не запоминает историю перемещения из состояния в состояние. Однако в ArgoUML имеются средства, которые дополняют возможности описания предыстории поведения объекта на основе введения в рассмотрение так называемых исторических состояний с помощью

инструментов и , которые описаны в таблице 7.1.

2. В каждый момент времени автомат может находиться в одном и только в одном из своих состояний. Причем, если не происходит никаких событий, то автомат может находиться в отдельном состоянии как угодно долго.
1. Переходы из одного состояния в другое являются мгновенными, а длительность нахождения автомата в том или ином состоянии, а также время достижения того или иного состояния никак не специфицируются. Поэтому время на диаграмме состояний присутствует в неявном виде. Однако в ArgoUML имеются средства, которые отмечают синхронные состояния с помощью инструмента , описанного в таблице 7.1.
4. Количество состояний автомата должно быть обязательно конечным.
5. Граф автомата не должен содержать изолированных состояний и переходов. Это условие означает, что для каждого из состояний, кроме начального, должно быть определено предшествующее состояние. Каждый переход должен обязательно соединять два состояния автомата. Допускается переход из состояния в себя, такой переход еще называют «петлей».
6. Автомат не должен содержать конфликтующих переходов, т. е. таких переходов из одного и того же состояния, когда объект одновременно может перейти в два и более последующих состояния (кроме случая параллельных подавтоматов).

Инструменты для построения диаграммы состояний показаны на рисунке 7.2, а их описание приведено в таблице 7.1.

Рисунок 7.2 – Инструменты для построения диаграммы состояний

Таблица 7.1 – Описание инструментов диаграммы состояний

Инструмент	Описание
	Простое состояние
	Композитное (составное) состояние
	Новый переход (New Tansition)
	Состояние синхронизации
Submachine State	Состояние – ссылка на вложенный автомат
	Состояние -заглушка

Stub State	
	Начальное состояние
	Конечное состояние
 Junction	Ветвление. Используется для расщепления одного входящего перехода на несколько исходящих. Выполняется тот переход, для которого в момент перехода сторожевое условие принимает значение «истина». Предопределенное условие, обозначаемое как «else», можно определить только для одного перехода, который выполняется, если для других переходов сторожевые условия принимают значение «ложь».
	Выбор (Choice). Выбор используется для расщепления входящего перехода на несколько исходящих. Выполняется тот переход, для которого в момент перехода сторожевое условие принимает значение «истина». Если для нескольких переходов условие принимает значение «истина», то выбор выполняемого перехода производится случайно. Предопределенное условие, обозначаемое как «else», можно определить только для одного перехода, который выполняется, если для других переходов сторожевые условия принимают значение «ложь».
	Разветвление
	Соединение
	Недавнее историческое состояние (shallow history state)
	Давнее историческое состояние (deep history state). Используется для запоминания всех подсостояний любого уровня вложенности для текущего подавтомата.
	Вызов события как триггера для перехода. Существует 4 типа вызовов: событие вызова; событие изменения; событие сигнала и событие времени (таблица 1.2).
	Сторожевое условие (guard condition) записывается в квадратных скобках после события-триггера и представляет собой логическое 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.
	Вызов действия, которое приводит к переходу из одного состояния в другое. Существует 7 видов: действие вызова; действие создания; действие уничтожения; действие возврата; действие отправки; неинтерпретируемое действие; новое последовательное действие (таблица 7.3)

Ниспадающее меню для выбора событий вызова появляется при наведении курсора на пиктограмму  . Оно содержит 4 варианта событий вызова, которые описаны в таблице 7.2.

Таблица 7.2 - Инструменты для выбора событий вызова

Инструмент	Описание
	Событие вызова
	Событие изменения
	Событие сигнала
	Событие времени

Ниспадающее меню для выбора варианта вызова действия появляется при наведении курсора на пиктограмму  . Оно содержит 7 вариантов, которые описаны в таблице 7.3.

Таблица 7.3 - Инструменты для выбора вызова действия

Инструмент	Описание
	Действие вызова
	Действие создания
	Действие уничтожения
	Действие возврата
	Действие отправки
	Конечное состояние
	Неинтерпретируемое действие
	Новое последовательное действие

Состояние (state) характеризуется значением таких свойств элементов системы, которые отражают динамический или функциональный аспект ее поведения. Чтобы добавить новое простое состояние нужно нажать на пиктограмму  на панели инструментов, которая показана на рисунке 7.2. После перемещения курсора на панели редактирования в то место, где нужно разместить новое состояние, необходимо щелкнуть левой кнопкой. После этого на панели редактирования появится фигура, изображающая простое состояние (рисунок 7.3). Красной волнистой линией подчеркнуто место, предназначенное для ввода имени состояния. Состояние разделено горизонтальной линией на две части: верхняя часть будет содержать имя состояния, нижняя часть – список его внутренних действий.

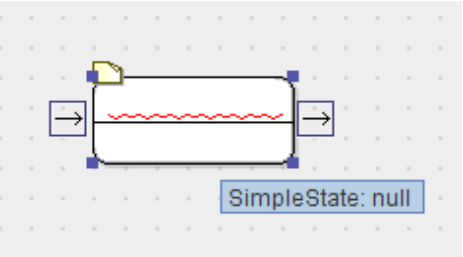


Рисунок 7.3 – Добавление нового простого состояния на диаграмму

После двойного щелчка указателем на красной волнистой линии можно добавить на диаграмму имя состояния. Имена внутренних действий: entry (входное), do (выполняемое), exit (выходное) можно добавить с помощью инструментов «Действие при входе», «Действие при выходе» и «Деятельность выполнения», расположенных на панели деталей, – рисунок 7.4. Тип действий можно выбрать из ниспадающего списка на панели деталей после нажатия на пиктограмму  .

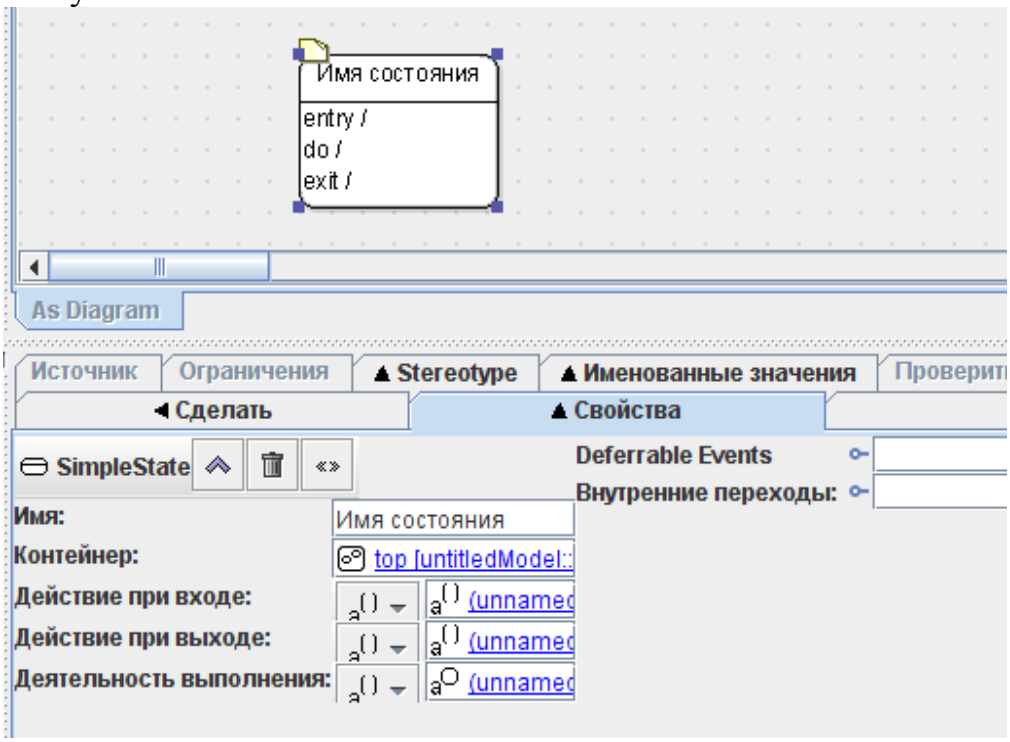


Рисунок 7.4 – Добавление списка внутренних действий простого состояния

Названия внутренних действий вводятся после двойного щелчка в области списка внутренних действий в нижней части прямоугольника, изображающего простое состояние, на панели редактирования.

Пример диаграммы состояний для примера, описывающего работу электронной почты, представлен на рисунке 7.5.

На диаграмме (рисунок 7.5) показано начальное состояние, соответствующее пиктограмме  , три простых состояния «Активизация почтовой программы», «Ввод пароля», «Загрузка почты» и конечное состояние, соответствующее пиктограмме  .

Пиктограммой  на диаграмме изображен инструмент (ветвление), описанный в таблице 7.1. Он использован для расщепления перехода, исходящего от состояния «Ввод пароля», на два перехода. Один из них, направленный к состоянию «Загрузка почты», выполняется, если сторожевое условие [пароль введен верно] принимает значение «истина». Этот переход на диаграмме обозначен [guard]. Предопределенное условие, обозначаемое как «else», выполняется, если сторожевое условие [пароль введен верно] принимает значение «ложь».

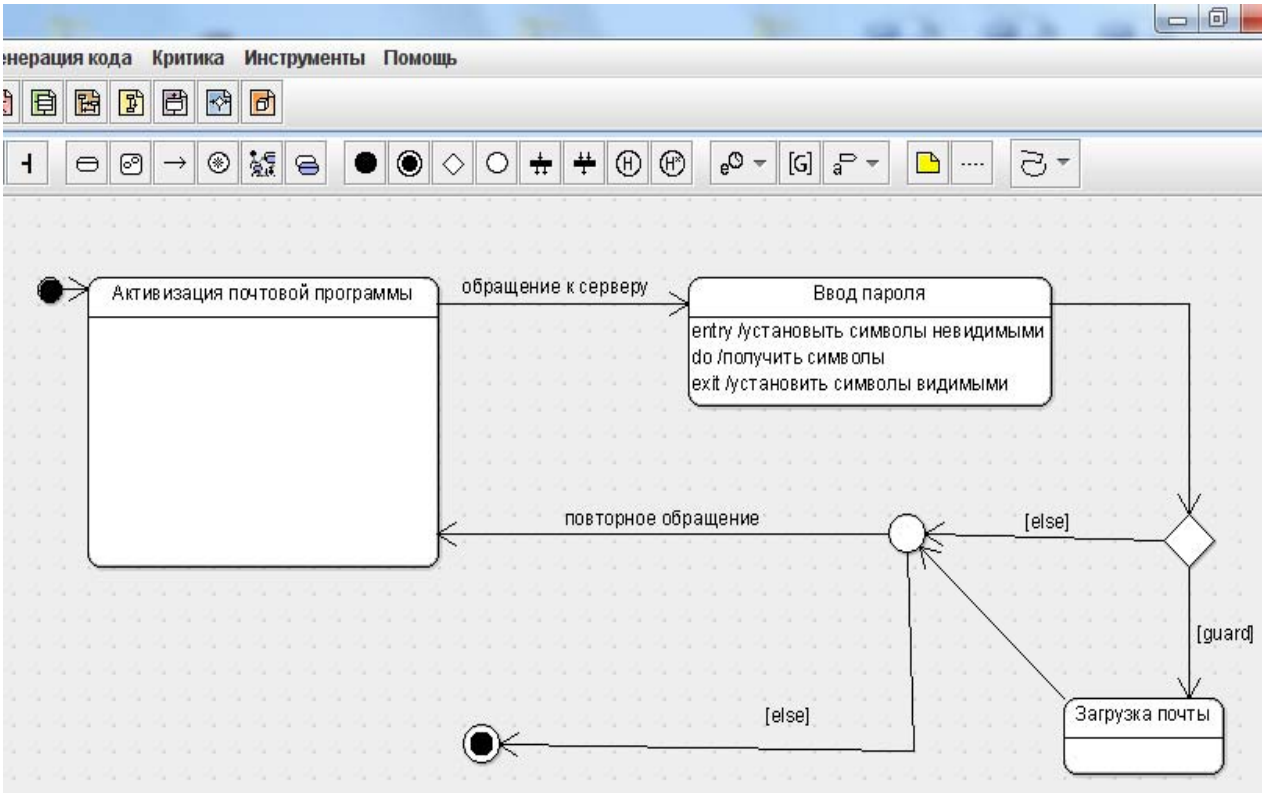


Рисунок 7.5 – Диаграмма состояний

Пиктограммой «Выбор»  на диаграмме изображен инструмент, описанный в таблице 7.1. Он использован для выбора либо повторного обращения к почтовой программе, либо для выхода.

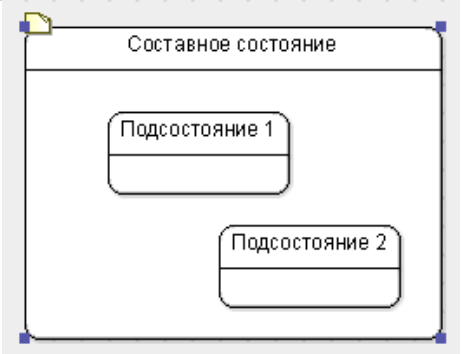


Рисунок 7.6 – Составное состояние

Составное состояние (composite state) создается с помощью

инструмента, описанного в таблице 7.1. Это такое состояние, внутри которого находятся вложенные в него подсостояния (substate). Между составным состоянием и подсостояниями имеет место отношение композиции.

Подсостояния могут быть последовательными и параллельными. Чтобы построить на диаграмме параллельные подсостояния, необходимо область внутри составного состояния разделить горизонтальной штриховой линией на горизонтальные полосы, т.е. выделить новый конкурентный участок. Для этого поместив курсор в область, занятую прямоугольником составного состояния, нужно щелкнуть правой кнопкой мыши и в ниспадающем меню (рисунок 7.7) выбрать команду «New Concurrent Region».

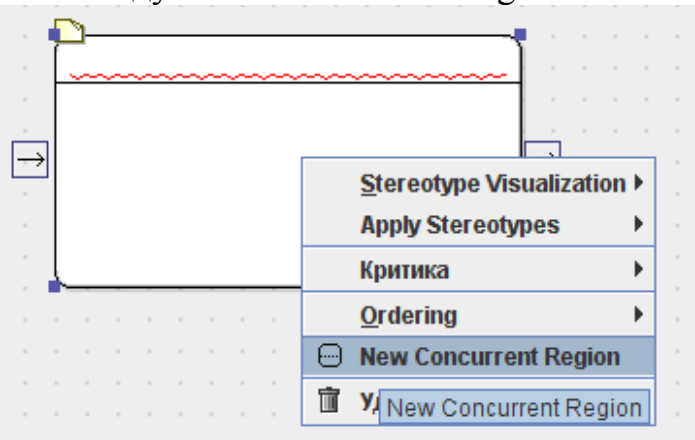


Рисунок 7.7 – Выбор команды «New Concurrent Region»

На рисунке 7.8 область внутри составного состояния разделена штриховой линией на два параллельных процесса.

Параллельные подсостояния (concurrent substates) позволяют специфицировать два и более подавтомата, которые могут выполняться параллельно внутри составного события. Параллельные подсостояния в свою очередь могут состоять из последовательных. Например, Подсостояние 1 и Подсостояние 2 на рисунке 7.8 являются последовательными подсостояниями подавтомата 1. Для подавтомата 2 последовательные подсостояния – это 3 и 4.

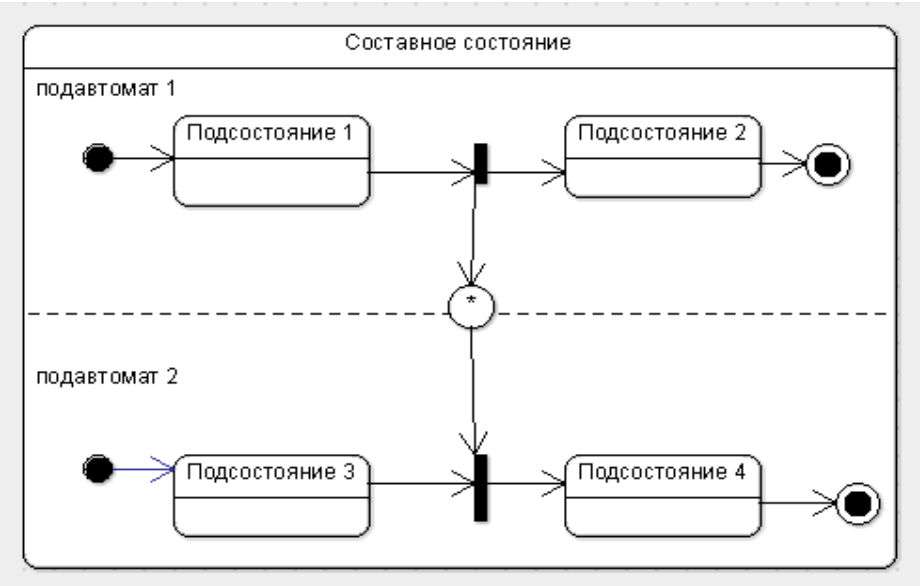


Рисунок 7.8 – Диаграмма состояний, содержащая параллельные подавтоматы

Для учета синхронизации наступления отдельных событий в языке UML имеется специальное псевдосостояние, которое называется синхронизирующим состоянием. На диаграмме показано, что переходы между Подсостоянием 1 и Подсостоянием 2 подавтомата 1 и между Подсостоянием 3 и Подсостоянием 4 подавтомата 2 синхронизированы (символ  на пересечении перехода и штриховой линии).

Задание 1. Необходимо изучить предметную область, соответствующую вашему варианту задания. Выделите состояния, характерные для предметной области, проанализируйте возможные переходы между состояниями и события, которые приводят к переходам состояний. Проанализируйте возможность реализации составных состояний с последовательными и параллельными переходами. Постройте диаграмму состояний.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно построенную диаграмму состояний.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы состояний ArgoUML?
2. Какие виды состояний можно показать на диаграмме состояний с помощью ArgoUML?
3. Каким образом можно разделить переход на несколько исходящих?

Лабораторная работа № 8
ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ

Цель: изучить порядок построения диаграммы деятельности с помощью ArgoUML.

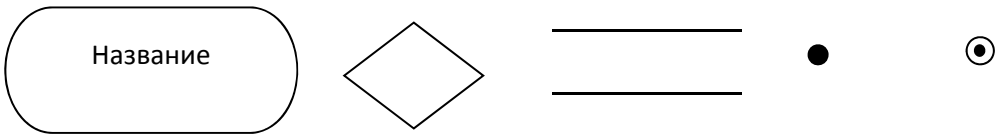
Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграммы деятельности (activity diagram) отображают особенности алгоритмической и логической реализации выполняемых системой операций. При структурном подходе к разработке информационных систем для отражения алгоритмических решений традиционно использовались блок-схемы алгоритмов.

Под *деятельностью* в данном случае понимают задачу (операцию), которую необходимо выполнить вручную или с помощью средств автоматизации. Каждому варианту использования соответствует своя последовательность задач. В теоретическом плане диаграммы деятельности являются обобщенным представлением алгоритма, реализующего анализируемый вариант использования. На диаграмме деятельность обозначается прямоугольником с закругленными углами (рисунок 8.1, а).

Диаграммы деятельностей позволяют описывать альтернативные и параллельные процессы. Для обозначения альтернативных процессов используют ромб (рисунок 8.1, б), условие указывают над ним слева или справа, а альтернативы «да», «нет» - рядом с соответствующими выходами. С помощью этого же блока можно построить циклический процесс. Множественность активации деятельности обозначают символом «*», помещенным рядом со стрелкой активации деятельности, и при необходимости уточняют надписью вида «для каждой строки». Для обозначения параллельных процессов используют линейки синхронизации (рисунок 8.1, в), причем условие синхронизации можно уточнить, указав его на диаграмме.



а б в г д

Рисунок 8.4 – Условные обозначения диаграммы деятельности:
а – деятельность; б – выбор; в – линейная синхронизация; г – начало; д – конец

На рисунке 8.2 показано, что «Деятельность 1» и «Деятельность 2» могут выполняться параллельно. На этапе определения спецификаций имеет смысл уточнять только варианты использования, краткое описание которых недостаточно для понимания сущности решаемых проблем. Диаграммы

деятельностей, таким образом, можно использовать вместо описания вариантов использования или как дополнение к ним.

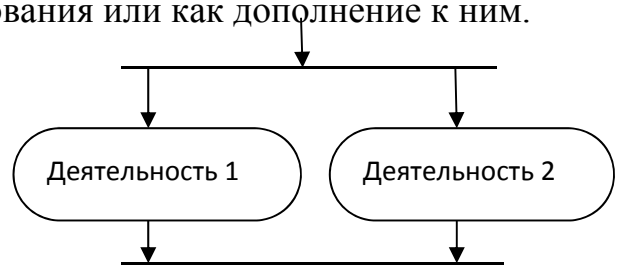


Рисунок 8.2 – Диаграмма деятельности с указанием параллельности процессов

Диаграммы деятельности описывают последовательность выполняемых операций и отображают особенности процедурного и синхронного управления. Их можно считать частным случаем диаграмм состояний. В ArgoUML панель инструментов для построения диаграммы деятельности (рисунок 8.3) имеет сходство с панелью инструментов для построения диаграммы состояний. На обеих диаграммах присутствуют обозначения состояний, переходов, сторожевого условия, ветвления, начального и конечного состояний. Отличие заключается в семантике состояний, которые используются для представления действий. Каждое состояние деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние на диаграмме деятельности происходит после завершения операции в предыдущем состоянии.

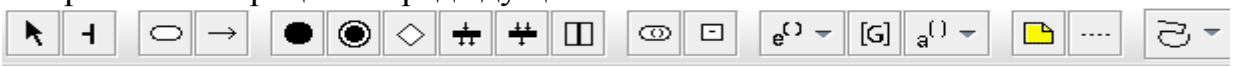


Рисунок 8.3 – Панель инструментов для построения диаграммы деятельности

В таблице 8.1 описаны только те инструменты для построения диаграммы деятельности, которые отличаются от инструментов для построения диаграммы состояний.

Таблица 8.1 – Инструменты для построения диаграммы деятельности

Инструмент	Описание
	Новое состояние деятельности
	Дорожки (swimlanes). Применяется для разделения состояния действия на диаграмме деятельности на отдельные группы, которые отделяются друг от друга вертикальными линиями.
	Вызов состояния
	Объект перехода между состояниями

В зависимости от степени детализации диаграммы деятельностей так же, как диаграммы классов, используют на разных этапах разработки. На этапе анализа требований и уточнения спецификаций диаграммы деятельностей позволяют конкретизировать основные функции разрабатываемого программного обеспечения.

Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. Простейшая диаграмма деятельности показана на рисунке 8.4.

Рисунок 8.4 – Диаграмма деятельности

Состояние деятельности изображается фигурой, ограниченной параллельными горизонтальными линиями и боковыми дугами. Для добавления нового состояния деятельности используется инструмент с пиктограммой  .

Для представления параллельных процессов на диаграммах деятельности предусмотрены специальные инструменты: «Разветвление»  и «Соединение»  . На рисунке 8.5 показана диаграмма деятельности, на которой параллельно выполняются процессы, соответствующие третьему и четвертому состояниям действиям.

Другим средством ArgoUML для представления параллельных процессов является инструмент «Дорожки» (swimlanes)  . Он позволяет моделировать работу предприятий (бизнес-процессы), предполагающую взаимодействие между отделами предприятия.

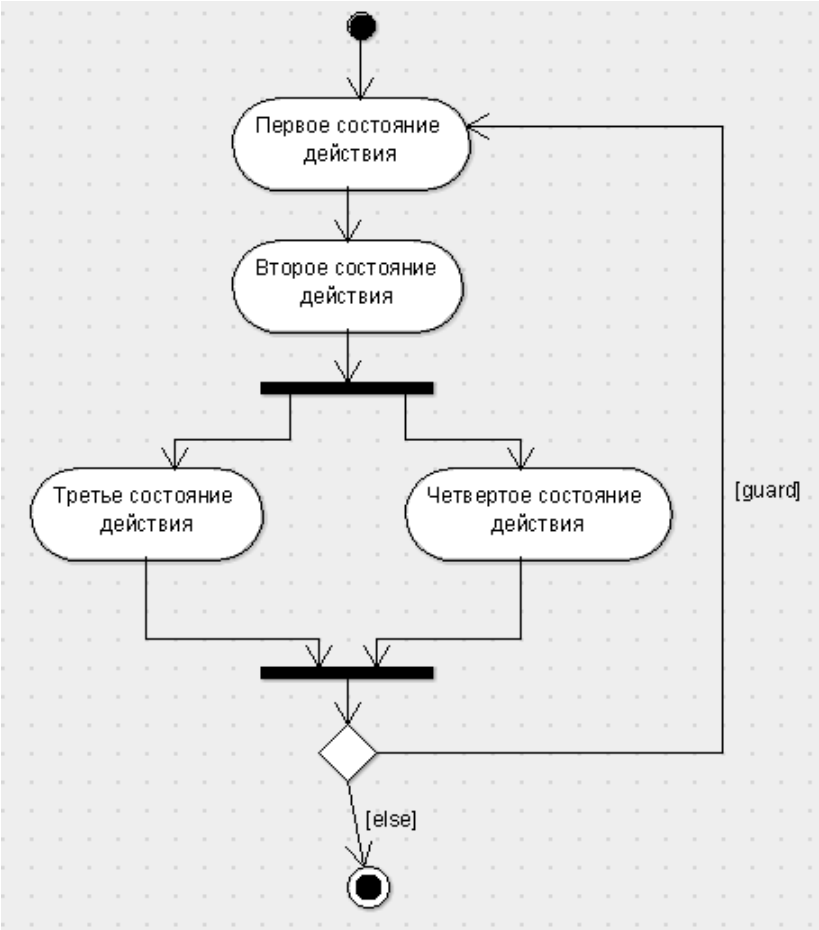


Рисунок 8.5 – Диаграмма деятельности с последовательным и параллельным выполнением операций

На рисунке 8.6 показана диаграмма деятельности, в которой использован инструмент «Дорожки» для моделирования работы двух отделов предприятия.

Инструмент  можно применять многократно для получения необходимого количества дорожек. Дорожки на панели редактирования размещаются рядом и имеют одинаковую длину.

Чтобы добавить надписи на диаграмму деятельности необходимо использовать щелчок правой кнопкой мыши и ниспадающее меню (рисунок 8.7).

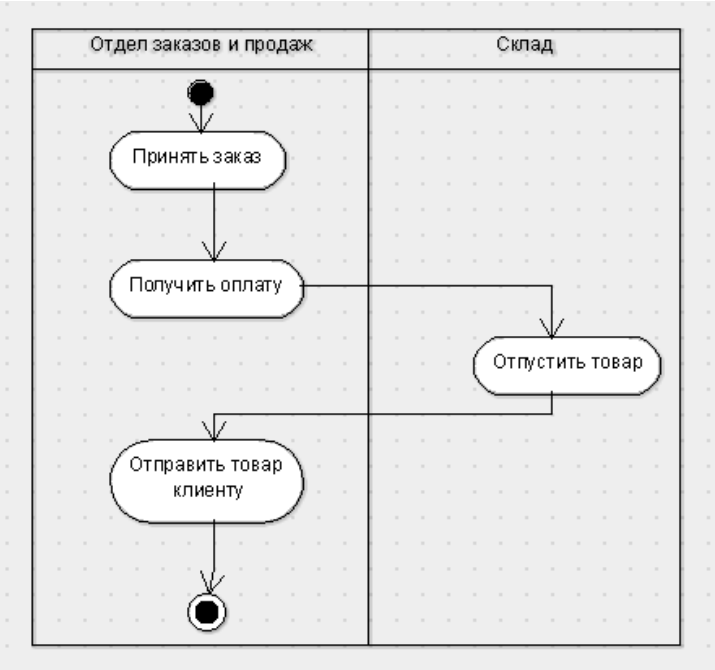


Рисунок 8.6 - Диаграмма деятельности с параллельными дорожками

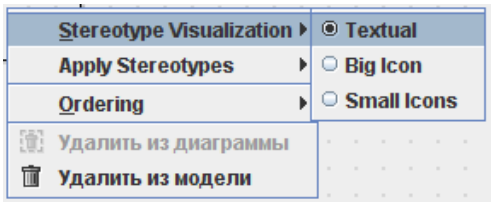


Рисунок 8.7 – Ниспадающее меню для добавления надписей

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Объекты можно явно указать на диаграмме деятельности (рисунок 8.8).

Для графического представления объектов используется прямоугольник класса, с тем отличием, что имя объекта подчеркивается. На диаграмме деятельности с дорожками расположение объекта может иметь некоторый дополнительный смысл. А именно, если объект расположен на границе двух дорожек, то это может означать, что переход к следующему состоянию действия в соседней дорожке ассоциирован с готовностью некоторого документа (объект в некотором состоянии). Если же объект целиком расположен внутри дорожки, то и состояние этого объекта целиком определяется действиями данной дорожки.

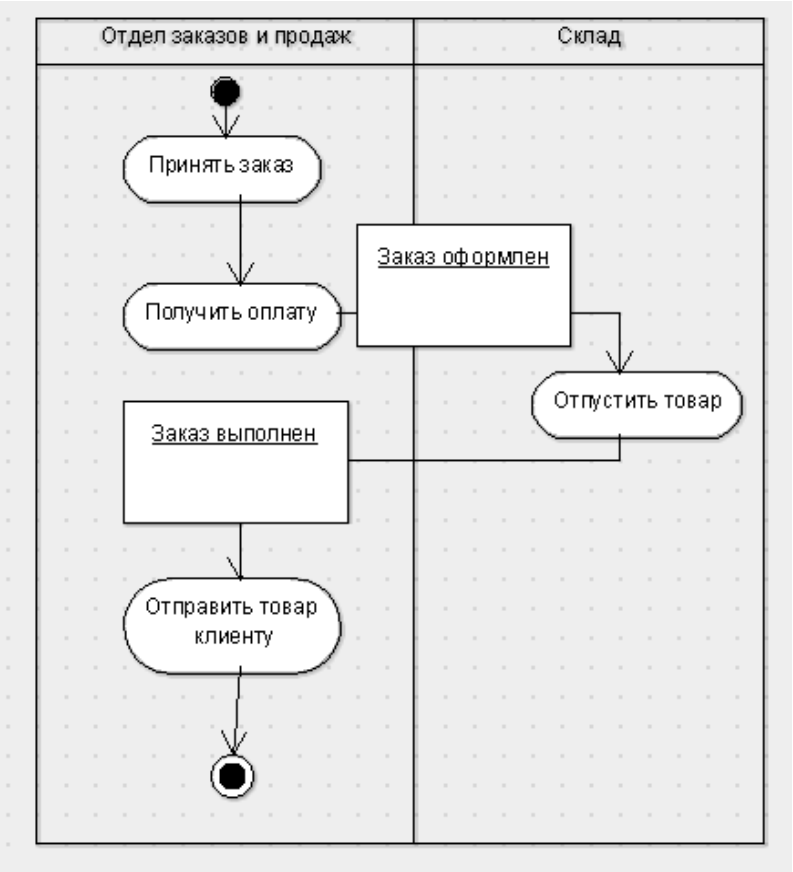


Рисунок 8.8 – Диаграмма деятельности с параллельными дорожками и объектом

Задание 1. Необходимо изучить предметную область, соответствующую вашему варианту задания, которое представлено в таблице 1.1. Выделите действия, характерные для предметной области, проанализируйте возможные переходы между состояниями. Проанализируйте возможность реализации последовательных и параллельных действий. Постройте диаграмму деятельности.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно построенную диаграмму деятельности.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы деятельности ArgoUML?
2. Какими средствами ArgoUML можно представить последовательные и параллельные операции?
3. Охарактеризуйте инструмент «Дорожки» ArgoUML?

Лабораторная работа № 9
ДИАГРАММЫ РАЗВЕРТЫВАНИЯ

Цель: изучить порядок построения диаграммы развертывания с помощью ArgoUML.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграмма развертывания (Deployment diagram) в UML моделирует *физическое* развертывание артефактов на узлах. Например, чтобы описать web-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, web-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, web-приложение, база данных), и как различные части этого комплекса соединяются друг с другом.

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

1. Узел устройства.
2. Узел среды выполнения.

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Панель инструментов для построения диаграммы развертывания представлена на рисунке 9.1. Описание инструментов представлено в таблице 9.1.

Рисунок 9.1 – Инструменты для построения диаграммы развертывания

Диаграмма развертывания показывает наличие физических соединений - маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime).

Таблица 9.1 – Описание инструментов диаграммы развертывания

Инструмент	Описание
	Узел (узел-тип)
	Узел - экземпляр
	Компонент
	Компонент-экземпляр
	Обобщение (от ребенка к родителю)
	Реализация (от класса к интерфейсу)
	Зависимость (от зависимого элемента)
	Ассоциация (шесть видов ассоциаций представлены на рисунке 7.3)
	Объект
	Связь

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом. В языке UML понятие узла может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы. Узлы на диаграмме развертывания могут представляться как в качестве типов (рисунок 9.2, а), так и в качестве экземпляров (рисунок 9.2, б).

а) б)

Рисунок 9.2 – Изображение узлов на диаграмме развертывания

Внутри узлов на диаграмме развертывания могут быть размещены исполняемые компоненты. На рисунке 9.3 показана диаграмма развертывания, на которой внутри узлов расположены исполняемые компоненты. Так как в ArgoUML нет инструмента для изображения интерфейса, то на диаграмме развертывания показан объект – экземпляр интерфейса.

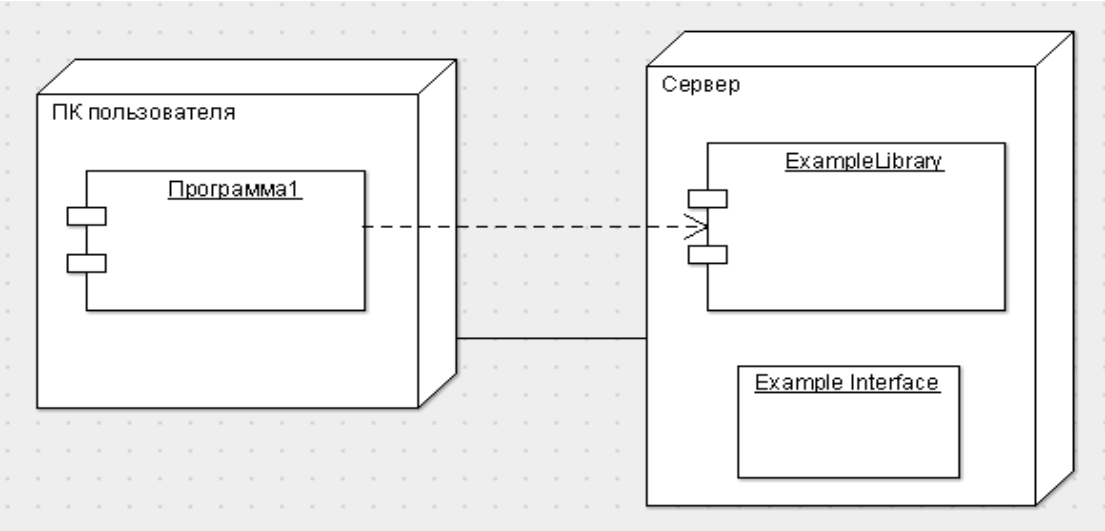


Рисунок 9.3 – Изображение исполняемых компонентов внутри узлов на диаграмме развертывания

В качестве отношений между узлами на диаграммах развертывания можно представить физические соединения между и зависимости между узлами и компонентами. На рисунке 9.4 показана диаграмма развертывания для компьютерной сети. Комментарий добавлен с помощью инструмента , связь комментария с узлом добавлена с помощью инструмента .

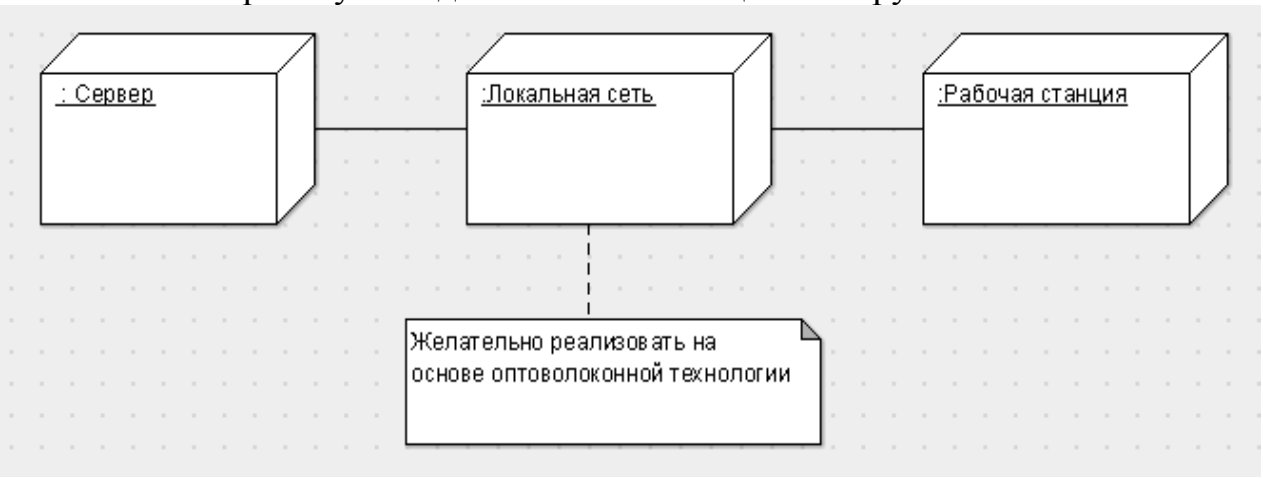


Рисунок 9.4 – Изображение локальной сети на диаграмме развертывания

Если на узле развернуто большое количество компонентов, то их можно показать не внутри узла, а в виде отношения зависимости (рисунок 9.5).

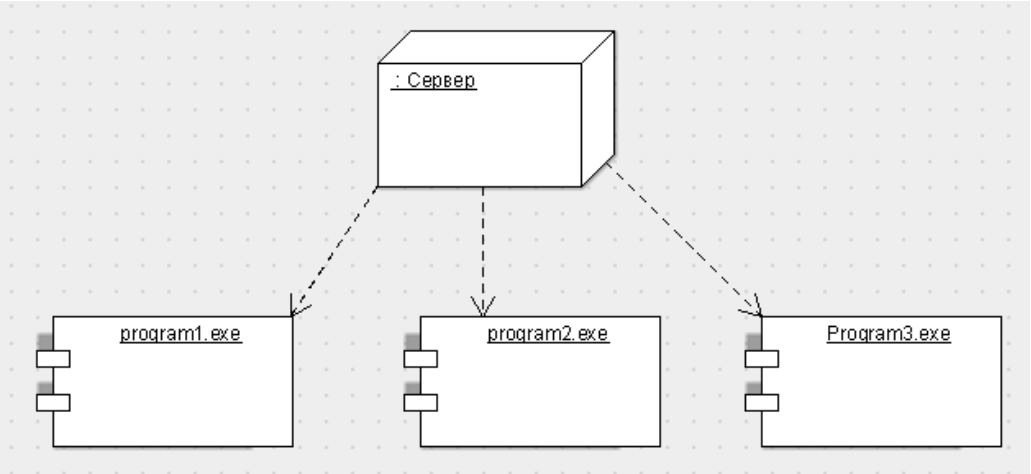


Рисунок 9.5 – Диаграмма развертывания с использованием отношения зависимости

Задание 1. Необходимо изучить предметную область, соответствующую вашему варианту задания, которое представлено в таблице 1.1. Выделите характерные для предметной области компоненты программы, существующие лишь на этапе ее исполнения, проанализируйте, какие аппаратные или программные компоненты можно представить как узлы. Проанализируйте возможность реализации программных или аппаратных решений для моделирования предметной области. Постройте диаграмму развертывания.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно построенную диаграмму развертывания.

Вопросы для защиты работы

1. Какие команды реализованы в панели инструментов диаграммы развертывания ArgoUML?
2. Какими средствами ArgoUML можно представить сложные зависимости между узлами и компонентами?
3. Охарактеризуйте отличие между узлом-типом и узлом-экземпляром в ArgoUML.

РАЗДЕЛ 3. ПРИНЯТИЕ РЕШЕНИЙ В СИСТЕМНОЙ ИНЖЕНЕРИИ

Лабораторная работа №10
ОПИСАНИЕ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

Цель: изучить состав, содержание и процедуры формирования основных документов, которые создаются в процессе типового проектирования ИС.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Описание деятельности предприятия является очень сложной задачей, которую в зависимости от цели рассматривают на различных уровнях. Например, согласно Нестерову А.К.[Нестеров А.К. Общая характеристика предприятия // Энциклопедия Нестеровых - https://odiplom.ru/lab/obszaya-harakteristika-predpriyatiya.html] Общая характеристика предприятия структурно включает в себя три элемента: описание предприятия и его организационно-правовой формы; анализ специфики деятельности, ассортимента продукции или услуг; организационную структуру предприятия и функции подразделений. В то же время, в общую характеристику предприятия могут быть включены основные экономические показатели деятельности предприятия, финансовые результаты и т.п. Общая характеристика предприятия может также включать описание основных подсистем, которые существуют в компании. В связи с этим следует иметь ввиду, что современное предприятие является сложной системой, обладающей многочисленными переплетенными подсистемами и каналами информационно-аналитических, административно-управленческих, материально-финансовых, производственно-технологических потоков. С целью выявления факторов, мешающих взаимодействию подсистем и каналов, рекомендуется выполнять анализ всех подсистем в комплексе. Информацию для проведения такого анализа находят в таких документах как Устав, Учредительный договор, Положения и др., в которых описываются процедуры прохождения информационно-аналитических, административно-управленческих, материально-финансовых, производственно-технологических потоков. Основными типами подсистем предприятия являются: информационно-аналитическая, собирающая и обрабатывающая информацию, полезную для успешного функционирования предприятия: маркетинговая, исследовательская, планово-экономическая, информационная, бухгалтерская; административно-управленческая, обеспечивающая управление и координацию различных подсистем: координации и регламента, стимулирования, продвижения; производственно-технологическая, обеспечивающая освоение новых технологий и успешное техническое функционирование производства: технологическая, производственная, опытных разработок; материально-финансовая, обеспечивающая

функционирование материально - финансовых потоков: продаж, закупок, складская, финансовая.

Задания.

1. Повторить раздел лекционного материала по данной тематике.
2. Рассмотреть с преподавателем вариант задания.
3. Привести текстовое описание основной деятельности компании, на основе которой будет выполняться лабораторная работа. При выполнении дальнейших заданий исходить из условия, что компания намеревается реализовать проект изменений в архитектуре предприятия.

Описание деятельности предприятия необходимо начинать с формулировки характеристики хозяйствующей единицы, включающую:

– краткую историческую справку о создании организации, установление правового статуса, т.е. организационно-правовой формы ее в зависимости от форм собственности (государственная, муниципальная, частная, индивидуальная, семейная, смешанная, общество с ограниченной ответственностью, акционерные общества закрытого и открытого типа, объединения предприятий, их филиалы и представительства, предприятия, созданные на основе аренды и выкупа имущества трудовым коллективом и др.);

– анализ всех видов деятельности, необходимых для функционирования организации;

– описание целей, масштаба и основного вида деятельности организации с учетом характеристики и факторов внешней среды, в которой она действует, специфику выпускаемой продукции или оказываемых услуг.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно составленное текстовое описание основной деятельности предприятия.

Вопросы для защиты работы

1. Какую информацию следует включить в краткую историческую справку о предприятии?
2. Какие подсистемы деятельности предприятия являются основными?
3. Охарактеризуйте информационные потоки на предприятии.

Лабораторная работа №11
АНАЛИЗ АРХИТЕКТУРЫ ПРЕДПРИЯТИЯ

Цель: Изучить состав, содержание и процедуры формирования основных документов, научиться применять методы системного анализа и моделирования для анализа архитектуры предприятий.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Организационная структура предприятия – это системообразующий элемент предприятия как целостного организма, определяющий его функционирование. Она содержит иерархию подчинения и связи структурных единиц, которые реализуют производственные бизнес-процессы

На практике применяют следующие модели:

- линейная модель: каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности;
- функциональная модель: «одно подразделение = одна функция»;
- линейно-функциональная модель: ступенчатая иерархическая;
- процессная модель: «одно подразделение = один процесс»;
- матричная модель: «один процесс или один проект = группа сотрудников из разных функциональных подразделений»;
- дивизиональная;
- множественная (смешанная);
- модель, ориентированная на контрагента: «одно подразделение = один контрагент (клиент или клиентская группа, поставщик, подрядчик и прочее), модель применяется в случае, если рынок контрагента, ограниченный. Например, в случае, если число потребителей сильно ограничено, целесообразно применить модель, ориентированную на клиента или клиентскую группу: «одно подразделение = один клиент».

Задания.

1. Описать деятельность предприятия и используемые им информационные технологии; построить модель оргструктуры предприятия. Для выполнения работы самостоятельно выбрать инструмент с поддержкой методологии моделирования архитектуры предприятия, например, Archimate.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно описанную организационную структуру предприятия.

Вопросы для защиты работы

1. Охарактеризуйте модели, применяемые для анализа архитектуры предприятия: линейную модель: функциональную модель линейно-функциональную модель, процессную модель, матричную модель.

Лабораторная работа №12
СТРАТЕГИЯ РАЗВИТИЯ АРХИТЕКТУРЫ ПРЕДПРИЯТИЯ

Цель: Научиться разрабатывать стратегию развития архитектуры предприятия.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Понятие «архитектура бизнеса» тесно связано со структурой предприятия, его отраслевой принадлежностью, производственной ориентацией и прочими характеристиками. В результате начало постепенно формироваться широкое представление и об архитектуре предприятия в целом, неразрывно связанное прежде всего с используемыми информационными технологиями и, в частности, с информационными системами.

Модель бизнес-мотивации, включает в себя цели (определенные в блоке «внутренние бизнес-процессы»), заинтересованные стороны, драйверы (то, что стимулирует решение проблемы) и оценки. Она позволяет выделять существующие проблемы компании, которые являются источником бизнес-потребностей.

Задания. По согласованию с преподавателем выполнить один из перечисленных вариант заданий:

1.Сформировать мотивационную модель целевой архитектуры предприятия для выбранного кейса. Основанием для мотивационной модели будут сформированная модель внешнего окружения компании, а также выявленные основные драйверы изменений. Для формализации драйверов можно использовать метод SWOT-анализа. Для выполнения данного варианта задания можно использовать нотацию ArchiMate. 5.

2. Выявить заинтересованные стороны (стейкхолдеры) и построить матрицу «Интерес – влияние».

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно выполненный отчет по варианту задания.

Вопросы для защиты работы

1. Охарактеризуйте метод SWOT-анализа.
2. Охарактеризуйте методы работы со стейкхолдерами.

Лабораторная работа №13
РАЗРАБОТКА И АНАЛИЗ АРХИТЕКТУРЫ ПРЕДПРИЯТИЯ

Цель: Научиться моделировать, анализировать и совершенствовать бизнес-процессы, разрабатывать и анализировать архитектуру предприятия.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Понятие «архитектура бизнеса» тесно связано со структурой предприятия, его отраслевой принадлежностью, производственной ориентацией и прочими характеристиками. В результате начало постепенно формироваться широкое представление и об архитектуре предприятия в целом, неразрывно связанное прежде всего с используемыми информационными технологиями и, в частности, с информационными системами.

Верхнеуровневая (обзорная) модель архитектуры предприятия (АП) отражает ядро архитектуры в наиболее общем виде. Такая модель включает все три слоя, которые входят в ядро, но рассматривает их очень абстрактно, отражая лишь наиболее важные архитектурные объекты. Модель фактически является представлением многослойного ракурса.

Верхнеуровневая (обзорная) модель АП формируется для всех архитекторов, работающих на предприятии. Именно эта модель позволяет архитекторам понять, с чем им предстоит работать в дальнейшем. Поскольку полное описание архитектуры редко представляется возможным, именно верхнеуровневая модель АП служит стартовой позицией, с которой будет происходить выбор области для подробного описания.

Формируются как текущая, так и целевая верхнеуровневая модель АП. Важно, чтобы целевая модель была согласованной и в ней отсутствовала избыточная сложность, в противном случае архитектурный проект только заложит фундамент для будущих проблем организации. Кроме того, целевая модель должна быть гибкой — в этом случае последующие проекты по разработке архитектуры будут выполняться с меньшими затратами.

Задания

1.Рассмотреть с преподавателем вариант задания.

2.Сформировать верхнеуровневую модель текущего состояния (включает верхнеуровневую модель бизнес-слоя, слоя информационных систем, технологического слоя), целевую верхнеуровневую модель архитектуры предприятия (включает аналогичные слои), опираясь на ADM. Выполнить gap-анализ между целевой и текущими моделями архитектуры. Для выполнения данного задания требуется использовать нотацию ArchiMate.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно выполненный материал по выбранному варианту задания.

Вопросы для защиты работы

1. Охарактеризуйте gap-анализ между целевой и текущими моделями архитектуры.

1. Охарактеризуйте три слоя, которые входят в ядро АП.

Лабораторная работа №14
ТЕСТ GOOGLE UNIT (GTEST)

Цель: Изучить метод разработки и разработать тест Unit Test.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Рамки тестирования Google C ++ основаны на архитектуре xUnit. Это кросс - платформенная система, которая обеспечивает автоматическое обнаружение тестов. Другими словами, нам не нужно перечислять все тесты в нашем тестовом наборе вручную. Он поддерживает богатый набор утверждений, таких как фатальные утверждения (ASSERT_), не фатальные утверждения (EXPECT_) и тест завершения, который проверяет, что программа прерывается ожидаемо.

Пошаговое выполнение лабораторной работы.

Первоначально у нас есть проект, возведение числа в квадрат:

```
// simplemath.h
#include <cmath>
Double cvadrat (double d)
{
return pow (d, 3);
}
// SimpleMath.cpp: определяет точку входа для консольного приложения.
#include "simplemath.h"
int main ()
{
cvadrat (10);
return 0;
}
```

Тестирование:

Шаг 1. Загрузите тест Google (gtest)

Загрузите gtest-1.7.0-rc1.zip из модуля Google C ++ Unit Test, а затем извлеките его.

Давайте посмотрим на каталог C: \ GTEST \ gtest-1.7.0, чтобы узнать, какие файлы есть.

В папке src есть все исходные файлы gtest, и позже нам нужно добавить каталог include в путь include.

Шаг 2. Скомпилируйте gtest в статическую библиотеку.

1. Создайте новый проект статической библиотеки с именем GoogleTest. Add-> New Project-> Win32 Project-> Статическая библиотека без предварительно скомпилированного заголовка.
2. Щелкните правой кнопкой мыши наш новый проект GoogleTest. На страницах свойств добавьте include path: C: \ GTEST \ gtest-1.7.0 и C: \ GTEST \ gtest-1.7.0 \ include.
3. Добавить исходные файлы с помощью Add-> Существующие элементы: C: \ GTEST \ GTEST-1.7.0 \ SRC \ gtest_all.cc

C: \ GTEST \ gtest-1.7.0 \ src \ gtest_main.cc.
4. Создайте GoogleTest в статической библиотеке.
В процессе сборки у нас могут быть некоторые ошибки, связанные с шаблоном класса:
VC ++ 2012 не поддерживает и не будет поддерживать вариационные шаблоны; следовательно, его реализация стандартной библиотеки пытается подделать их с использованием препроцессорных перегрузок и специализаций. Количество параметров шаблона faux variadic по умолчанию равно 5 - проблема в том, что gtest пытается создать экземпляр std :: tuple <> с целым числом аргументов в 10 шаблонов. - Google Test в Visual Studio 2012.
Таким образом, мы должны установить _VARIADIC_MAX = 10 для определений препроцессора в C / C ++. 7
Теперь, построим его снова:
1> ------ Rebuild Все начато: Проект: GoogleTest, Конфигурация: Отладка Win32 ------
1> gtest_main.cc
1> gtest-all.cc
1> Создание кода ...
1> GoogleTest.vcxproj -> c: \ users \ khyuck \ documents \ visual studio 2012 \ Projects \ SimpleMath \ Debug \ GoogleTest.lib
========== Перестроить все: 1 удалось, 0 не удалось, 0 пропущено ==========

Шаг 3. Создание проекта единичного тестирования. Теперь пришло время создать единый тестовый проект.
1. Щелкните правой кнопкой мыши Solution-> Add-> New Project с именем unittest_SimpleMath как консоль Win32. Мы только что добавили третий вариант решения:
2. Нам нужно добавить два пути, как это было сделано на шаге 2:
Щелкните правой кнопкой мыши на нашем новом проекте, unittest_SimpleMath.
На страницах свойств добавьте include path:
C: \ GTEST \ gtest-1.7.0 и C: \ GTEST \ gtest-1.7.0 \ include.
3. Этот проект требует дополнительного пути к первоначальному проекту (SimpleMath), который мы хотим протестировать.
4. Давайте добавим новые ссылки (GoogleTest и SimpleMath) в unittest_SimpleMath.
Щелкните правой кнопкой мыши на unittest_SimpleMath -> Ссылки ...
В разделе «Свойства» -> Добавить новые ссылки ...
5. Был разработан наш проект Unit Test.
Шаг 4. Создайте тестовый пример
Теперь нам нужно создать тестовый пример.
Введите следующие строки кода:
// unittest_SimpleMath.cpp: Определяет точку входа для консольного приложения.

```
#include "gtest / gtest.h"
#include "simplemath.h"
TEST (testMath, myCubeTest)
{
EXPECT_EQ (1000, cvadrat (10));
}
```

Здесь мы проверяем cvadrat () функцию, которую мы писали ранее, и сравниваем вывод 10 ^ 2 с 1000, используя макрос EXPECT_EQ.
Если мы запустим unittest_SimpleMath, мы получим результат теста:
В результате мы видим, что тест пройден успешно и без каких - либо ошибок.

Задания

1.Рассмотреть с преподавателем вариант задания.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать самостоятельно выполненный материал по выбранному варианту задания.

Вопросы для защиты работы

1. Охарактеризуйте методы тестирования программных компонентов.
2. Охарактеризуйте методы верификации объектно-ориентированных программ.

Лабораторная работа 15
ДИАГРАММЫ ПЕРЕХОДОВ СОСТОЯНИЙ

Цель: изучить построение диаграммы переходов состояний программного обеспечения, активно не взаимодействующего с окружающей средой, а также активно взаимодействующего с окружающей средой.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграмма переходов состояний является графической формой представления конечного автомата - математической абстракции, используемой для моделирования детерминированного поведения технических объектов или объектов реального мира.

На этапе анализа требований и определения спецификаций диаграмма переходов состояний демонстрирует *поведение* разрабатываемой программной системы при получении управляющих воздействий. Под *управляющими воздействиями* или сигналами в данном случае понимают управляющую информацию, получаемую системой извне. Например, управляющими воздействиями считают команды пользователя и сигналы датчиков, подключенных к компьютерной системе. Получив такое управляющее воздействие, разрабатываемая система должна выполнить определенные действия и или остаться в том же состоянии, или перейти в другое состояние взаимодействия с внешней средой.

Для построения диаграммы переходов состояний необходимо в соответствии с теорией конечных автоматов определить: основные состояния, управляющие воздействия (или условия перехода), выполняемые действия и возможные варианты переходов из одного состояния в другое. Условные обозначения, используемые при построении диаграмм переходов состояний, показаны на рисунке 15.1.

а б в

Рисунок 15.1 – Условные обозначения диаграмм переходов состояний: а – терминальное состояние, б – промежуточное состояние, в - переход

Если программная система в процессе функционирования активно не взаимодействует с окружающей средой (пользователем или датчиками), например, использует примитивный интерфейс и выполняет некоторые вычисления по заданным исходным данным, то диаграмма переходов состояний демонстрирует только последовательно выполняемые переходы: из исходного состояния в состояние ввода данных, затем после выполнения вычислений - в состояние вывода и, наконец, в состояние завершения работы

(рисунок 15.2).

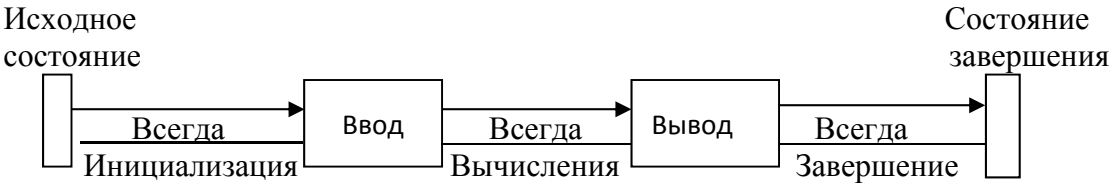


Рисунок 15.2 – Диаграмма переходов состояний ПО, активно не взаимодействующего с окружающей средой

Для интерактивного программного обеспечения с развитым пользовательским интерфейсом основные управляющие воздействия - команды пользователя, для программного обеспечения реального времени - сигналы от датчиков и|или оператора производственного процесса. Общим для этих типов программного обеспечения является наличие состояния ожидания, когда программное обеспечение приостанавливает работу до получения очередного управляющего воздействия. Для интерактивного программного обеспечения наиболее характерно получение команд различных типов (рисунок 15.3), а если это еще и программное обеспечение реального времени - однотипных сигналов (либо от многих датчиков, либо требующих продолжительной обработки).

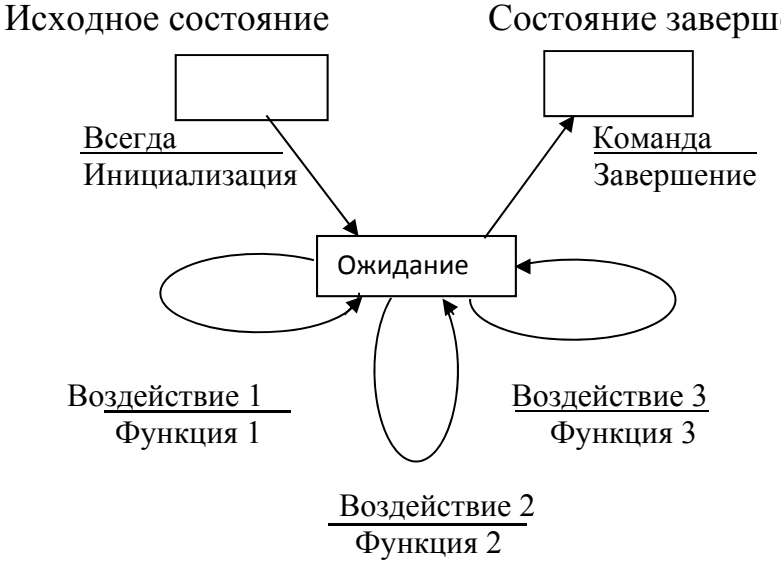


Рисунок 15.3 - Диаграмма переходов состояний ПО, активно взаимодействующего с окружающей средой

Методика и порядок выполнения работы

Изучите материал лекций. Определить основные состояния, управляющие воздействия (или условия перехода) для моделирования работы турникета или домофона.

Задание. Составить диаграмму переходов состояний для моделирования

программного обеспечения встроенного микропроцессора турникета.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;

2) диаграммы, построенных при выполнении задания.

Вопросы для защиты работы

1. В каких случаях целесообразно использовать диаграммы переходов состояний?
2. Для каких диаграмм характерно наличие состояния ожидания?
3. Какие условные обозначения используются для построения диаграммы переходов состояний?

Лабораторная работа № 16
ДИАГРАММЫ ПОТОКОВ ДАННЫХ

Цель: изучить правила построения диаграмм потоков данных.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Диаграммы потоков данных (DFD, Data Flow Diagram) – представляют собой сеть связанных между собой работ. Их удобно использовать для описания документооборота и обработки информации.

DFD описывает:

– функции обработки информации – работы;

– документы, объекты, сотрудников или отделы, участвующие в процессе обработки информации;

– внешние ссылки, которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;

– таблицы для хранения документов (хранилища данных).

Потоки данных являются механизмами, использующимися для моделирования передачи информации (или физических компонентов) из одной части системы в другую. Потоки изображаются на схеме именованными стрелками, ориентация которых указывает направление движения информации. Стрелки могут подходить к любой грани работы и могут быть двунаправленными для описания взаимодействия типа команда-ответ.

Назначение процесса состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы.

В основе модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных и потока данных.

Хранилище данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилища – это «срезы» потоков данных во времени. Информация, содержащаяся в хранилище, может использоваться в любое время после её определения при этом данные могут выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое. Хранилище данных - абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. Физически это может быть база данных, файл, таблица в оперативной памяти, картотека на бумаге и т. п.

Внешняя сущность представляет собой сущность вне контекста системы, являющуюся источником или приемником данных системы. Предполагается, что объекты, представленные внешними сущностями, не должны участвовать ни в какой обработке. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Внешняя сущность - материальный объект или физическое лицо,

выступающие в качестве источников или приемников информации, например, заказчики, персонал, поставщики, клиенты, банк и т. п.

Диаграммы потоков данных позволяют специфицировать как функции разрабатываемого программного обеспечения, так и обрабатываемые им данные. При использовании этой модели систему представляют в виде иерархии диаграмм потоков данных, описывающих асинхронный процесс преобразования информации с момента ввода в систему до выдачи пользователю. На каждом следующем уровне иерархий происходит уточнение процессов, пока очередной процесс не будет признан элементарным.

Модели потоков данных были независимо предложены сначала Е. Йорданом (1975), затем Ч. Гейном и Т. Сарсоном (1979). На этих моделях основаны классические методологии структурного анализа и проектирования программного обеспечения соответственно Йордана-Де Марка и Гейна-Сарсона. Та же модель используется в методологии структурного анализа и проектирования SSADM (Structured Systems Analysis and Design Method), принятой в Великобритании в качестве национального стандарта разработки информационных систем.

Процесс - преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Каждый процесс в системе имеет свой номер и связан с исполнителем, который осуществляет данное преобразование. Как в случае функциональных диаграмм, физически преобразование может осуществляться компьютерами, вручную или специальными устройствами. На верхних уровнях иерархии, когда процессы еще не определены, вместо понятия «процесс» используют понятия «система» и «подсистема», которые обозначают соответственно систему в целом или ее функционально законченную часть.

Поток данных — процесс передачи некоторой информации от источника к приемнику.

Физически процесс передачи информации может происходить по кабелям под управлением программы или программной системы или вручную при участии устройств или людей вне проектируемой системы.

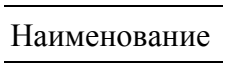
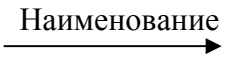
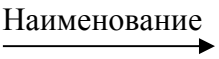
Таким образом, диаграмма иллюстрирует как потоки данных, порожденные некоторыми внешними сущностями, трансформируются соответствующими процессами (или подсистемами), сохраняются накопителями данных и передаются другим внешним сущностям — приемникам информации. В результате мы получаем сетевую модель хранения|обработки информации.

Для изображения диаграмм потоков данных традиционно используют два вида нотаций: нотации Йордана и Гейна-Сарсона (таблица 16.1).

Построение иерархии диаграмм потоков данных начинают с диаграммы особого вида - контекстной диаграммы, которая определяет наиболее общий вид системы. На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации без указания исполнителей, т. е. описывают интерфейс между системой и

внешним миром. Обычно начальная контекстная диаграмма имеет форму звезды.

Таблица 16.1 – Нотации Йордана и Гейна-Сарсона для диаграммы потоков данных

Понятие	Нотация Йордана	Нотация Гейна-Сарсона
Внешняя сущность		
Система, подсистема, процесс		
Накопитель данных		
Поток		

Над линией потока, направление которого обозначают стрелкой, указывают, какая конкретно информация в данном случае передается. На рисунке 16.1 показан пример диаграммы потоков данных для создания записи клиента.

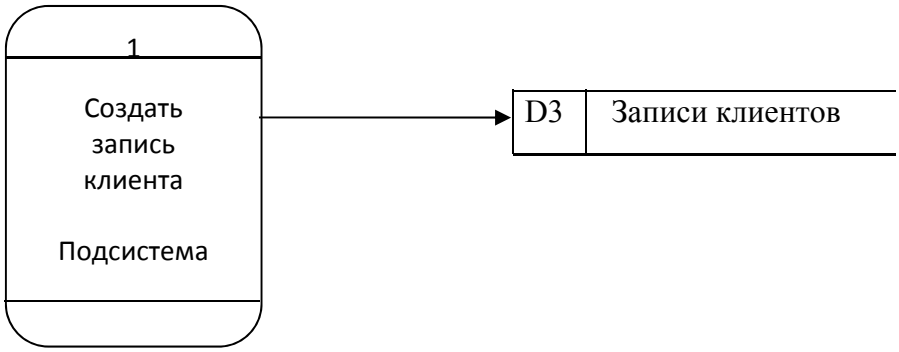


Рисунок 16.1 – Диаграмма потока данных (нотация Гейна-Сарсона)

Если проектируемая система содержит большое количество внешних сущностей (более 10-ти), имеет распределенную природу или включает уже существующие подсистемы, то строят иерархии контекстных диаграмм.

При разработке контекстных диаграмм происходит детализация функциональной структуры будущей системы, что особенно важно, если разработка ведется несколькими коллективами разработчиков.

Полученную таким образом модель системы проверяют на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами).

На следующем этапе каждую подсистему контекстной диаграммы детализируют при помощи диаграмм потоков данных. В процессе детализации соблюдают правило балансировки – при детализации подсистемы можно использовать компоненты только тех подсистем, с которыми у разрабатываемой подсистемы существует информационная связь (т. е. с которыми она связана потоками данных).

Решение о завершении детализации процесса принимают в следующих случаях:

- процесс взаимодействует с 2-3-мя потоками данных;
- возможно описание процесса последовательным алгоритмом;
- процесс выполняет единственную логическую функцию преобразования входной информации в выходную.

На недетализируемые процессы составляют спецификации, которые должны содержать описание логики (функций) данного процесса. Такое описание может, выполняться: на естественом языке, с применением структурированного естественного языка (псевдокодов), с применением таблиц и деревьев решений, в виде схем алгоритмов, в том числе flow-форм и диаграмм Насси-Шнейдермана.

Для облегчения восприятия процессы детализируемой подсистемы нумеруют, соблюдая иерархию номеров: так процессы, полученные при детализации процесса или подсистемы «1», должны нумероваться «1.1», «1.2» и т. д. Кроме этого желательно размещать на каждой диаграмме от 3 до 6-7 процессов и не загромождать диаграммы деталями, не существенными на данном уровне.

Декомпозицию потоков данных необходимо осуществлять параллельно с декомпозицией процессов.

Окончательно разработку модели выполняют в два этапа.

1 этап - построение контекстной диаграммы - включает выполнение следующих действий:

- классификацию множества требований и организацию их в основные функциональные
- группы — процессы;
- идентификацию внешних объектов - внешних сущностей, с которыми система должна быть связана;
- идентификацию основных видов информации - потоков данных, циркулирующей между системой и внешними объектами;
- предварительную разработку контекстной диаграммы;
- изучение предварительной контекстной диаграммы и внесение в нее

изменений по результатам ответов на возникающие при изучении вопросы по всем ее частям;

- построение контекстной диаграммы путем объединения всех процессов предварительной диаграммы в один процесс, а также группирования потоков.

2 этап - формирование иерархии диаграмм потоков данных – включает для каждого уровня:

- проверку и изучение основных требований по диаграмме соответствующего уровня (для первого уровня - по контекстной диаграмме);
- декомпозицию каждого процесса текущей диаграммы потоков данных с помощью детализирующей диаграммы или, если некоторую функцию сложно или невозможно выразить комбинацией процессов, построение спецификации процесса;
- добавление определений новых потоков в словарь данных при каждом появлении их на диаграмме;
- проведение ревизии с целью проверки корректности и улучшения наглядности модели после построения двух-трех уровней.

Полная спецификация процессов включает также описание структур данных, используемых как при передаче информации в потоке, так и при хранении в накопителе. Описываемые структуры данных могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что соответствующие элементы данных в структуре могут отсутствовать.

Альтернатива означает, что в структуру может входить один из перечисленных элементов.

Итерация означает, что элемент может повторяться некоторое количество раз.

Кроме того, для данных должен быть указан тип: непрерывное или дискретное значение. Для непрерывных данных могут определяться единицы измерений, диапазон значений, точность представления и форма физического кодирования. Для дискретных - может указываться таблица допустимых значений.

Полученную законченную модель необходимо проверить на полноту и согласованность. Под согласованностью модели в данном случае понимают выполнение для всех потоков данных правила сохранения информации: все поступающие куда-либо данные должны быть считаны и записаны.

Для представления управляющих процессов в проектируемых системах можно применить диаграммы управляющих потоков данных, которые используют понятия: управляющий процесс, управляющий поток данных и, возможно, хранилище управляющих данных.

Управляющий процесс получает с помощью управляющих потоков некоторую информацию о ситуации в системе и инициирует посредством управляющего потока соответствующие процессы.

На диаграммах управляющих потоков данных используют те же обозначения, что и для обычных потоков, но изображают их пунктирной линией. Дополнительно может быть указан тип управляющего потока:

- T-поток (Trigger Flow - тригерный поток) - поток управления, который может только «включать» процесс, следующий управляющий сигнал опять «включит» процесс, даже если процесс уже активен;
- A-поток (Activator Flow - активирующий поток) - поток управления, который может как «включать», так и «выключать» управляемый процесс - если процесс включен, то следующий сигнал его выключит;
- E|D-поток (Enable/Disable Flow - переключающий поток) - поток управления, который может включать процесс сигналом по одной (E) линии и выключать - сигналом по другой (D) линии.

При необходимости тип потока данных (управляющий или обычный) можно изменять. Для этого используют специальное обозначение - узел изменения типа потока данных (рисунок 16.2). К этому узлу поток подходит как поток данных, а выходит из него как управляющий поток.

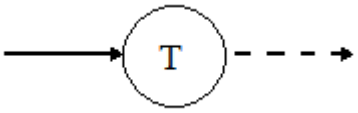


Рисунок 16.2 - Узел изменения типа потока данных

Задание. Изучить теоретическое обоснование. По согласованию с преподавателем выбрать предметную область для построения диаграммы. Для построения диаграммы потоков данных можно использовать средства текстового процессора Ms Word или любой графический редактор.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать контекстную диаграмму и детализирующую диаграмму потоков данных

Вопросы для защиты работы:

1. Каким образом для представления управляющих процессов в проектируемых системах можно применить диаграммы управляющих потоков данных?
2. Каким образом в диаграммах управляющих потоков данных отображается управляющий процесс?
3. Каким образом в диаграммах управляющих потоков данных отображается управляющий поток данных?
4. Каким образом в диаграммах управляющих потоков данных отображается хранилище управляющих данных?

РАЗДЕЛ 4. СТАНДАРТЫ В СИСТЕМНОЙ ИНЖЕНЕРИИ

Лабораторная работа № 17
СТАНДАРТЫ IDEF. IDEF()-ДИАГРАММЫ

Цель: Изучить стандарты IDEF и методы построения функциональных диаграмм.

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

В настоящий момент к семейству IDEF можно отнести следующие стандарты:

– IDEF0 – Function Modeling – методология функционального моделирования. С помощью наглядного графического языка IDEF0 изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков – в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы. Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique);

– IDEF1 – Information Modeling – методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи;

– IDEF1X (IDEF1 Extended) – Data Modeling – методология построения реляционных структур (баз данных), относится к типу методологий «Сущность-связь» (ER–Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;

– IDEF2 – Simulation Model Design – методология динамического моделирования развития систем. В связи с весьма серьезными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе. В настоящее время присутствуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных сетей Петри» (CPN – Color Petri Nets);

– IDEF3 – Process Description Capture — Документирование технологических процессов. IDEF3 — методология документирования процессов, происходящих в системе (например, на предприятии), описываются сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 – каждая функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3;

– IDEF4 – Object-Oriented Design — методология построения объектно-ориентированных систем, позволяют отображать структуру объектов и заложенные принципы их взаимодействия, тем самым позволяя анализировать

и оптимизировать сложные объектно-ориентированные системы;

– IDEF5 – Ontology Description Capture — Стандарт онтологического исследования сложных систем. С помощью методологии IDEF5 онтология системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы и производится её оптимизация;

– IDEF6 – Design Rationale Capture — Обоснование проектных действий. Назначение IDEF6 состоит в облегчении получения «знаний о способе» моделирования, их представления и использования при разработке систем управления предприятиями. Под «знаниями о способе» понимаются причины, обстоятельства, скрытые мотивы, которые обуславливают выбранные методы моделирования. Проще говоря, «знания о способе» интерпретируются как ответ на вопрос: «почему модель получилась такой, какой получилась?» Большинство методов моделирования фокусируются на собственно получаемых моделях, а не на процессе их создания. Метод IDEF6 акцентирует внимание именно на процессе создания модели;

– IDEF7 – Information System Auditing — Аудит информационных систем. Этот метод определён как востребованный, однако так и не был полностью разработан;

– IDEF8 – User Interface Modeling — Метод разработки интерфейсов взаимодействия оператора и системы (пользовательских интерфейсов). Современные среды разработки пользовательских интерфейсов в большей степени создают внешний вид интерфейса. IDFE8 фокусирует внимание разработчиков интерфейса на программировании желаемого взаимного поведения интерфейса и пользователя на трех уровнях: выполняемой операции (что это за операция); сценарии взаимодействия, определяемом специфической ролью пользователя (по какому сценарию она должна выполняться тем или иным пользователем); и, наконец, на деталях интерфейса (какие элементы управления, предлагает интерфейс для выполнения операции);

– IDEF9 – Scenario-Driven IS Design (Business Constraint Discovery method) — Метод исследования бизнес ограничений был разработан для облегчения обнаружения и анализа ограничений в условиях которых действует предприятие. Обычно, при построении моделей описанию ограничений, оказывающих влияние на протекание процессов на предприятии уделяется недостаточное внимание. Знания об основных ограничениях и характере их влияния, закладываемые в модели, в лучшем случае остаются неполными, несогласованными, распределенными нерационально, но часто их вовсе нет. Это не обязательно приводит к тому, что построенные модели нежизнеспособны, просто их реализация столкнется с непредвиденными трудностями, в результате чего их потенциал будет не реализован. Тем не менее в случаях, когда речь идет именно о совершенствовании структур или

адаптации к предсказываемым изменениям, знания о существующих ограничениях имеют критическое значение;

– IDEF10 – Implementation Architecture Modeling — Моделирование архитектуры выполнения. Этот метод определён как востребованный, однако так и не был полностью разработан;

– IDEF11 – Information Artifact Modeling. Этот метод определён как востребованный, однако так и не был полностью разработан;

– IDEF12 – Organization Modeling — Организационное моделирование. Этот метод определён как востребованный, однако так и не был полностью разработан;

– IDEF13 – Three Schema Mapping Design — Трёхсхемное проектирование преобразования данных. Этот метод определён как востребованный, однако так и не был полностью разработан;

– IDEF14 – Network Design — Метод проектирования компьютерных сетей, основанный на анализе требований, специфических сетевых компонентов, существующих конфигураций сетей. Он обеспечивает поддержку решений, связанных с рациональным управлением материальными ресурсами, что позволяет достичь существенной экономии.

Основной принцип методологии IDEF – это представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе. В IDEF0 все, что происходит в системе и ее элементах, называется функциями. Каждой функции ставится в соответствие блок. На диаграмме IDEF0 блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются стрелками, входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась. Диаграммы IDEF0 основаны на простой графике блоков и стрелок, метки для описания блоков и стрелок выполняются на естественном языке. Последовательная декомпозиция диаграмм строится по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение.

Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется.

Основные определения (понятия) методологии и языка IDEF0.

1. Блок: прямоугольник, содержащий имя и номер и используемый для описания функции.
2. Ветвление: разделение стрелки на два или большее число сегментов.
3. Внутренняя стрелка: входная, управляющая или выходная стрелка, концы которой связывают источник и потребителя, являющиеся блоками одной диаграммы. Отличается от граничной стрелки.

4. Входная стрелка: класс стрелок, которые отображают вход IDEF0-блока, то есть данные или материальные объекты, которые преобразуются функцией в выход. Входные стрелки связываются с левой стороной блока IDEF0.
5. Выходная стрелка: класс стрелок, которые отображают выход IDEF0-блока, то есть данные или материальные объекты, произведенные функцией. Выходные стрелки связываются с правой стороной блока IDEF0.
6. Глоссарий: список определений для ключевых слов, фраз и аббревиатур, связанных с узлами, блоками, стрелками или с моделью IDEF0 в целом.
7. Граничная стрелка: стрелка, один из концов которой связан с источником или потребителем, а другой не присоединен ни к какому блоку на диаграмме. Отображает связь диаграммы с другими блоками системы и отличается от внутренней стрелки.
8. Декомпозиция: разделение моделируемой функции на функции – компоненты.
9. Дерево узлов: представление отношений между родительскими и дочерними узлами модели IDEF0 в форме древовидного графа.
10. Диаграмма A-0: специальный вид (контекстной) диаграммы IDEF0, состоящей из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизмы, вместе с формулировками цели модели и точки зрения, с которой строится модель.
11. Диаграмма: часть модели, описывающая декомпозицию блока.
12. Диаграмма-иллюстрация (FEO): графическое описание, используемое, для сообщения специфических фактов о диаграмме IDEF0. При построении диаграмм FEO можно не придерживаться правила IDEF0.
13. Дочерний блок: блок на дочерней (порожденной) диаграмме.
14. Дочерняя диаграмма: диаграмма, детализирующая родительский (порождающий) блок.
15. Имя блока: глагол или глагольный оборот, помещенный внутри блока и описывающий моделируемую функцию.
16. Интерфейс: разделяющая граница, через которую проходят данные или материальные объекты; соединение между двумя или большим числом компонентов модели, передающее данные или материальные объекты от одного компонента к другому.
17. Код ICOM: аббревиатура (Input - Вход, Control - Управление, Output - Выход, Mechanism – Механизм), код, обеспечивающий соответствие граничных стрелок дочерней диаграммы со стрелками родительского блока; используется для ссылок.
18. Контекст: окружающая среда, в которой действует функция (или комплект функций на диаграмме).
19. Контекстная диаграмма: диаграмма, имеющая узловой номер A-n ($n \geq 0$), которая представляет контекст модели, Диаграмма A-0, состоящая из одного блока, является необходимой (обязательной) контекстной диаграммой; диаграммы с узловыми номерами A-1, A-2,... - дополнительные контекстные диаграммы.

20. Метка стрелки: существительное или оборот существительного, связанные со стрелкой или сегментом стрелки и определяющие их значение.
21. Модель IDEF0: графическое описание системы, разработанное с определенной целью и с выбранной точки зрения. Комплект одной или более диаграмм IDEF0, которые изображают функции системы с помощью графики, текста и глоссария.
22. Номер блока: число (0 – 6), помещаемое в правом нижнем углу блока и однозначно идентифицирующее блок на диаграмме.
23. Перечень узлов: список, часто ступенчатый, показывающий узлы модели IDEF0 в упорядоченном виде. Имеет то же значение и содержание, что и дерево узлов (п. 9).
24. Примечание к модели: текстовый комментарий, являющийся частью диаграммы IDEF0 и используемый для записи факта, не нашедшего графического изображения.
25. Родительская диаграмма: диаграмма, которая содержит родительский блок.
26. Родительский блок: блок, который подробно описывается дочерней диаграммой.
27. Связывание/развязывание: объединение значений стрелок в составное значение (связывание в «пучок»), или разделение значений стрелок (развязывание «пучка»), выраженные синтаксисом слияния или ветвления стрелок.
28. Сегмент стрелки: сегмент линии, который начинается или заканчивается на стороне блока, в точке ветвления или слияния, или на границе (несвязанный конец стрелки).
29. Семантика: значение синтаксических компонентов языка.
30. Синтаксис: Структурные компоненты или характеристики языка и правила, которые определяют отношения между ними.
31. Слияние: объединение двух или большего числа сегментов стрелок в один сегмент.
32. С-номер: номер, создаваемый в хронологическом порядке и используемый для идентификации диаграммы и прослеживания ее истории; может быть использован в качестве ссылочного выражения при определении конкретной версии диаграммы.
33. Стрелка: направленная линия, состоящая из одного или нескольких сегментов, которая моделирует открытый канал или канал, передающий данные или материальные объекты от источника (начальная точка стрелки), к потребителю (конечная точка с «наконечником»). Имеется 4 класса стрелок: входная стрелка, выходная стрелка, управляющая стрелка, стрелка механизма (включает стрелку вызова).
34. Стрелка вызова: вид стрелки механизма, который обозначает обращение из блока данной модели (или части модели) к блоку другой модели (или другой части той же модели) и обеспечивает связь между моделями или между разными частями одной модели.

35. Стрелка механизма: класс стрелок, которые отображают механизмы IDEF0, то есть средства, используемые для выполнения функции; включает специальный случай стрелки вызова. Стрелки механизмов связываются с нижней стороной блока IDEF0.
36. Стрелка, помещенная в туннель (туннельная стрелка): стрелка (со специальной нотацией), не удовлетворяющая обычному требованию, согласно которому каждая стрелка на дочерней диаграмме должна соответствовать стрелкам на родительской диаграмме.
37. Текст: любой текстовый (не графический) комментарий к графической диаграмме IDEF0.
38. Тильда: небольшая ломаная (волнистая) линия, используемая для соединения метки с конкретным сегментом стрелки или примечания модели с компонентом диаграммы.
39. Точка зрения: указание на должностное лицо или подразделение организации, с позиции которого разрабатывается модель.
40. Узел: блок, порождающий дочерние блоки; родительский блок.
41. Узловая ссылка: код, присвоенный диаграмме, для ее идентификации и определения положения в иерархии модели; формируется из сокращенного имени модели и узлового номера диаграммы с дополнительными расширениями.
42. Узловой номер диаграммы: часть узловой ссылки диаграммы, которая соответствует номеру родительского блока.
43. Узловой номер: код, присвоенный блоку и определяющий его положение в иерархии модели; может быть использован в качестве подробного ссылочного выражения.
44. Управляющая стрелка: класс стрелок, которые в IDEF0 отображают управления, то есть условия, при выполнении которых выход блока будет правильным. Данные или объекты, моделируемые как управления, могут преобразовываться функцией, создающей соответствующий выход. Управляющие стрелки связываются с верхней стороной блока IDEF0.
45. Функция: деятельность, процесс или преобразование (моделируемые блоком IDEF0), идентифицируемое глаголом или глагольной формой, которая описывает, что должно быть выполнено.
46. Цель: краткая формулировка причины создания модели.

В основе методологии IDEF0 лежат четыре основных понятия, описанные в разделе 13 лекций:

- функциональный блок (Activity Box);
- интерфейсная дуга (Arrow) или стрелка;
- декомпозиция (Decomposition);
- глоссарий (Glossary).

Функциональный блок показан на рисунке 17.1. Внутри каждого блока размещается его имя и номер. Имя должно быть активным глаголом или глагольным оборотом, описывающим функцию. Номер блока размещается в

правом нижнем углу. Номера блоков используются для их идентификации на диаграмме и в соответствующем тексте.

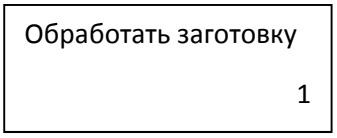


Рисунок 17.1 – Блок

Стрелка (дуга) формируется из одного или более отрезков прямых линий и наконечника на одном конце. Отрезки стрелок могут быть прямыми или ломаными; в последнем случае горизонтальные и вертикальные отрезки стрелки сопрягаются дугами, имеющими угол 90°. Правила построения стрелок показаны на рисунке 17.2.

Концы стрелок должны касаться внешней границы функционального блока, но не должны пересекать ее. Стрелки должны присоединяться к блоку на его сторонах, а не в углах.

Каждая сторона функционального блока имеет стандартное значение и однозначно определяет роль присоединенной стрелки. Стрелки, входящие в левую сторону блока обозначают входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелки, входящие в блок сверху, соответствуют управлению. Стрелки, выходящие справа, обозначают выходы, т.е. данные или материальные объекты, произведенные функцией.

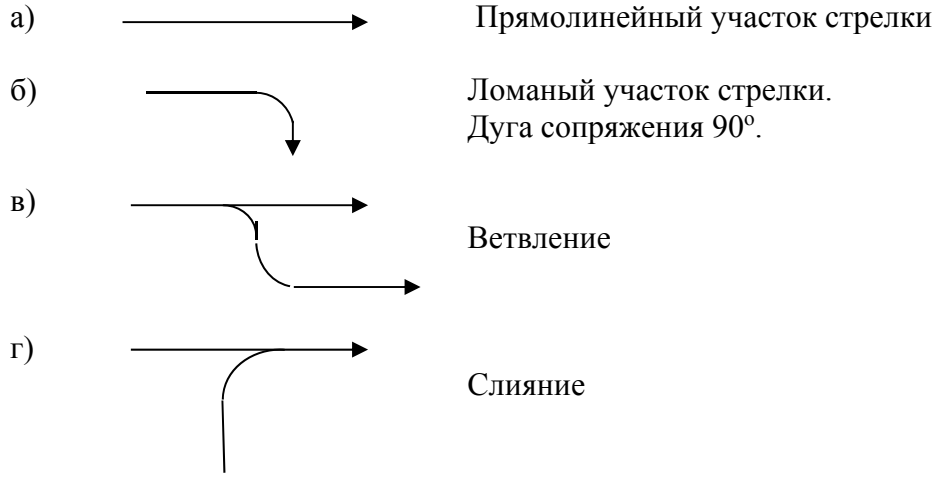


Рисунок 17.2 – Построение стрелок

Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Другие средства могут наследоваться из родительского блока. Стрелки механизма, направленные вниз, являются стрелками вызова. Стрелки вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок). Стандартное расположение стрелок показано на рисунке 17.3.

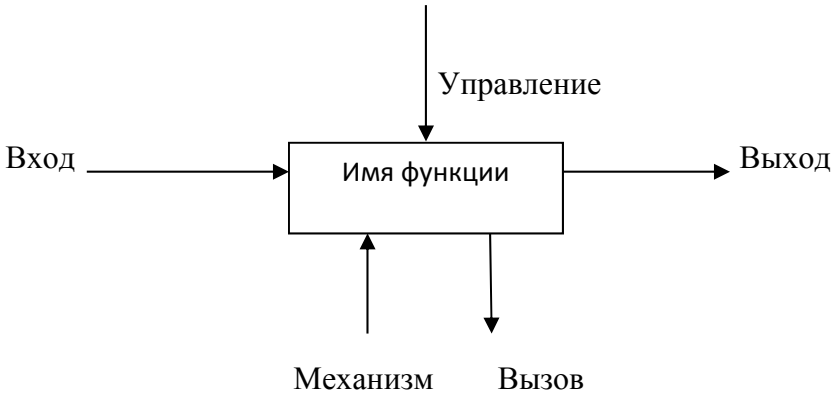


Рисунок 17.3 – Стандартное расположение стрелок

Примеры имен функций (функциональных блоков):

- производить детали;
- планировать ресурсы;
- наблюдать за выполнением;
- проектировать систему;
- эксплуатировать;
- разработать чертежи;
- изготовить деталь;
- проверять деталь.

Стрелки идентифицируют данные или материальные объекты, необходимые для выполнения функции или производимые ею. Каждая стрелка должна быть помечена существительным или оборотом существительного, например:

- спецификации;
- отчет об испытаниях;
- бюджет;
- конструкторские требования;
- конструкция детали;
- директива;
- инженер-конструктор
- плата в сборе;
- требования.

В метках стрелок не должны использоваться такие общеупотребительные термины, как: функция, вход, управление, выход, механизм, вызов.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется А-0. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя – общее для

всего проекта. Это же справедливо и для всех стрелок диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта. Диаграмма А-0 устанавливает область моделирования и ее границу. Пример диаграммы А-0 показан на рисунке 17.4.

Контекстная диаграмма А-0 также должна содержать краткие утверждения, определяющие точку зрения должностного лица или подразделения, с позиций которого создается модель, и цель, для достижения которой ее разрабатывают. Изменение точки зрения, приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект. Формулировка цели выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

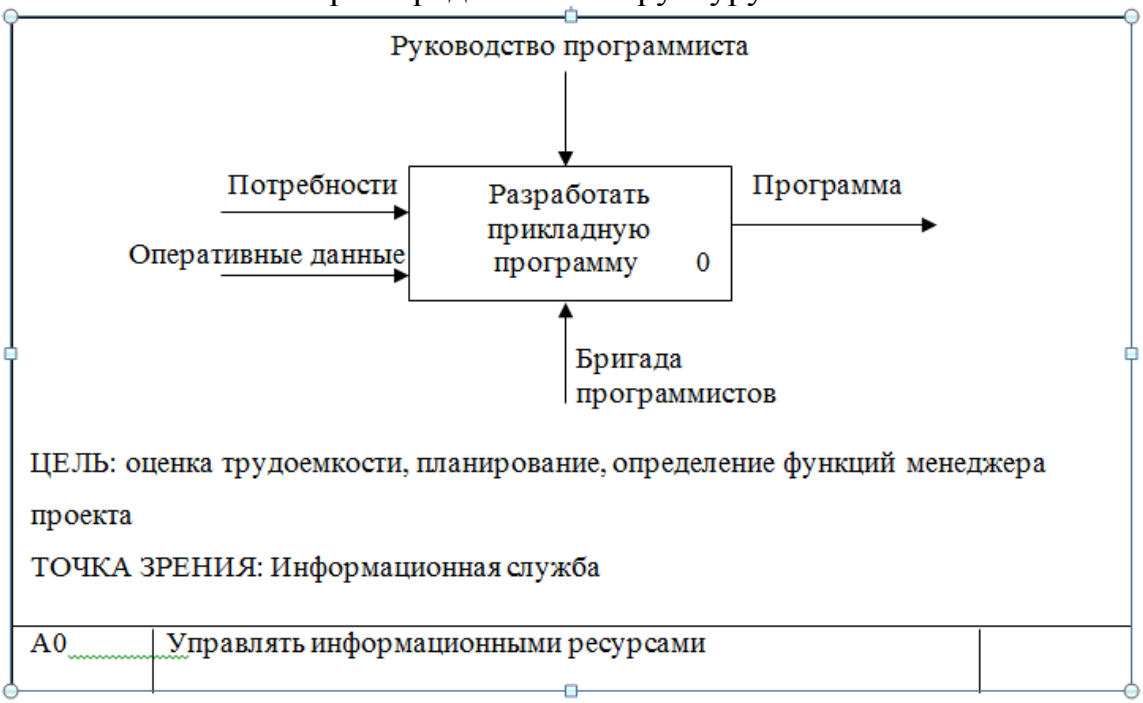


Рисунок 17.4 – Контекстная диаграмма

Наиболее важные свойства объекта обычно выявляются на верхних уровнях иерархии. По мере декомпозиции функции (функционального блока) верхнего уровня и разбиения на подфункции, эти свойства уточняются. Каждая подфункция может быть представлена декомпозицией на элементы следующего уровня, и так происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Каждая подфункция моделируется отдельным блоком. Каждый родительский блок подробно описывается дочерней диаграммой на более низком уровне. Все дочерние диаграммы не должны выходить за пределы области контекстной диаграммы верхнего уровня.

Дочерняя диаграмма

Функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции путем создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части путем создания дочерней диаграммы следующего, более низкого уровня. Каждая дочерняя диаграмма содержит дочерние блоки и стрелки, обеспечивающие дополнительную детализацию родительского блока.

Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Поэтому дочернюю диаграмму можно рассматривать как вложенную в свой родительский блок. Эта структура показана на рисунке 17.5.

Родительская диаграмма

Родительской называется диаграмма, которая содержит один или более родительских блоков. Каждая не контекстная диаграмма является также дочерней диаграммой, поскольку, по определению, она подробно описывает некоторый родительский блок. Таким образом, любая диаграмма может быть как родительской диаграммой (содержать родительские блоки), так и дочерней (подробно описывать собственный родительский блок). Аналогично, блок может быть как родительским (подробно описываться дочерней диаграммой) так и дочерним (появляющимся на дочерней диаграмме).

В методологии IDEF0 существует 6 (шесть) типов отношений между блоками в пределах одной диаграммы:

- доминирование;
- управление;
- выход - вход;
- обратная связь по управлению;
- обратная связь по входу;
- выход – механизм.

Доминирование определяется взаимным расположением блоков на диаграмме. Предполагается, что блоки, расположенные на диаграмме выше и левее, доминируют над блоками, расположенными ниже и правее. Доминирование подразумевает влияние, которое один блок оказывает на другие блоки диаграммы. Остальные пять отношений описывают связи между блоками и изображаются соответствующими стрелками.

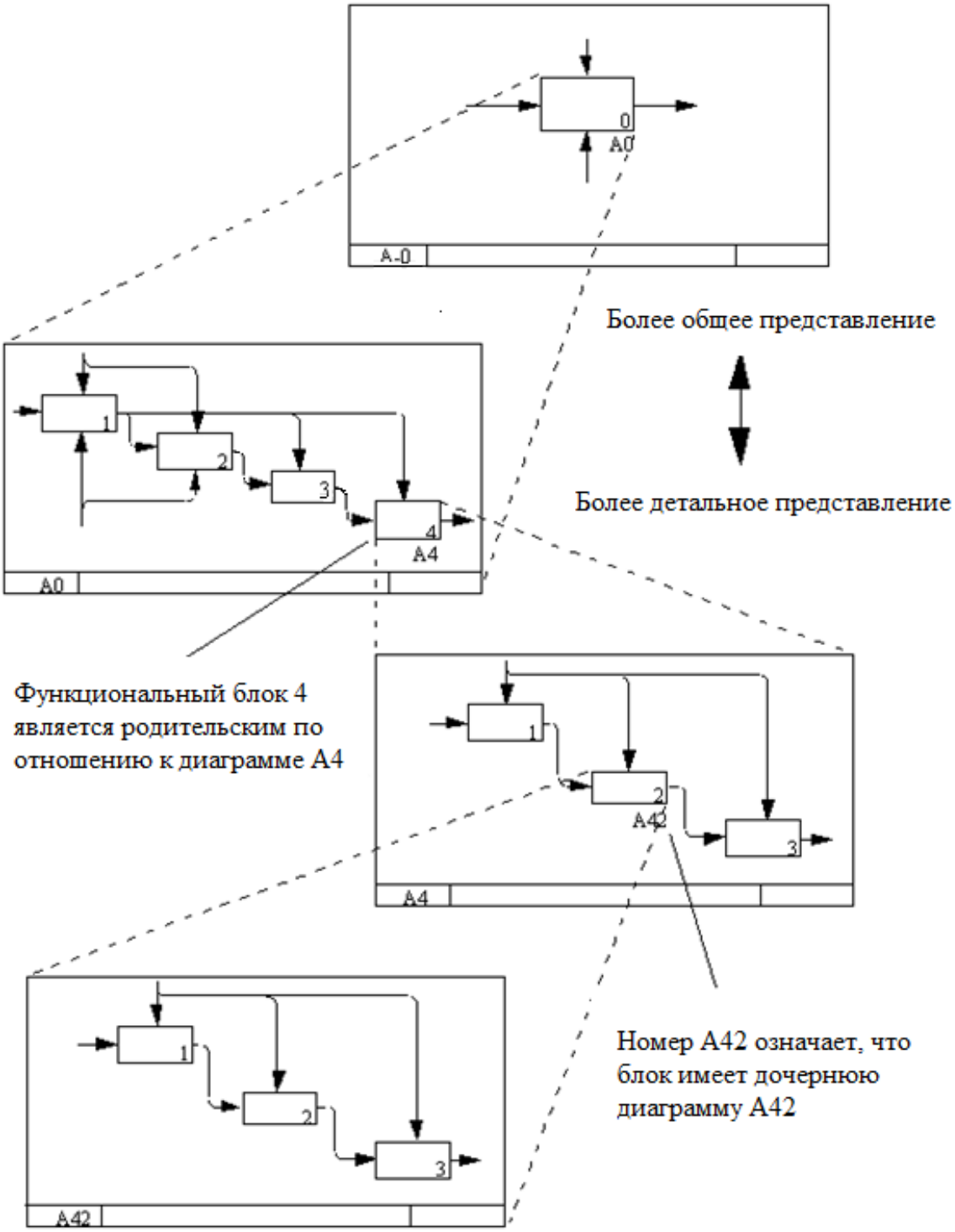


Рисунок 17.5 - Декомпозиция функциональных блоков

Стрелки, помещенные в «туннель»

Туннель - круглые скобки в начале и|или окончании стрелки. Туннельные стрелки означают, что данные, выраженные этими стрелками, не рассматриваются на родительской диаграмме и|или на дочерней диаграмме. Стрелка, помещенная в туннель там, где она присоединяется к блоку, означает, что данные, выраженные этой стрелкой, не обязательны на следующем уровне декомпозиции. Стрелка, помещаемая в туннель на свободном конце, означает,

что выраженные ею данные отсутствуют на родительской диаграмме. Туннель используется, чтобы не загромождать диаграмму.

Правила построения диаграмм

1. В составе модели должна присутствовать контекстная диаграмма А-0, которая содержит только один блок.
2. Блоки на диаграмме должны располагаться по диагонали – от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров. Блоки на диаграмме, расположенные вверху слева доминируют над блоками, расположенными внизу справа. Расположение блоков на листе диаграммы отражает авторское понимание доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.
3. Не контекстные диаграммы должны содержать не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм на уровне, доступном для чтения, понимания и использования.
4. Каждый блок не контекстной диаграммы получает номер, помещаемый в правом нижнем углу; порядок нумерации - от верхнего левого к нижнему правому блоку (номера от 1 до 6).
5. Каждый блок, подвергнутый декомпозиции, должен иметь ссылку на дочернюю диаграмму; ссылка (например, узловой номер, С-номер или номер страницы) помещается под правым нижним углом блока.
6. Имена блоков (выполняемых функций) и метки стрелок должны быть уникальными. Если метки стрелок совпадают, это значит, что стрелки отображают тождественные данные.
7. При наличии стрелок со сложной топологией целесообразно повторить метку для удобства ее идентификации.
8. Следует обеспечить максимальное расстояние между блоками и поворотами стрелок, а также между блоками и пересечениями стрелок для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные стрелки.
9. Блоки всегда должны иметь хотя бы одну управляющую и одну выходную стрелку, но могут не иметь входных стрелок.
10. Если одни и те же данные служат и для управления, и для входа, вычерчивается только стрелка управления. Этим подчеркивается управляющий характер данных и уменьшается сложность диаграммы.
11. Максимально увеличенное расстояние между параллельными стрелками облегчает размещения меток, их чтение и позволяет проследить пути стрелок.
12. Стрелки связываются (сливаются), если они представляют сходные данные и их источник не указан на диаграмме.
13. Обратные связи по управлению должны быть показаны стрелками, расположенными сверху и над блоком. Обратные связи по входу должны быть показаны стрелками, расположенными внизу и под блоком. Так же показываются обратные связи посредством механизма. Таким образом

обеспечивается показ обратной связи при минимальном числе линий и пересечений.

Задание 1. По согласованию с преподавателем выбрать предметную область для построения функциональной диаграммы. Составить контекстную функциональную диаграмму для выбранной предметной области. Для построения диаграммы можно использовать средства текстового процессора Ms Word, любой графический редактор, или средство проектирования Ramus.

Задание 2. Произвести декомпозицию функциональных блоков для диаграммы, построенной по заданию 1. Получить декомпозицию с количеством блоков более трех.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;

2) функциональной контекстной диаграммы, построенной при выполнении задания 1.

3) функциональной диаграммы, содержащей дочерние и родительские блоки, построенной при выполнении задания 2.

Вопросы для защиты работы:

1. Перечислите основные компоненты функциональной диаграммы.
2. Какие диаграммы называются контекстными?
3. Какие диаграммы называются родительскими?
4. Какие диаграммы называются дочерними?
5. Каким образом на функциональных диаграммах отображается доминирование одних функциональных блоков над другими?

Лабораторная работа 18
РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Цель: изучить правила написания технического задания (ТЗ) на разработку программного обеспечения; изучить ГОСТ 19.201–78; разработать документ «Техническое задание».

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

На техническое задание существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». В соответствии с этим стандартом техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки.

При необходимости допускается в техническое задание включать приложения. Рассмотрим пример: Разработать техническое задание на создание системы «Учет успеваемости студентов». Система предназначена для оперативного учета успеваемости студентов в сессию деканом, заместителями декана по курсам и сотрудниками деканата. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому. Текст технического задания приведен ниже.

1. ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку системы учета успеваемости студентов, предназначенной для сбора и хранения информации о ходе сдачи экзаменационной сессии. Предполагается, что использовать данную систему будут сотрудники деканата, декан и его заместители. Во время сессии необходимо получение оперативной информации о ходе ее сдачи студентами, однако выполнение такого контроля вручную требует значительного времени. Автоматизированная система учета успеваемости позволит улучшить качество контроля сдачи сессии со стороны куратора и деканата и обеспечит получение сведений о динамике работы каждого студента, группы и курса в целом. Кроме того, хранение информации о сдаче сессий в течение всего времени обучения позволит осуществлять автоматическую генерацию справок о прослушанных курсах и приложений к диплому выпускника.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Система разрабатывается на основании приказа декана факультета и в соответствии с планом мероприятий по совершенствованию учебного процесса на 2012-2013 учебный год.

3. НАЗНАЧЕНИЕ

Система предназначена для хранения и обработки сведений об успеваемости студентов учебных групп факультета в течение всего срока обучения. Обработанные сведения об успеваемости студентов могут быть использованы для оценки успеваемости каждого студента, группы, курса и факультета в целом.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1. Требования к функциональным характеристикам.

4.1.1. Система должна обеспечивать возможность выполнения следующих функций:

- инициализацию системы (ввод списков групп, перечней изучаемых дисциплин в
- соответствии с учебными планами и т. п.);
- ввод и коррекцию текущей информации о ходе сдачи сессии конкретными
- студентами;
- хранение информации об успеваемости в течение времени обучения студента;
- получение сведений о текущем состоянии сдачи сессии студентами.

4.1.2. Исходные данные:

- списки студентов учебных групп;
- учебные планы кафедр - перечень предметов и контрольных мероприятий по каждому предмету;
- расписания сессий;
- текущие сведения о сдаче сессии каждым студентом.

4.1.3. Результаты:

- итоги сдачи сессии конкретным студентом;
- итоги сдачи сессии студентами конкретной группы;
- процент успеваемости по всем студентам группы при сдаче конкретного предмета в целом на текущий момент;
- проценты успеваемости по всем группам специальности на текущий момент;
- проценты успеваемости по всем группам курса на текущий момент;
- проценты успеваемости по всем курсам и в целом по факультету на текущий момент;
- список задолжников группы на текущий момент;
- список задолжников курса на текущий момент.

4.2. Требования к надежности

4.2.1.Предусмотреть контроль вводимой информации.

4.2.2.Предусмотреть блокировку некорректных действий пользователя при работе с системой.
4.2.3.Обеспечить целостность хранимой информации.
4.3. Требования к составу и параметрам технических средств
4.3.1.Система должна работать на IBM совместимых персональных компьютерах.
4.3.2.Минимальная конфигурация:

- тип процессора - Pentium и выше;
- объем оперативного запоминающего устройства - 32 Мб и более.

4.4. Требования к информационной и программной совместимости
Система должна работать под управлением семейства операционных систем Win 32 (Windows 95, Windows 98, Windows 2000, Windows NT и т. п.).
5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ
5.1.Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии.
5.2.Программная система должна включать справочную информацию о работе и подсказки пользователю.
5.3.В состав сопровождающей документации должны входить:
5.3.1.Пояснительная записка на 25-30 листах, содержащая описание разработки.
5.3.2.Руководство системного программиста.
5.3.3.Руководство пользователя.
5.3.4.Графическая часть на трех листах формата А1:
5.3.4.1.Схема структурная программной системы.
5.3.4.2.Диаграмма компонентов данных.
5.3.4.3.Формы интерфейса пользователя.

Задание. Составить техническое задание на разработку программного обеспечения для информационной системы по своему варианту.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:
1) титульного листа;
2) технического задания.

Вопросы для защиты работы

1. Назовите, какой раздел технического задания можно считать основным и почему.
2. Какую информацию должны содержать остальные разделы?
3.В чем основная сложность разработки технического задания?

Лабораторная работа 19
ДОКУМЕНТИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ. РАЗРАБОТКА РУКОВОДСТВА АДМИНИСТРАТОРА

Цель: изучить стандарты и правила документирования программных продуктов, разработать документ «Руководство администратора».

Формируемые компетенции: УК-2.1; УК-2.2; УК- 2.3; УК-3.1; УК- 3.2; ОПК-5.1; ОПК-6.1; ОПК -6.2; ОПК- 8.1; ОПКД- 5.1; ОПКД- 5.2; ОПКД- 6.1; ОПКД- 6.2

Теоретическая часть:

Основу отечественной нормативной базы в области документирования программных средств составляет комплекс стандартов Единой системы программной документации (ЕСПД).

В состав ЕСПД входят:

- основополагающие и организационно-методические стандарты;
- стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных;
- стандарты, обеспечивающие автоматизацию разработки программных документов.

Стандарты ЕСПД (как и другие ГОСТы) подразделяются на группы (таблица 19.1).

Таблица 19.1 - Группы ЕСПД

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7	Резервные группы
8	
9	Прочие стандарты

Обозначение стандарта ЕСПД должно состоять из:

- из числа 19 (присвоенных классу стандартов ЕСПД);
- одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной в таблице;
- двузначного числа (после тире), указывающего год регистрации стандарта.

Перечень документов ЕСПД очень обширен, в него, в частности, входят следующие ГОСТы:

ГОСТ 19.001-77 ЕСПД. Общие положения.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77 ЕСПД. Стадии разработки.

ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и

оформлению.
ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

2.1. Руководство администратора

2.1.1 Общие сведения о руководстве администратора

Руководство администратора должно обеспечить возможность администратору системы управлять ее функционированием.

Документация - Системная или программная|эксплуатационная.

Предмет - Система и|или программные средства для управления ее работой

Аудитория - ответственный пользователь системы, обеспечивающий ее целевое применение (*администратор*).

2.1.2. Цели и задачи

Работа администратора многопользовательской системы, как правило, заключается в управлении учетными записями других пользователей, предоставлении им полномочий на доступ к данным и выполнение операций, а также в исправлении сделанных ими ошибок.

2.1.3. Содержание документа

В Руководстве администратора системы обязательно должны быть описаны:

- назначение и порядок применения системы;
- общие принципы и логика работы системы;
- обязанности администратора и связанные с ними операции;
- обязательность, регулярность и очередность выполнения всех операций;
- порядок выполнения каждой операции;
- проблемы в работе системы и способы их решения.

Руководство по административному модулю программного или программно-аппаратного комплекса содержит примерно те же сведения, но в более общем виде. Например, в нем должно быть объяснено, как создать учетную запись пользователя, но не может быть указано, когда это следует делать. Такая конкретика возникает только при внедрении продукта и отражается в технологических инструкциях или регламентах.

2.1.4. Методика и стиль изложения

При составлении руководства администратора особое внимание следует уделить описанию системы прав доступа и логике их назначения пользователям.

2.1.5. Типовая структура

Структура руководства администратора существенным образом зависит от того, как устроена система, и какого обслуживания она требует.

Поэтому приведенные здесь структуры следует рассматривать не как типовые, а, скорее, как типичные.

Структура руководства администратора системы:

– Назначение системы.
– Принципы функционирования системы.
– Обязанности и задачи администратора.
– Обслуживание системы.
– Настройка параметров работы системы.
– Ведение нормативно-справочной информации.
– Учетные записи пользователей и управление ими.
– Назначение пользователям прав доступа.
– Загрузка и выгрузка данных.
– Проблемы в работе системы и способы их решения.

Структура руководства по административному модулю программного или аппаратно-программного комплекса:

– Общие сведения о комплексе.
– Функционирование комплекса в рамках системы.
– Интерфейс пользователя административного модуля.
– Задачи по обслуживанию системы.
– Настройка параметров работы системы.
– Ведение нормативно-справочной информации.
– Учетные записи пользователей и управление ими.
– Назначение пользователям прав доступа.
– Загрузка и выгрузка данных.
– Типичные проблемы в работе системы и способы их решения.

В разделе 2 можно рассмотреть несколько наиболее типичных случаев применения комплекса и перечислить основные обязанности администратора системы в каждом из них.

Методика и порядок выполнения работы

Изучить теоретический материал, представленный в лекциях, в разделе 24 «Стандартизация и сертификация программного обеспечения», а также теоретическое обоснование к данной лабораторной работе. Составить документ «Руководство администратора» для того же варианта, что и при разработке Технического задания в лабораторной работе 1.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) руководства администратора.

Вопросы для защиты работы:

1. Что представляет собой ЕСПД?
2. Какие стандарты входят в ЕСПД?
3. Какова цель документирования ПО?
4. Для кого предназначено руководство администратора?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Болодурина И.П., Проектирование компонентов распределенных информационных систем [Электронный ресурс]: учеб. пособие/ И.П. Болодурина, Оренбург:ОГУ, 2012
2. URL: http://www.iprbookshop.ru/30122.html
3. Гриценко Ю.Б., Методические указания по выполнению лабораторных работ, практических занятий и организации самостоятельной работы студентов по дисциплине «Архитектура предприятия»: [Электронный ресурс], Томск, 2012
4. URL:http://aoi.tusur.ru/upload/methodical_materials/MU_Arkhitektura_predprijatija_2012 Magistry_file__325_1392.pdf
5. Архитектура предприятия. Практика для экзамена. URL:https://www.studocu.com/en/document/finansovyy-universitet-pri-pravitelьstve-rf/arkhitektura-predpriyatiya/summaries/arkhitektura-predpriyatiya-praktika-dlya-ekzamena/2595929/view
6. http://odiplom.ru/lab/obszaya-harakteristika-predpriyatiya.html
7. URL: https://megalektsii.ru/s3230t2.htm
8. http://filling-form.ru/blank_dov/70916/index.html?page=4
9. Тестирование от компании Google https://testing.googleblog.com/2008
10. Google testing framework (gtest) https://habr.com/post/119090/
11. Романова А.Н., Одинцова Б.Е.. Информационные системы в экономике: Учебное пособие/; 2-е изд.; перераб. и доп. - М.: Вузовский учебник, 2008. - 411с. - (Вузовский учебник).
12. Косяков А., Уильям Н. Свит, Сэмюэль Дж. Сеймур, Стивен М. Бимер., «Системная инженерия. Принципы и практика» – М.:ДМК Пресс,2014- 624 с.
13. Лоусон Г., Путешествие по системному ландшафту. — ДМК-Пресс, 2013.
14. Батоврин В.К. Толковый словарь по системной и программной инженерии: учеб. Пособие для ВУЗов. М.: ДМК Пресс, 2012. 280с.
15. Батоврин В.К. Стандарты системной инженерии: серия докладов (зеленых книг) в рамках проекта «Промышленный и технологический форсайт Российской Федерации» / В.К. Батоврин; под ред. М.С. Липецкой, К.А. Ивановой; Фонд «Центр стратегических разработок «Северо-Запад». — СПб., 2012. — Вып. 4. — 64 с. — (Серия докладов в рамках проекта «Промышленный и технологический форсайт Российской Федерации») https://www.csr-nw.ru/files/csr/file_content_1269.pdf
16.Леоненков А.В. Самоучитель UML 2. СПБ: БХВ-СПб, 2007. - 576 с.
17. Руководство к СВОДУ ЗНАНИЙ ПО УПРАВЛЕНИЮ ПРОЕКТОМ (РУКОВОДСТВО PMBOK® Шестое издание). https://biconsult.ru/files/datavault/PMBOK-6th-Edition-Ru.pdf
18. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. Издательство: ДМК Пресс, 2007 г.

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

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
К САМОСТОЯТЕЛЬНОЙ РАБОТЕ СТУДЕНТОВ

по дисциплине
«Системная инженерия»

Направление подготовки 09.04.02 «Информационные системы и технологии»
Направленность (профиль) «Искусственный интеллект, математическое моделирование и суперкомпьютерные технологии в разработке информационных систем»

Ставрополь, 2022 г.

ВВЕДЕНИЕ

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторной работы теоретического материала, а также подготовку отчета по выполненным на лабораторном и практическом занятии заданиям.

Подготовленный материал оформляется в виде отчета. Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине «Системная инженерия» имеют целью:

- закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;
- привить практические навыки при проведении анализа и разработки сложных систем.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Системная инженерия».

Современные стандарты подготовки магистров предусматривают значительный объем внеаудиторной работы. Освоение программы курса предполагает получение как теоретических знаний по проблемам системной инженерии, так и формирование навыков практической работы. Поэтому самостоятельная работа в рамках курса ориентирована как на теоретический, так и на практический аспект.

Самостоятельная работа студентов – это выполнение теоретических и практических заданий студентами по усвоению изучаемой дисциплины «Системная инженерия».

Целью самостоятельной работы является закрепление и углубление знаний, полученных студентами на лекциях, подготовка к текущим практическим занятиям, промежуточным формам контроля знаний и к итоговому контролю.

Дидактические цели самостоятельных занятий:

- формирование профессиональных умений;
- формирование умений и навыков самостоятельного умственного труда;
- мотивирование регулярной целенаправленной работы по освоению специальности;
- развитие самостоятельности мышления;
- формирование убежденности, волевых черт характера, способности к самоорганизации;
- овладение практическими навыками по применению информационных технологий при разработке сложных систем.

Самостоятельная работа включает те разделы курса, которые не получили достаточного освещения на лекциях по причине ограниченности лекционного времени и большого объема изучаемого материала.

Методическое обеспечение самостоятельной работы по дисциплине состоит из:

1) определения учебных вопросов, которые студенты должны изучить самостоятельно;

2) списка необходимой учебной литературы, обязательной для проработки и изучения;

3) списка дополнительной научной литературы, ГОСТов и нормативно-справочной литературы, к которой студенты могут обращаться по желанию, если у них возникает интерес в данной теме;

Самостоятельная работа студента – это особым образом организованная деятельность, включающая в свою структуру такие компоненты, как:

- уяснение цели и поставленной учебной задачи;
- четкое и системное планирование самостоятельной работы;
- поиск необходимой учебной и научной информации;

- освоение собственной информации и ее логическая переработка;
- использование методов учебной и научно-исследовательской работы для решения поставленных задач;
- выработка собственной позиции по поводу полученной задачи;
- представление, обоснование и защита полученного решения;
- проведение самоанализа и самоконтроля

Виды самостоятельных работ по учебной дисциплине:

- работа с понятийным аппаратом;
- подготовка к лекциям;
- подготовка к лабораторным работам;
- проектирование фрагментов исследовательской деятельности;
- анализ практик системной инженерии;
- подготовка научного доклада.

Контроль знаний студентов включает формы текущего и итогового контроля. Текущий контроль осуществляется в процессе изучения курса и включает в себя оценку работы студентов на практических занятиях (круглый стол, собеседование, групповые научные проекты, тестирование) и лабораторных работах.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

Комплексное изучение предлагаемой студентам учебной дисциплины «Системная инженерия» предполагает овладение материалами лекций, учебников, программы, творческую работу студентов в ходе проведения практических занятий, лабораторных работ, а также систематическое выполнение заданий для самостоятельной работы.

В ходе лекций раскрываются основные вопросы в рамках рассматриваемых тем, делаются акценты на наиболее сложные и интересные положения изучаемого материала, которые должны быть приняты студентами во внимание. Материалы лекций являются основой для подготовки студентов к практическим занятиям.

Работа с конспектом лекций

Просмотрите конспект сразу после занятий. Отметьте материал конспекта лекций, который вызывает затруднения для понимания. Попытайтесь найти ответы на затруднительные вопросы, используя предлагаемую литературу. Если самостоятельно не удалось разобраться в материале, сформулируйте вопросы и обратитесь на текущей консультации или на ближайшей лекции за помощью к преподавателю.

Каждую неделю отводите время для повторения пройденного материала, проверяя свои знания, умения и навыки по контрольным вопросам и тестам.

Выполнение лабораторных работ по дисциплине «Системная инженерия»

На первом занятии получите у преподавателя задания по курсу, планы подготовки к лабораторным работам. Перед лабораторными работами изучите теорию вопроса, предполагаемого к изучению, ознакомьтесь с целью и методикой выполнения работы.

Целью лабораторных работ является:

- овладение навыкам практической работы;
- изучение современных инструментальных средств и технологий, программного и аппаратного обеспечения информационных систем.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО НАПИСАНИЮ И ОФОРМЛЕНИЮ ДОКЛАДА

1. Общие положения

1.1. Доклад, как вид самостоятельной работы в учебном процессе, способствует

формированию навыков исследовательской работы, расширяет познавательные интересы, учит критически мыслить.

1.2. При написании доклада по заданной теме студент составляет план, подбирает основные источники.

1.3. В процессе работы с источниками систематизирует полученные сведения, делает выводы и обобщения.

1.4. К докладу по крупной теме могут привлекать несколько студентов, между которыми распределяются вопросы выступления.

2. Выбор темы доклада

2.1. Тематика доклада обычно определяется преподавателем, но в определении темы инициативу может проявить и студент.

2.2. Прежде чем выбрать тему доклада, автору необходимо выявить свой интерес, определить, над какой проблемой он хотел бы поработать, более глубоко ее изучить.

3. Этапы работы над докладом

3.1. Формулирование темы, причем она должна быть не только актуальной по своему значению, но и оригинальной, интересной по содержанию.

3.2. Подбор и изучение основных источников по теме (как правильно, при разработке доклада используется не менее 8-10 различных источников).

3.3. Составление списка использованных источников.

3.4. Обработка и систематизация информации

3.5. Разработка плана доклада.

3.6. Написание доклада.

3.7. Публичное выступление с результатами исследования.

4. Структура доклада:

- титульный лист

- оглавление (в нем последовательно излагаются названия пунктов доклада, указываются страницы, с которых начинается каждый пункт);

- введение (формулирует суть исследуемой проблемы, обосновывается выбор темы, определяются ее значимость и актуальность, указываются цель и задачи доклада, дается характеристика используемой литературы);

- основная часть (каждый раздел ее, доказательно раскрывая отдельную проблему или одну из ее сторон, логически является продолжением предыдущего; в основной части могут быть представлены таблицы, графики, схемы);

- заключение (подводятся итоги или дается обобщенный вывод по теме доклада, предлагаются рекомендации);

- список использованных источников

5. Структура и содержание доклада

5.1. Введение - это вступительная часть научно-исследовательской работы. Автор должен приложить все усилия, чтобы в этом небольшом по объему разделе показать актуальность темы, раскрыть практическую значимость ее, определить цели и задачи эксперимента или его фрагмента.

5.2. Основная часть. В ней раскрывается содержание доклада. Как правило, основная часть состоит из теоретического и практического разделов. В теоретическом разделе раскрываются история и теория исследуемой проблемы, дается критический анализ литературы и показываются позиции автора. В практическом разделе излагаются методы, ход, и результаты самостоятельно проведенного эксперимента или фрагмента.

В основной части могут быть также представлены схемы, диаграммы, таблицы, рисунки и т.д.

5.3. В заключении содержатся итоги работы, выводы, к которым пришел автор, и рекомендации. Заключение должно быть кратким, обязательным и соответствовать поставленным задачам.

5.4. Список использованных источников представляет собой перечень использованных книг, статей, фамилии авторов приводятся в алфавитном порядке, при этом все источники даются под общей нумерацией литературы. В исходных данных источника указываются фамилия и инициалы автора, название работы, место и год издания.

5.5. Приложение к докладу оформляются на отдельных листах, причем каждое должно иметь свой тематический заголовок и номер, который пишется в правом верхнем углу, например: «Приложение 1»

6. Требования к оформлению доклада

6.1. Объем доклада может колебаться в пределах 5-15 печатных страниц; все приложения к работе не входят в ее объем.

6.2. Доклад должен быть выполнен грамотно, с соблюдением культуры изложения.

6.3. Обязательно должны иметься ссылки на используемую литературу.

6.4. Должна быть соблюдена последовательность написания библиографического аппарата.

7. Должна быть подготовлена презентация доклада

8. Критерии оценки доклада

- актуальность темы исследования;
- соответствие содержания теме;
- глубина проработки материала; правильность и полнота использования источников;
- соответствие оформления доклада стандартам.

По усмотрению преподавателя доклады могут быть представлены на семинарах, научно-практических конференциях, а также использоваться как зачетные работы по пройденным темам.

Примерные темы докладов по дисциплине «Системная инженерия»

2 семестр

Раздел 1. Основные понятия системной инженерии

1. Проблемы разработки сложных систем.
2. Особенности Российского информационного рынка.
3. Проблема подготовки системных инженеров в РФ.
4. Тенденции развития нормативной базы системной инженерии.

Раздел 2. Практики системной инженерии

1. Обязательные процессы системной инженерии.
2. Цифровая экономика.
3. Эксплуатационные требования.
4. Непрерывный менеджмент рисков. ISO/IEC 24765

3 семестр

Раздел 3. Принятие решений в системной инженерии

1. Изменение жизненного цикла ПО при использовании CASE-технологий.
2. Блокчейн.
3. Проблема сопровождения программных средств.
4. Анализ потребностей при зарождении новой системы.
5. Проблема морального устаревания традиционных систем.
6. Моделирование на протяжении разработки новой системы.

Раздел 4. Стандарты в системной инженерии

1. Стандарт датацентрического информационного моделирования и интеграции данных ISO 15926.
2. Госпрофиль ВОС.
3. Международная стандартизация и проблемы информационной совместимости.
4. Цели и задачи стандартизации в Российской Федерации.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО РАБОТЕ С ЛИТЕРАТУРОЙ И ИСТОЧНИКАМИ

Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Системная инженерия», которые прямо или косвенно относятся к изучаемой теме.

При изучении литературы и источников студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, необходимо выписать их в виде цитат с полным указанием библиографических источников.

Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам.

Основой технологии интенсификации обучения на платформе цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Системной инженерии».

Работа с учебно-иллюстрационными материалами имеет следующие этапы.

1. Изучение теоретических основ учебного материала в аудитории: изложение преподавателем изучаемого материала студентам с объяснением по опорному конспекту;

2. Самостоятельная работа: индивидуальная работа студентов по опорному конспекту; фронтальное закрепление по блокам опорного конспекта.

3. Первое повторение - воспроизведение содержания заданной темы опорного конспекта по памяти.

4. Устное проговаривание материала опорного конспекта – необходимый этап внешне речевой деятельности при усвоении учебного материала.

5. Второе повторение – взаимоопрос и взаимопомощь студентов друг другу.

Применение учебно-иллюстрационных материалов позволяет обобщить сложный по содержанию материал, активизировать мыслительную деятельность студентов.

Необходимо помнить, что главное для студента в самостоятельной работе с рекомендуемой литературой и источниками - это формирование своего индивидуального стиля, который может стать основой в будущей профессиональной деятельности.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЭКЗАМЕНУ

Контроль и оценка знаний студентов являются неотъемлемой составной частью учебно-воспитательного процесса в вузе. Экзаменационная сессия – самая ответственная пора в студенческой жизни: она является итогом большой работы студентов в семестре. Экзамен – это метод проверки знаний студентов по части или полному курсу учебной дисциплины, произведенный путем постановки устных или письменных вопросов. Он дает

объективную официально фиксируемую оценку успехов студентов за определенный отрезок времени.

Экзамен преследует многогранную цель: во-первых, это – проверка знаний студента. Во-вторых, экзамены сами по себе являются важным звеном в овладении наукой, в-третьих, это продолжение учебного процесса; наконец, он имеют большое значение как фактор стимулирования глубокого изучения предмета.

Подготовка к экзамену состоит из двух взаимосвязанных этапов. Первый – систематический труд на протяжении семестра, учебного года, охватывающий все формы учебного процесса: лекции, изучение и конспектирование рекомендованной литературы, активное участие в семинарских (практических) и лабораторных занятиях.

Второй – подготовка непосредственно перед экзаменом. Она позволяет студентам за сравнительно короткий отрезок времени охватить всю перспективу изученного и лучше понять основные закономерности и явления.

Ограниченность времени (3-4 дня) для непосредственной подготовки к экзамену по предмету требует от студентов спокойно, без нервозной суеты и спешки еще раз внимательно продумать изученный в течение семестра материал, тщательно отработать вопросы, недостаточно изученные или плохо понятые, с тем, чтобы по возможности устранить все пробелы в своих знаниях.

Готовиться надо по строго продуманному графику, последовательно переходя от темы к теме, не пропуская ни одну из них.

Сложные вопросы, недостаточно уясненные в процессе подготовки к экзамену, необходимо записать и получить на них разъяснения у преподавателей во время предэкзаменационных консультаций.

Следует заметить, что студенты, которые не ходят на консультации из-за отсутствия, по их мнению, сложных вопросов, поступают опрометчиво. На консультациях очень часто лектор не только отвечает на заданные вопросы, но и по собственной инициативе разъясняет наиболее трудные разделы курса.

Итак, основной задачей подготовки студентов к экзамену является систематизация знаний учебного материала, его творческое осмысливание. Важнейшим учебным пособием на этом этапе работы студента является собственный конспект прослушанных лекций и самостоятельно проработанных тем курса.

В соответствии с положением об экзаменах их, как правило, принимают преподаватели, читающие курс лекций.

Получив билет, студент должен хорошо продумать содержание поставленных вопросов. Значительное число неудачных ответов объясняется неясным пониманием поставленной проблемы. Правильное понимание вопроса обеспечит успех при ответе на него. При подготовке к ответу на билет нужно составить развернутый план по каждому вопросу

Отвечая на вопросы экзаменационного билета, не следует спешить. Надо излагать материал спокойно, не торопясь, владеть собой, следить за построением фраз. Следует избегать подходов издалека, общих рассуждений. Рекомендуется строить ответы четко, последовательно, конкретно, по возможности исчерпывающе. Вместе с тем весьма желательно быстро и правильно иллюстрировать свой ответ примерами.

От экзаменующегося требуется: определение понятий, обоснование выдвинутых положений, свободное оперирование фактическим материалом, дать альтернативные подходы по отдельным проблемам. Логичность, стройность, литературная грамотность изложения являются неотъемлемыми чертами полноценного ответа. Нельзя при ответе допускать ни излишней краткости, переходящей в схематизм, ни многословия. И то, и другое не оправдано. Краткость не дает преподавателю возможности понять, владеет ли студент учебным материалом, а многословие может показать, что студент не умеет акцентировать внимание на главном и говорит слишком расплывчато.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная литература:

1 Батоврин, В. К. Системная и программная инженерия. Словарь-справочник Электронный ресурс : Учебное пособие для вузов / В. К. Батоврин. - Системная и программная инженерия. Словарь-справочник,2019-04-19. - Саратов : Профобразование, 2017. - 280 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0129-7, экземпляров неограниченно. Режим доступа: http://www.iprbookshop.ru/63956.html

2 Маглинец, Ю.А. Анализ требований к автоматизированным информационным системам Электронный ресурс : учебное пособие / Ю.А. Маглинец. - Анализ требований к автоматизированным информационным системам,2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 191 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-94774-865-9

Дополнительная литература:

1 Кутузов, А.С. Введение в функциональный анализ: учебное пособие. Москва|Берлин: Директ- Медиа, 2020

2 Дроздова, В. И. (Северо-Кавказский федеральный университет). Системная инженерия: учебное пособие : Направление подготовки 09.04.02 - Информационные системы и технологии. Магистерские программы "Управление данными", "Информационные системы и мультимедиа-технологии в сфере высшего образования", "Робототехнические системы" / В. И. Дроздова, М. Г. Романенко. - Ставрополь : СКФУ, 2016. - 90 с.

3 Меерович, М.И., Шрагина, Л.И. Системное мышление: формирование и развитие: учебное пособие. Москва: СОЛОН-Пресс, 2019

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1 Системная инженерия : Учеб.-метод. пособие : Направление подготовки. 09.04.02 Информационные системы и технологии. Квалификация (степень) выпускника - магистр.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля):

1 ЭБС «ZNANIUM.COM»: [сайт]. URL:http://znanium.com/

2 http://www.intuit.ru – Сайт национального открытого университета «Интуит»

3 Издательство «Открытые системы»: [сайт]. URL: https://www.osp.ru/

4 Научно-техническая библиотека ДГТУ: [сайт]. URL:http://ntb.donstu.ru

5 http://khpi-iip.mipk.kharkiv.edu/library/case/leon /Леоненков А.В. Самоучитель UML

6 http://gradschools.com/search-programs/system-engineering – Сайт учебных материалов по системной инженерии

7 http://argouml.tigris.org – Сайт, описывающий среду объектно-ориентированного моделирования ARGOUML