**МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РФ**

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Пензенский государственный университет»

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по учебной (ознакомительной) практике

Выполнил: студент Мельхов А. А.

Направление: 09.03.02

Группа: 19ВИ1

Руководитель практики: к.т.н., доцент Самуйлов С. В.

Работа защищена

Оценка

Пенза, 2020

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную (ознакомительную) практику

Задание 1

1. Исследовать современные технологии объектно-ориентированного анализа и проектирования информационных систем. Изучить нотацию и семантику языка UML. Сделать обзор особенностей применения данного языка в качестве средства объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

2.Провести анализ предметной области «ВУЗ». Для заданной предметной области построить диаграмму вариантов использования и диаграмму последовательности.

Задание 2.

Изучить информационную технологию обработки текстовой информации и технической документации. Сделать обзор ключевых возможностей и особенностей применения данной технологии в различных программных средах. Изучить стандарт ГОСТ 2.105−95 «Общие требования к текстовым документам». Оформить отчет по практике в соответствии с ГОСТ 2.105−95 «Общие требования к текстовым документам».

Задание 3. Получить сертификат открытого университета ИНТУИТ (Pascal, UML).

Содержание

[Содержание 3](#_Toc45315055)

[Введение 4](#_Toc45315056)

[1. Объектно-ориентированный анализ и проектирование 6](#_Toc45315057)

[1.1. Некоторые нотации ООАП 7](#_Toc45315058)

[1.2. История языка UML 8](#_Toc45315059)

[1.3. Канонические диаграммы языка UML 12](#_Toc45315060)

[1.4. Особенности графического изображения диаграмм языка UML 12](#_Toc45315061)

[1.5. Диаграмма вариантов использования 16](#_Toc45315062)

[1.6. Диаграмма последовательности 17](#_Toc45315063)

[2. Анализ предметной области «ВУЗ» 20](#_Toc45315064)

[2.1. Диаграмма вариантов использования предметной области «ВУЗ» 20](#_Toc45315065)

[2.2. Диаграмма последовательности предметной области «ВУЗ» 21](#_Toc45315066)

[3. Технология обработки текстовой информации 23](#_Toc45315067)

[Заключение 26](#_Toc45315068)

[Список использованных источников 27](#_Toc45315069)

[Приложение А. Результаты прохождения обучающего курса 28](#_Toc45315070)

Введение

В период с 29.06.19. по 12.07.19. Мельхов Антон Алексеевич, студент Пензенского Государственного Университета, группы 19ВИ1 проходил учебную практику. Учебная практика ставит перед собой цели исследовать современные технологии объектно-ориентированного анализа и проектирования информационных систем. Объектом практической работы был язык графического описания UML.

**UML -** язык [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), для [моделирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения [организационных структур](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0).

**UML** является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

Главными в разработке UML были следующие цели:

* предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий разрабатывать осмысленные модели и обмениваться ими;
* предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;
* обеспечить независимость от конкретных языков программирования и процессов разработки;
* обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма);
* стимулировать рост рынка объектно-ориентированных инструментальных средств;
* интегрировать лучший практический опыт.

Язык UML находится в процессе стандартизации, проводимом OMG (Object Management Group) – организацией по стандартизации в области объектно-ориентированных методов и технологий, в настоящее время принят в качестве стандартного языка моделирования и получил широкую поддержку в индустрии ПО.

Язык UML принят на вооружение практически всеми крупнейшими компаниями – производителями ПО (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase и др.). Кроме того, практически все мировые производители CASE-средств, помимо Rational Software (Rational Rose), поддерживают UML в своих продуктах (Paradigm Plus 3.6, System Architec, Microsoft Visual Modeler for Visual Basic, Delphi, PowerBuilder и др.). Создатели UML представляют его как язык для определения, представления, проектирования и документирования программных систем, организационно-экономических, технических и др.

За время выполнения практического задания, мы должны разобраться в ООАиП, познакомиться с возможными нотациями, составить несколько диаграмм для предметной области «ВУЗ», используя UML, узнать про текстовые редакторы и правила оформления текста, ознакомиться с ГОСТ 2.105−95 «Общие требования к текстовым документам», оформить отчет по практике в соответствии с ним, а также пройти в Национальном Открытом Университете ИНТУИТ (НОУ ИНТУИТ) два курса (Pascal, UML).

1. Объектно-ориентированный анализ и проектирование

Необходимость анализа предметной области до начала написания программы была осознана задолго до появления методологии объектно-ориентированного анализа и проектирования. При выполнении масштабных программных проектов было замечено, что, например, процесс разработки компьютерной базы данных существенно отличается от написания программного кода для решения вычислительной (даже весьма алгоритмически сложной) задачи. При проектировании базы данных возникает необходимость в предварительной разработке информационной модели предметной области - концептуальной схемы, которая отражала бы взаимосвязи между объектами предметной области, существенные с точки зрения поддержания целостности данных и реализации пользовательских поисковых запросов.

Объектный подход к разработке сложных программных систем безусловно предполагает, что непосредственное программирование (написание кода) начинается далеко не сразу, а этапы анализа и моделирования предметной области, предшествующие программированию, не менее важны и сложны.

Объектный подход применяется на всех основных стадиях жизненного цикла ПО и включает в себя три ключевых понятия:

* OOA (object oriented analysis) - объектно-ориентированный анализ.
* OOD (object oriented design) - объектно-ориентированное проектирование.
* OOP (object oriented programming) - объектно-ориентированное программирование.

**Объектно-ориентированный анализ** - это методология анализа предметной области, при которой требования к проектируемой системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области.

**Объектно-ориентированное проектирование** - это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

**Объектно-ориентированное программирование** - это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Предметом рассмотрения нашей дисциплины являются первые две составляющие объектного подхода – объектно-ориентированный анализ и объектно-ориентированное проектирование (далее – ООАП). [1]

* 1. Некоторые нотации ООАП

**Объектно-ориентированный анализ и проектирование** (ООАП, Object-Oriented Analysis/Design) - технология разработки программных систем, в основу которых положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов

Методология ООАП тесно связана с концепцией автоматизированной разработки программного обеспечения Computer Aided Software Engineering, CASE). К первым CASE-средствам отнеслись с определенной настороженностью. Со временем появились как восторженные отзывы об их применении, так и критические оценки их возможностей. Причин для столь противоречивых мнений было несколько. Первая из них заключается в том, что ранние CASE-средства были простой надстройкой над системой управления базами данных (СУБД). Визуализация процесса разработки концептуальной схемы БД имеет немаловажное значение, тем не менее, она не решает проблем создания приложений других типов. [2]

* 1. История языка UML

Объектно-ориентированные языки моделирования появились в период с середины 70-х до конца 80-х годов, когда исследователи, поставленные перед необходимостью учитывать новые возможности объектно-ориентированных языков программирования и требования, предъявляемые все более сложными приложениями, вынуждены были начать разработку различных альтернативных подходов к анализу и проектированию.

С 1989 по 1994 год число различных объектно-ориентированных методов возросло с десяти более чем до пятидесяти. Тем не менее, многие пользователи испытывали затруднения при выборе языка моделирования, который бы полностью соответствовал их потребностям, что послужило причиной так называемой «войны методов». В результате этих войн появилось новое поколение методов, среди которых особое значение приобрели языки Booch, созданный Грейди Бучем (Grady Booch), OOSE (Object-Oriented Software Engineering), разработанный Айваром Джекобсоном (Ivar Jacobson) и ОМТ (Object Modeling Technique), автором которого является Джеймс Рамбо (James Rumbaugh). Кроме того, следует упомянуть языки Fusion, Шлаера-Меллора (Shlaer-Mellor) и Коада-Йордона (Coad-Yourdon). Каждый из этих методов можно считать вполне целостным и законченным, хотя любой из них имеет не только сильные, но и слабые стороны.

Выразительные возможности метода Буча особенно важны на этапах проектирования и конструирования модели. OOSE великолепно приспособлен для анализа и формулирования требований, а также для высокоуровневого проектирования. ОМТ-2 оказался особенно полезным для анализа и разработки информационных систем, ориентированных на обработку больших объемов данных.

Критическая масса новых идей начала формироваться к середине 90-х годов, когда Грейди Буч (компания Rational Software Corporation), Айвар Джекобсон (Objectory) и Джеймс Рамбо (General Electric) предприняли попытку объединить свои методы, уже получившие мировое признание как наиболее перспективные в данной области. Являясь основными авторами языков Booch, OOSE и ОМТ, партнеры попытались создать новый, унифицированный язык моделирования и руководствовались при этом тремя соображениями.

Во-первых, все три метода, независимо от желания разработчиков, уже развивались во встречном направлении. Разумно было продолжать эту эволюцию вместе, а не по отдельности, что помогло бы в будущем устранить нежелательные различия и, как следствие, неудобства для пользователей.

Во-вторых, унифицировав методы, проще было привнести стабильность на рынок инструментов объектно-ориентированного моделирования, что дало бы возможность положить в основу всех проектов единый зрелый язык, а создателям инструментальных средств позволило бы сосредоточиться на более продуктивной деятельности.

Наконец, следовало полагать, что подобное сотрудничество приведет к усовершенствованию всех трех методов и обеспечит решение задач, для которых любой из них, взятый в отдельности, был не слишком пригоден.

Начав унификацию, авторы поставили перед собой три главные цели:

* моделировать системы целиком, от концепции до исполняемого артефакта, с помощью объектно-ориентированных методов;
* решить проблему масштабируемости, которая присуща сложным системам, предназначенным для выполнения ответственных задач;
* создать такой язык моделирования, который может использоваться не только людьми, но и компьютерами.

Изобретение языка для объектно-ориентированного анализа и проектирования не слишком отличается от разработки языка программирования. Во-первых, требовалось ограничить задачу. Следует ли включать в язык возможность спецификации требований? Должен ли язык позволять визуальное программирование? Во-вторых, было необходимо найти точку равновесия между выразительной мощью и простотой. Слишком простой язык ограничил бы круг решаемых с его помощью задач, а слишком сложный мог ошеломить неискушенного разработчика. Кроме того, при объединении существующих методов приходилось учитывать наличие уже разработанных с их помощью продуктов. Внесение слишком большого числа изменений могло бы оттолкнуть уже имевшихся пользователей, а сопротивляясь развитию языка, авторы потеряли бы возможность привлекать новых пользователей и делать язык более простым и удобным для применения. Создавая UML, разработчики старались найти оптимальное решение этих проблем.

Официально создание UML началось в октябре 1994 года, когда Рамбо перешел в компанию Rational Software, где работал Буч. Первоначальной целью было объединение методов Буча и ОМТ. Первая пробная версия 0.8 Унифицированного Метода (Unified Method), как его тогда называли, появилась в октябре 1995 года. Приблизительно в это же время в компанию Rational перешел Джекобсон, и проект UML был расширен с целью включить в него язык OOSE. В результате совместных усилий в июне 1996 года вышла версия 0.9 языка UML. На протяжении всего года создатели занимались сбором отзывов от основных компаний, работающих в области конструирования программного обеспечения. За это время стало ясно, что большинство таких компаний сочло UML языком, имеющим стратегическое значение для их бизнеса. В результате был основан консорциум UML, в который вошли организации, изъявившие желание предоставить ресурсы для работы, направленной на создание полного определения UML.

Версия 1.0 языка появилась в результате совместных усилий компаний Digital Equipment Corporation, Hewlett Packard, I-Logix, Intellicprp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments и Unisys. UML 1.0 оказался хорошо определенным, выразительным, мощным языком, применимым для решения большого количества разнообразных задач. В январе 1997 года он был представлен Группе по управлению объектами (Object Management Group, OMG) на конкурс по созданию стандартного языка моделирования.

Между январем и июнем 1997 года консорциум UML расширился, в него вошли практически все компании, откликнувшиеся на призыв OMG, а именно: Andersen Consulting, Ericsson, ObjecTime Limited, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software и Taskon. Чтобы формализовать спецификации UML и координировать работу с другими группами, занимающимися стандартизацией, под руководством Криса Кобрина (Cris Kobryn) из компании MCI Systemhouse и Эда Эйкхолта (Ed Eykholt) из Rational была организована семантическая группа. Пересмотренная версия UML (1.1) была снова представлена на рассмотрение OMG в июле 1997 года. В сентябре версия была утверждена на заседаниях Группы по анализу и проектированию и Комитета по архитектуре OMG, a 14 ноября 1997 года принята в качестве стандарта на общем собрании всех членов OMG.

Дальнейшая работа по развитию UML проводилась Группой по усовершенствованию (Revision Task Force, RTF) OMG под руководством Криса Кобрина. В июне 1998 года вышла версия UML 1.2, а осенью 1998 – UML 1.3. [3]

Формальная спецификация версии UML 2.0 опубликована в августе 2005 года. Семантика языка была значительно уточнена и расширена для поддержки методологии Model Driven Development — MDD. Последняя версия UML 2.5 опубликована в июне 2015 года.

UML 2.4.1 принят в качестве международного стандарта ISO/IEC 19505-1, 19505. [4]

* 1. Канонические диаграммы языка UML

Диаграммы рисуются для визуализации системы с разных точек зрения, затем они отображаются в систему. Обычно диаграмма дает неполное представление элементов, которые составляют систему. Хотя один и тот же элемент может появляться во всех диаграммах, на практике он появляется только в некоторых диаграммах. Теоретически диаграмма может содержать любую комбинацию предметов и отношений, на практике ограничиваются малым количеством комбинаций, которые соответствуют пяти представлениям архитектуры ПС. По этой причине UML включает девять видов диаграмм:

* диаграммы классов;
* диаграммы объектов;
* диаграммы Use Case (диаграммы прецедентов);
* диаграммы последовательности;
* диаграммы сотрудничества (кооперации);
* диаграммы схем состояний;
* диаграммы деятельности;
* компонентные диаграммы;
* диаграммы размещения (развертывания). [5]
  1. Особенности графического изображения диаграмм языка UML

Большинство перечисленных выше диаграмм являются в своей основе графами специального вида, состоящими из вершин в форме геометрических фигур, которые связаны между собой ребрами или дугами. Поскольку информация, которую содержит в себе граф, имеет в основном топологический характер, ни геометрические размеры, ни расположение элементов диаграмм (за некоторыми исключениями, такими как диаграмма последовательностей с метрической осью времени) не имеют принципиального значения.

Для диаграмм языка UML существуют три типа визуальных обозначений, которые важны с точки зрения заключенной в них информации:

* Связи, которые представляются различными линиями на плоскости. Связи в языке UML обобщают понятие дуг и ребер из теории графов, но имеют менее формальный характер.
* Текрт, который содержится внутри границ отдельных геометрических фигур на плоскости. При этом форма этих фигур (прямоугольник, эллипс) соответствует некоторым элементам языка UML (класс, вариант использования) и имеет фиксированную семантику.
* Графические символы, изображаемые вблизи от тех или иных визуальных элементов диаграмм.
* Таким образом, в языке UML используется четыре основных вида графических конструкций:
* Значки или пиктограммы. Значок представляет собой графическую фигуру фиксированного размера и формы. Она не может увеличивать свои размеры, чтобы разместить внутри себя дополнительные символы. Значки могут размещаться как внутри других графических конструкций, так и вне их. Примерами значков могут служить окончания связей элементов диаграмм или некоторые другие дополнительные обозначения (украшения).
* Графические символы на плоскости. Такие двумерные символы изображаются с помощью некоторых геометрических фигур и могут иметь различную высоту и ширину с целью размещения внутри этих фигур других конструкций языка UML. Наиболее часто внутри таких символов помещаются строки текста, которые уточняют семантику или фиксируют отдельные свойства соответствующих элементов языка UML. Информация, содержащаяся внутри фигур, имеет важное значение для конкретной модели проектируемой системы, поскольку регламентирует реализацию соответствующих элементов в программном коде.
* Пути, которые представляют собой последовательности из отрезков линий, соединяющих отдельные графические символы. При этом концевые точки отрезков линий должны обязательно соприкасаться с геометрическими фигурами, служащими для обозначения вершин диаграмм, как принято в теории графов. С концептуальной точки зрения путям в языке UML придается особое значение, поскольку они являются простыми топологическими сущностями. С другой стороны, отдельные части пути или сегменты могут не существовать сами по себе вне содержащего их пути. Пути всегда соприкасаются с другими графическими символами на обеих границах соответствующих отрезков линий. Как отмечалось выше, пути могут иметь в качестве окончания или терминатора специальную графическую фигуру – значок, который изображается на одном из концов линий, являющихся сегментами этого пути.
* Строки текста. Служат для представления различных видов информации в некоторой грамматической форме. Последний необходим для получения полной информации о модели. Например, строки текста в различных секциях обозначения класса могут соответствовать атрибутам этого класса или его операциям. На использование строк накладывается важное условие – семантика всех допустимых символов должна быть заранее определена в языке UML или служить предметом его расширения в конкретной модели.

При графическом изображении диаграмм следует придерживаться следующих основных рекомендаций:

* Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области. Речь идет о том, что в процессе разработки диаграммы необходимо учесть все сущности, важные с точки зрения контекста данной модели и диаграммы. Отсутствие тех или иных элементов на диаграмме служит признаком неполноты модели и может потребовать ее последующей доработки.
* Все сущности на диаграмме модели должны быть одного концептуального уровня. Здесь имеется в виду согласованность не только имен одинаковых элементов, но и возможность вложения отдельных диаграмм друг в друга для достижения полноты представлений. В случае достаточно сложных моделей систем желательно придерживаться стратегии последовательного уточнения или детализации отдельных диаграмм.
* Вся информация о сущностях должна быть явно представлена на диаграммах. Речь идет о том, что, хотя в языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию (например, в случае неявного указания видимости атрибутов и операций классов), необходимо стремиться к явному указанию свойств всех элементов диаграмм.
* Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем использовании на практике. Например, наличие замкнутых путей при изображении отношений агрегирования или композиции приводит к ошибкам в программном коде, который будет реализовывать соответствующие классы. Наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен также приводит к неоднозначной интерпретации и может служить источником проблем. [5]
  1. Диаграмма вариантов использования

**Диаграмма вариантов использования** в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

**Прецедент** — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

**Основное назначение диаграммы** — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Для отражения модели прецедентов на диаграмме используются:

* рамки системы – прямоугольник с названием в верхней части и эллипсами внутри. Часто может быть опущен без потери полезной информации,
* актёр – стилизованный человечек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью. Актёры не могут быть связаны друг с другом,
* прецедент – эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым актёрами результатам.

Часть дублирующийся информации в модели прецедентов можно устранить указанием связей между прецедентами:

* ***обобщение прецедента*** — стрелка с незакрашенным треугольником (треугольник ставится у более общего прецедента),
* ***включение прецедента*** — пунктирная стрелка со стереотипом «include»,
* ***расширение прецедента*** — пунктирная стрелка со стереотипом «extend» (стрелка входит в расширяемый прецедент, в дополнительном разделе которого может быть указана *точка расширения* и, возможно в виде комментария, условие расширения)

При работе с вариантами использования важно помнить несколько простых правил:

* каждый прецедент относится как минимум к одному действующему лицу;
* каждый прецедент имеет инициатора;
* каждый прецедент приводит к соответствующему результату.[5]
  1. Диаграмма последовательности

**Диаграмма последовательности** – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами. На данной диаграмме объекты располагаются слева направо.

Как было сказано выше, взаимодействие между актёрами отображается при помощи специальных стрелок, передающих управление от отправителя (от кого идёт стрелка) к получателю (тот, к кому направлена стрелка). Стрелки демонстрируют ход сценария и те события, которые происходят во время анализируемого прецедента.

Всего существует 5 видов стрелок:

* Синхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Пока проводимое актёром-получателем действие не будет завершено (соответственно, не будет получено ответное сообщение), актёр-отправитель теряет возможность производить какие-либо действия. Графически изображается как стрелка с закрашенным треугольником, после которой идёт прямоугольник, отражающий деятельность объекта, в конце которого находится ответное сообщение.
* Ответное сообщение – данное сообщение является ответом на синхронное сообщение. Обычно, содержит какое-либо возвращаемое изначальному актёру-отправителю значение, также возвращающее ему управление (возможность действовать).
* Асинхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Основное отличие от синхронного сообщения состоит в том, что актёр-отправитель не теряет возможности совершать другие действия.
* Потерянное сообщение – сообщение без адресата (есть отправитель, нет получателя).
* Найденное сообщение – сообщение без отправителя.

Последние два вида стрелок (взаимодействий) используются крайне редко. В основном они используются для демонстрации взаимодействия имеющихся объектов в данном прецеденте с внешними системами. [6]

1. Анализ предметной области «ВУЗ»

Основной деятельностью Высшего Учебного Заведения (ВУЗ) является подготовка студентов по специальностям. Ежегодно огромное количество выпускников школ желает поступить в Университет, для оценки знаний был создан Единый Государственный Экзамен (ЕГЭ). При зачислении в ВУЗ в первую очередь смотрят на набранные баллы на экзаменах. Из-за этого, необходимость отсортировать абитуриентов так же входит в деятельность ВУЗа.

* 1. Диаграмма вариантов использования предметной области «ВУЗ»

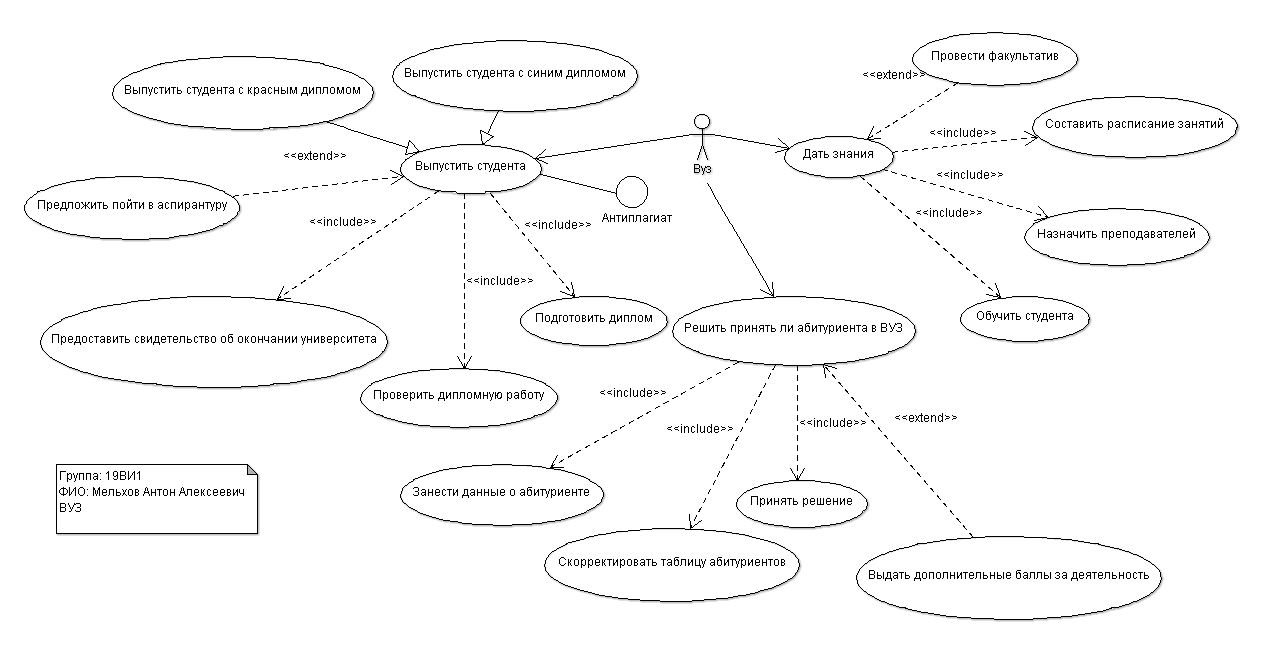
В результате анализа предметной области была составлена диаграмма вариантов использования (Рисунок 1).

Рисунок 1 - Диаграмма вариантов использования

С помощью связи включение прецедента соединены такие варианты использования (ВИ): «Подготовить диплом», «Проверить дипломную работу», «Предоставить свидетельство об окончании университета» и другие. На этой диаграмме так же присутствует интерфейс «Антиплагиат». Ещё видны прецеденты, расширяющие возможности: «Провести факультатив», «Выдать дополнительные баллы за деятельность» и т.д. Есть ВИ со связью обобщения прецедента: «Выпустить студента с красным дипломом» и «Выпустить студента с синим дипломом». Все вместе они составляют актёра – «ВУЗ». С тремя возможными вариантами использования:

* Выпустить студента
* Дать знания
* Решить принять ли абитуриента
  1. Диаграмма последовательности предметной области «ВУЗ»

В ходе работы, была составлена диаграмма последовательности (Рисунок 2), на ней изображен процесс составления плана обучения в упрощённой форме.

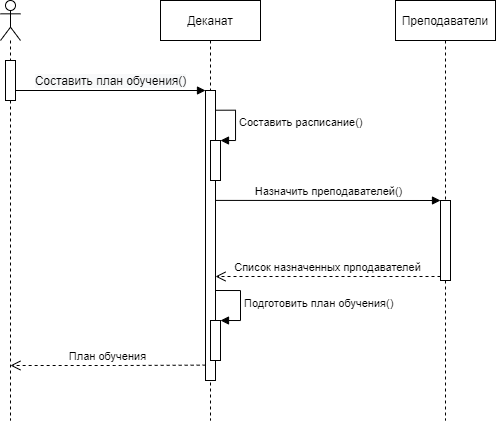


Рисунок 2 - Диаграмма последовательности для «ВУЗ»

На этой диаграмме актёр отправляет синхронный запрос в отдел «Деканат» на составления плана обучения. Который, в свою очередь, составляет расписание, отправляет синхронный запрос к «Преподавателям», чтобы назначить преподавателей и получить список от них, а затем сам подготавливает план обучения и возвращает его актёру.

1. Технология обработки текстовой информации

В современном нельзя обойтись без работы с информацией на компьютере. Основными помощниками в этом являются текстовые редакторы. Компьютеры отлично помогают ускорить работу, быстро отредактировать текст, исправить ошибки. Существует несколько типов редакторов текста. Например, можно воспользоваться на компьютере или ноутбуке блокнотом. Но у него ограниченные возможности, поэтому в нем удобнее всего писать просто заметки или подготавливать HTML-документы. Есть еще один редактор — WordPad. Помимо обычного редактирования информации в нем можно менять размер шрифта, выделение курсивом и подчеркивание. Существует еще один вид текстовых редакторов. Они многофункциональные. В них можно работать не только с текстом, но и с диаграммами, схемами, звуками. Их называют текстовыми процессорами. Текстовые процессоры имеют множество возможностей. Они включают в себя вставку, замену, копирование, перемещение разных фрагментов. В них можно составлять списки, менять абзацы, настраивать расположение на странице. Есть удобная функция проверки грамматики и пунктуации. В текстовых процессорах можно построить любую таблицу или диаграмму. Еще одно преимущество- возможность просмотреть, как будет отражаться текст на бумаге перед его печатью. Рассмотрим некоторые из них.

**Редактор Лексикон**

Он предназначен для работы с документами на английском и русском языках. Такие редакторы подходят для любой компании, потому что есть возможность открыть сразу несколько окон. Следовательно, менять и редактировать текст будет очень удобно. Есть функция макропоследовательности. Программа запоминает определенные действия, и их можно воспроизвести одним нажатием кнопки.

**Microsoft Word**

Этот редактор знаком почти каждому человеку. Он входит в пакет Microsoft Office. Работать в нем можно на 22 языках. Система работает очень автоматизировано. Может сразу проверять орфографию и пунктуацию, переносить слова, переносить текст. Есть функция автосохранения за определенный промежуток времени. Удобно использовать поиск слова или фразы. Также а Microsoft Word есть расписание и календарь. Еще одно его преимущество — возможность работы с большим количеством текста. **Notepad++**

Этот редактор поддерживает сразу несколько языков программирования. В нем можно редактировать сразу несколько документов. Notepad ++ обладает функцией завершения слова, которое вы набираете. Как и в ворде, здесь можно пользоваться поиском и заменой слов. Окна динамически изменяются во время просмотра. Текст, создаваемый в редакторе может редактироваться и форматироваться. Редактированием является именно преобразование текста. Форматирование тоже преобразовывает элемент, но меняя его форму. Форматирование может быть осуществлено с помощью изменения параметров страницы или форматирования абзацев и символов.  
[7]

Пояснительная записка и другие материалы оформляться в соответствии с требованиями государственных стандартов, действующих на момент представления работы к защите. Общие требования к оформлению текстовых документов, а также требования к использованию в них формул, рисунков и таблиц, определены межгосударственными стандартами:

* ГОСТ 7.32–2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»;
* ГОСТ 2.105–95 «Общие требования к текстовым документам»;
* ГОСТ Р 7.0.5–2008 «Библиографическая ссылка. Общие требования и правила составления»;
* ГОСТ 7.1–2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;
* ГОСТ 7.80–2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».

В основном, используются следующие параметры:

* Для текста – шрифт «Times New Roman», размер шрифта – 14, межстрочный интервал 1,5, в параметрах абзаца отступ слева – 0 см, отступ справа – 0 см, интервал перед – 0 пт, интервал после 0 пт.
* Для названия глав, разделов и подразделов – выравнивание по левому краю и оформление полужирным шрифтом «Times New Roman» размер шрифта – 14, межстрочный интервал 1,5, в параметрах абзаца отступ слева – 0 см, отступ справа – 0 см, интервал перед – 10 пт, интервал после 10 пт.
* Для ненумерованных заголовков – не имеют абзацного отступа (красной строки), выравниваются по центру и оформляются полужирным шрифтом «Times New Roman» размер шрифта – 14, межстрочный интервал 1,5, в параметрах абзаца отступ слева – 0 см, отступ справа – 0 см, интервал перед – 10 пт, интервал после 10 пт.

Все страницы работы должны быть пронумерованы последовательно арабскими цифрами. Нумерация страниц должна быть сквозной от титульного листа (на титульном листе и задании на работу нумерация не ставится, но в общее количество листов включается) до последнего листа текста, включая иллюстративный материал (таблицы, графики, диаграммы и т. п.), расположенный внутри текста, а также приложения.

Все рисунки должны иметь подпись, находящуюся под ним.

Заключение

В ходе выполнения учебной практики, мы узнали, что такое объектно-ориентированный анализ и проектирование, некоторые виды нотации, ознакомились с историей создания UML, разобрали виды диаграмм, познакомились с несколькими из них, а именно: диаграмма вариантов использования, диаграмма последовательности, составили для предметной области «ВУЗ» вышеуказанные диаграммы, приобрели знания по технологии обработки текстовой информации, которая основывается на ГОСТ.

С помощью полученных знаний, были пройдены тесты и получены сертификаты (Приложение А), а также оформлен отчёт по практическому заданию.

Список использованных источников

1. <https://studopedia.ru/3_33047_ob-ektno-orientirovanniy-analiz-i-proektirovanie.html>
2. НОИ ИНТУИТ <https://www.intuit.ru/studies/courses/32/32/lecture/1000?page=2>
3. <http://www.maksakov-sa.ru/ModelUML/IstorUML/index.html>
4. <https://ru.wikipedia.org/wiki/UML#%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F>
5. Леоненков А. В., Самоучитель UML / Леоненков А. В., Самоучитель UML, BHV, 2007 г.
6. <https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8>
7. <http://interesting-information.ru/2015/07/texnologii-obrabotki-tekstovoj-informacii-kratko>

Приложение А. Результаты прохождения обучающего курса



Рисунок 3 – Pascal



Рисунок 4 - UML