МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

Дисциплина: «Технологии проектирования информационных систем»

**Отчет по лабораторной работе №11**

Создание диаграммы вариантов использования.

проект: «Авиакасса»

Выполнили: ст. гр. 10702217 Тарасенко И.В. Косякова Д.Д.

Принял: доц. Сидорик В.В.

Минск 2020

**Цель:**

Построить диаграмму вариантов использования для бизнес-процесса «Авиакасса» в Rational Rose.

**Задачи:**

Научиться применять язык UML для моделирования и проектирования информационных систем.

Изучить программное средство Rational Rose.

**Теоретические сведения:**

Моделирование приложений на UML.

Обзор case-средств объектно-ориентированного анализа и проектирования. Rational Rose, общая характеристика и функциональные возможности.

Основные структурные компоненты. Интерфейс Rational Rose.

Основные типы сущностей: действующие лица и варианты использования.

Типы отношений: зависимости, ассоциации, обобщении.

Нотация диаграммы использования.

Выделение вариантов использования как ключ к моделированию.

Семантика вариантов использования.

Сценарий.

* **Моделирование приложений на UML.**

Моделирование – построение и изучение моделей реально существующих объектов, процессов или явлений с целью получения этих явлений, интересующих исследователя.

UML (Unified Modeling Language) преднaзнaчен для упрощения общения и взaимодейcтвия учacтников проектa, cокрaщения времени нa объяcнение и уcвоение информaции, облегчения документировaния.

UML - грaфичеcкaя нотaция, преднaзнaченнaя для опиcaния и моделировaния процеccов, протекaющих в ходе рaзрaботки. Диaгрaммы дaнной нотaции рaзличaютcя по типaм и опиcывaют рaзличные acпекты рaзрaботки. Рaзличaют 2 оcновных типa UML-диaгрaмм: cтруктурные и поведенчеcкие[1].

Cтруктурные диaгрaммы отрaжaют элементы, из которых cоcтоит cиcтемa. Поведенчеcкие модели опиcывaют процеccы, протекaющие в cиcтеме.

У кaждого языкa моделировaния cущеcтвует cвой cловaрь. Cловaрь UML cоcтоит из cледующих элементов:

Сущноcти - это aбcтрaкции, которые являютcя оcновными элементaми моделей;

Отношения - это cвязи между cущноcтями;

Диaгрaммы - это отрaжение взaимодейcтвия cущноcтей и отношений.

* **Обзор case-средств объектно-ориентированного анализа и проектирования.**

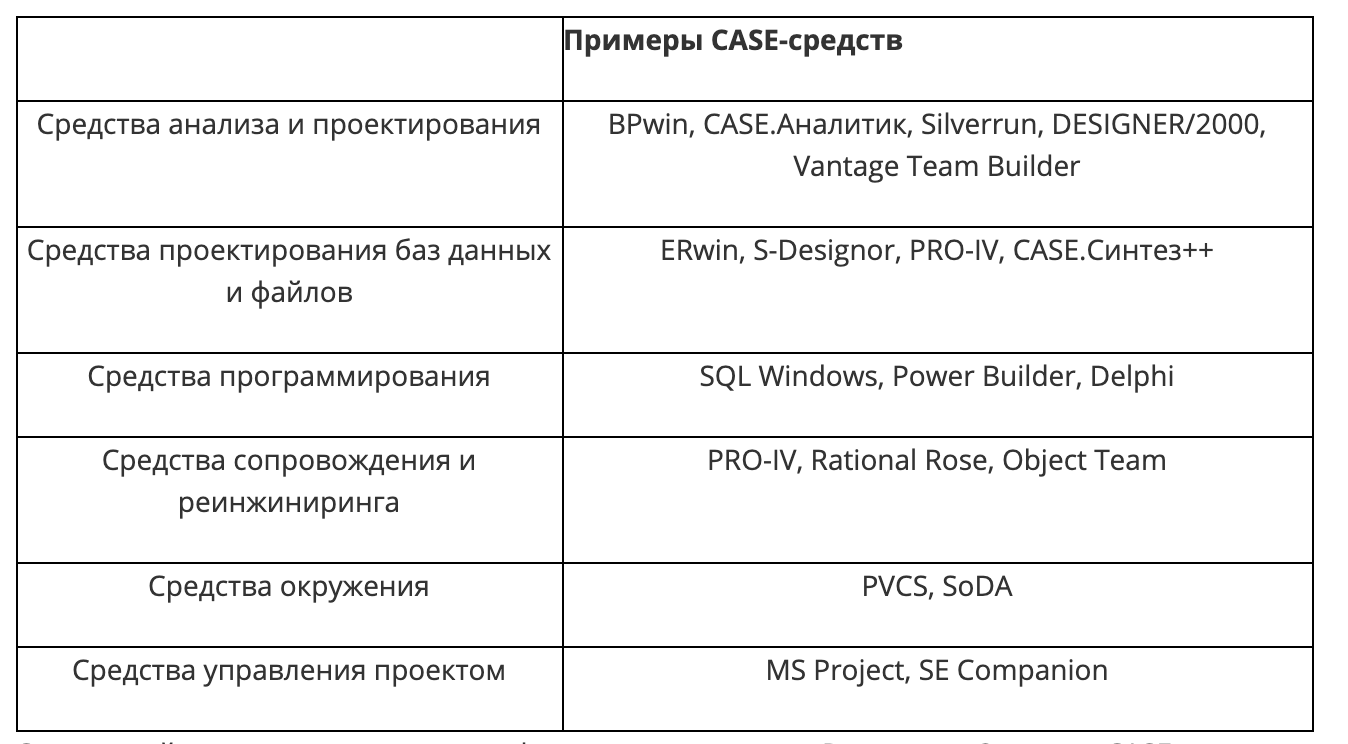
CASE-средство - на английском языке звучит как Computer-Aided Software Engineering, - специальный набор применяемой техники, а также методов программной инженерии при создании программного продукта, помогающий обеспечить отсутствие ошибок, высокое качество, а также простое обслуживание программного продукта.

Главной целью CASE-средств является увеличение производительности труда разработки, а также облегчение работы разработчиков программного продукта.

CASE-средство состоит прежде всего из:

* методологии - задает единый графический язык, а также правила и методы работы с ним;
* графических редакторов - используются при построении диаграмм;
* генератора - генерирует исходный код для различных платформ;
* репозитория - база данных, хранящая результаты работы разработчиков.

Таблица 1. Примеры CASE-средств, классифицированных по функциональной направленности.

****

* **Rational Rose, общая характеристика и функциональные возможности.**

CASE-средство IBM Rational Rose со времени своего появления претерпело серьезную эволюцию, и в настоящее время представляет собой современный интегрированный инструментарий для проектирования архитектуры, анализа, моделирования и разработки программных систем. Именно в IBM Rational Rose язык UML стал базовой технологией визуализации и разработки программных систем, что определило популярность и стратегическую перспективность этого инструментария[2].

Наиболее характерные функциональные особенности этой программы заключаются в следующем:

* интеграция с MS Visual Studio 6, которая включает поддержку на уровне прямой и обратной генерации кодов и диаграмм Visual Basic и Visual С++ с использованием ATL (Microsoft Active Template Library), Web-Классов, DHTML и протоколов доступа к различным базам данных;
* непосредственная работа (инжиниринг и реинжиниринг) с исполняемыми модулями и библиотеками форматов EXE, DLL, TLB, OCX.
* поддержка технологий MTS (Microsoft Transaction Server) и ADO (ActiveX Data Objects) на уровне шаблонов и исходного кода, а также элементов технологии Microsoft - COM+ (DCOM);
* полная поддержка компонентов CORBA и J2EE, включая реализацию технологии компонентной разработки приложений CBD (Component-Based Development), языка определения интерфейса IDL (Interface Definition Language) и языка определения данных DDL (Data Definition Language);
* полная поддержка среды разработки Java-приложений, включая прямую и обратную генерацию классов Java формата JAR, а также работу с файлами формата CAB и ZIP.
* **Основные структурные компоненты.**

В составе Rational Rose можно выделить 6 основных структурных компонент: репозиторий, графический интерфейс пользователя, средства просмотра проекта (browser), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генератор кодов (индивидуальный для каждого языка) и анализатор для С++, обеспечивающий реинжиниринг - восстановление модели проекта по исходным текстам программ.

Репозиторий представляет собой объектно-ориентированную базу данных. Средства просмотра обеспечивают "навигацию" по проекту, в том числе, перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т. д. Средства контроля и сбора статистики дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. Генератор отчетов формирует тексты выходных документов на основе содержащейся в репозитории информации.

Средства автоматической генерации кодов программ на языке С++, используя информацию, содержащуюся в логической и физической моделях проекта, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на языке С++. Анализатор кодов С++ реализован в виде отдельного программного модуля. Его назначение состоит в том, чтобы создавать модули проектов в форме Rational Rose на основе информации, содержащейся в определяемых пользователем исходных текстах на С++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать, какая информация должна быть включена в формируемую модель и какие элементы выходной модели следует выводить на экран. Таким образом, Rational Rose/С++ обеспечивает возможность повторного использования программных компонент.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

* диаграммы классов;
* диаграммы состояний;
* диаграммы сценариев;
* диаграммы модулей;
* диаграммы процессов;
* спецификации классов, объектов, атрибутов и операций
* заготовки текстов программ;
* модель разрабатываемой программной системы.

Последний из перечисленных документов является текстовым файлом, содержащим всю необходимую информацию о проекте (в том числе необходимую для получения всех диаграмм и спецификаций).

* **Интерфейс Rational Rose.**

Рабочий интерфейс программы IBM Rational Rose 2003 состоит из различных элементов, основными из которых являются(рис.1):

* главное меню;
* стандартная панель инструментов ;
* специальная панель инструментов;
* окно браузера проекта;
* рабочая область изображения диаграммы или окно диаграммы;
* окно документации;
* окно журнала.

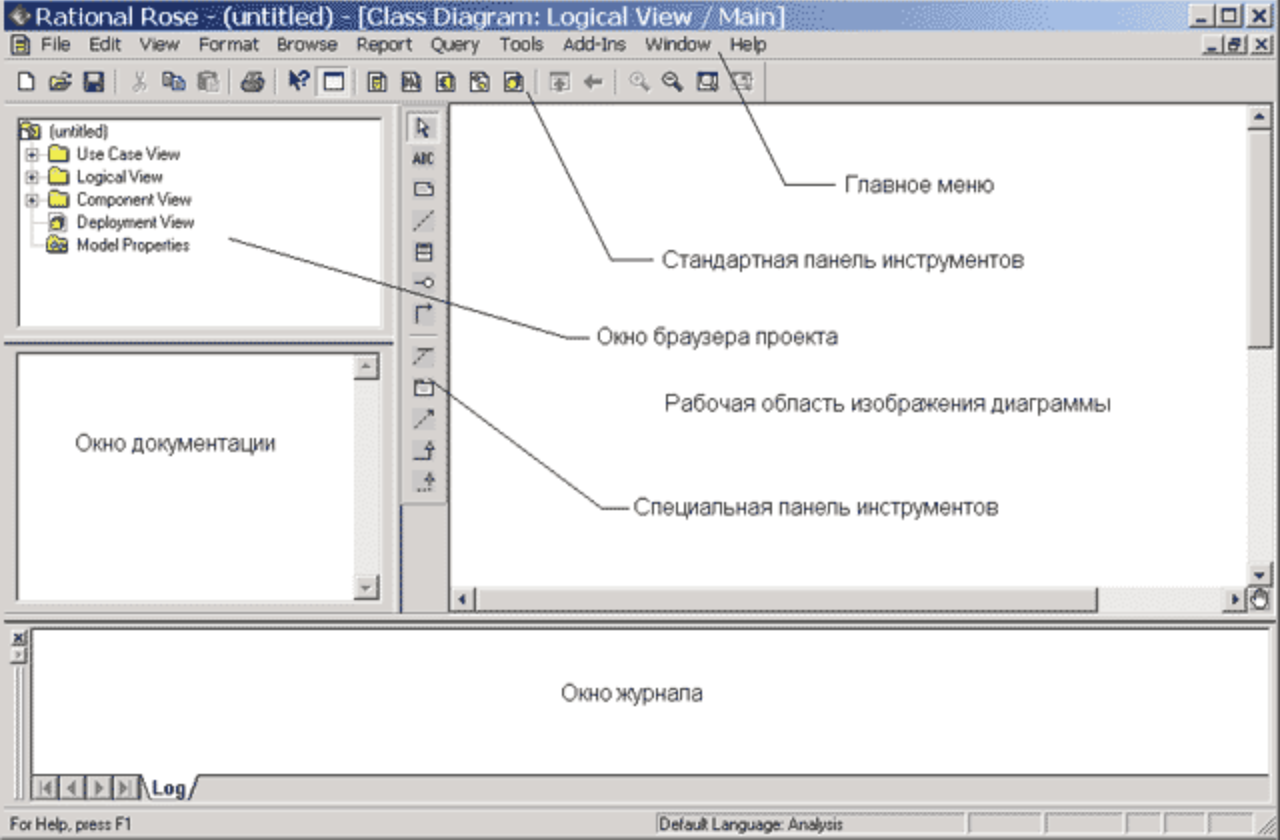
****

Рис.1 - Общий вид рабочего интерфейса CASE-средства IBM Rational Rose 2003

****

Рис.2 – Панель ToolboxIBM Rational Rose 2003

* **Основные типы сущностей: действующие лица и варианты использования. Выделение вариантов использования как ключ к моделированию. Семантика вариантов использования.**

Вопрос о выделении (или идентификации) действующих лиц при составлении модели — один из самых болезненных. Неудачный выбор действующих лиц может отрицательно повлиять на всю модель в целом. Здесь легко впасть в крайность: объявить, что имеется одно действующее лицо (внешний мир), взаимодействующее со всеми вариантами использования или, наоборот, придумать искусственных действующих лиц для каждого варианта использования. Формального метода идентификации действующих лиц не существует. Перечислим некоторые приемы, которые полезно, по нашему мнению, иметь в виду при выделении действующих лиц и покажем применение этих приемов на нашем сквозном примере. Для начала укажем более детальное определение действующего лица.

Действующие лица (actor) служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждое действующее лицо может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением действующего лица на диаграммах является фигурка человечка под которой записывается конкретное имя действующего лица (рис. 2).



Рис. 3 – Графическое обозначение действующего лица

Конструкция или стандартный элемент языка UML вариант использования применяется для спецификации общих особенностей поведения системы или любой другой сущности предметной области без рассмотрения внутренней структуры этой сущности[3]. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим действующим лицом. Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов. Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами (рис. 3).



Рис. 4 - Графическое обозначение варианта использования

* **Типы отношений: зависимости, ассоциации, обобщении.**
* Отношение зависимости.

Является наиболее общей формой отношения в языке UML. Все другие типы отношений можно считать частным случаем данного отношения. Отношение зависимости показывает, что изменение одного класса влечет изменение другого класса. Чаще всего применяется, когда один класс использует другой в качестве аргумента. Изображается пунктирной линией со стрелкой, направленной от зависимого класса к независимому[4].

Для создания отношения зависимости следует выбрать кнопку Dependency or instantiaties на панели Toolbox, затем щелкнуть мышкой по зависимому классу и не отпуская кнопки мыши перетащить стрелку на независимый класс.

Для данного типа отношения можно задать следующие свойства: будет ли иметь зависимый класс доступ к не-public элементам независимого класса (Friendship required), видимость отношения (Export Control), начальную (Multiplicity from) и конечную мощность(Multiplicity) отношения.

* Отношение ассоциации.

Поскольку отношения ассоциации, агрегации и композиции отличаются очень незначительно, то рассмотрим работу с ними в общем. Отношение ассоциации показывает, что один класс каким-то образом связан с другим классом (аналог связи в диаграмме «Сущность-Связь»). Изображается сплошной линией, соединяющей классы.

Для создания одного из приведенных выше отношений следует выбрать кнопку Unidirectional Association на панели Toolbox, затем щелкнуть мышкой по одному классу и не отпуская кнопки мыши перетащить стрелку на другой класс. По умолчанию создается однонаправленная ассоциация, т.е. класс, от которого идет стрелка, знает об общих атрибутах и операциях второго класса, но не наоборот. Для создания двунаправленной ассоциации следует вызвать для отношения контекстное меню и выбрать пункт Navigable(рис. 4).



Рис. 4 - Пример графического представления отношения ассоциации между актером и вариантом использования

* Отношение обобщения.

Это отношение между общей сущностью (суперклассом, или родителем) и ее конкретным воплощением (субклассом, или потомком). Обобщения иногда называют отношениями типа "является", имея в виду, что одна сущность является частным выражением другой, более общей. Обобщение означает, что объекты класса-потомка могут использоваться всюду, где встречаются объекты класса-родителя, но не наоборот. Изображается в виде линии с большой незакрашенной стрелкой.

Для создания отношения обобщения следует выбрать кнопку Generalization на панели Toolbox, затем щелкнуть мышкой по классу-потомку и не отпуская кнопки мыши перетащить стрелку на класс-родитель. Для данного типа отношения можно задать следующие свойства: будет ли и иметь класс-потомок доступ к не-public элементам класса-родителя (Friendship required), видимость отношения (Export Control), будет ли наследоваться только одна копия класса-родителя (Virtual inheritance).

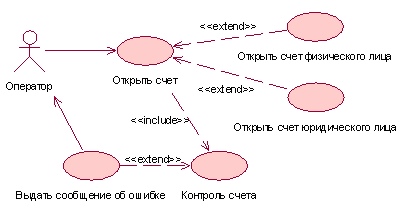


Рис. 5 - Пример графического изображения отношения обобщения между вариантами использования

* **Нотация диаграммы использования.**

На диаграмме вариантов использования можно отобразить следующие элементы нотации UML, доступные в панели элементов(табл. 2).

Таблица 2 Элементы нотации на диаграмме вариантов использования.

| **Элемент/Нотация** | **Предназначение** |
| --- | --- |
|  | Участник (Actor) |
|  | Вариант (Use case) |
|  | Граница (Boundary |
|  | Ненаправленная ассоциация (Undirected communication association) |
|  | Направленная ассоциация (Directed communication association) |
|  | Обобщение (Generalization) |
|  | Зависимость (Dependency) |
|  | Точка изгиба связей (Point) |
|  | Комментарий (Note) |
|  | Коннектор комментария (Note connector) |

* **Сценарий.**

Сценарий – это текстовое описание каждого варианта использования. В сценарии определяются взаимодействующие действующие лица, предусловия, постусловия, шаги выполнения и альтернативный ход выполнения.После определения сценариев возможно улучшение модели вариантов использования путем выявления новых (реальных) вариантов использования, находящихся в отношении расширения, использования или наследования с идеальными вариантами использования.

**Общее задание бизнес-процесса “Магазин”**

**Задание 1. Создание диаграммы вариантов использования.**

Результат представлен на рисунке 1.

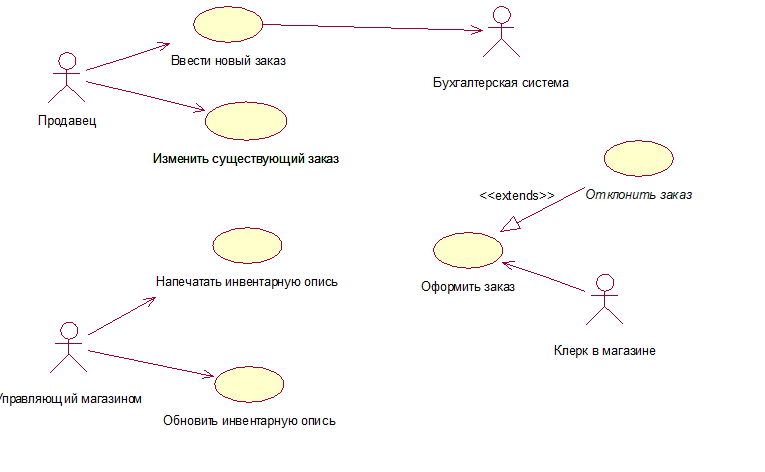


Рис.1– Диаграмма вариантов использования для бизнес-процесса «Магазин».

**Индивидуальное задание бизнес-процесса “Авиакасса”**

**Задание 2. Создание диаграммы вариантов использования.**

Результат представлен на рисунке 2.

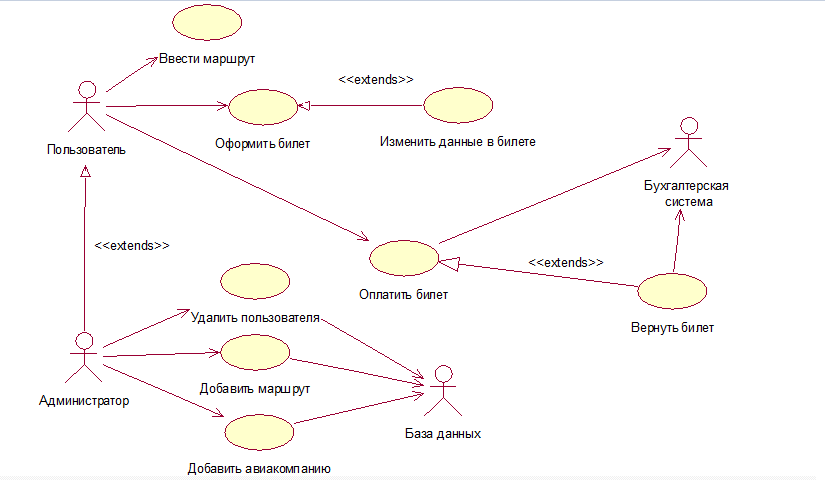
****

Рис.2 – Диаграмма вариантов использования для бизнес-процесса «Авиакасса».

Действующие лица:

* Пользователь – имеет право ввести маршрут, оформить билет, изменить данные в билете, олатить билет, вернуть билет.
* Администратор – имеет право добавить маршрут, удалить пользоваетля, добавить авиакомпанию.
* База данных – совокупность данных о маршрутах,авиакомпаниях, пользователях.
* Бухгалтерская система – система с данными об оплатах билетов.

**Вывод:** в ходе выполнения лабораторной работы были разработаны диаграммы вариантов использования для бизнес-процессов «Магазин» и «Авиакасса» в среде Rational Rose.

**Контрольные вопросы:**

1. **Назначение диаграммы вариантов использования.**

Для описания функционального назначения системы или, другими словами, то, что система будет делать в процессе своего функционирования. Визуальное моделирование с использованием нотации UML можно представить как процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной бизнес-системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что бизнес-система должна делать в процессе своего функционирования.

1. **Цели разработки диаграммы вариантов использования.**

* на начальных этапах проектирования системыопределить общие границы и контекст моделируемой предметной области;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

1. **Элементы диаграммы вариантов использования. Действующие лица.**

Это любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство или программа, которые могут служить источником воздействия на моделируемую систему так, как определит сам разработчик.

Пример на рисунке 3:

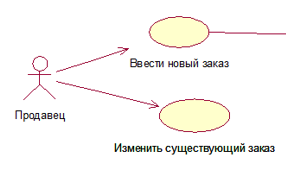


Рис.3 – Пример действующего лица

1. **Элементы диаграммы вариантов использования. Отношения.**

Отношения описывают взаимодействие экземпляров одних действующих лиц и вариантовиспользованиясэкземплярамидругих действующих лиц, и вариантов. Одно действующее лицо может взаимодействовать с несколькими вариантами использования. В этом случае это действующее лицо обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими действующими лицами, предоставляя для всех них свой сервис.

Виды отношений: отношение ассоциации (associationrelationship), отношение расширения (extendrelationship), отношение обобщения (generalizationrelationship), отношение включения (includerelationship).

1. **Элементы диаграммы вариантов использования. UseCase.**

Use case служит для описания сервисов, которые система предоставляет действующему лицу. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий.

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами (рис. 8).

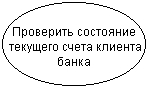


Рис.4 – Отдельный вариант использования

1. **Какие отношения могут быть между действующими лицами? Ответ подтвердите диаграммами.**

Между действующими лицами может быть только отношение обобщения



Рис.5 – Отношение обобщения

1. **Какие отношения могут быть между вариантами использования? Ответ подтвердите диаграммами. На диаграммах обязательно должно быть действующее лицо (лица).**

Между вариантами использования могут быть только отношения включения и расширения

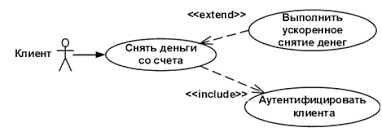


Рис.6 – Отношение включения и расширения

1. **Какие отношения могут быть между действующими лицами и вариантами использования? Ответ подтвердите диаграммами.**

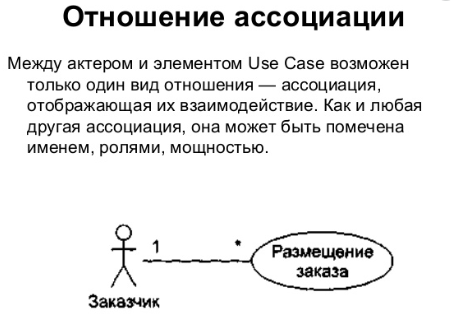


Рис.7 – Отношение ассоциации

**Список литературы:**

1. Методические указания к лабораторной работе №11 – [Электронный ресурс] – Google Drive – [https://drive.google.com/drive/folders/17tKay6k54Zahkh9OYJo2v-Qp81y1jZrp](https://drive.google.com/drive/folders/17tKay6k54Zahkh9OYJo2v-Qp81y1jZrp%20–%2028.04.2020) Дата обращение 28.04.2020
2. Работа с классами в Rational Rose- [Электронный ресурс] - <http://khpi-iip.mipk.kharkiv.edu/library/case/lab/ref/classes.html> Дата обращения 28.04.2020
3. Рабочий интерфейс Rational Rose.– [Электронный ресурс] - <https://studfile.net/preview/3675069/page:3/> Дата обращение 28.04.2020
4. Объектно-ориентированные CASE-средства (Rational Rose) – [Электронный ресурс] - <http://www.interface.ru/fset.asp?Url=/rational/oocase.htm> Дата обращение 28.04.2020

**Глоссарий:**

1. **UML (Unified Modeling Language)** – унифицированный язык моделирования является графическим языком для визуализации, описания параметров, конструирования и документирования различных систем (программ в частности).
2. **Диаграмма вариантов использования (Use case diagram)** – диаграмма, на которой отражены отношения, существующие между актёрами и вариантами использования.
3. **Вариант использования (Use case)** – это описание множества возможных последовательностей действий (событий), приводящих к значимому для действующего лица результату.
4. **Сценарий (Senario)** – конкретная последовательность действий.
5. **Действующее лицо (Actor)** – любая внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.
6. **Отношение (Relationship)** – семантическая связь между отдельными элементами модели.
7. **Отношение расширения (Extension relationship)** – определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров.