МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование информационных систем**

**Лабораторная работа №7**

**Тема:** «Объектно-ориентированное моделирование. UML-диаграммы поведения»

**Цель:** Изучение методологии объектно-ориентированного моделирования средствами UML. Получение дополнительных навыков проектирования моделей информационной системы с применением возможностей UML диаграмм.

Выполнила:

студентка 4 курса 4 группы

Миневич Кристина Викторовна

Минск 2023

Описание Microsoft Visio

Diagrams.net (ранее draw.io) – это бесплатное кроссплатформенное программное обеспечение для рисования графиков с открытым исходным кодом, разработанное на HTML5 и JavaScript. Его интерфейс можно использовать для создания таких диаграмм, как блок-схемы, каркасы, UML-диаграммы, организационные диаграммы и сетевые диаграммы.

Это отличный графический редактор, позволяющий работать с диаграммами и схемами. Программа содержит мощный набор инструментов, который будет полезен для работы. Приложение может применяться в разных сферах. Его используют it-специалисты, менеджеры, аналитики.

Программа, имеющая множество полезных инструментов, которые отлично подходят для управления каким-либо проектом. Приложение располагает мощным арсеналом средств. Причем от пользователя не требуются какие-либо профессиональные знания в технической или изобразительной области, а также связанные с этим навыки. Вы можете использовать готовые шаблоны, фигуры и элементы, с помощью которых добьетесь нужного результата.

Какие-либо навыки работы в Diagrams.net для этого не нужны. Разобраться в программе сможет даже неопытный и начинающий пользователь. Приложение имеет удобный и понятный интерфейс. Вы сможете легко создавать нужные диаграммы и схемы, необходимые для решения конкретных задач.

Инструменты Diagrams.net позволяют создавать диаграммы в один клик – перед этим достаточно внести все необходимые данные, и вы легко выполните нужную задачу. Программа позволяет предоставить информацию из готовой диаграммы – это делается с помощью формирования определенного отчета.

Приложение позволяет создавать чертежи, которые отличаются очень высоким уровнем информативности. Здесь используются различные элементы. Для каждой части чертежа составляются подробные комментарии.

Diagrams.net позволяет выполнять масштабирование проекта. В качестве основы для построения схем могут использоваться системы для автоматического проектирования.

Пользователи могут выполнять экспорт и импорт данных, который осуществляется между программами, входящими в пакет Office. Это увеличивает уровень удобства при работе с приложением. В программе также предусмотрена возможность для получения справки по работе с Visio. Здесь содержатся специальные подсказки и советы.

Ход выполнения работы

Unified Modeling Language (UML) — унифицированный язык моделирования. Расшифруем: *modeling* подразумевает создание модели, описывающей объект. *Unified* (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. UML описывает объект в едином заданном синтаксисе, поэтому где бы вы не нарисовали диаграмму, ее правила будут понятны для всех, кто знаком с этим графическим языком — даже в другой стране.

Одна из задач UML — служить средством коммуникации внутри команды и при общении с заказчиком. Давайте рассмотрим возможные варианты использования диаграмм:

Проектирование. UML-диаграммы помогут при моделировании архитектуры больших проектов, в которой можно собрать как крупные, так и более мелкие детали и нарисовать каркас (схему) приложения. По нему впоследствии будет строиться код.

Реверс-инжиниринг — создание UML-модели из существующего кода приложения, обратное построение. Может применяться, например, на проектах поддержки, где есть написанный код, но документация неполная или отсутствует.

Из моделей можно извлекать текстовую информацию и генерировать относительно удобочитаемые тексты — документировать. Текст и графика будут дополнять друг друга.

Все рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо, некоторым образом, реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Чтобы пояснить отличие логического и физического представлений, рассмотрим в общих чертах процесс разработки некоторой программной системы. Ее исходным логическим представлением могут служить структурные схемы алгоритмов и процедур, описания интерфейсов и концептуальные схемы баз данных. Однако для реализации этой системы необходимо разработать исходный текст программы на некотором языке программирования (C++, Pascal, Basic/VBA, Java). При этом уже в тексте программы предполагается такая организация программного кода, которая предполагает его разбиение на отдельные модули.

Тем не менее исходные тексты программы еще не являются окончательной реализацией проекта, хотя и служат фрагментом его физического представления. Очевидно, программная система может считаться реализованной в том случае, когда она будет способна выполнять функции своего целевого предназначения. А это возможно, только если программный код системы будет реализован в форме исполняемых модулей, библиотек классов и процедур, стандартных графических интерфейсов, файлах баз данных. Именно эти компоненты являются необходимыми элементами физического представления системы.

Таким образом, полный проект программной системы представляет собой совокупность моделей логического и физического представлений, которые должны быть согласованы между собой. В языке UML для физического представления моделей систем используются так называемые диаграммы реализации (implementation diagrams), которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания. Особенности построения первой из них рассматриваются в этой главе, а второй - в следующей.

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма деятельности – мощный инструмент, который интенсивно используется при создании ИС.

В зависимости от, поставленной перед нами задачи мы создаем диаграмму деятельности, используя тот набор элементов, который необходим для отражения определенного уровня детализации.

Таким образом, диаграмма деятельности может применяться как для описания бизнес-процесса, так и функциональных требований к системе.

Цель концептуального описания - показать целостную картину бизнес-процессов предметной области.

На данной диаграмме деятельности мы можем наблюдать процесс добавления пользователя в друзья.

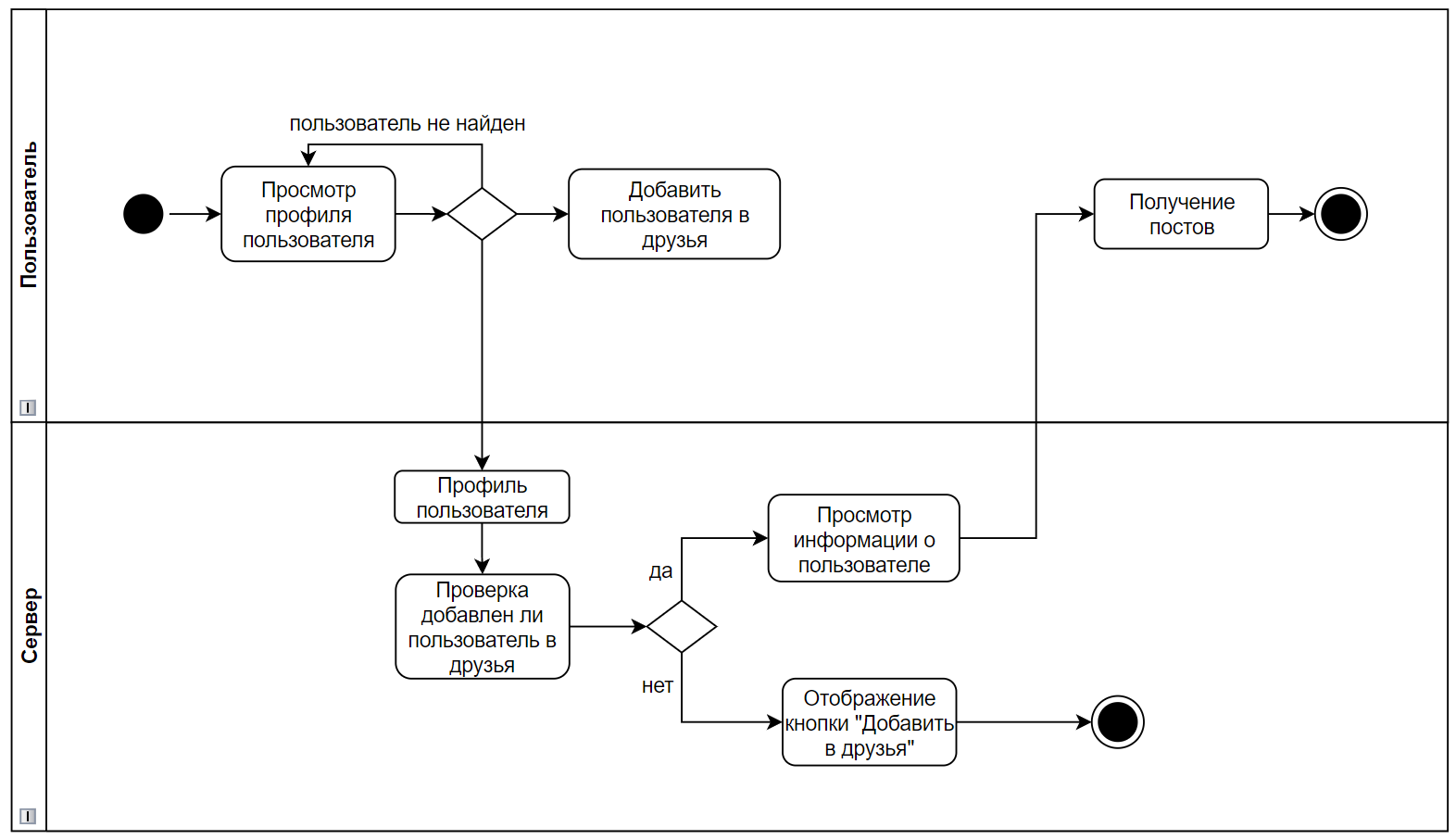


Рисунок 7.1 – UML Диаграмма деятельности

Для диаграммы состояний я выбрала непосредственно сам просмотр постов.

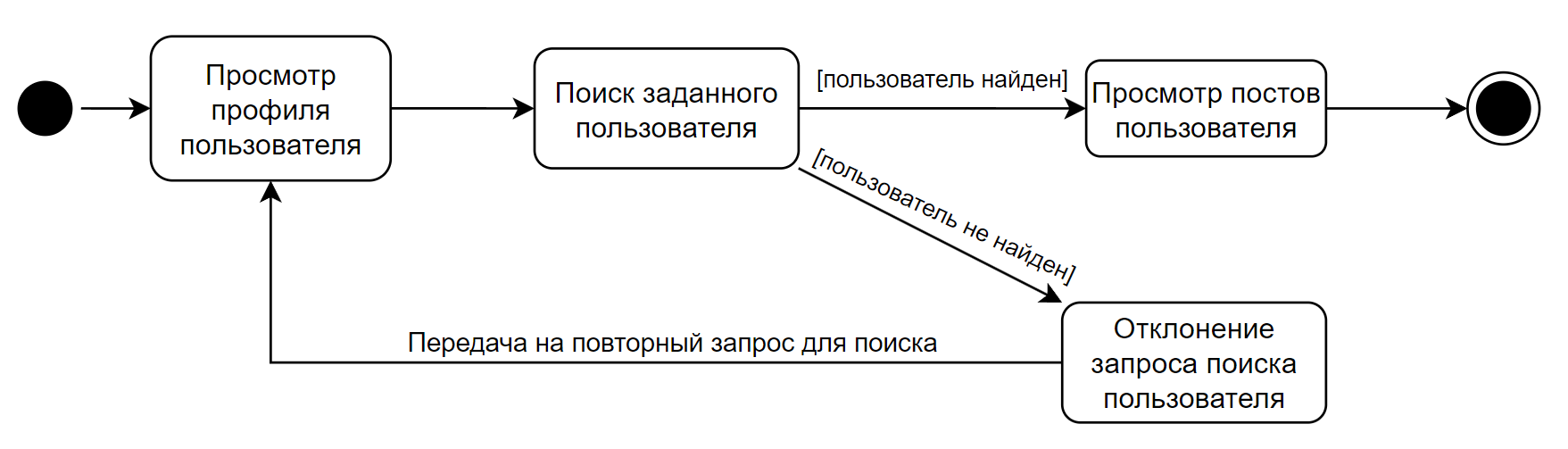


Рисунок 7.2 – UML Диаграмма состояний

**Вывод:** таким образом, в ходе выполнения работы были разработаны диаграмма деятельности и диаграммы состояний.

Ответы на вопросы

1. **Укажите виды диаграмм поведения.**

Всего существует 3 вида диаграмм поведения:

* диаграмма деятельности;
* диаграмма состояний;
* диаграмма вариантов использования.

1. **Опишите назначение диаграммы деятельности.**

Диаграмма деятельности — UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

1. **Опишите основные нотации, которые используются на диаграмме состояний.**

| **Элемент/Нотация** | **Предназначение** |
| --- | --- |
| Пример | Класс (Class) |
| Пример | Состояние (State) |
| Пример | Состояние (StateEx) |
| Пример | Составное состояние (Composite state) |
| Пример | Разделитель (Concurrent state) |
| Пример | История (History) |
| Пример | Глубокая история (Deep history) |
| Пример | Начальное состояние (Start state) |
| Пример | Конечное состояние (Final state) |
| ПримерПример | Синхронизатор/разветвитель (Complex transition) |
| Пример | Переход (Transition) |
| Пример | Сообщение (Event message) |
| Пример | Точка изгиба связей (Point) |
| Пример | Комментарий (Note) |
| Пример | Коннектор комментария (Note connector) |

1. **Укажите виды связей между объектами на диаграмме последовательностей.**

* *Синхронное сообщение* — отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Пока проводимое получателем действие не будет завершено (соответственно, не будет получено ответное сообщение), отправитель теряет возможность производить какие-либо действия. Графически изображается как сплошная линия со стрелкой в виде закрашенного треугольника, 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 отправителю значение, также возвращающее ему управление (возможность действовать). Графически изображается пунктирной линией с открытой стрелкой.
* *Асинхронное сообщение* — отправитель передаёт ход управления получателю, которому необходимо провести в прецеденте некоторое действие. Основное отличие от синхронного сообщения состоит в том, что отправитель не теряет возможности совершать другие действия. Графически изображается сплошной линией с открытой стрелкой.
* *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.

1. **Какая диаграмма позволяет моделировать параллельные вычисления?**

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.