Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование программного обеспечения»

Лабораторная работа №4

«Разработка Use-Case диаграмм»

Студент: Стрелковская В. А.

ФИТ 3 курс 2 группа

Преподаватель: Якубенко К. Д.

# **Тема и цель работы**

1. Описание понятий Unified Process (UP) и UML:

Unified Process (UP) (Унифицированный процесс) – это гибкий и итеративный процесс разработки программного обеспечения, направленный на управление жизненным циклом продукта.

UP основывается на принципах объектно-ориентированного проектирования и активно использует Unified Modeling Language (UML) для описания архитектуры и взаимодействий в системе.

UP поддерживает итеративную разработку, где проект разбивается на серии циклов (итераций), каждая из которых содержит все стадии: анализ требований, проектирование, реализация и тестирование. Одним из наиболее популярных вариантов UP является Rational Unified Process (RUP).

UML (Unified Modeling Language, унифицированный язык моделирования) – это стандартный язык для визуального моделирования программных систем. UML используется для создания диаграмм, которые помогают разработчикам и аналитикам описывать архитектуру системы, поведение и взаимодействие объектов, процессов и компонентов.

UML охватывает несколько типов диаграмм, каждая из которых описывает определённые аспекты системы (структурные и поведенческие).

2. Основные диаграммы UML 2.0:

UML 2.0 включает 14 типов диаграмм, которые можно разделить на две категории: структурные и поведенческие диаграммы.

Структурные диаграммы:

* Диаграмма классов (Class diagram) – отображает классы, их свойства, методы и отношения между ними.
* Диаграмма компонентов (Component diagram) – описывает физическую структуру системы и связи между компонентами.
* Диаграмма объектов (Object diagram) – показывает конкретные экземпляры классов в определённый момент времени.
* Диаграмма развёртывания (Deployment diagram) – иллюстрирует развертывание системы на физической инфраструктуре.
* Диаграмма пакетов (Package diagram) – группирует классы и другие элементы в пакеты.
* Диаграмма композитной структуры (Composite Structure diagram) – описывает внутреннюю структуру класса и взаимодействие его частей.

Поведенческие диаграммы:

* Диаграмма вариантов использования (Use case diagram) – описывает взаимодействие пользователей (акторов) с системой через различные варианты использования (use cases).
* Диаграмма последовательностей (Sequence diagram) – отображает обмен сообщениями между объектами в процессе выполнения сценария.
* Диаграмма деятельности (Activity diagram) – описывает поток управления между действиями внутри системы.
* Диаграмма состояний (State machine diagram) – показывает состояния объекта и переходы между ними в зависимости от событий.
* Диаграмма взаимодействия (Communication diagram) – описывает взаимодействие объектов через обмен сообщениями.
* Диаграмма тайминга (Timing diagram) – детализирует изменения состояния объекта или взаимодействие компонентов во времени.
* Диаграмма взаимодействия компонентов (Interaction overview diagram) – объединяет несколько диаграмм взаимодействия для описания сложных процессов.
* Диаграмма кооперации (Collaboration diagram) – альтернативный способ представления взаимодействия объектов, подчеркивая их совместную работу.

3. CASE-средства, поддерживающие создание UML-диаграмм:

CASE-средства (Computer-Aided Software Engineering) предназначены для автоматизации процессов проектирования и документирования программных систем. Вот несколько популярных инструментов для создания UML-диаграмм:

* Rational Rose – продукт компании IBM, который поддерживает UML и часто используется в рамках RUP.
* Enterprise Architect – мощное средство для проектирования и моделирования систем с поддержкой UML.
* StarUML – бесплатный инструмент с открытым исходным кодом для создания UML-диаграмм.
* Visual Paradigm – CASE-инструмент с широкими возможностями для создания UML-диаграмм и интеграции с различными методологиями.
* ArgoUML – ещё одно бесплатное средство для моделирования UML-диаграмм.
* Lucidchart – онлайн инструмент для визуализации, поддерживающий создание UML-диаграмм.

4. Назначение диаграммы вариантов использования

Диаграмма вариантов использования (Use Case Diagram) предназначена для описания взаимодействий пользователя (актера) с системой.

Её цель – показать функциональные требования системы через действия (варианты использования), которые пользователь может выполнить, взаимодействуя с системой. Варианты использования помогают ответить на вопрос «Что делает система?», но не «Как это реализовано?». Это позволяет сосредоточиться на потребностях пользователей и основных функциях системы, оставляя детали реализации на более поздние этапы разработки. Такая структура способствует более эффективному взаимодействию между разработчиками и заинтересованными сторонами, обеспечивая общее понимание целей и задач системы.

Основные элементы диаграммы вариантов использования:

* Акторы – пользователи или внешние системы, взаимодействующие с системой.
* Варианты использования – Описывают действия, которые акторы могут выполнить. Каждый вариант использования представляет собой сценарий, который описывает последовательность действий, выполняемых актором в системе для достижения определенной цели.
* Отношения – связи между актором и вариантами использования, а также зависимости между вариантами использования (например, <<extend>> или <<include>>).

5. Нотации, используемые для построения Use-Case диаграммы:

Для построения Use-Case диаграммы используются следующие нотации:

* Актор – изображается в виде "человечка", представляющего пользователя или другую систему, взаимодействующую с системой.
* Вариант использования (use case) — изображается в виде эллипса, в который вписывается название функции или действия, выполняемого актором.
* Связь актор-диаграмма – показывается в виде линии, соединяющей актера и соответствующий вариант использования.
* Обобщение акторов – изображается стрелкой с пустым треугольником на конце, обозначая иерархию акторов (например, один актор является подтипом другого).

Отношения include и extend:

* <<include>> – пунктирная стрелка с надписью "include", указывающая, что один вариант использования всегда включает другой;
* <<extend>> – пунктирная стрелка с надписью "extend", показывающая, что один вариант использования может расширить другой при определённых условиях.

Эти элементы создают общую картину функциональности системы с точки зрения конечного пользователя.

# **Описание функциональных требований**

Функциональны требования к системе можно разделить на требования к функционалу для различных ролей приложения – пользователя, гостя, администратора.

1. Функционал для пользователя:

Для зарегистрированных пользователей приложение предлагает широкий спектр возможностей, направленных на удобство и улучшение пользовательского опыта:

* просмотр расписания сеансов и фильмов;
* просмотр информации о фильмах;
* онлайн покупка билетов на сеансы;
* выбор мест в зале при покупке билетов;
* просмотр истории бронирований и приобретенных билетов;
* авторизация пользователя.

1. Функционал для администратора:

Администраторы играют ключевую роль в управлении и поддержании работы приложения. Их функционал включает:

* управление расписанием сеансов и фильмами;
* добавление, редактирование и удаление информации о фильмах (включая название, описание, жанр и актеров);
* управление ценами на билеты.

1. Функционал для гостя:

Гостям предоставляется доступ к ограниченному набору функций, что позволяет им ознакомиться с приложением и принять решение о регистрации:

* регистрация;
* просмотр расписания сеансов и фильмов;
* просмотр информации о фильмах;
* поиск фильмов и сеансов по критериям.

# **Описание программных средств**

Для построения диаграмм IDEF0 использовался веб-ресурс Draw.io, разрабатываемый компанией JGraph Ltd. и направленный на построение диаграмм. Адрес веб-ресурса – https://www.drawio.com. Данный ресурс доступен на всех платформах, имеющих веб-браузер и доступ в Интернет.

Draw.io предлагает интуитивно понятный интерфейс, который позволяет легко создавать и редактировать диаграммы. В функционал данного ресурса входит широкий спектр возможностей, включая:

* построение графиков и смысловых карт: пользователи могут визуализировать свои идеи и концепции, создавая понятные и наглядные схемы;
* UML-диаграммы: этот инструмент поддерживает создание различных типов UML-диаграмм, что делает его полезным для разработчиков программного обеспечения и системных аналитиков;
* диаграммы Венна: draw.io позволяет создавать диаграммы Венна для визуального представления пересечений и различий между наборами данных;
* Agile и Kanban доски: инструмент поддерживает методологии Agile, что позволяет командам эффективно управлять проектами и отслеживать прогресс;
* графики мозговых штурмов: пользователи могут организовывать свои идеи и генерировать новые концепции в удобной и визуально привлекательной форме;
* диаграммы архитектур технических систем: draw.io предоставляет возможности для создания сложных архитектурных схем, что полезно для инженеров и проектировщиков.

Одним из ключевых преимуществ draw.io является возможность совместной работы в реальном времени, что позволяет командам эффективно сотрудничать и вносить изменения одновременно. Пользователи могут делиться своими диаграммами с коллегами и работать над проектами совместно, не беспокоясь о лицензиях или ограничениях платформ.

Кроме того, draw.io обеспечивает высокий уровень конфиденциальности и безопасности данных, позволяя пользователям хранить свои диаграммы локально или в облачных хранилищах по своему выбору. Это делает инструмент идеальным для команд, стремящихся к безопасному управлению своей информацией.

Таким образом, использование draw.io для построения диаграмм IDEF0 не только облегчает процесс визуализации, но и предоставляет мощные инструменты для поддержки различных методологий и повышения эффективности работы команды.

# **Описание практического задания**

В системе выделяются три основные роли пользователей, каждая из которых имеет свои уникальные возможности:

* гость – эта роль предоставляет базовые функции, такие как возможность входа в систему или регистрации, а также доступ к просмотру списков фильмов и сеансов. Гость также может ознакомиться с информацией о кинотеатре;
* пользователь – эта роль включает функции, доступные Гостю, но дополнительно предоставляет возможность покупки билетов. Пользователь, при желании, может просматривать историю своих бронирований. Кроме того, у него есть возможность управлять своим профилем, что включает в себя изменение личных данных и настроек, а также возможность выхода из системы;
* администратор – эта роль также включает возможности Гостя, но и предоставляет дополнительные функции управления системой. Администратор может редактировать информацию о фильмах и пользователях, что позволяет поддерживать актуальность данных. Кроме того, он имеет доступ к просмотру всех совершенных бронирований, что дает возможность анализировать посещаемость и предпочтения клиентов.

Иерархия актеров подсистемы:

* "Гость" – базовая роль, предоставляющая минимальные возможности взаимодействия с системой;
* "Пользователь" – расширение роли "Гость", обладающее дополнительными функциями, которые делают взаимодействие с системой более полноценным и удобным;
* "Администратор" – расширение роли "Гость", наделенное полномочиями, необходимыми для управления и контроля над системой.

Для наглядного понимания взаимодействий между пользователями и системой представлена диаграмма Use Case, или сценарий использования, который является текстовым описанием конкретного взаимодействия. Данная диаграмма помогает визуализировать функциональные требования системы и включает в себя все основные действия, которые могут выполнять каждый из актеров. Она демонстрирует, как пользователи будут взаимодействовать с системой и какие функции доступны для каждой роли.

Кроме того, диаграмма позволяет выявить ключевые сценарии, которые наиболее важны для пользователей, и служит основой для дальнейшего анализа требований. Использование диаграммы позволяет команде разработки лучше понять контекст использования системы и обеспечить соответствие функциональности ожиданиям пользователей.

Важность диаграммы Use Case заключается также в том, что она способствует выявлению потенциальных проблем и недостатков на ранних стадиях разработки, что позволяет значительно снизить риск дорогостоящих изменений в будущем. Она служит эффективным средством для коммуникации между различными заинтересованными сторонами, включая бизнес-аналитиков, разработчиков и конечных пользователей, и помогает создать общее понимание функциональности системы.

Диаграмма представлена на рисунке 4.1.

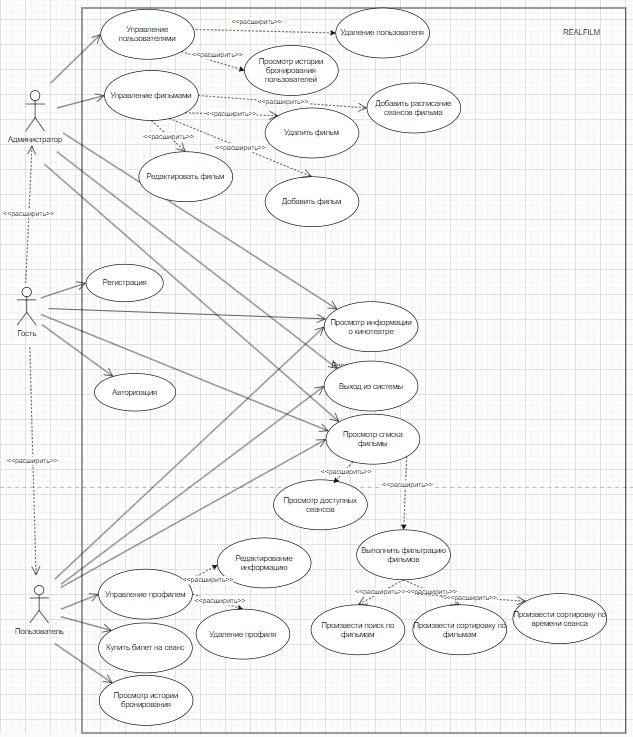


Рисунок 4.1 – Use-case-диаграмма для сервиса “REALFILM”

По итогам лабораторной работы были получены важные навыки проектирования функциональности системы с применением UML. Создание Use Case диаграммы для кинотеатра не только помогло выделить группы пользователей системы, но и дало возможность глубже понять их потребности и ожидания.