Практическое задание №1

Создание диаграмм вариантов использования, диаграммы деятельности

Целью задания является изучение объектно-ориентированного анализа и моделирования бизнес-процессов в исследуемой предметной области с помощью языка UML.

В ходе выполнения задания для выбранной (или заданной) предметной области изучается процесс построения диаграммы вариантов использования (диаграммы прецедентов) и диаграммы деятельности с помощью CASE-средства, поддерживающего язык UML.

Общая последовательность выполнения задания №1 включает определенные этапы (разделы).

1. Постановка задачи.

   Выбор темы для моделирования (предметной области). Исследование системы: сформулировать цель моделирования, определить объекты и выполняемые функции.

2. Характеристика и возможности CASE-средства. Рабочий интерфейс программы и создание проекта.
3. Построение диаграммы вариантов использования (диаграммы прецедентов, Use Case Diagram).
4. Построение диаграммы деятельности (Activity Diagram).

В результате выполнения задания и построения диаграмм следует сохранить модель в отдельном файле, а также подготовить и оформить отчет, содержащий последовательность разработки моделей и их элементов, необходимые комментарии, пояснения и краткий вывод в заключении.

В названии файла модели и отчета по заданию необходимо использовать фамилию студента и номер работы.

Далее приводится демонстрационный пример построения моделей основных бизнес-процессов, описывающих деятельность некоторого небольшого интернет-магазина.

1. Постановка задачи. Определение предметной области

Интернет-магазин занимается продажей некоторых товаров. Потенциальный покупатель просматривает каталог и делает заказ. Предполагается, что потенциальный клиент, открывая сайт магазина, может нажать кнопку просмотра (или загрузки) каталога, далее может положить понравившийся товар в корзину, изменить корзину и, приняв решение о покупке товаров, перейти из корзины к оформлению заказа.

Для того чтобы корректно создать систему, отвечающую всем требованиям заказчика, необходимо точно представлять ее основные бизнес-функции и выяснить предъявляемые к системе требования. Для этого необходимо провести обследование компании и построить ее полную бизнес-модель.

Однако в данном примере не рассматривается общение с заказчиком, поэтому нет возможности провести такое обследование. В данном случае будет использоваться демонстрационный пример и обобщенное описание.

1.1. Краткое описание работы системы

Каждый товар в каталоге описывается артикулом, категорией, ценой и фото с кратким описанием.

Покупатель может загрузить каталог товаров. Каталог содержит разделы (категории), имеет блочную структуру, состоит из набора товаров с фото, ценой и характеристиками. Покупатель складывает понравившиеся товары в корзину, при этом выбирая количество необходимого товара данного артикула.

Корзину можно изменить: просмотреть, удалить товар, изменить количество позиций одного артикула, вернуться в каталог.

Когда покупатель делает заказ, он вводит свои личные данные, телефон и оплачивает его по банковской карте (если заказ не оплачен, то он и не сделан).

После того как сделан заказ, его можно забрать со склада через 1 рабочий день. Данные о заказе поступают сотруднику магазина, назовем его сотрудником отдела продаж, он проверяет наличие товаров и передает его кладовщику на комплектацию. Кладовщик, собрав заказ, делает отметку о готовности.

Заказ выдается со склада кладовщиком. Кладовщик выдает заказ и отмечает в системе, что заказ выдан.

Магазин не занимается доставкой заказов, не делает скидок. Для того чтобы ограничить масштаб задачи, не рассматривается система снабжения

магазина новыми товарами. Этим занимается другая система, назовем ее Склад. Информация о проданных товарах (т.е. сделанных заказах) поступает также в систему Склад.

Поскольку на протяжении от создания до выдачи заказа, он проходит разные стадии, то будет разумно ввести понятие статуса заказа. Сотрудники магазина могут статус заказа изменять, а покупатель может проследить за сборкой заказа. В таком случае наша система предоставляет еще одну функцию: узнать статус заказа.

2. Характеристика и возможности CASE-средства. Рабочий интерфейс программы. Создание проекта

2.1. Характеристика и возможности CASE-средства. Рабочий интерфейс программы

Целью моделирования рассматриваемой системы является знакомство с нотацией UML и изучение приемов работы в выбранном (определенном) CASE-средстве моделирования и проектирования.

В данной работе используется CASE-средство моделирования и проектирования **StarUML** v2.8.0 (http://staruml.io/download).

- Дать краткую характеристику, возможности и описать элементы интерфейса программы **StarUML**.

2.2. Создание проекта

Для создания нового проекта в **StarUML** можно выбрать **Rational** из списка предложенных подходов. Проект моделей будет иметь четыре представления: Use Case, Logical, Component и Deployment. Данный подход по структуре представлений наиболее соответствует методологии Rational Unified Process (RUP), которая поддерживает итеративный процесс разработки информационных систем (рисунок 2.1).

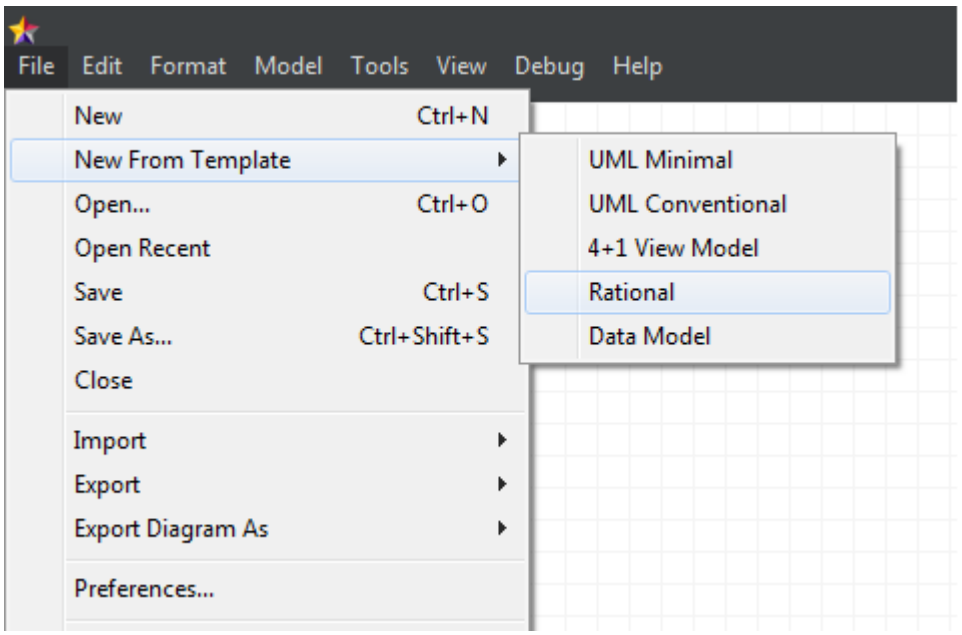

Рисунок 2.1 – Выбор шаблона Rational

Подход Rational выбирается для удобства дальнейшей работы. Сохраните созданный вами проект под соответствующим именем.

3. Диаграмма вариантов использования (Use Case Diagram)

Поведение системы (т.е. функциональность, которую она обеспечивает) описывают с помощью функциональной модели, которая отображает системные прецеденты (use cases, случаи использования), системное окружение (действующих лиц, актеров actors) и связи между ними.

Диаграмма вариантов использования (диаграмма прецедентов, use case diagram) — это диаграмма, на которой изображаются отношения между актерами и вариантами использования.

С помощью этой диаграммы можно:

- Определить общие **границы и контекст** моделируемой предметной области на начальных этапах проектирования системы;
- 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.

Диаграмма вариантов использования (прецедентов) представляет собой граф, в вершинах которого расположены актеры или прецеденты, связи между вершинами – это разного вида отношения.

Актером (действующее лицо, actor) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне.

Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. На диаграммах вариантов использования актер изображается в виде человечка, под которым записывается его имя (рисунок 3.1).

Рисунок 3.1 – Действующее лицо (актер)

Вариант использования (прецедент, use case) — описание множества последовательности действий (включая варианты), выполняемых системой для того, чтобы актер мог получить определенный результат.

Каждый вариант использования должен быть независимым в том смысле, что если он всегда выполняется совместно с другим вариантом использования, то, скорее всего, это один прецедент, а не два, либо они связаны отношением включения или расширения, о чем речь пойдет позже. Как следует из определения, прецедент (или вариант использования) должен обладать результирующей ценностью для актера, актер должен получить некоторый результат исполнения прецедента. Скорее всего, после исполнения прецедента в системе произойдут некоторые изменения: появятся новые данные, изменится поведение. Каждый вариант использования должен исполняться от начала до конца.

Прецедент описывает, **что делает** система, но никак **не уточняет, как** она это делает. Заметим, что диаграмма прецедентов не отображает последовательность, в которой будут выполняться варианты использования. На диаграмме прецедент изображается в виде эллипса.

Имя прецедента может состоять из нескольких слов и знаков препинания (за исключением двоеточия), как правило, имя выбирают в виде словосочетания или глагольного выражения (рисунок 3.2).

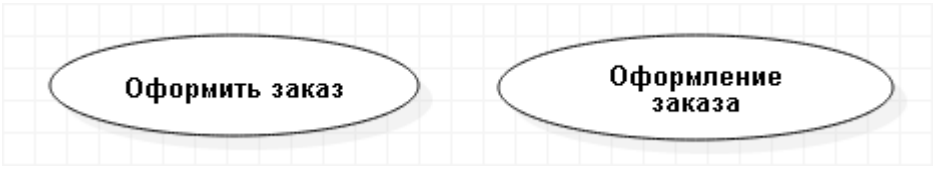

Рисунок 3.2 – Варианты использования (прецеденты)

Одним из наиболее важных (и дорогостоящих) этапов проектирования информационных систем является этап определения требований к системе. Если требования заказчика информационной системы разработчиками будут определены не корректно, то в итоге заказчик может получить совсем не ту систему, которую он ожидал.

Моделирование прецедентов и актеров помогает нам лучше понять требования, предъявляемые к системе, и согласовать их с заказчиком с помощью демонстрации и обсуждения диаграммы прецедентов. Прецеденты и актеры – это отражение требований к системе, они показывают, кто и для чего будет использовать будущую систему.

Разрабатывая диаграммы Use Case, придерживаются определенных правил:

1. Не моделируют связи между действующими лицами. По определению они находятся вне сферы действия системы. Связи между ними не относятся к ее компетенции.
2. Не соединяют стрелкой непосредственно два варианта использования (кроме связей использования и расширения). Диаграмма описывает только, какие варианты использования доступны системе, а не порядок их выполнения.
3. Каждый вариант использования должен быть инициирован действующим лицом. Всегда должна быть стрелка, начинающаяся на действующем лице и заканчивающаяся на варианте использования (кроме связей использования и расширения).
4. Не рисуют стрелки от одного варианта к другому для изображения потока информации.
5. Предполагается, что БД располагается на более низком уровне (слое), находящемся под диаграммой. С помощью одного варианта использования можно вводить данные в базу, а получать их – с помощью другого.

Пример. Определим актеров и прецеденты системы заказов магазина «Style». Напомним, что покупатель делает заказ, складывая товары в корзину. Возможна только одна форма оплаты: банковской картой по интернету, невозможно оформление заказа без оплаты. Заказ имеет статус: оплачен, передан на комплектацию, собран, получен. Статус заказа изменяется автоматически либо сотрудником магазина. Покупатель может узнать статус своего заказа по уникальному номеру заказа.

Система не занимается поставками товаров в магазин. Этим занимается другая система, назовем ее Склад.

Таким образом, с нашей системой взаимодействуют покупатель, сотрудники магазина и внешняя система Склад. С нашей системой взаимодействуют сотрудник отдела продаж, который проверяет оплату заказа и отправляет его на комплектацию, и кладовщик, который собирает заказ и выдает его покупателю. С точки зрения бизнеса – это две разных должности, выполняющих разные функции, но с точки зрения системы они играют одну роль сотрудника, изменяющего статус заказа покупателя с использованием программного обеспечения моделируемой системы. В этом смысле для системы нет разницы между сотрудником отдела продаж и кладовщиком. Выбирая

действующих лиц, нужно помнить о том, что мы должны отразить их роль, а не должность. Введем обобщающее сотрудников действующее лицо – Сотрудник. Другой пример: сотрудник магазина «Style» (положим, кладовщик) может выступать в роли сотрудника и общаться с системой как сотрудник магазина, а может выступать и в роли покупателя, сделав заказ в магазине. Не смотря на то, что физически это один человек, он выступает в роли двух актеров: покупателя и сотрудника. Итак, актеры системы заказов магазина «Style» будут следующие:

Покупатель, Сотрудник, Система Склад.

Покупатель использует нашу систему для того, чтобы заказать вещи, он просматривает каталог, добавляет понравившиеся ему товары в корзину, открывает корзину, удаляет из нее товары или изменяет их количество и, наконец, может оформить свой заказ, при этом его оплатив. В конечном итоге результат использования системы покупателем будет получен, если он выполнил все эти действия от начала до конца. Поэтому не будем разделять заказ товаров на несколько прецедентов, а выделим только один: Заказ товаров.

Покупатель, сделав заказ в магазине «Style», может в дальнейшем узнавать статус своего заказа, это тоже случай использования системы, назовем его Получение информации о заказе.

Сотрудник должен изменять статус сделанного заказа, для него определим прецедент Управление статусом заказа.

Система Склад должна получать информацию о сделанных заказах для возможности управления наличием товаров на складе, для нее также должен быть доступен прецедент Получение информации о заказе.

Итак, прецеденты системы заказов магазина «Style»: Заказ товаров, Управление статусом заказа, Получение информации о заказе.

3.1. Отношения между прецедентами и актерами

Связи и взаимоотношения, существующие между элементами модели, в UML описываются с помощью отношений, изображаемых на диаграммах.

Отношение (relationship) — это семантическая связь между отдельными элементами модели.

Между актерами и прецедентами диаграммы вариантов использования могут существовать разного рода отношения, показывающие, что экземпляры действующих лиц и вариантов использования взаимодействуют друг с другом.

Действующие лица могут взаимодействовать с несколькими прецедентами и между собой, равно как и прецеденты могут быть связаны между собой особого типа отношениями. В основном на диаграммах прецедентов используются следующие типы отношений:

- ассоциации (association relationship);
- включения (include relationship);
- расширения (extend relationship);
- обобщения (generalization relationship).

Ассоциация – это структурное отношение, показывающее, что объект неким образом связан с другим объектом.

Отношение этого типа используется не только на диаграммах прецедентов, но и на других диаграммах. Если между элементами модели показано отношение ассоциации, то это означает, что между ними существует семантическая связь. Ассоциативное отношение может быть направленным. В этом случае направление связи показывает, кто является инициатором. Если отношение направленно от актера к прецеденту, то это означает, что актер инициирует выполнение прецедента.

Пример. Покупатель в системе заказов магазина «Style» инициирует выполнение прецедента Заказ товаров (рисунок 3.3).

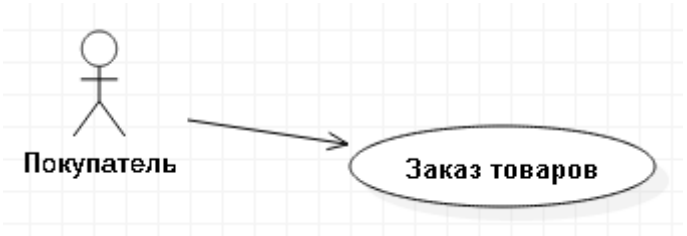

Рисунок 3.3 – Отношение ассоциации между актером и прецедентом

Между прецедентами также возможны взаимоотношения, которые описываются отношениями двух типов: включения и расширения (дополнения).

Включение (include) говорит о том, что исходный прецедент явным образом включает в себя поведение целевого.

Другими словами, включение создается, когда один прецедент использует другой. При этом исполнение базового прецедента невозможно без исполнения используемого. Изображается включение в виде пунктирной стрелки с надписью <<include>>, которая направлена от базового элемента к используемому.

Пример. В системе заказов магазина «Style» невозможен заказ товаров без оплаты. На диаграмме прецедентов это можно отразить так, как это показано на рисунке 3.4.

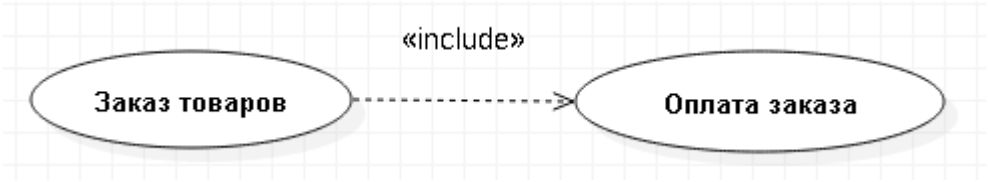

Рисунок 3.4 – Отношение включения между прецедентами

Расширение (extend) показывает, что целевой прецедент расширяет поведение исходного прецедента.

Используемый прецедент выполняется не всегда вместе с базовым, а только при выполнении дополнительных условий, таким образом, расширяя функциональность базового элемента. Изображается расширение пунктирной стрелкой с надписью <<extend>>, направленной от используемого варианта использования к базовому.

Пример. При заказе товаров в системе заказов магазина «Style» покупатель может изменить содержание корзины перед тем, как оформить заказ окончательно, а может оставить корзину без изменений. Изменение корзины – это опция, которую на диаграмме вариантов использования мы можем изобразить с помощью расширения (рисунок 3.5).

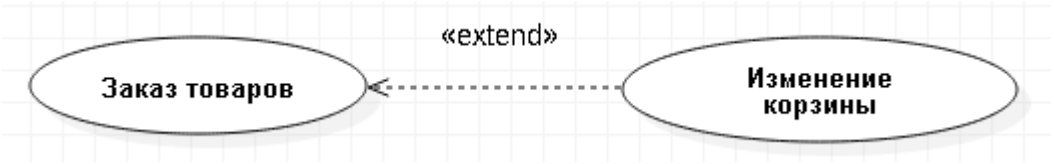

Рисунок 3.5 – Отношение расширения между прецедентами

Обозначения отношений <<include>> и <<extend>> есть не что иное, как обозначения стереотипов, которые широко используются в UML для создания новых элементов модели путем расширения функциональности базовых элементов.

Стереотип (Stereotype) – это механизм, позволяющий систематизировать элементы модели с помощью категорий. С помощью стереотипов мы можем создавать своего рода подтипы типов. Это позволяет UML иметь минимальный набор элементов, которые могут быть дополнены при необходимости для создания связующих базовых элементов в системе. В UML стереотип обозначается именем, которое записывается в двойных угловых скобках: **<<имя стереотипа>>.** В UML мы можем создавать собственные стереотипы на основе уже имеющихся типов, но также существуют и стандартные, заранее определенные стереотипы нотации UML. Так, отношение зависимости (о котором мы еще будем говорить) расширяется для прецедентов и актеров с помощью двух стереотипов <<include>> и <<extend>>.

Ассоциация – это коммуникативное отношение, которое соответствует стереотипу <<communicate>>, который, впрочем, всегда опускается.

Два и более актера могут иметь общие свойства, т.е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.

Обобщение (Generalization) – это отношение между общей сущностью и ее конкретным воплощением. На диаграммах обобщение обозначается стрелкой с не закрашенным треугольником на конце, направленной от частного элемента к общему.

Пример. Для изменения статуса заказов в магазине «Style» с проектируемой системой будут работать сотрудник отдела продаж и кладовщик. На диаграмме мы можем показать с помощью отношения обобщения взаимосвязь между актером Сотрудник и актерами Менеджер и Кассир (рисунок 3.6).

Рисунок 3.6 – Отношение обобщения между актерами

Актеры, прецеденты и отношения – это основные элементы нотации диаграмм вариантов использования. Диаграмма вариантов использования помогает отобразить основные требования к моделируемой системе и обеспечить взаимопонимание функциональности системы между разработчиком и заказчиком. Можно построить одну, главную диаграмму прецедентов, на которой будут отражены границы системы (актеры) и ее основная функциональность (прецеденты). Для более подробного представления системы допускается построение вспомогательных диаграмм прецедентов.

3.2. Построение диаграммы прецедентов

В StarUML диаграммы прецедентов (вариантов использования) располагаются в представлении **Use Case View**. Если в навигаторе модели (обозреватель Model Explorer) щелкнуть два раза по имени диаграммы, то откроется ее рабочее поле (рисунок 3.7).

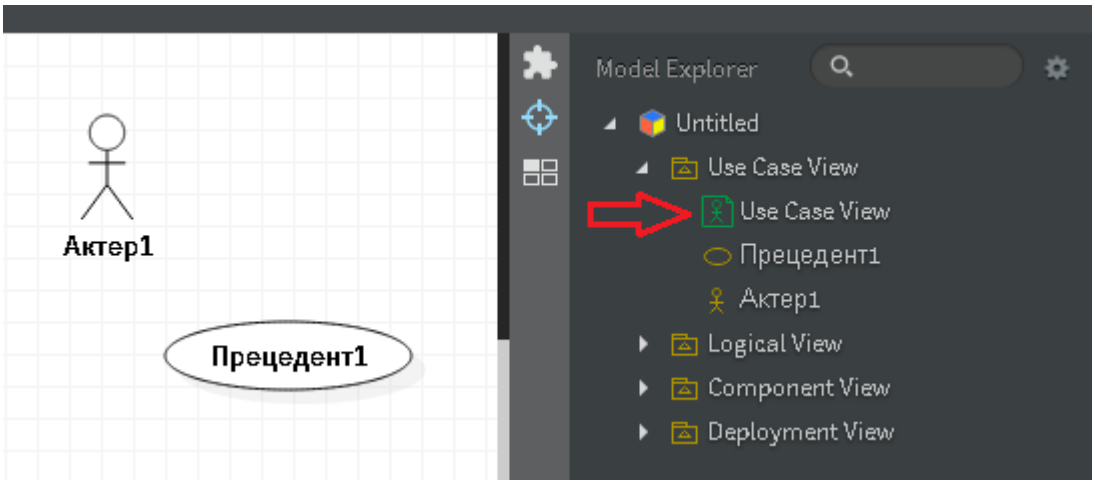

Рисунок 3.7 – Главная диаграмма прецедентов в представлении Use Case View

Для того чтобы создать прецедент, щелкните по овальному символу прецедента на панели элементов слева от рабочего поля диаграммы, а затем щелкните по тому месту на рабочем поле диаграммы, в которое вы хотите поместить прецедент. Аналогичным образом создается актер.

Когда элемент помещается на поле диаграммы, он становится доступен для редактирования имени и некоторых свойств. В выделенное поле введите новое имя прецедента или актера (см. рисунок 3.7).

Для создания отношения между элементами диаграммы щелкните по изображению соответствующего отношения на панели элементов справа, а затем проведите линию от одного элемента к другому, удерживая левую кнопку мыши.

Чтобы удалить элемент с диаграммы достаточно щелкнуть левой кнопкой мыши по этому элементу, а затем нажать кнопку Delete, либо щелкнуть правой кнопкой мыши по элементу и в контекстном меню выбрать команду Delete.

Обратите внимание, что элемент был удален с диаграммы, но не из модели (рисунок 3.8). Его можно найти в навигаторе модели (обозреватель модели Model Explorer), несмотря на то, что на диаграмме он больше не отображается (элемент «Актер1» не отображается в рабочей области).

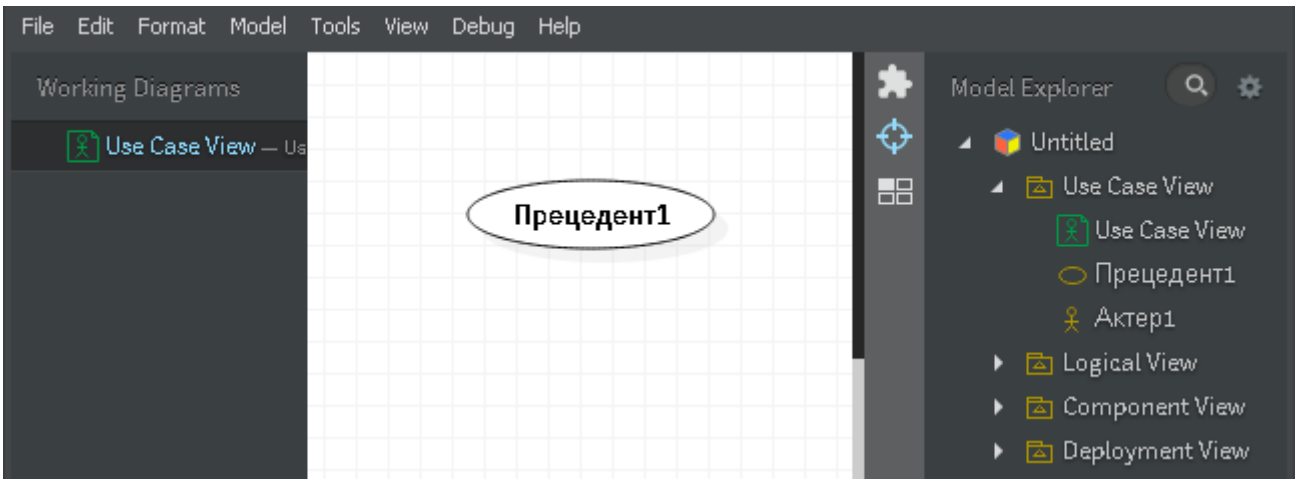

Рисунок 3.8 – Элемент «Актер1» удален из рабочей области диаграммы, но отображается в обозревателе модели

Если мы передумали и решили вернуть элемент на диаграмму, то это можно сделать, перетащив его из навигатора модели на поле диаграммы.

Для того чтобы **удалить элемент из модели** нужно щелкнуть по нему на диаграмме или по его изображению в навигаторе модели правой кнопкой мыши и в контекстном меню выбрать пункт **Delete from Model**. Элемент будет полностью удален.

Описанные выше способы добавления и удаления элементов и отношений могут быть использованы для построения диаграмм любых типов. Все описанные операции доступны также из главного меню StarUML.

Пример. Для системы заказов магазина «Style» мы определили актеров Покупатель, Сотрудник, Система Склад и прецеденты Заказ товаров, Управление статусом заказа, Получение информации о заказе. Построим основную диаграмму прецедентов (рисунок 3.9).

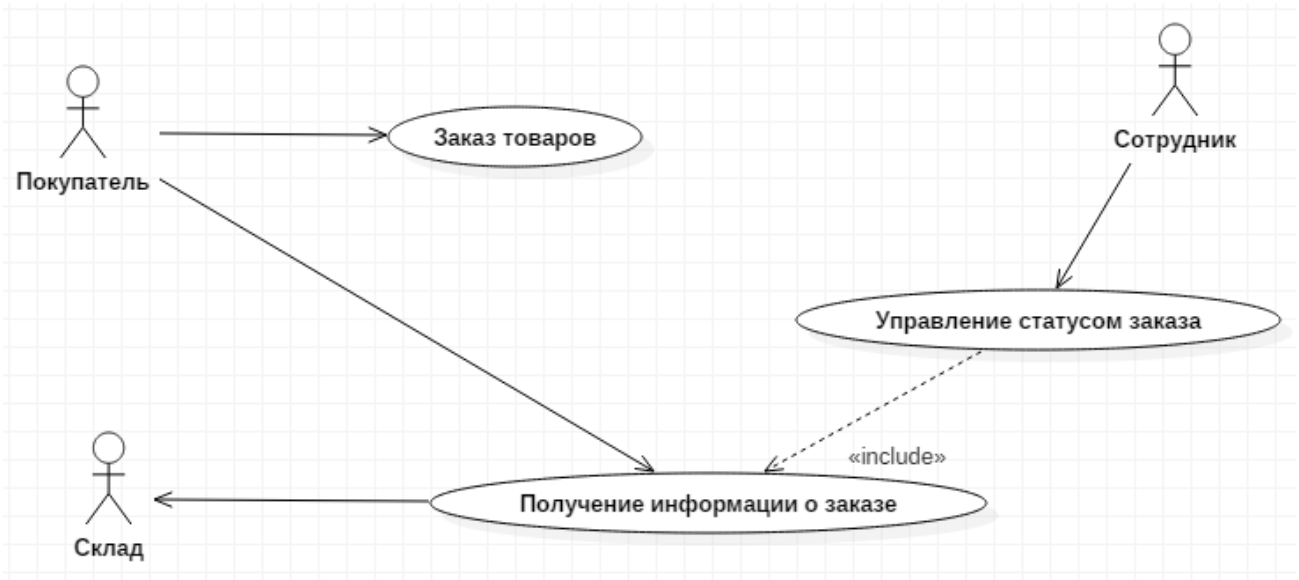

Рисунок 3.9 – Основная диаграмма вариантов использования системы заказов

Для актера Покупатель и прецедента Заказ товаров установили отношение направленной ассоциации: Заказ товаров инициализируется Покупателем. Сотрудник имеет возможность управлять статусом заказа, при этом он непременно участвует в прецеденте Получение информации о заказе. Направленную ассоциацию от Получение информации о заказе к актеру Система Склад можно понимать, как автоматическую передачу данных из моделируемой системы в систему снабжения товарами Склад.

В модель нужно включить краткое описание каждого актера или прецедента, делается это для того, чтобы между разработчиком и заказчиком системы не оставалось «белых пятен» и расхождений в понимании функциональности системы и ролей взаимодействующих с ней актеров. Для каждого актера описывается роль, которую он играет в системе, а для каждого прецедента – его назначение и функциональность. Также можно уточнить, каким актером запускается прецедент.

3.3. Документирование элементов модели

В StarUML добавление описания к элементам модели делается следующим образом. Выделите элемент модели, щелкнув по нему мышкой, и откройте редактор Documentation. Введите описание элемента в окно документирования (рисунок 3.10).

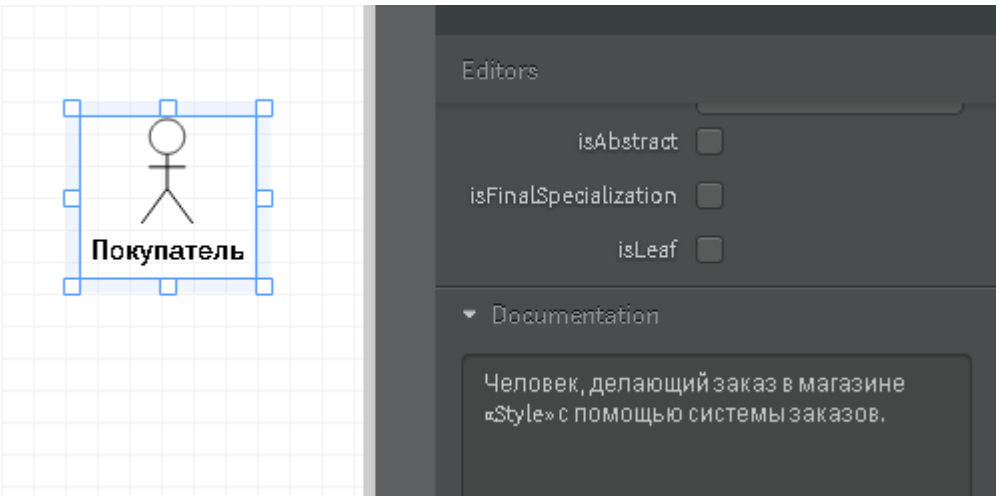

Рисунок 3.10 – Документирование элемента модели в StarUML

Все элементы модели должны быть задокументированы. Описанный выше способ подходит для любого элемента любой диаграммы.

Пример. Для актеров и прецедентов системы заказов магазина «Style» сделаем краткое описание. Покупатель – это человек, делающий заказ в магазине «Style» с помощью системы заказов.

Сотрудник – это все сотрудники магазина «Style», которые могут получать информацию о сделанных заказах и изменять статус заказа в системе в зависимости от того шага, на котором находится обработка данного заказа.

Система Склад – это внешняя система, которая получает информацию о сделанных в магазине «Style» заказах для того, чтобы обеспечить учет наличия товаров на складе и снабжение товарами.

Заказ товаров – этот прецедент запускается покупателем для того, чтобы оформить заказ в магазине «Style». Состоит из просмотра каталога, добавления товаров в корзину, просмотра корзины, изменения содержания корзины и оформления заказа, включая оплату.

Управление статусом заказа – этот прецедент используется сотрудниками магазина для изменения статуса заказа в процессе его обработки.

Получение информации о заказе – прецедент используется всеми актерами для просмотра информации о заказе.

3.4. Создание дополнительной диаграммы прецедентов

Для того чтобы создать еще одну диаграмму (любого типа), например, детализирующую прецедент, необходимо в обозревателе Model Explorer щелкнуть правой кнопкой мыши по папке **Use Case View** и в появившемся контекстном меню выбрать **Add Diagram**, затем выбрать из списка требуемую диаграмму.

Например, можно создать дополнительную диаграмму прецедентов, выбрав пункт **Use Case Diagram** (рисунок 3.11).

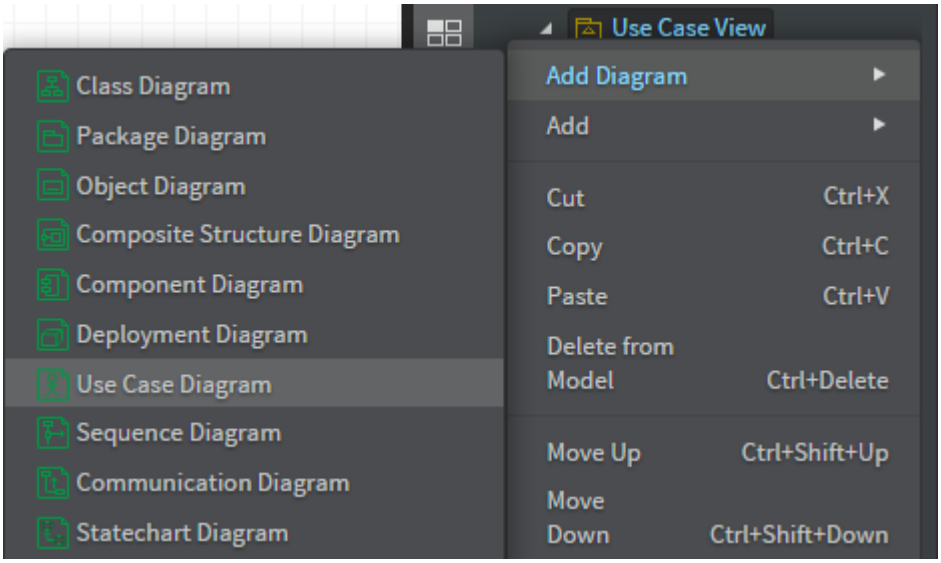

Рисунок 3.11 – Создание дополнительной диаграммы прецедентов

Пример. Наиболее значимым для данной системы и ее актеров прецедентом является прецедент Заказ товаров.

Для него следует построить дополнительную диаграмму прецедентов, поясняющую этот вариант использования (рисунок 3.12).

Рисунок 3.12 – Диаграмма вариантов использования, поясняющая прецедент «Заказ товаров»

Созданной дополнительной диаграмме вариантов использования назначьте соответствующее имя (рисунок 3.13).

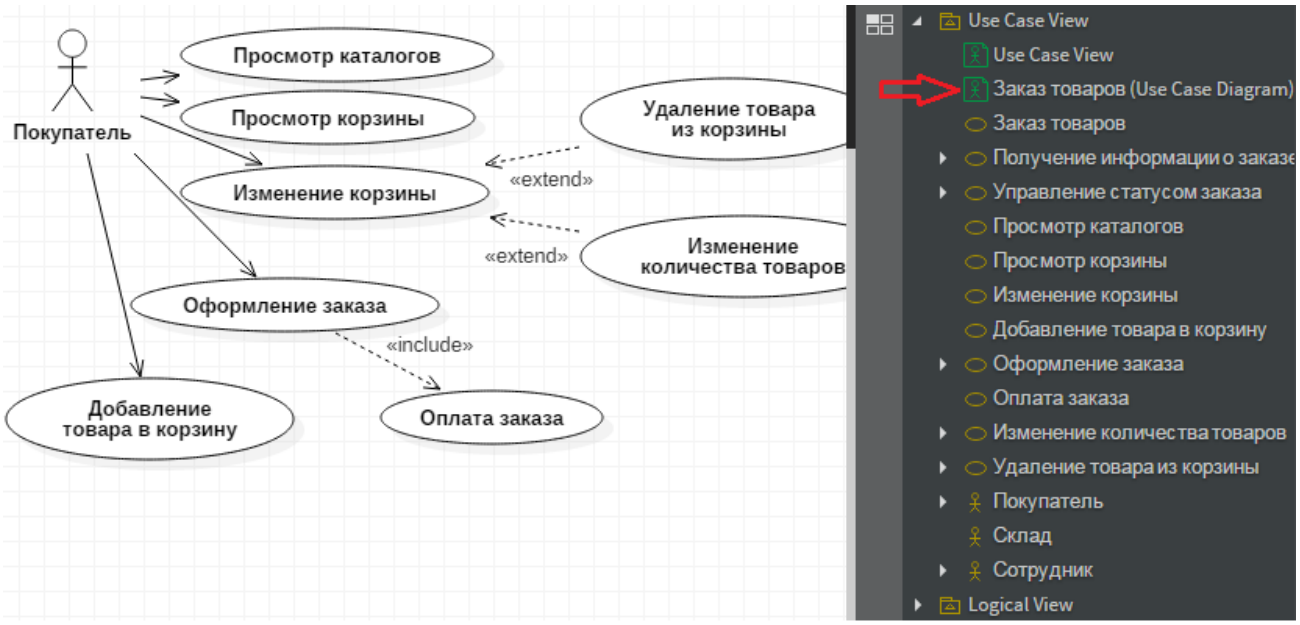

Рисунок 3.13 – Название «Заказ товаров (Use Case Diagram)» для дополнительной диаграммы

4. Посторенние диаграммы деятельности

Диаграммы деятельности обеспечивают еще один способ моделирования потока событий. С помощью текстового описания можно рассказать о потоке, но трудно будет понять логику событий в сложных и запутанных потоках с множеством альтернативных ветвей.

Диаграммы деятельности создаются также на разных этапах жизненного цикла системы для отражения последовательности выполнения операций.

4.1. Основные элементы диаграмм деятельности

Рассмотрим основные элементы нотации диаграмм деятельностей. На них иллюстрируются деятельности, переходы между ними, элементы выбора и синхронизации.

Деятельностью называется исполнение определенного поведения в потоке управления системы. В UML деятельность изображается в виде скругленного прямоугольника с текстовым описанием внутри.

Пример. Деятельность обозначает некоторый шаг (этап) процесса. В прецеденте Заказ товаров одним из таких шагов может быть Добавить товар в корзину (рисунок 4.1).

Рисунок 4.1 – Деятельность

Переход показывает, как поток управления переходит от одной деятельности к другой. Обычно переход осуществляется по завершении деятельности (рисунок 4.2).

Пример. В нашем примере выполняя Заказ товаров покупатель может Открыть корзину и Удалить товар из нее. Это две разные деятельности, переход к удалению товара возможен только после открытия корзины.

Рисунок 4.2 – Переход между деятельностями

Два состояния на диаграмме деятельности – **начальное и конечное** – определяют продолжительность потока. Начальное состояние обязательно должно быть отмечено на диаграмме, оно определяет начало потока. Конечных состояний может быть несколько или не одного. Оно определяет точку завершения потока. Конечных состояний может быть несколько, но начальное должно быть только одно. Начальное состояние изображается жирной точкой, а конечное – жирной точкой в окружности (рисунок 4.3).

Рисунок 4.3 – Обозначения начального (Initial) и конечного (Final) состояний

При моделировании управляющих потоков системы часто бывает необходимо показать места их разделения на основе **условного выбора**. Выбор на диаграмме показывается ромбом, помещенным на переходе. Ограничительные условия, от которых зависит выбор направления перехода, помещаются обычно над ромбом. В нотации UML условия записываются в квадратных скобках: [условие].

Пример. Если все товары, которые хочет заказать покупатель, добавлены в корзину, то покупатель может просмотреть корзину и оформить заказ. Условие перехода от деятельности Добавить товар в корзину к Просмотреть корзину на диаграмме можно показать так, как это изображено на рисунке 4.4.

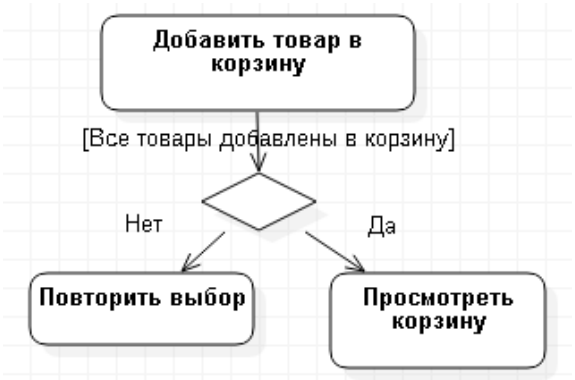

Рисунок 4.4 – Условие перехода между деятельностями

Синхронизация – это способ показать, что две или более ветвей потока выполняются параллельно. Деятельности, помещенные между двумя жирными линиями на диаграмме деятельности, исполняются синхронно, одновременно.

Пример. После оплаты заказа покупателем система присваивает заказу уникальный номер и отсылает подтверждение заказа на электронную почту покупателя. Эти две деятельности можно выполнить синхронно. Как это изображается на диаграмме, показано на рисунке 4.5.

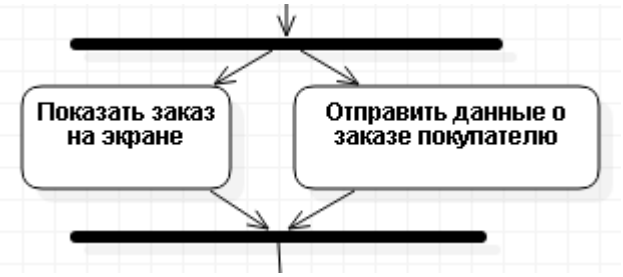

Рисунок 4.5 – Линии синхронизации

Секции делят диаграмму деятельности на несколько участков. Это нужно для того, чтобы показать, кто отвечает за выполнение деятельности и в каком порядке. Если деятельность находится на секции с именем Покупатель, то этот актер и выполняет ее.

Пример. Секция актера Покупатель изображена на рисунке 4.6.

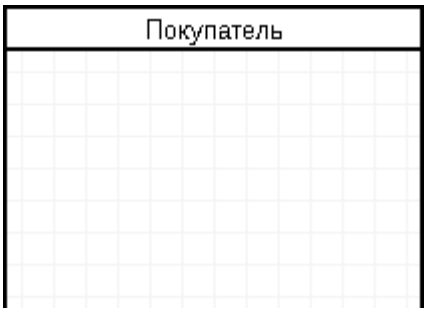

Рисунок 4.6 – Секция

4.2. Создание диаграммы деятельности

Чтобы построить диаграмму деятельности для некоторого прецедента в StarUML, нужно щелкнуть правой кнопкой мыши по этому прецеденту, в выпавшем контекстном меню выбрать пункт Add Diagram, затем в появившемся списке выбрать **Activity Diagram** (рисунок 4.7).

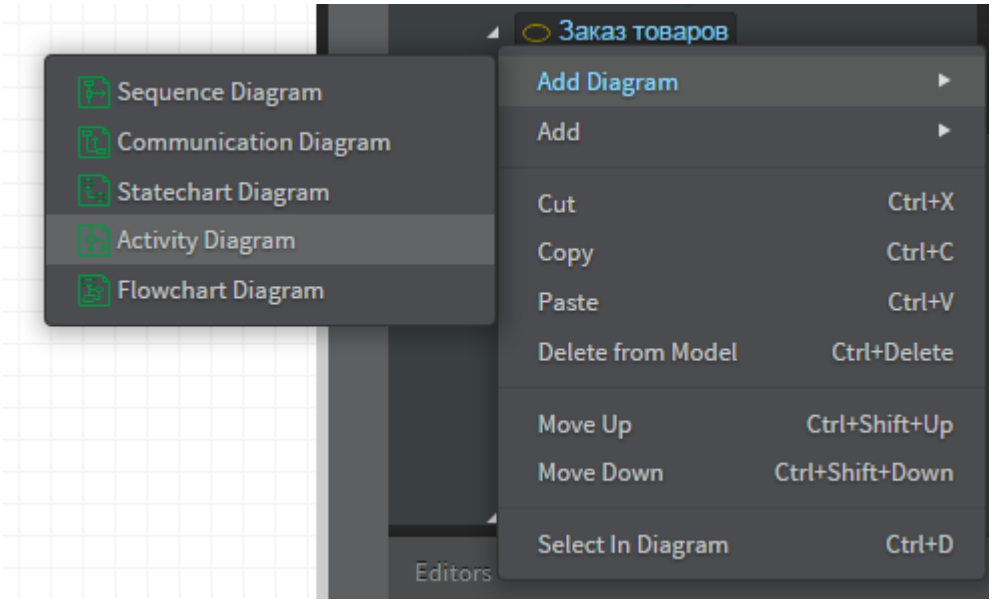

Рисунок 4.7 – Добавление диаграммы деятельности

Поле для создания диаграммы деятельности появится в окне программы, изменится панель инструментов слева, и новая диаграмма отобразится на навигаторе модели.

Пример. Построим диаграмму деятельности для дополнительного прецедента Оформить заказ актера Покупатель (рисунок 4.8). Оформление заказа включает указание своих личных контактных данных, электронной почты и оплату заказа. Оформление начинается из корзины покупателя, когда он выбирает опцию «Оформить заказ».

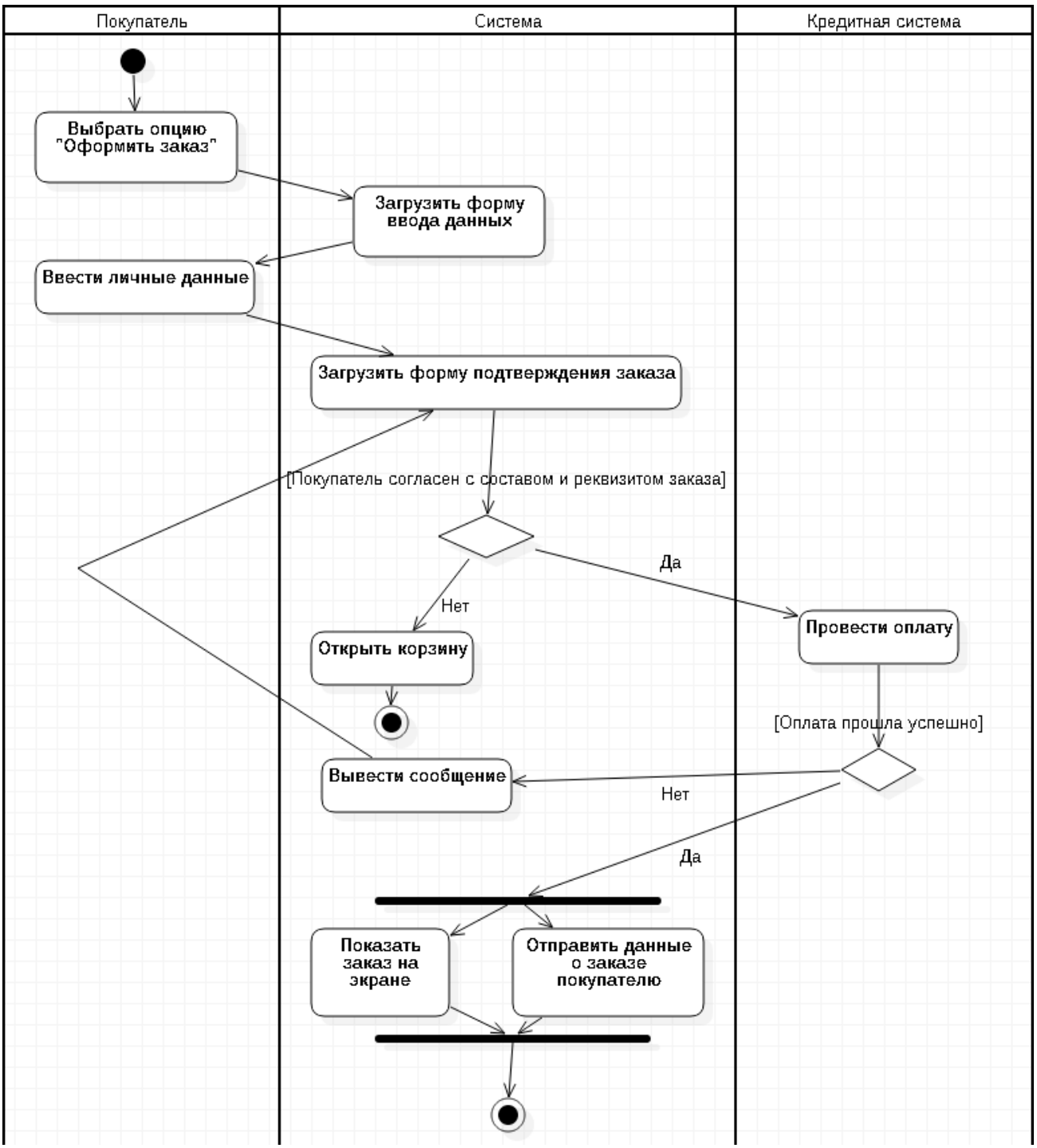

Рисунок 4.8 – Диаграмма деятельности прецедента «Оформить заказ»