МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ»

М.Д. Гочияева

УНИФИЦИРОВАННЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ UML

Учебно-методическое пособие для обучающихся 3 курса
по направлению подготовки 09.03.04 Программная инженерия

Черкесск 2024 г.

УДК 004.434:004.94
ББК 32.973.2
Г 74

Рассмотрено на заседании кафедры «Прикладная информатика»
Протокол №1 от 31.08.2023 г.
Рекомендовано к изданию редакционно-издательским советом СКГА
Протокол №26 от 29.09.2023 г.

Рецензенты: Кочкарова П.А. – к.ф-м.н., доцент кафедры ПИ

Г 74 **Гочияева, М.Д**. Унифицированный язык модели UML: учебно-методическое пособие для обучающихся 3 курса по направлению подготовки 09.03.04 Программная инженерия / М.Д. Гочияева. – Черкесск, БИЦ СКГА, 2024. – 24 с.

Учебно-методическое пособие предназначено обучающимся 3 курса направления подготовки 09.03.04 «Программная инженерия» для аудиторных и самостоятельных занятий по дисциплине предметам «Унифицированный язык моделирования UML

В пособии описываются основные элементы нотации диаграмм UML, на конкретном примере рассматривается процесс визуального моделирования информационных систем с применением CASE-инструмента StarUML, приведены варианты индивидуальных заданий.

УДК 004.434:004.94
ББК 32.973.2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для обучающихся 3 курса направления подготовки 09.03.04 «Программная инженерия» и направлено на изучение унифицированного языка моделирования UML (Unified Modeling Language). UML является стандартным языком визуального моделирования, который широко используется в процессе разработки программного обеспечения. Знание и понимание основ UML является важным навыком для будущих программных инженеров, так как позволяет эффективно проектировать, документировать и анализировать сложные программные системы.

Данное пособие содержит теоретический материал, практические задания и рекомендации для самостоятельной работы, которые позволят обучающимся:

1. Изучить основные концепции и принципы унифицированного языка моделирования UML.
2. Освоить методы и техники построения различных типов UML-диаграмм.
3. Научиться применять UML для проектирования и документирования программных систем.
4. Развить навыки анализа и проектирования программного обеспечения с использованием UML.

Материалы пособия могут быть использованы как во время аудиторных занятий под руководством преподавателя, так и для самостоятельной работы обучающихся. Пособие также может быть полезно для обучающихся, изучающих смежные дисциплины, связанные с разработкой программного обеспечения.

ВИЗУАЛЬНОЕ МОДЕЛИРОВАНИЕ И UML

UML – это язык визуального моделирования систем. Моделирование систем с помощью UML предполагает построение ряда взаимосвязанных диаграмм. Для сопровождения процесса построения, анализа и документирования модели, а также проверки модели и генерации программных кодов разработчики используют специально для этих целей созданные CASE-инструменты проектирования систем.

В общем смысле CASE (Computer-Aided Software Engineering) – это набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Существует достаточно много CASE-инструментов моделирования и проектирования систем и баз данных (не только с помощью UML). В данном учебном пособии для примера моделирования системы выбран программный инструмент моделирования StarUML.

Данная программная платформа имеет свободную лицензию и доступна для установки с официального сайта StarUML.

Визуальным моделированием (visual modeling) называется способ представления идей и проблем реального мира с помощью моделей.

Модель – это абстракция, описывающая суть сложной проблемы или структуры без акцента на несущественных деталях, тем самым делая ее более понятной. Разработка программного обеспечения - не исключение. При построении сложной системы строятся ее абстрактные визуальные модели.

В настоящее время в области проектирования информационных систем с успехом применяется визуальное моделирование с помощью унифицированного языка моделирования UML.

Унифицированный язык моделирования (Unified Modeling Language, UML) является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению.

С помощью UML можно детально описать систему, начиная разработку с концептуальной модели с ее бизнес-функциями и процессами, а также описать особенности реализации системы, такие как классы программного обеспечения системы, схему базы данных. Используя UML, мы также можем разрабатывать сложные системы быстро и качественно.

Как язык графического визуального моделирования UML имеет свою нотацию – принятые обозначения. Нотация обеспечивает семантику языка, является способом унификации обозначений визуального моделирования, обеспечивает всестороннее представление системы, которое сравнительно легко и свободно воспринимается человеком.

Моделирование с помощью UML осуществляется поэтапным построением ряда диаграмм, каждая из которых отражает какую-то часть или сторону системы либо ее замысла.

Диаграмма – это графическое представление множества элементов. Обычно диаграмма изображается в виде графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы подчиняются нотации UML и изображаются в соответствии с ней.

Основные диаграммы UML:

– вариантов использования (use case diagram);
– классов (class diagram);
– кооперации (collaboration diagram);
– последовательности (sequence diagram);
– состояний (statechart diagram);
– деятельности (activity diagram);
– компонентов (component diagram);
– развертывания (deployment diagram).

Построения этих диаграмм достаточно для полного моделирования системы.

СОЗДАНИЕ НОВОГО ПРОЕКТА В STARUML

Основная структурная единица в StarUML – это проект. Проект сохраняется в одном файле в формате XML с расширением «.UML». Проект может содержать одну или несколько моделей и различные представления этих моделей (View) – визуальные выражения информации, содержащейся в моделях. Каждое представление модели содержит диаграммы – визуальные образы, отображающие определенные аспекты модели.

Новый проект будет автоматически создан при запуске программы StarUML. При этом вам будет предложено в диалоговом окне выбрать один из подходов (Approaches), поддерживаемых StarUML (рисунок 1).

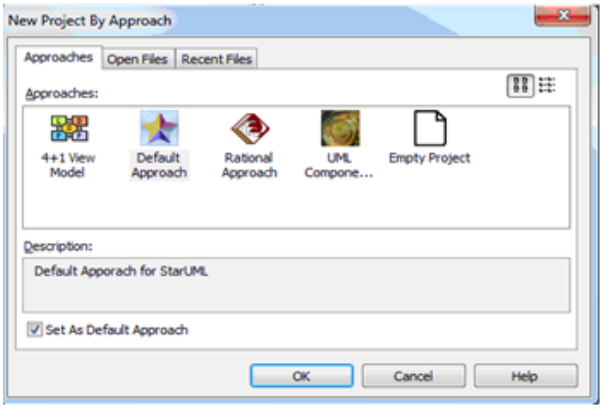

Рисунок 1–Выбор подхода

Существуют различные методологии моделирования информационных систем, компании-разработчики систем также могут разрабатывать свои методологии. Следовательно, на начальной стадии проектирования

необходимо определить основные положения методологии или выбрать одну из уже существующих. Для того чтобы согласовать между собой различные элементы и этапы моделирования, StarUML предлагает концепцию подходов.

После того как мы выбрали один из предложенных подходов, появится основное окно программы (рисунок 2).

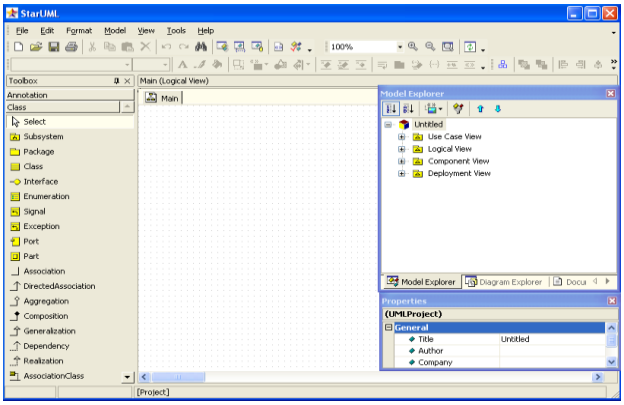

Рисунок 2–Основное окно программы

В верхней части окна расположено главное меню, кнопки быстрого доступа. Слева расположена панель элементов (Toolbox) с изображениями элементов диаграммы. Элементы соответствуют типу выбранной диаграммы. В центре находится рабочее поле диаграммы, на котором она может быть построена с использованием соответствующих элементов панели инструментов.

Справа находится инспектор модели, на котором можно найти вкладки навигатора модели Model Exploer, навигатора диаграмм Diagram Exploer, окно редактора свойств Properties, окно документирования элементов модели Documentation и редактор вложений Attachments. Внешний вид инспектора модели с вкладками представлен ниже (рисунок 3).

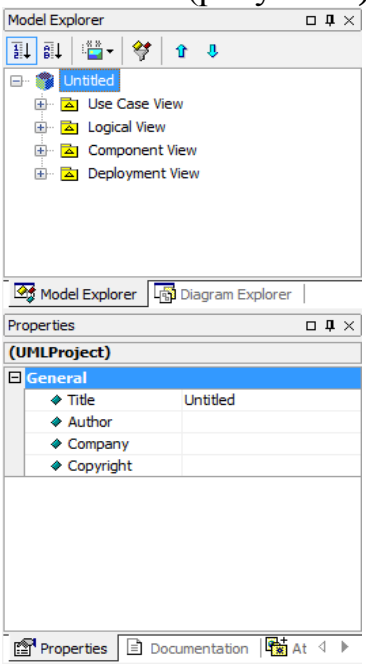

Рисунок 3–Инспектор модели

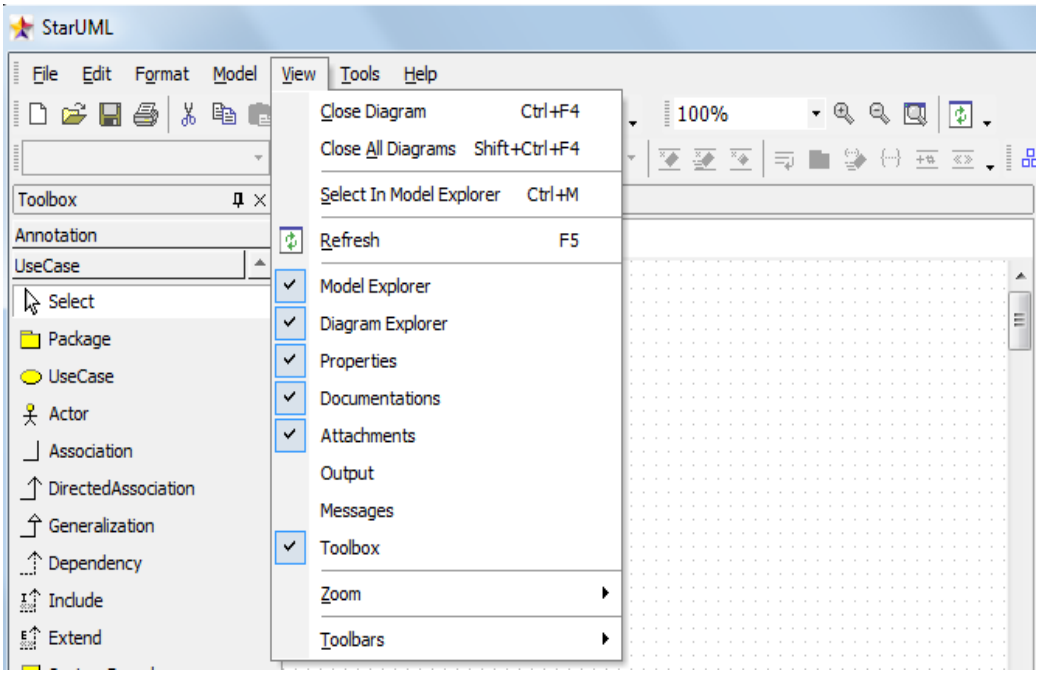

Рисунок 4–Пункт меню View. Управление видом инспектора модели

Управлять видом инспектора модели, панели элементов, закрывать и открывать редакторы инспектора можно с помощью пункта меню View (рисунок 4). Если рядом с пунктом меню стоит «галочка», этот элемент активен и его можно видеть в окне программы или открыть на доступных вкладках инспектора модели.

Иерархическая структура проекта отображается справа на навигаторе модели (Model Exploer). В зависимости от выбранного подхода на навигаторе модели будут отображены различные пакеты представлений модели. Каждый пакет представления будет содержать элементы моделей и диаграмм, которые мы создадим.

Если при создании нового проекта моделирования мы выберем подход Rational Approach, то при таком подходе в навигаторе будут присутствовать четыре пакета представлений модели системы (рисунок 5):

Use Case View – представление требований к системе, описывает, что система должна делать;

Logical View – логическое представление системы, описывает, как система должна быть построена;

Component View – представление реализации, описывает зависимость между программными компонентами;

Deployment View – представление развертывания, описывает аппаратные элементы, устройства и программные компоненты.

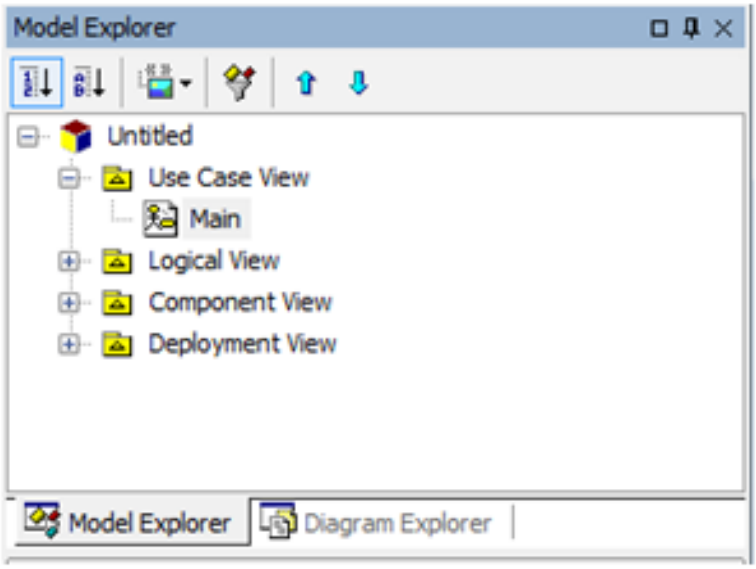

Рисунок 5–Навигатор модели

Сейчас каждое представление содержит одну диаграмму с именем Main. Если щелкнуть по ней два раза, то откроется рабочее поле этой диаграммы и соответствующая панель инструментов.

Пример. Если щелкнуть два раза по диаграмме Main представления Use Case View, то откроется рабочее этой диаграммы и ее панель элементов (рисунок 6).

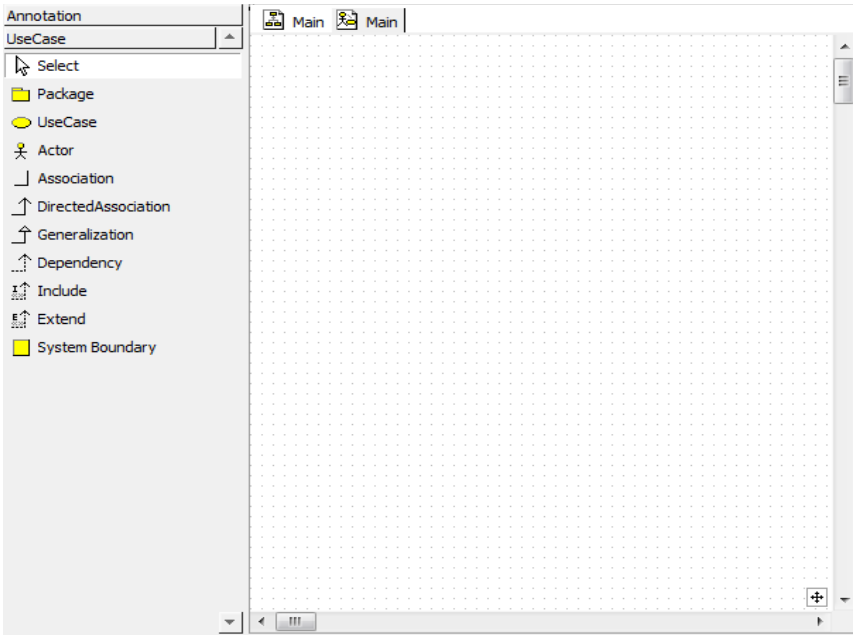

Рисунок 6–Рабочее поле диаграммы прецедентов Main и ее панель элементов

Ниже иерархии представлений отображаются свойства выделенного элемента модели или диаграммы (в данном случае свойства диаграммы Main, так как она выделена в навигаторе модели) (рисунок 7).

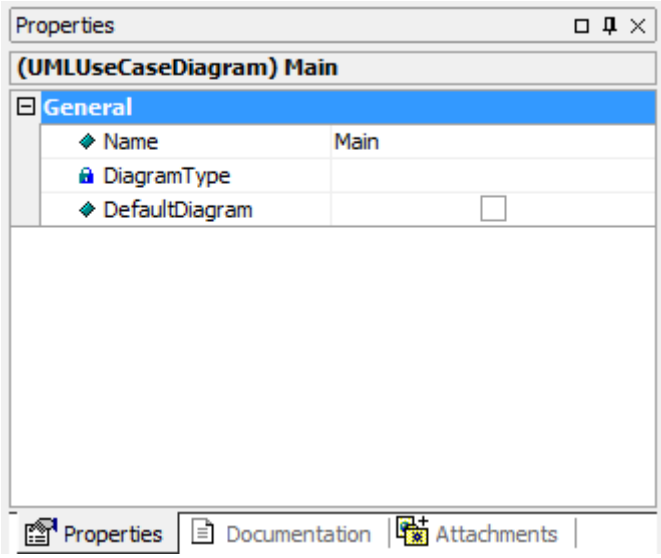

Рисунок 7–Редактор свойств

Рассмотрим визуальное моделирование систем с использованием UML на примере проектирования системы заказов интернет-магазина «Style».

ПОСТАНОВКА ЗАДАЧИ

Магазин занимается продажей одежды и обуви различных брендов. Покупатель просматривает каталог и делает заказ. Предполагаем, что потенциальный клиент заходит на сайт магазина, он 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, мы должны абсолютно четко представить себе ее основные бизнес-функции и выяснить предъявляемые к системе требования. Для этого необходимо провести обследование компании и построить ее полную бизнес-модель. Поскольку наш пример является придуманным, мы не можем провести такое обследование и не имеем возможности общаться с заказчиком, то мы будем опираться на придуманное нами словесное описание системы.

Описание работы системы

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

Покупатель может загрузить каталог товаров. Каталог не содержит разделы, имеет блочную структуру, состоит из набора товаров с фото, ценой и размерами. Покупатель складывает понравившиеся товары в корзину, при этом выбирая размер и 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, мы не рассматриваем систему 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.

Создание проекта

Создадим новый проект в StarUML, выбрав Rational Approach из списка предложенных подходов. Наша модель будет иметь четыре представления: Use Case, Logical, Component и Deployment. Данный подход по структуре представлений на наш взгляд наиболее соответствует методологии Rational Unified Procces (RUP), которая поддерживает итеративный процесс разработки информационных систем. Настоящее пособие не претендует на полное соответствие этапам RUP: целью моделирования нашей системы является знакомство с нотацией UML и изучение приемов работы в CASE-средстве проектирования и моделирования StarUML. Мы выбираем подход Rational Approach для удобства дальнейшей работы. Сохраните созданный вами проект.

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

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 (use cases diagrams).

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 8).

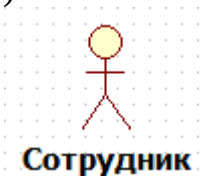

Рисунок 8–Действующее лицо (актер)

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. 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 9).

Рисунок 9–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.

Пример. Определим актеров и прецеденты системы заказов магазина.

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 (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 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, 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: Заказ товаров, Управление статусом заказа, Получение информации о заказе.

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 инициирует выполнение прецедента Заказ товаров (рисунок 10).

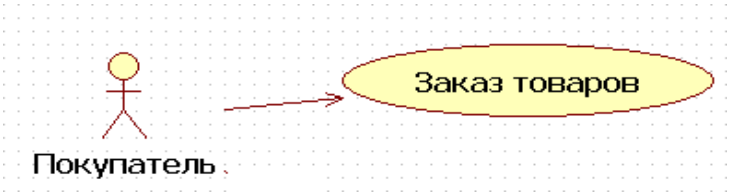

Рисунок 10–Отношение ассоциации между актером и прецедентом

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 невозможен заказ товаров без оплаты. На диаграмме прецедентов это можно отразить так, как это показано на рисунке 11.

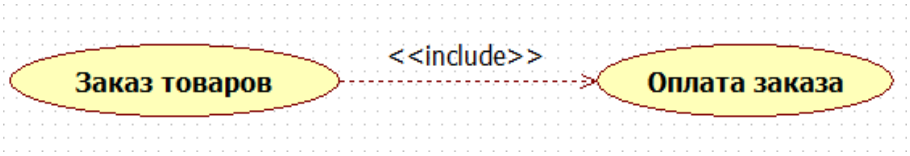

Рисунок 11–Отношение включения между прецедентами

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

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 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 12).

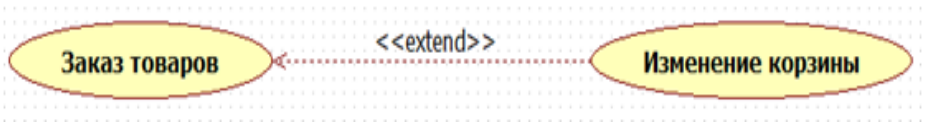

Рисунок 12–Отношение расширения между прецедентами

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 категоризировать элементы модели.

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 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 Сотрудник отдела продаж и Кладовщик (рисунок 13).

Рисунок 13–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.

Построение диаграммы прецедентов в StarUML

В StarUML главная диаграмма прецедентов называется Main и располагается в представлении Use Case. Если в навигаторе модели щелкнуть два раза по имени этой диаграммы, то откроется ее рабочее поле. 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 (рисунок 14).

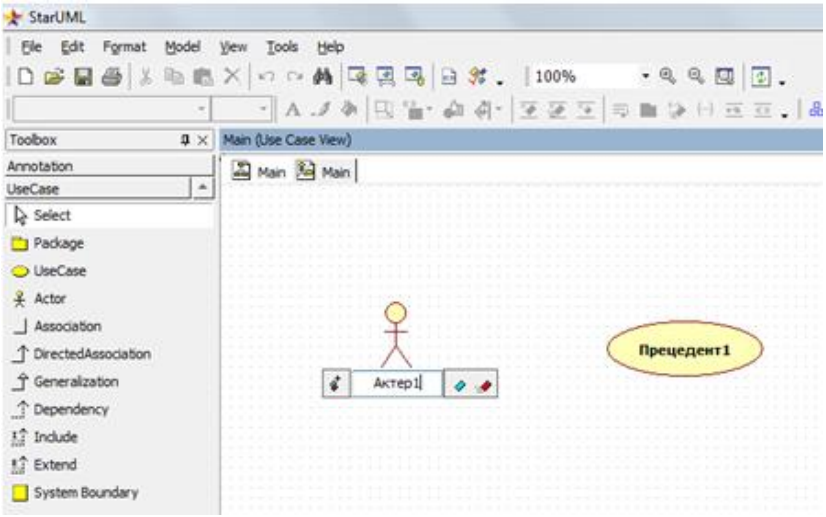

Рисунок 14–Именование элементов в StarUML

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 (рисунок 15).

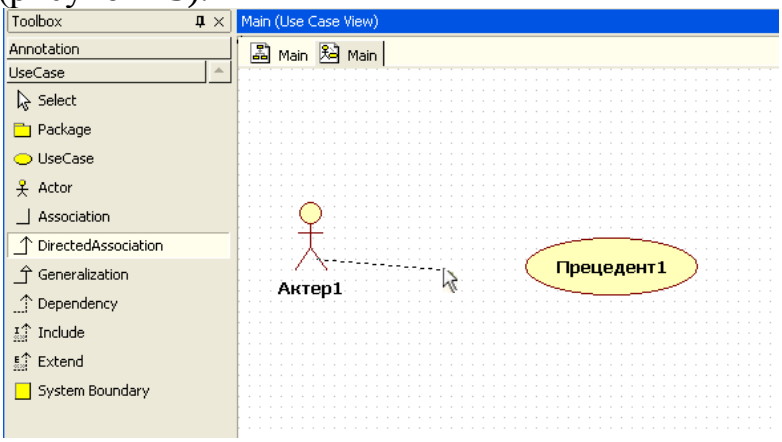

Рисунок 15–Создание отношений между элементами в StarUML

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 Edit -> Delete (рисунок 16).

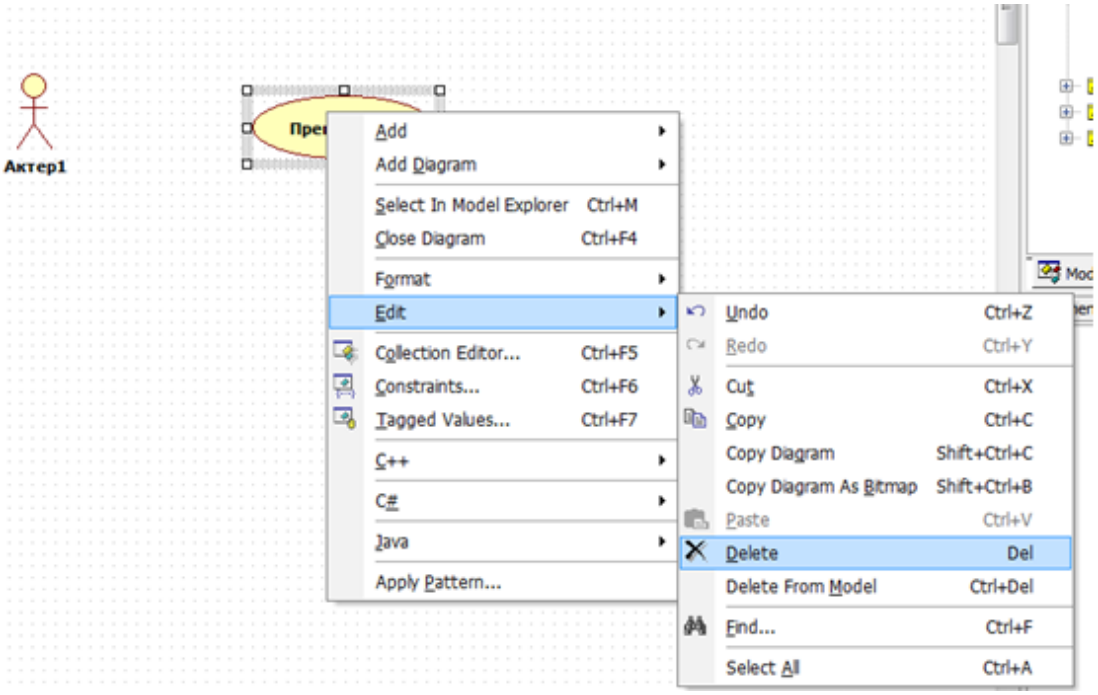

Рисунок 16–Удаление элемента диаграммы в StarUML

Обратите внимание, что элемент был удален с диаграммы, но не из модели (рисунок 17)! Его можно найти в навигаторе модели, не смотря на то, что на диаграмме он больше не отображается (элемент Прецедент1):

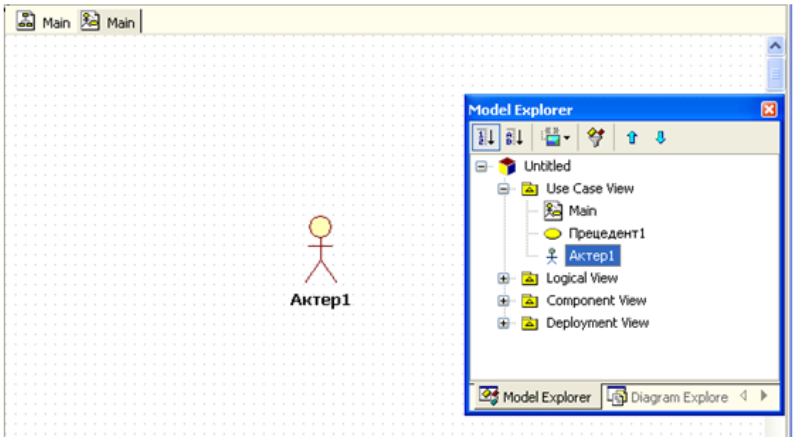

Рисунок 17–Элемент Прецедент1 удален с поля диаграммы, но отображается в навигаторе модели

Если мы передумали и решили вернуть элемент на диаграмму, то это можно сделать, перетащив его с 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. Мы не будем в дальнейшем заострять на этом внимание читателя. Заметим, что все описанные операции доступны также из главного меню StarUML.

Пример. Для системы заказов магазина мы определили актеров Покупатель, Сотрудник, Система Склад и прецеденты Заказ товаров, Управление статусом заказа, Получение информации о заказе. Построим основную диаграмму прецедентов (рисунок 18).

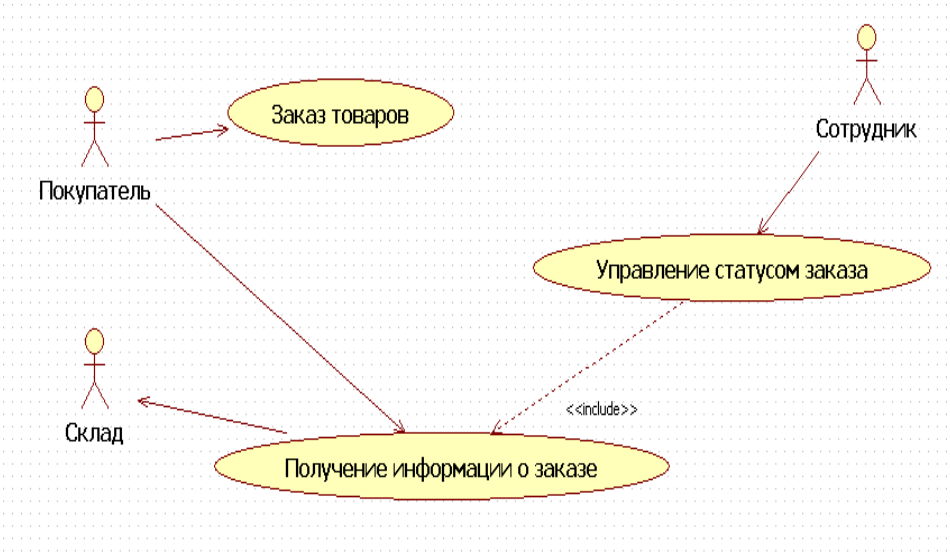

Рисунок 18–Основная диаграмма вариантов использования системы заказов магазина

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.

Документирование элементов модели в StarUML

В StarUML добавление описания к элементам модели делается следующим образом. Выделите элемент модели, щелкнув по нему мышкой, и откройте редактор Documentation. Если он не отображается справа на одной из вкладок инспектора модели, то откройте его, используя меню View → Documentation. Напротив пункта Documentation должна стоять галочка. Введите описание элемента в окно документирования (рисунок 19).

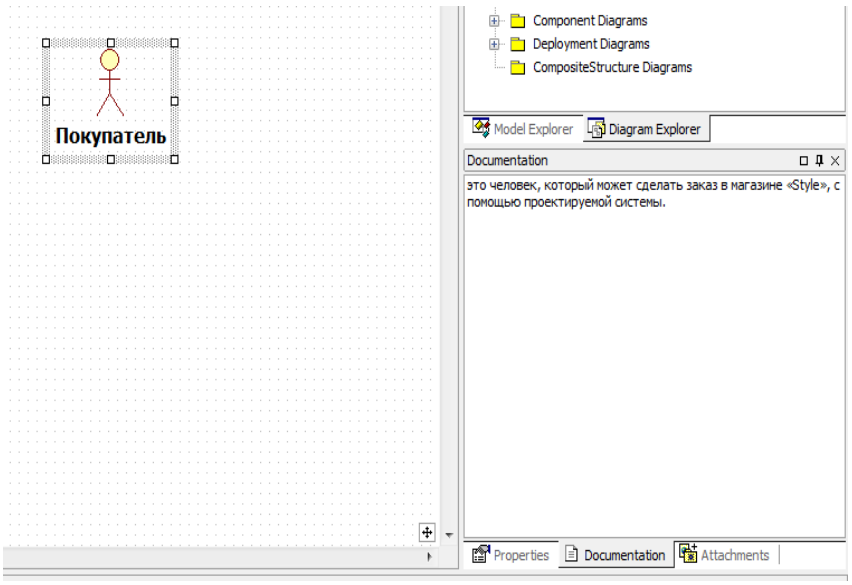

Рисунок 19–Документирование элемента модели в StarUML

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

Пример. Для актеров и прецедентов системы заказов магазина сделаем краткое описание.

Покупатель – это человек, который может сделать заказ в магазине с помощью проектируемой системы.

Сотрудник – это все сотрудники магазина, 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 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. 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.

Для того чтобы создать еще одну диаграмму (любого типа), например, детализирующую прецедент, щелкните правой кнопкой мыши по папке Use Case Model и в появившемся контекстном меню выберите Add Diagram, затем выберите из списка диаграмму, которую вы хотите добавить. Например, можно создать дополнительную диаграмму прецедентов, выбрав пункт Use Case Diagram (рисунок 20).

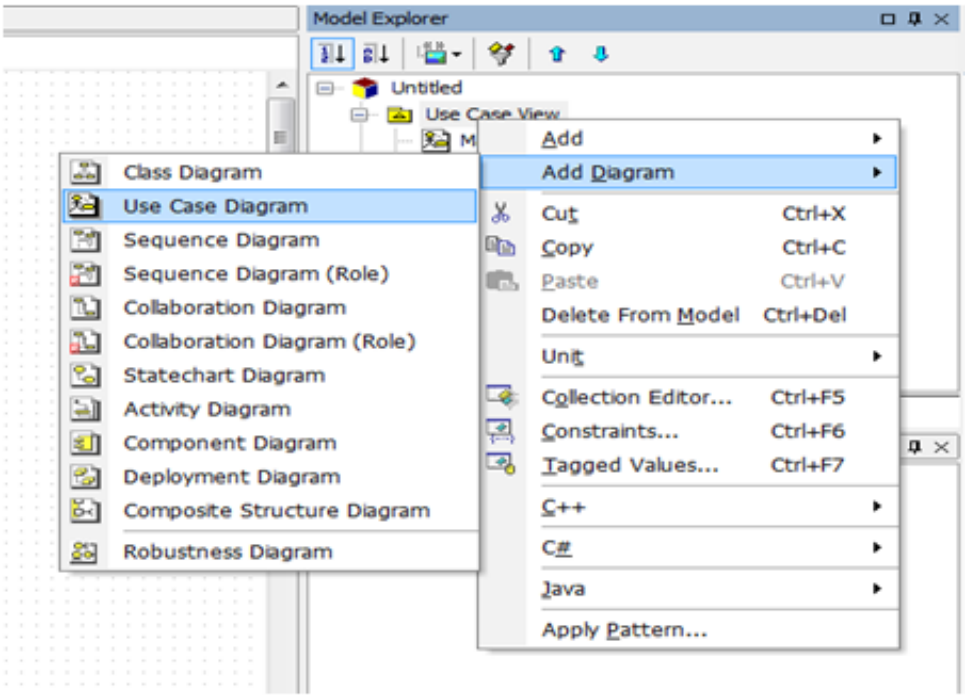

Рисунок 20–Создание дополнительной диаграммы прецедентов

Пример. Наиболее значимым для данной системы и ее актеров прецедентом является прецедент Заказ товаров. Для него мы построим дополнительную диаграмму прецедентов, поясняющую этот вариант использования (рисунок 21).

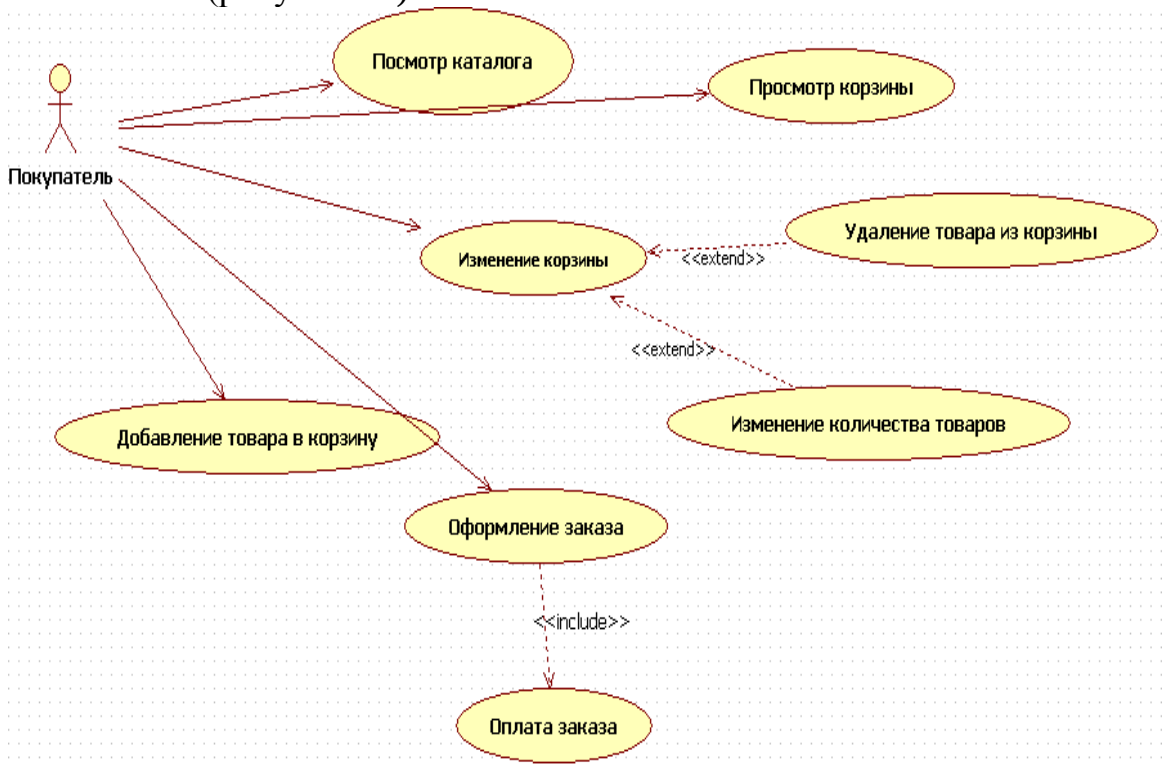

Рисунок 21–Диаграмма вариантов использования, поясняющая основной прецедент

КЕЙС-ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНЫХ ПРОЕКТОВ

1. Проектирование системы интернет-заказов товаров магазина электроники.
2. Проектирование системы электронной записи клиентов нотариальной конторы.
3. Проектирование системы интернет-заказов у поставщиков автозапчастей.
4. Проектирование электронной системы учета оценок студентов
5. Проектирование электронной системы распределения нагрузки преподавателей.
6. Проектирование информационной системы страховой компании.
7. Проектирование системы учета кадров на предприятии.
8. Проектирование электронной системы заказа книг в библиотеке.
9. Проектирование системы бронирования для проката автомобилей.
10. Проектирование системы учета рекламы в эфире телеканала.
11. Проектирование системы электронного расписания работы телеканала.
12. Проектирование электронной системы сдачи в аренду торговых площадей.
13. Проектирование системы учета технического обслуживания станков.
14. Проектирование информационной системы турфирмы.
15. Проектирование системы покупки и бронирования билетов на поезд.
16. Проектирование информационной системы компании грузоперевозок.
17. Проектирование системы учета телефонных разговоров сотрудников.
18. Проектирование интернет-системы подачи заявок на оформление кредита.
19. Проектирование интернет-кабинета клиента банка.
20. Проектирование информационной системы агентства недвижимости.
21. Проектирование системы регистрации и контроля сообщений участников интернет- форума.
22. Проектирование системы доставки товаров из магазина.
23. Проектирование интернет-системы заказа и доставки пиццы.
24. Проектирование информационной системы детского сада.
25. Проектирование системы курсов дистанционного обучения.
26. Проектирование системы футбольных ставок.
27. Проектирование системы бронирования столиков и заказа блюд меню ресторана по интернету.
28. Проектирование системы обслуживания клиентов частной почтовой службы.
29. Проектирование системы маркетинга предприятия.
30. Проектирование информационной системы компании прямых продаж косметики.

СПИСОК ЛИТЕРАТУРЫ

1. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, И. Якобсон; перевод Н. Мухина. — 3-е изд. — Москва: Академия АйТи, ДМК Пресс, 2022. — 494 с. — ISBN 978-5-89818-247-2. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/126312.html
2. Бизнес-процессы: языки моделирования, методы, инструменты / Ф. Шёнталер, Г. Фоссен, А. Обервайс, Т. Карле; перевод А. Абдулнагимов [и др.]. — Москва: Альпина Паблишер, 2019. — 264 с. — ISBN 978-5-9614-2022-7. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/124474.html
3. Носова, Л. С. Case-технологии и язык UML: учебно-методическое пособие / Л. С. Носова. — 2-е изд. — Челябинск, Саратов: Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 67 с. — ISBN 978-5-4486-0670-0. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/81479.html
4. Практическое применение нотации визуального моделирования UML в бизнес процессах: учебное пособие / Д. В. Шлаев, С. Г. Шматко, Ю. В. Орел, А. А. Сорокин. — Ставрополь : АГРУС, 2022. — 72 с. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/129601.html
5. Пальмов, С. В. Методы и средства моделирования программного обеспечения: методические указания к лабораторным работам / С. В. Пальмов. — Самара: Поволжский государственный университет телекоммуникаций и информатики, 2016. — 33 с. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/71854.html
6. Розенберг, Д. Применение объектного моделирования с использованием UML и анализ прецедентов на примере книжного Internet-магазина / Д. Розенберг, К. Скотт; перевод А. А. Слинкина. — 2-е изд. — Москва: ДМК Пресс, 2022. — 159 с. — ISBN 978-5-89818-245-8. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/126310.html
7. Самуйлов, С. В. Объектно-ориентированное моделирование на основе UML: учебное пособие / С. В. Самуйлов. — Саратов: Вузовское образование, 2016. — 37 с. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/47277.html

ГОЧИЯЕВА Мадина Джаштуевна

УНИФИЦИРОВАННЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ UML

Учебно-методическое пособие для обучающихся 3 курса
по направлению подготовки 09.03.04 Программная инженерия

Корректор Чагова О.Х.
Редактор Чагова О.Х.

Сдано в набор 09.09.2024 г.
Формат 60х84/16
Бумага офсетная.
Печать офсетная.
Усл. печ. л.1,39
Заказ 4973
Тираж 100 экз.

Оригинал-макет подготовлен
в Библиотечно-издательском центре СКГА
369000, г. Черкесск, ул. Ставропольская, 36