Министерство образования Московской области

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ МОСКОВСКОЙ ОБЛАСТИ

ЛЮБЕРЕЦКИЙ ТЕХНИКУМ ИМЕНИ ГЕРОЯ СОВЕТСКОГО СОЮЗА,

ЛЁТЧИКА-КОСМОНАВТА Ю. А. ГАГАРИНА

ОТЧЕТ

по производственной практике

ПМ 02 «Осуществление интеграции программных модулей»

Костина Макара Руслановича

Специальность: 09.02.07 «Информационные системы и программирование»

Период практики с «15» декабря 2021 г. по «21» декабря 2021 г.

Группа № 185, курс 4

Руководители работы:

Жирнова Юлия Витальевна

Лобзин Роман Валентинович

Дзержинский 2021

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc92896391)

[Описание предметной области 4](#_Toc92896392)

[Изучение программных средств 6](#_Toc92896393)

[Проектирование диаграммы деятельности UML (activity diagram) 8](#_Toc92896394)

[Проектирование диаграммы последовательности UML (sequence diagram) 12](#_Toc92896395)

[Проектирование диаграммы состояний UML (statechart diagram) 16](#_Toc92896396)

[Выводы 21](#_Toc92896397)

[Список литературы 22](#_Toc92896398)

Введение

Производственная практика для студентов — это важная составляющая учебного процесса, позволяющая сориентироваться на рынке труда и найти себя в будущей профессии.

Основная цель производственной практики — закрепление и углубление теоретических знаний, полученных в процессе обучения, реализация адаптационных возможностей студента к новым условиям работы, а также выработка навыков и овладение профессиональными знаниями.

Для достижения данной цели требуется решить следующие задачи:

* Описание предметной области.
* Изучение программных средств.
* Проектирование диаграммы деятельности UML (activity diagram).
* Проектирование диаграммы последовательности UML (sequence diagram).
* Проектирование диаграммы состояний UML (statechart diagram).

В ходе производственной практики планируется освоение следующего вида деятельности: «Осуществление интеграции программных модулей» (ПК):

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

Описание предметной области

Акционерное Общество «МЕТТЭМ-БЕТАР» является официальным представителем ООО ПКФ «БЕТАР» — одного из крупнейших производителей приборов учета воды и газа, электрической энергии, тепла, компонентов систем водоснабжения.

Осуществляет оптовую и розничную продажу юридическим и физическим лицам следующих категорий товаров: счетчики воды бытовые, счетчики воды промышленные, счетчики газа, регуляторы давления, преобразователи воды магнитные, счетчики электричества, теплосчетчики, индикаторы магнитного поля, фитинги для труб. Кроме того, организация занимается оптовыми продажами широкого ассортимента фасованной таблетированной соли.

Процесс деятельности организация начинается с поступления заказа от клиента.

Аббревиатура UML расшифровывается как Unified Modeling Language, дословно переводится как «унифицированный язык моделирования». UML — это язык моделирования, который позволяет создавать структуры программных систем.

UML состоит из графических обозначений, диаграмм, которые помогают создать дизайн программных проектов. С помощью UML-диаграмм проектные группы коммуницируют между собой, составляют и проверяют архитектурный дизайн ПО.

Работа с UML-диаграммами — важная часть проекта, так как на этом этапе продумывается его структура. Проектирование помогает в дальнейшем не запутаться в коде, снизить количество ошибок и упростить работу.

UML имеет единый синтаксис, поэтому является международным языком. Диаграммы будут понятны любому человеку, знакомому с ним. Также стоит отметить, что UML используется для разработки широкого спектра программ от информационных систем масштаба предприятия до распределенных веб-приложений.

Основные цели дизайна UML:

* Проектирование. Благодаря UML разработчики получили возможность создавать модели различных процессов, анализировать, проектировать и внедрять программные системы, рисовать схемы приложений, по которым впоследствии пишется код.
* Обеспечение роста рынка объектно-ориентированных инструментов и развития отрасли.
* Создание UML таким образом, чтобы можно было работать с любым языком программирования и любому пользователю.
* Коммуникация внутри команды и при общении с заказчиком.

Изучение программных средств

Компьютерное построение диаграмм — это процесс создания масштабируемых диаграмм на ПК, которые могут использоваться в различных приложениях, таких как макетирование, организационные диаграммы, временные графики, планы этажей и даже прототипы пользовательских программных интерфейсов. Диаграммы состоят из фигур, объектов и трафаретов, которые при правильном объединении могут помочь спроецировать много полезной информации.

Современное компьютерное построение диаграмм основано на использовании векторных форм вместо растровых. Растровые фигуры или растровые изображения плохо масштабируются, и их разрешение и качество ухудшаются с каждым редактированием. Векторные же фигуры являются математическим представлением фигур и, следовательно, хорошо масштабируются. На векторные фигуры редактирование не влияет, и их наиболее целесообразно использовать, когда требуются точные измерения.

Программный продукт Microsoft Visio — векторный графический редактор для Windows, предназначенный для быстрой и качественной разработки графических документов любой сложности. С помощью него можно создавать схемы, блок-схемы, планы, технические проекты, модели, диаграммы, различные чертежи и многое другое.

MS Visio является наиболее популярным программным обеспечением для создания диаграмм, которое позволяет создавать диаграммы, визуализировать данные и моделировать процессы в привычном интерфейсе. Visio поставляется с массивом шаблонов и встроенных форм для создания практически любых диаграмм. Visio также позволяет пользователям определять свои собственные формы и импортировать их в чертеж.

Visio предназначен больше для корпоративного класса, поскольку домашним пользователям редко приходится использовать расширенные функции построения диаграмм. Однако многие пользователи покупают стандартную версию Visio для лучшей визуализации простых диаграмм, таких как генеалогическое дерево или планы этажей.

MS Visio тесно интегрируется с другими продуктами Microsoft Office, такими как Word, Excel и Access. Данные могут быть напрямую импортированы из этих программ и преобразованы в значимые диаграммы, которые изменяются в реальном времени в соответствии с данными.

Более новые версии Visio, от Visio 2013 и выше, поддерживают формат .vsdx, который обеспечивает улучшенное сжатие и многие другие функции по сравнению со старым форматом .vsd, поддерживаемым Visio 2010 и более ранними версиями.

Visio, входящий в состав пакета Office 365, постоянно обновляется.

Имеет следующие полезные функции:

* Визуализатор данных.

Data Visualizer помогает автоматически создавать диаграммы процессов из данных Excel. Готовые или настраиваемые шаблоны Excel, содержащие информацию, которая помещается в диаграммы, можно импортировать непосредственно в Visio с помощью визуализатора данных и преобразовывать в диаграмму Visio. Любые изменения, внесенные в исходный файл Excel, также отражаются на диаграмме Visio.

* Визуализация структур базы данных из исходных данных.

Последняя версия Visio позволяет осуществлять реверс-инжиниринг баз данных для создания их визуального представления. Visio может подключаться к различным базам данных, таким как MySQL, SQL Server, Oracle и т. д., и может обновлять диаграммы синхронно с изменениями в базе данных.

* Шаблоны для отраслевых диаграмм.

Последние обновления Visio для Office 365 поставляются с множеством новых стандартизированных шаблонов, предназначенных для таких отраслей, как бизнес-процессы, разработка программного обеспечения, информационные технологии и образование. Эти шаблоны предоставляются как Microsoft, так и другими третьими лицами, расширяющими возможности Visio.

Проектирование диаграммы деятельности UML (activity diagram)

Диаграмма деятельности (activity diagram) — диаграмма для демонстрации рабочего процесса некоторой деятельности, основанной на поэтапных действиях и действиях с поддержкой выбора и параллелизма.

Диаграмма деятельности — технология, позволяющая описывать логику процедур, бизнес-процессы и потоки работ.

Основным отличием диаграмм деятельности от блок-схем является активная поддержка параллельных процессов, что объясняет применение диаграммы деятельности для моделирования потоков работ.

Организации-дилеру поступают заявки на поставку продукции. На практике была спроектирована диаграмма выполнения заявок за день.

Основные шаги построения диаграммы деятельности:

* Добавление основных элементов.
* Работа с операциями.

Создадим диаграмму деятельности, используя шаблон UML Activity из раздела Software and Database программы Visio.

* Добавление основных элементов.

1. Добавим дорожки.

Дорожка (swimlane) — часть области диаграммы деятельности для отображения деятельностей, за которые отвечает конкретный объект (например, пользователь или организационное подразделение).

В нашем случае диаграмма содержит две дорожки: «Система» и «Дилер» (Рис. 1).

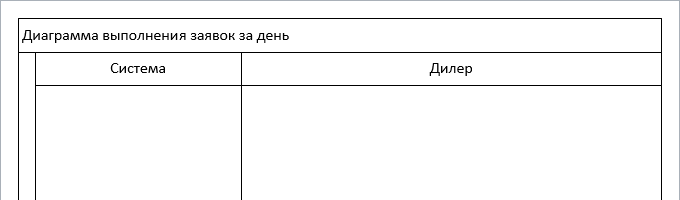


Рисунок 1. Добавление дорожек на диаграмму деятельности

2. Добавим узлы и операции.

Диаграмма деятельности должна начинаться с начального узла (activity initial node), из которого исходят поток или потоки при вызове деятельности извне, и завершаться конечным узлом (activity final node), который останавливает все потоки данной деятельности.

Операции (operations) являются ключевыми элементами диаграммы деятельности.

При проектировании диаграммы выполнения заявок из начального узла дорожки «Дилер» происходит вызов операции «Вывести список заявок на поставку за день» (Рис. 2).

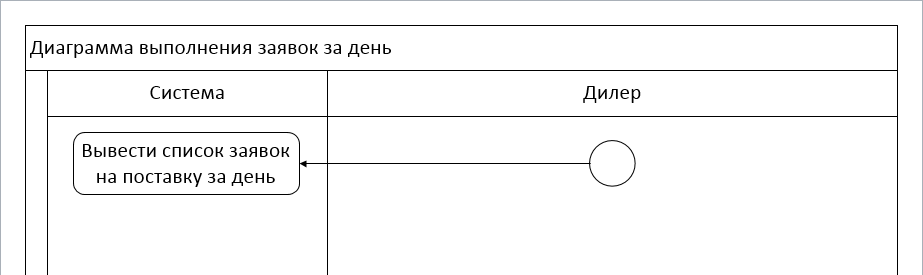


Рисунок 2. Начальный узел (справа) и операция (слева)

* Работа с операциями.

1. Отобразим условное поведение типа «Решение».

Решение (decision) имеет один входящий и несколько защищенных выходящих потоков. Защитой является условное выражение, помещенное в скобки. Так как при достижении решения выбирается только один из выходных потоков, защиты должны быть взаимоисключающими.

В нашей диаграмме условиями для принятия решения по операции «Вывести список заявок на поставку за день» являются варианты: «Нет заявок» и «Есть заявки». При условии «Нет заявок» решение завершается конечным узлом, который останавливает поток данной деятельности. При условии «Есть заявки» выбирается решение «Выбрать заявку» (Рис. 3).

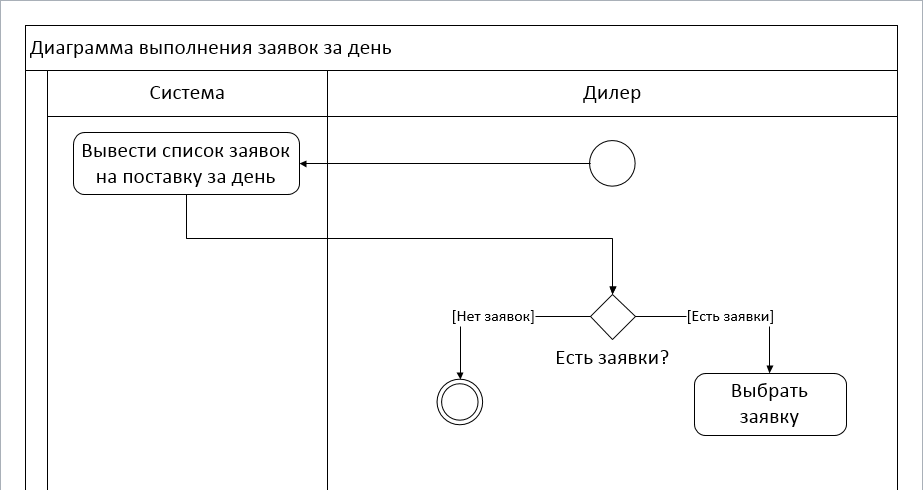


Рисунок 3. Условное поведение типа «Решение»

2. Отобразим условное поведение типа «Слияние».

Слияние (merge) означает завершение условного поведения, которое было начато решением (Рис. 4).

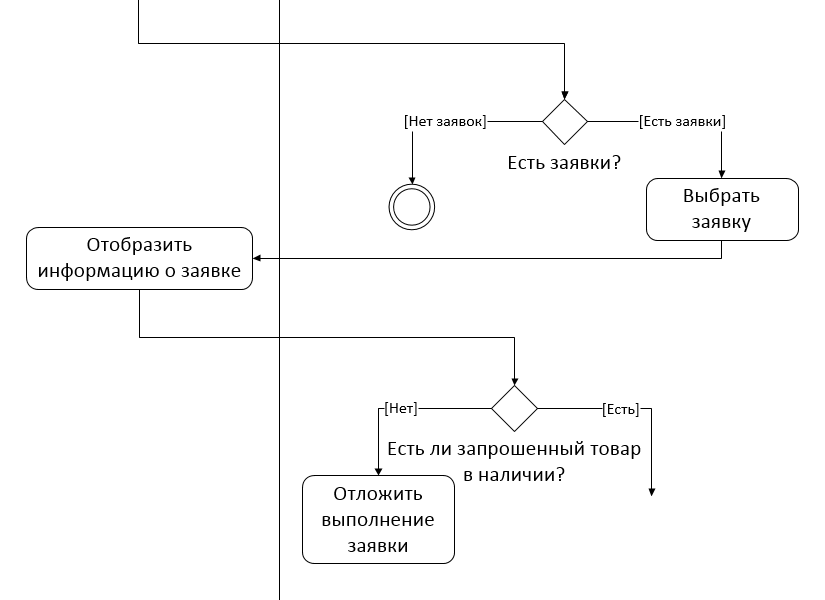


Рисунок 4. Условное поведение типа «Слияние»

3. Реализуем параллельные потоки.

Для реализации параллельных потоков используются точки разделения и точки слияния. Из точки разделения (fork node) выходят два и более потока, выполняющихся параллельно. Точка слияния (join node) синхронизирует потоки, т. к. каждый из них ждет достижения этой точки остальными потоками, после чего продолжается последовательное исполнение (Рис. 5).

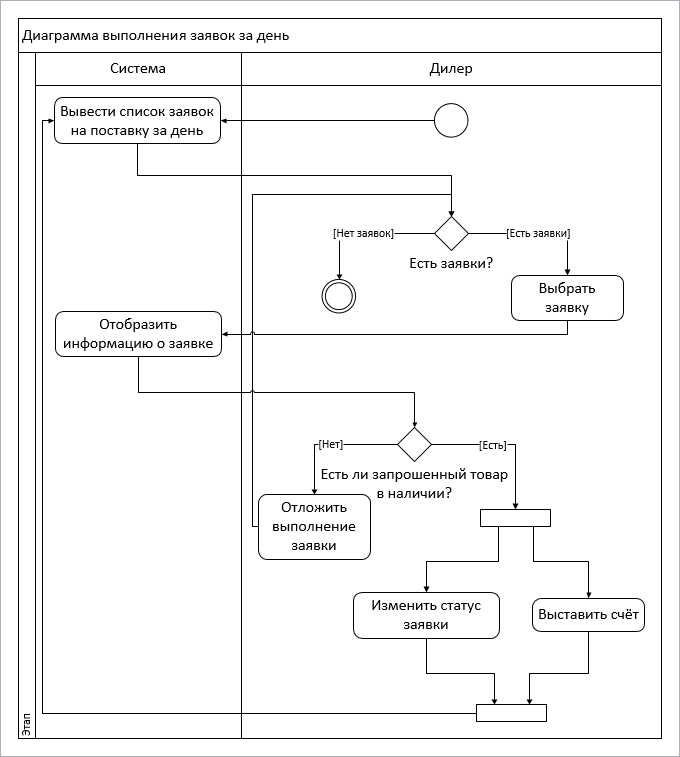


Рисунок 5. Реализация параллельных потоков

Проектирование диаграммы последовательности UML (sequence diagram)

Диаграмма последовательности (sequence diagram) — диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие актеров ИС в рамках какого-либо определенного прецедента.

Диаграмма последовательности позволяет наглядно увидеть взаимодействие отдельных объектов.

Актер (actor) — роль объекта вне системы, который прямо взаимодействует с ее частью — конкретным элементом.

Прецедент (use case, вариант использования) — описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды.

Диаграмма последовательности позволяет изобразить поведение нескольких объектов в рамках одного прецедента. Она удобна для представления взаимодействия объектов, но не для точного определения их поведения. Диаграмма показывает экземпляры объектов и сообщения, которыми обмениваются экземпляры в рамках одного прецедента.

В ходе производственной практики было выполнено построение диаграммы последовательности ответа дилера на заявку заказчика.

Основные шаги построения диаграммы последовательности:

* Добавление основных элементов.
* Работа с сообщениями.

Создадим диаграмму последовательности, используя шаблон UML Sequence из раздела Software and Database программы Visio.

* Добавление основных элементов.

1. Добавим объекты.

Объекты обычно подписываются в формате «объект:класс» и изображаются как в виде обычных прямоугольников, так и с использованием дополнительных обозначений.

В представленной ниже диаграмме объектами являются: запрос, обозначенный прямоугольником («Заявка»), а также Дилер и Заказчик, обозначенные элементом «Актер» (Рис. 6).

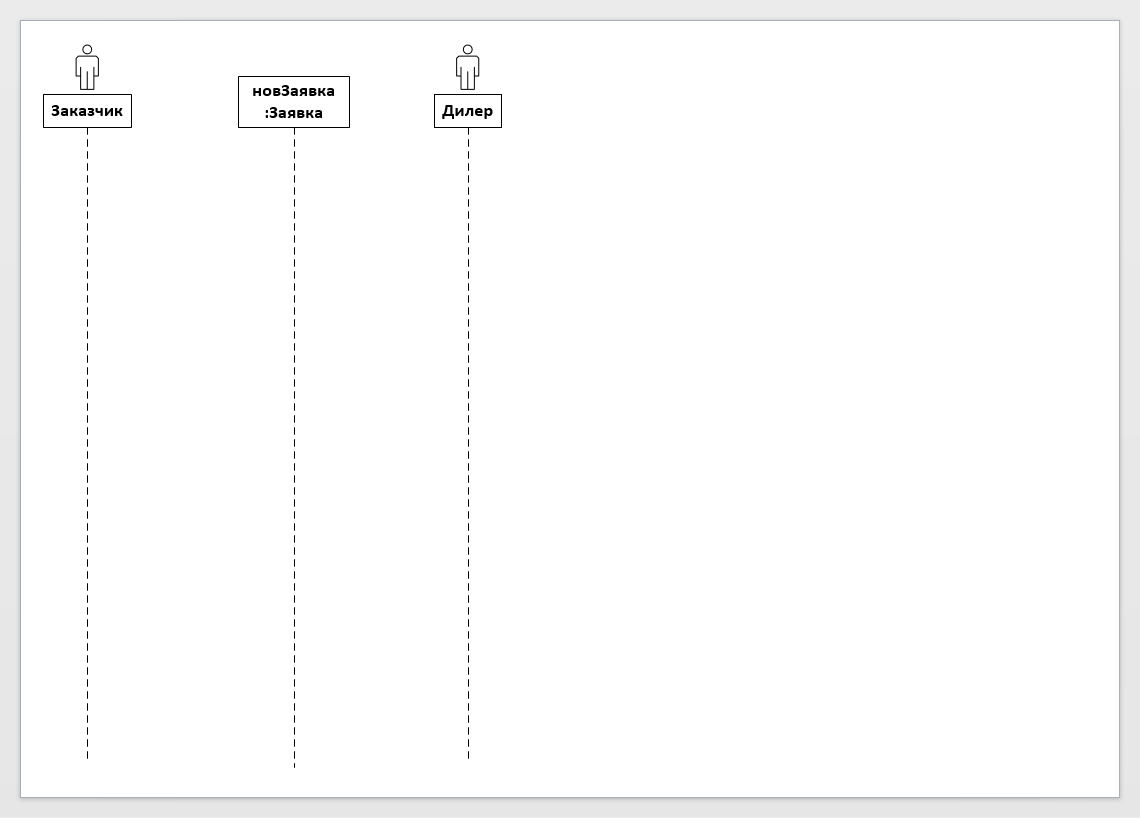


Рисунок 6. Объекты диаграммы последовательности

2. Добавим полосы активности на линии жизни (Рис. 7).

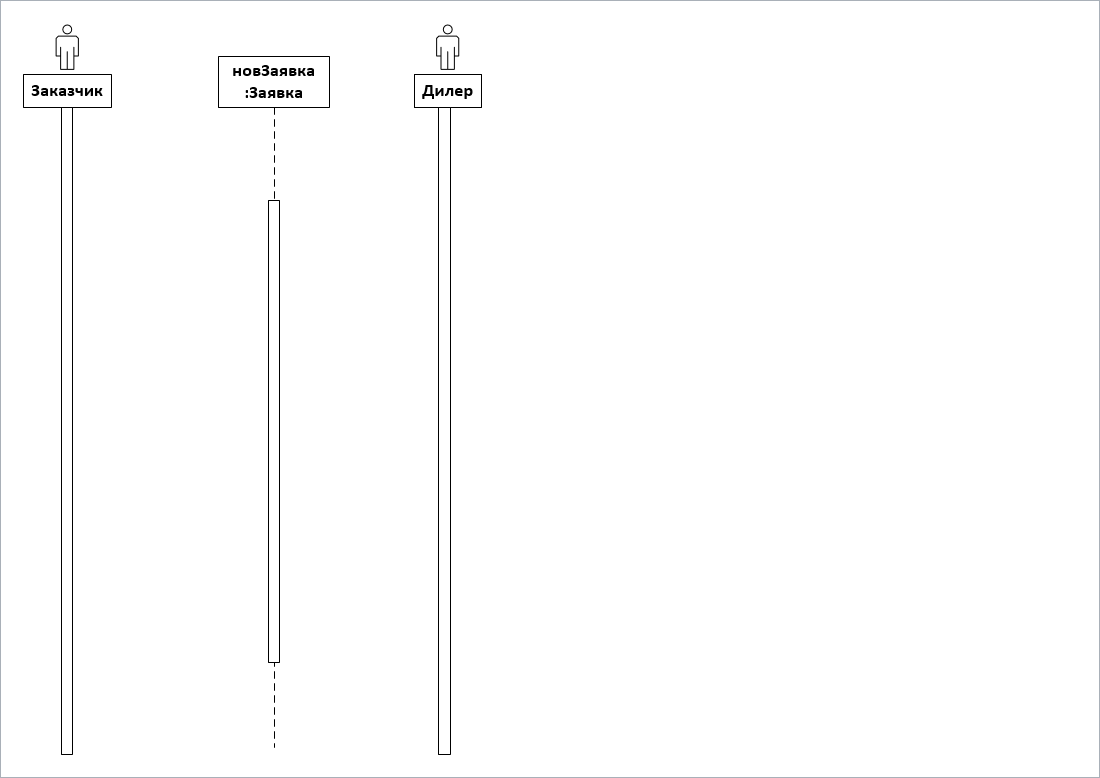


Рисунок 7. Полосы активности на линиях жизни объектов

Линия жизни (lifeline) идет вертикально вниз от каждого объекта и упорядочивает сообщения на странице таким образом, чтобы они читались сверху вниз. Каждая линия жизни имеет полосу активности (вертикальный прямоугольник), показывающую интервал активности участника при взаимодействии.

* Работа с сообщениями.

1. Отобразим основные взаимодействия.

Сообщения показывают взаимодействие между объектами в виде горизонтальной стрелки, концы которой лежат на линиях жизни. Направление стрелки указывает на адресата, а положение на линии жизни упорядочивает сообщения по времени. При создании нового объекта и применении конструктора можно не указывать имя сообщения, указав ключевое слово «нов» («new»).

На рисунке 8 изображены сообщения, показывающие взаимодействие между объектами.

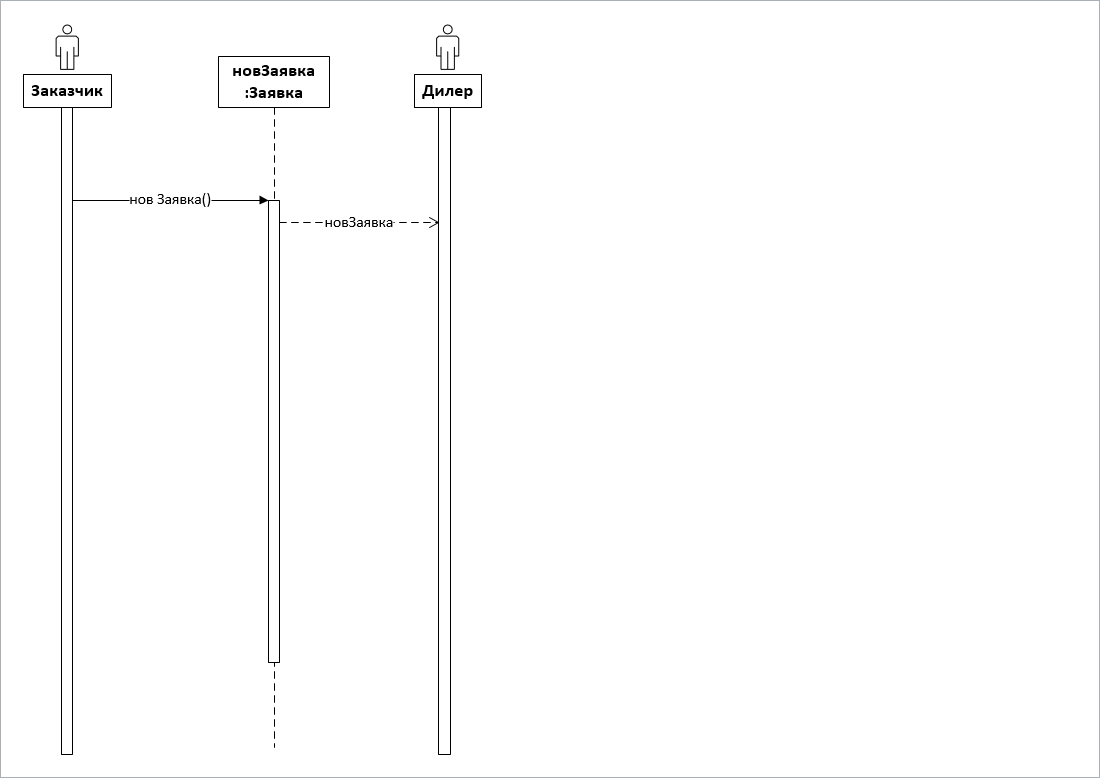


Рисунок 8. Сообщения в виде горизонтальных стрелок

2. Отобразим условия.

Условия, как и циклы, изображаются с помощью фреймов взаимодействий (interaction frames), позволяющих разметить диаграмму взаимодействия. Каждый фрейм представляет собой разделенную на несколько фрагментов область диаграммы, причем каждый фрейм имеет оператор, а каждый фрагмент может иметь защиту.

В нашей диаграмме для условной логики используется оператор «альт» («alt») и будет выполнено условие, защита которого имеет истинное значение. То есть, либо принятие заявки «ApplyRequest ()», либо откладывание заявки с указанием причины «DeclineRequest (String reason)» (Рис. 9).

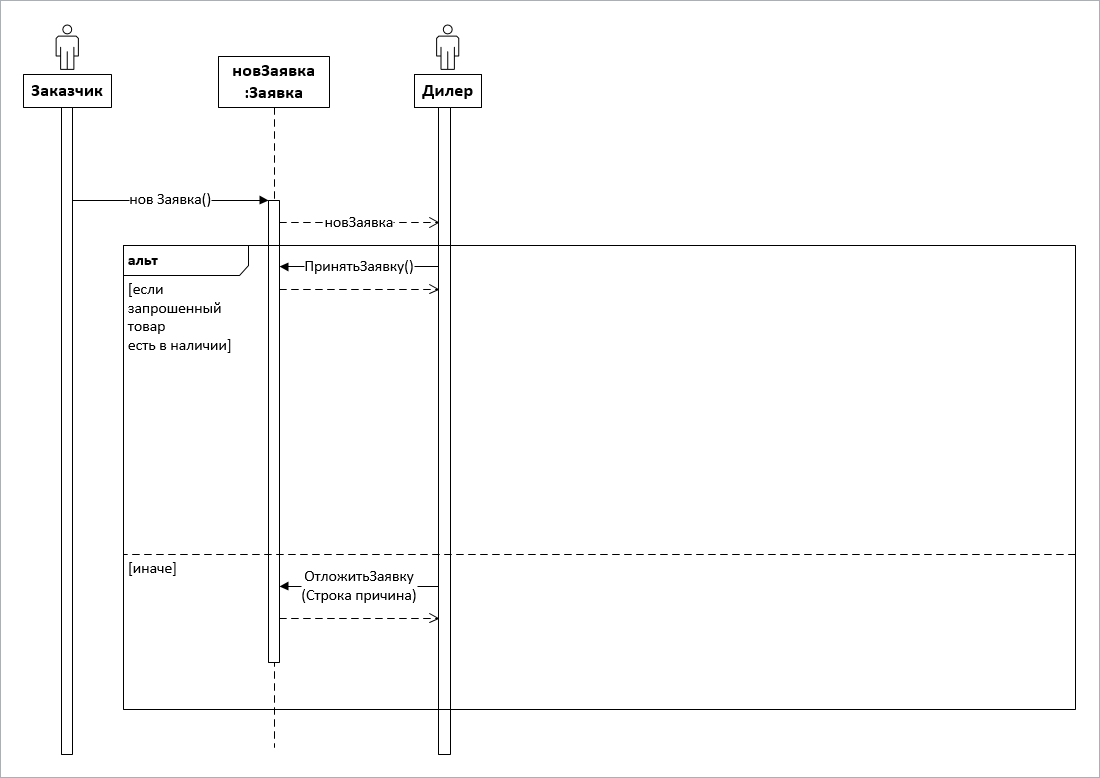


Рисунок 9. Отображение условий на диаграмме последовательности

3. Отобразим циклы.

Для отображения цикла применяется оператор «цикл» («loop») с единственным фрагментом, причем тело итерации помещается в защиту (Рис. 10).

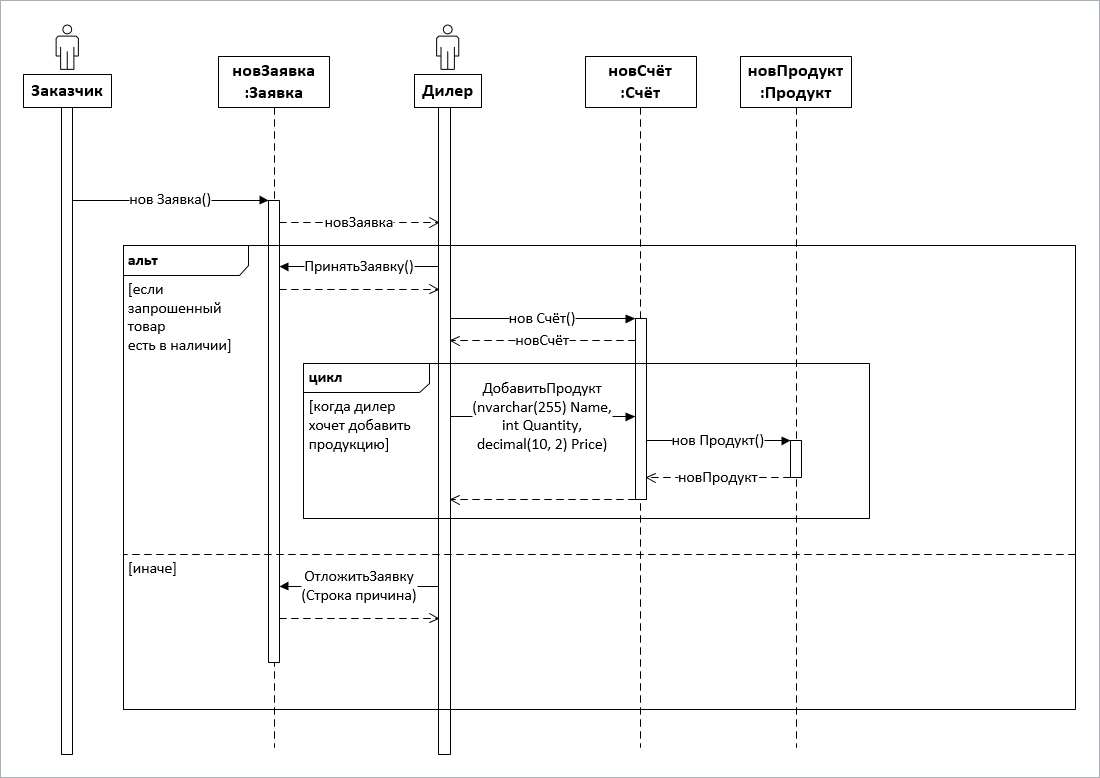


Рисунок 10. Отображение циклов на диаграмме последовательности

Проектирование диаграммы состояний UML (statechart diagram)

Диаграмма состояний (statechart diagram) — методология объектно-ориентированного проектирования, предназначенная для представления жизненного цикла объектов в реальном или абстрактном мире.

Диаграмма состояний позволяет наглядно описать поведение объекта.

Событие (event) — сообщение, которое возникает в различных точках исполняемого кода при выполнении определенных условий.

Диаграмма состояний позволяет описывать поведение системы.

В объектно-ориентированном подходе разрабатывается диаграмма состояний единственного класса, демонстрирующая поведение одного объекта в течение его жизни.

Состояние на диаграмме является более абстрактным понятием, чем состояние объекта (последнее есть комбинация всех данных из полей объекта).

Диаграмма позволяет проектировать различные способы реакции на события.

Рассмотрим построение диаграммы состояний заявки заказчика.

Основные шаги построения диаграммы состояний:

* Добавление состояний.
* Указание переходов.
* Добавление внутренних активностей.
* Указание подсостояний и суперсостояний.

Создадим диаграмму состояний, используя шаблон UML State Machine из раздела Software and Database программы Visio.

* Добавление состояний.

1. Добавим начальное псевдосостояние.

Начальное псевдосостояние (initial pseudostate) не является состоянием, но впоследствии будет иметь стрелку, указывающую на начальное состояние (Рис. 11).

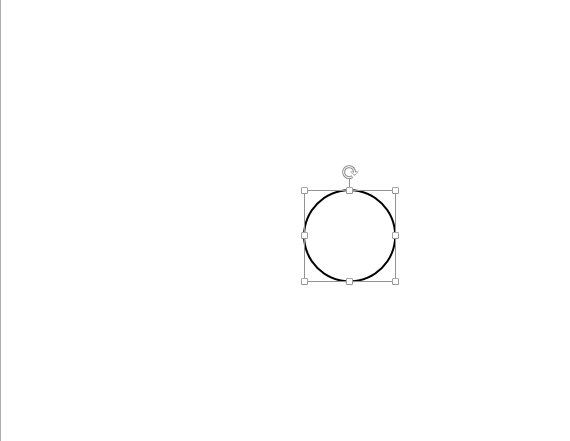


Рисунок 11. Начальное псевдосостояние

2. Далее добавим возможные состояния.

На диаграмме отображаются состояния, в которых объект может находиться продолжительное время. Состояние может быть прервано вследствие наступления определенного события.

На рисунке 12 изображены возможные состояния заявки: «Создано», «Принято», «Отложено», «Отменено».

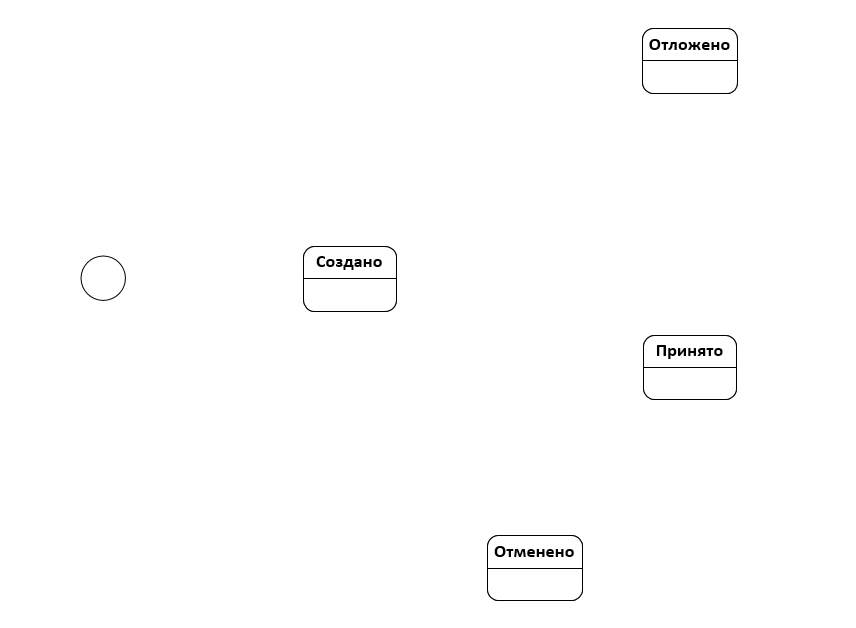


Рисунок 12. Добавление возможных состояний

* Указание переходов.

1. Отобразим основные переходы.

Переход (transition) означает перемещение из одного состояния в другое и изображается в виде линий, связывающих состояния (Рис. 13).

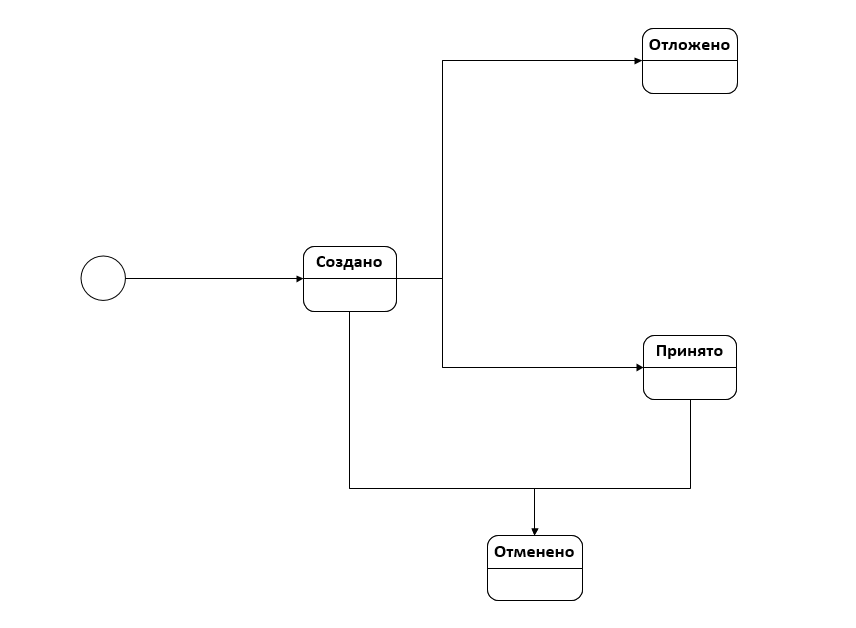


Рисунок 13. Отображение основных переходов на диаграмме состояний

2. Добавим метки.

Каждый переход имеет метку, состоящую из следующих необязательных частей:

Триггер-идентификатор — единственное событие, способное вызвать изменение состояния. Пропуск этой части означает, что переход происходит немедленно.

Защита — логическое условие, выполнение которого обязательно для осуществления перехода. Пропуск защиты означает, что в ответ на инициирующее событие переход всегда осуществляется.

Активность — поведение системы во время перехода. Пропуск активности означает, что в процессе перехода ничего не происходит.

На диаграмме рисунка 14 представлены методы «ПринятьЗаявку()» и «ОтложитьЗаявку()», которые служат примерами событий, вызывающих переходы между состояниями. Примером защиты является выражение «[Причина != null]», означающее, что при отклонении заявки всегда следует указывать причину (например, недостаточное количество на складе).

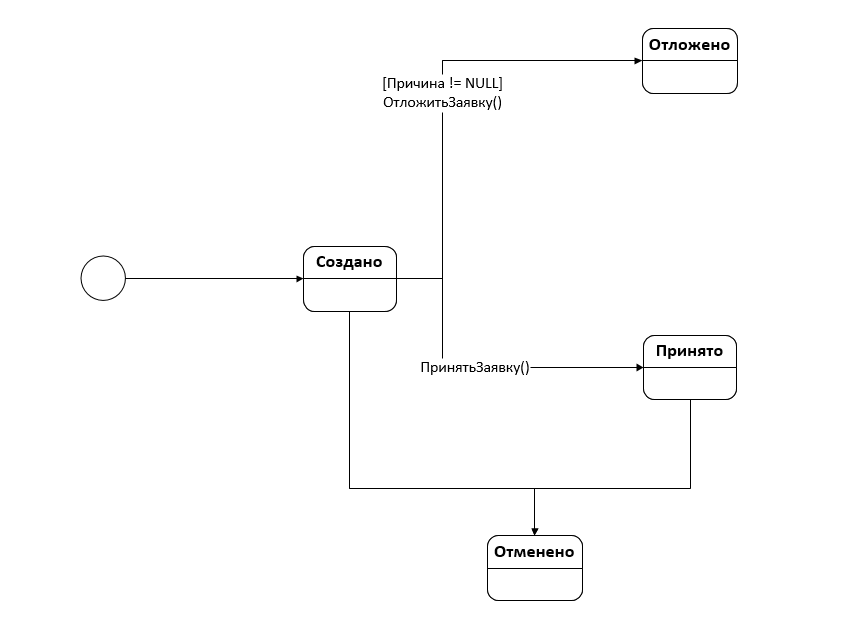


Рисунок 14. Добавление меток

* Добавление внутренних активностей.

Внутренние активности (internal activities) используются для описания действий объекта, совершаемых без перехода. Список основных действий включает следующие значения:

* Входное действие (entry) — действие, которое выполняется в момент входа в данное состояние.
* Выходное действие (exit) — действие, которое выполняется в момент выхода из данного состояния.
* Выполняющая деятельность (do) — действие, которое выполняется в течение всего времени нахождения объекта в данном состоянии.

С помощью внутренней активности «do/ПроверитьОтвет» заказчик может проверять, пришел ли ответ от дилера. Это показано на рисунке 15.

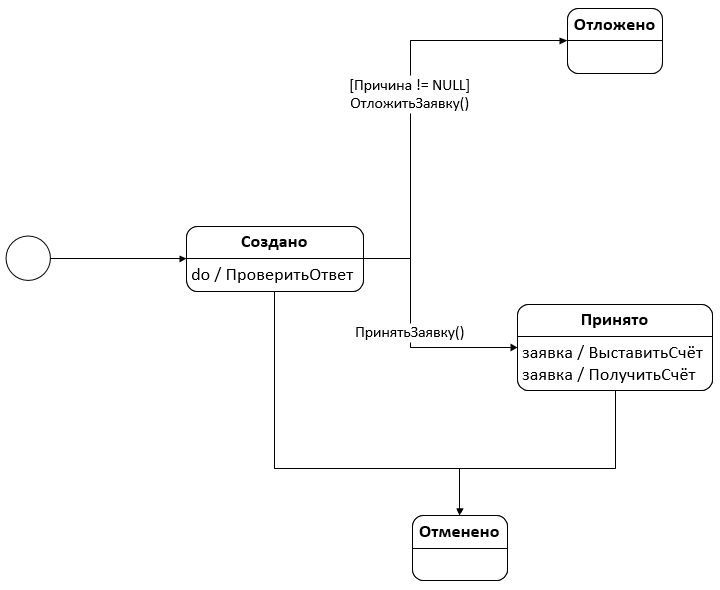


Рисунок 15. Добавление внутренних активностей на диаграмму состояний

* Указание подсостояний и суперсостояний.

В случае наличия у нескольких состояний общих переходов и внутренних активностей, само состояние можно превратить в подсостояние (substates), а их общее поведение перенести в суперсостояние (superstate).

Заявка может быть отменена как из состояния «Создано», так и из состояния «Принято». Эти состояния могут быть объединены в суперсостояние «Может быть отменено» (Рис. 16).

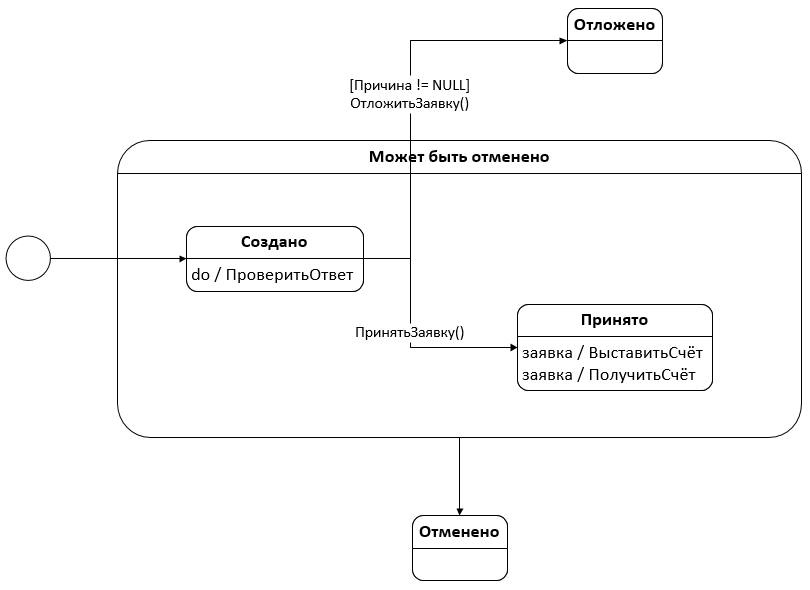


Рисунок 16. Указание подсостояний и суперсостояний

Выводы

По окончании производственной практики были решены следующие задачи:

* Описана предметная область.
* Изучены программные средства.
* Спроектирована диаграмма деятельности UML (activity diagram).
* Спроектирована диаграмма последовательности UML (sequence diagram).
* Спроектирована диаграмма состояний UML (statechart diagram).

В ходе этой практики были освоены следующие профессиональные компетенции:

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

Список литературы

1. GitHub репозиторий — URL https://github.com/KostinMakar
2. Обучение работе с Visio — URL https://support.microsoft.com/ru-ru/office/обучение-работе-с-visio-e058bcfa-1d90-4653-afc6-e84d54cf94a6
3. Программные решения для бизнеса — Национальная сборная Worldskills Россия — URL https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/
4. Продукция — URL http://betar.ru/catalog/
5. Счетчики воды бытовые, промышленные, Счетчики газа, тепла и электричества, Меттэм-Бетар, 8(985)764-9834, www.betar-moskva.ru — URL http://www.betar-moskva.ru/production.htm