Аве Кодер!  
  
Тебе пришла крутая идея продукта, но ты не хочешь увязнуть в коде и потерять целостную картинку из-за мелких деталей?   
  
Эта статья будет посвящена полезному, но порой ускользающему от молодой поросли знанию — диаграммам UML. Поговорим немного об истории и зачем диаграмм должно быть так много.  
  
UML — это сокращение от Unified Modeling Language, и, как мы знаем, он является стандартизированным языком моделирования, состоящим из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем, а также, к примеру, для бизнес-моделирования.  
  
UML представляет собой набор лучших инженерных практик, которые доказали свою эффективность в моделировании больших и сложных систем и является очень важной частью разработки объектно-ориентированного программного обеспечения.  
  
UML использует в основном графические обозначения, чтобы выразить дизайн программных проектов. Использование UML помогает проектным группам общаться, изучать потенциальные проекты и проверять архитектурный дизайн программного обеспечения.

**Происхождение UML**

Цель UML — предоставить стандартную нотацию, которая может использоваться всеми объектно-ориентированными методами, а также выбрать и интегрировать лучшие элементы нотаций-предшественников. UML был разработан для широкого спектра приложений. Следовательно, он предоставляет конструкции для широкого спектра систем и видов деятельности (например, распределенных систем, анализа, проектирования и развертывания систем).  
  
UML не возник на пустом месте, ему предшествовали несколько значимых событий, личностей и методологий. Например:

* Техника объектного моделирования OMT [James Rumbaugh 1991], которая была лучшей для анализа информационных систем с большим объемом данных.
* Booch [Grady Booch 1994] — отлично подходит для разработки и реализации. Грэди Буч много работал с языком Ада и был крупным игроком в разработке объектно-ориентированных методов для языка. Хотя метод Буча был сильным, нотация была воспринята менее хорошо, например, в его моделях преобладали формы облаков, что выглядело не очень аккуратно.
* OOSE (объектно-ориентированная программная инженерия [Ivar Jacobson 1992]) — модель, известная как модель прецедентов — это мощная методология для понимания поведения всей системы, область, где ООП традиционно была слабой.

В 1994 году Джим Рамбо, не путать с Джоном Рэмбо, хотя Джим тоже был крут, потому что был, на секундочку, создателем вышеупомянутой техники объектного моделирования, ошеломил мир программного обеспечения, когда он покинул General Electric и присоединился к Грэди Бучу в Rational Corp. Цель партнерства состояла в том, чтобы объединить их идеи в единый унифицированный метод (рабочее название для метода действительно было — «Единый метод»).  
  
К 1995 году создатель OOSE, Ивар Якобсон, также присоединился к Rational, и его идеи (в частности, концепция «прецедентов») были включены в новый унифицированный метод, который теперь называется Unified Modeling Language.  
  
В противовес всем известной “Банде Четырех”, Команда Румбо, Буча и Якобсона известна как «Три Амигоса».  
  
На UML также повлияли другие объектно-ориентированные нотации:

* Меллор и Шлаер [1998]
* Coad и Yourdon [1995]
* Вирфс-Брок [1990]
* Мартин и Оделл [1992]

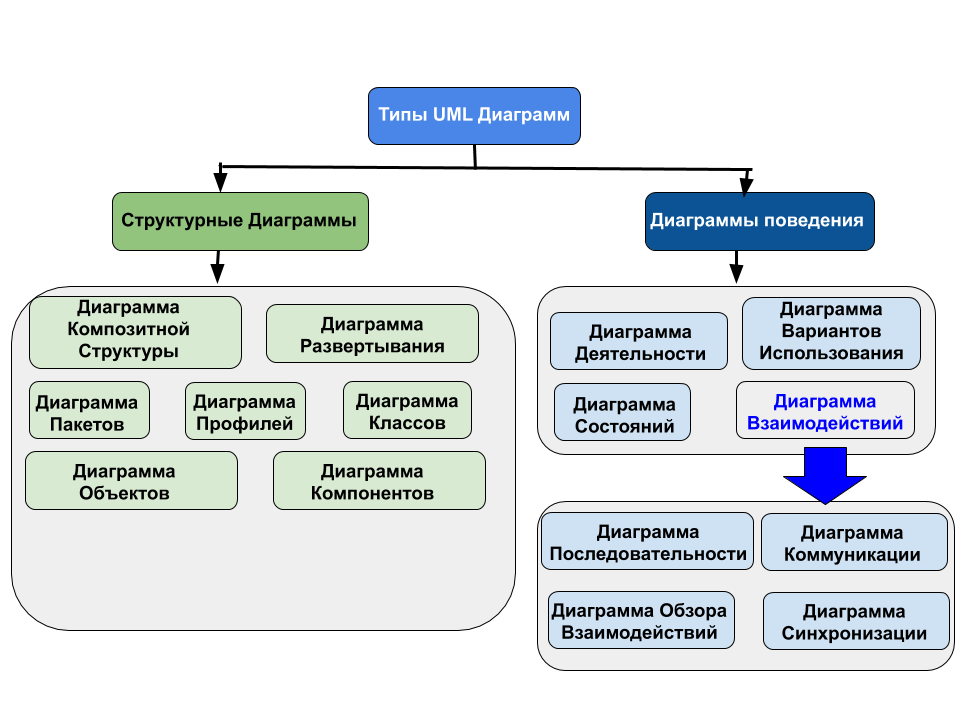
UML также включает в себя новые концепции, которых в то время не было в других основных методах, таких как механизмы расширения и язык ограничений.

**Почему UML?**

По мере того как стратегическая ценность программного обеспечения возрастала для многих компаний, отрасль искала методы для автоматизации производства программного обеспечения, а также для повышения качества и сокращения затрат и времени выхода на рынок.  
  
Эти методы включают технологию компонентов, визуальное программирование, шаблоны и структуры.  
  
Компании также ищут методы для управления сложностью систем по мере увеличения их масштаба.  
  
В частности, они признают необходимость решения повторяющихся архитектурных проблем, таких как физическое распределение, параллелизм, репликация, безопасность, балансировка нагрузки и отказоустойчивость.  
  
Кроме того, разработка под Web хоть и упрощает некоторые вещи, в целом, она усугубляет эти архитектурные проблемы.  
  
Унифицированный язык моделирования (UML) был разработан для удовлетворения этих потребностей.  
Основные цели дизайна UML:

* Предоставить пользователям готовый, выразительный язык визуального моделирования, чтобы они могли разрабатывать и обмениваться осмысленными моделями.
* Обеспечить механизмы расширяемости и специализации для расширения основных понятий.
* Быть независимым от конкретных языков программирования и процессов разработки.
* Обеспечить формальную основу для понимания языка моделирования.
* Поощрять рост рынка объектно-ориентированных инструментов.
* Поддержка высокоуровневых концепций разработки, таких как совместная работа, структуры, шаблоны и компоненты.
* Интегрировать лучшие практики.

Диаграммы UML подразделяют на два типа — это структурные диаграммы и диаграммы поведения.



Структурные диаграммы показывают статическую структуру системы и ее частей на разных уровнях абстракции и реализации, а также их взаимосвязь. Элементы в структурной диаграмме представляют значимые понятия системы и могут включать в себя абстрактные, реальные концепции и концепции реализации. Существует семь типов структурных диаграмм:

* Диаграмма составной структуры
* Диаграмма развертывания
* Диаграмма пакетов
* Диаграмма профилей
* Диаграмма классов
* Диаграмма объектов
* Диаграмма компонентов

Диаграммы поведения показывают динамическое поведение объектов в системе, которое можно описать, как серию изменений в системе с течением времени. А к диаграммам поведения относятся:

* Диаграмма деятельности
* Диаграмма прецедентов
* Диаграмма состояний
* Диаграмма последовательности
* Диаграмма коммуникаций
* Диаграмма обзора взаимодействия
* Временная диаграмма

Теперь пару слов о каждой из них

**Диаграмма классов**

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними.  
  
Три наиболее важных типа отношений в диаграммах классов (на самом деле их больше), это:  
  
Ассоциация, которая представляет отношения между экземплярами типов, к примеру, человек работает на компанию, у компании есть несколько офисов.  
  
Наследование, которое имеет непосредственное соответствие наследованию в Объектно-Ориентированном дизайне.  
  
Агрегация, которая представляет из себя форму композиции объектов в объектно-ориентированном дизайне.

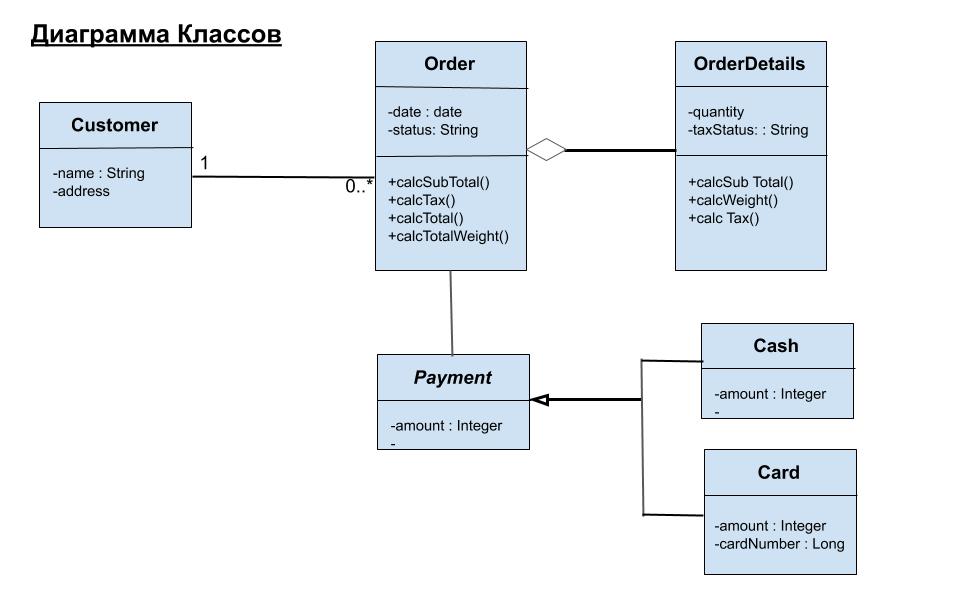


Диаграмма компонентов

На языке унифицированного моделирования диаграмма компонентов показывает, как компоненты соединяются вместе для формирования более крупных компонентов или программных систем.  
  
Она иллюстрирует архитектуры компонентов программного обеспечения и зависимости между ними.  
  
Эти программные компоненты включают в себя компоненты времени выполнения, исполняемые компоненты, а также компоненты исходного кода.

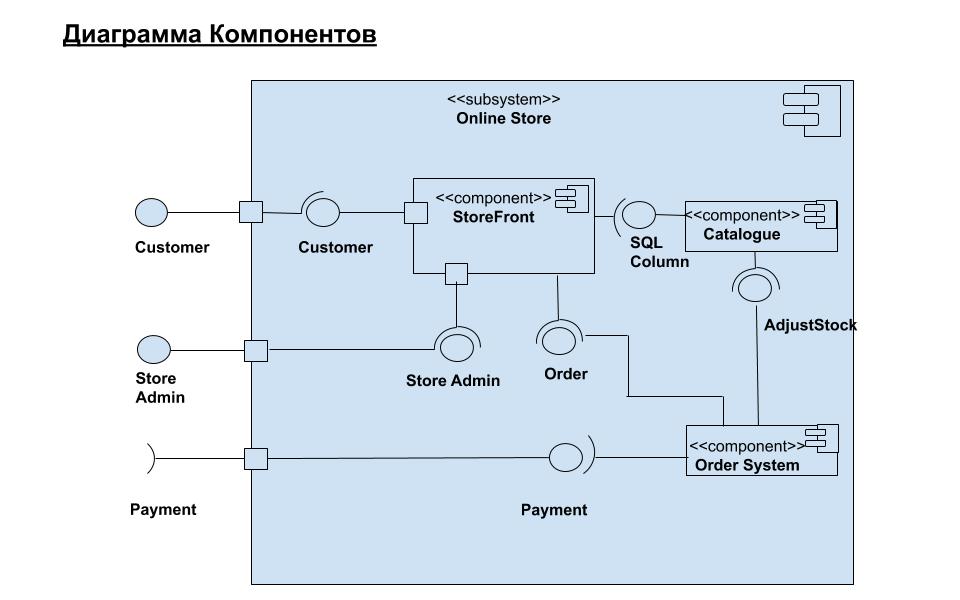


Диаграмма развертывания

Диаграмма развертывания помогает моделировать физический аспект объектно-ориентированной программной системы. Это структурная схема, которая показывает архитектуру системы, как развертывание (дистрибуции) программных артефактов.  
  
Артефакты представляют собой конкретные элементы в физическом мире, которые являются результатом процесса разработки.  
  
Диаграмма моделирует конфигурацию времени выполнения в статическом представлении и визуализирует распределение артефактов в приложении.  
  
В большинстве случаев это включает в себя моделирование конфигураций оборудования вместе с компонентами программного обеспечения, на которых они размещены.

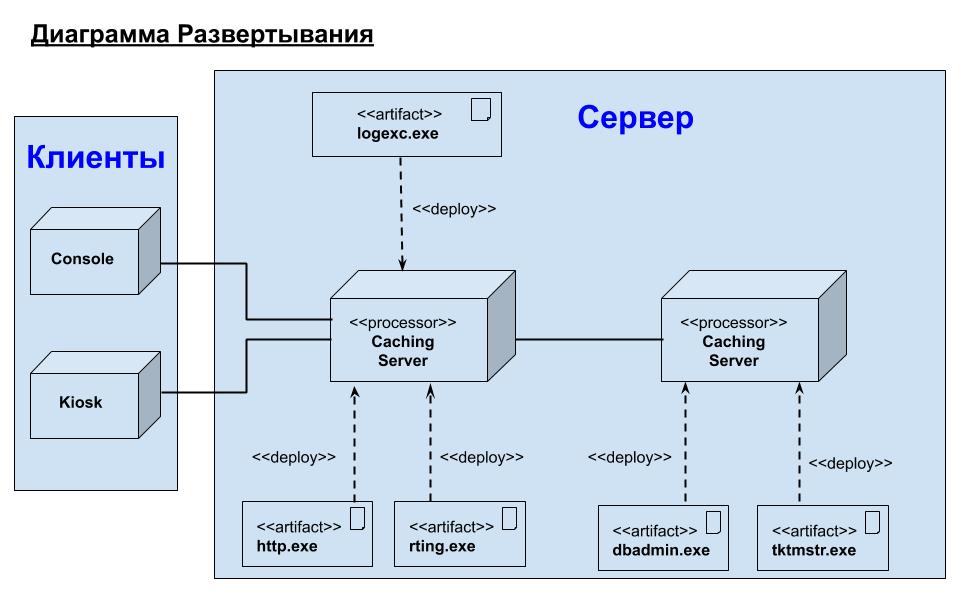


Диаграмма объектов

Статическая диаграмма объектов является экземпляром диаграммы класса; она показывает снимок подробного состояния системы в определенный момент времени. Разница в том, что диаграмма классов представляет собой абстрактную модель, состоящую из классов и их отношений.  
  
Тем не менее, диаграмма объекта представляет собой экземпляр в конкретный момент, который имеет конкретный характер. Использование диаграмм объектов довольно ограничено, а именно — чтобы показать примеры структуры данных.

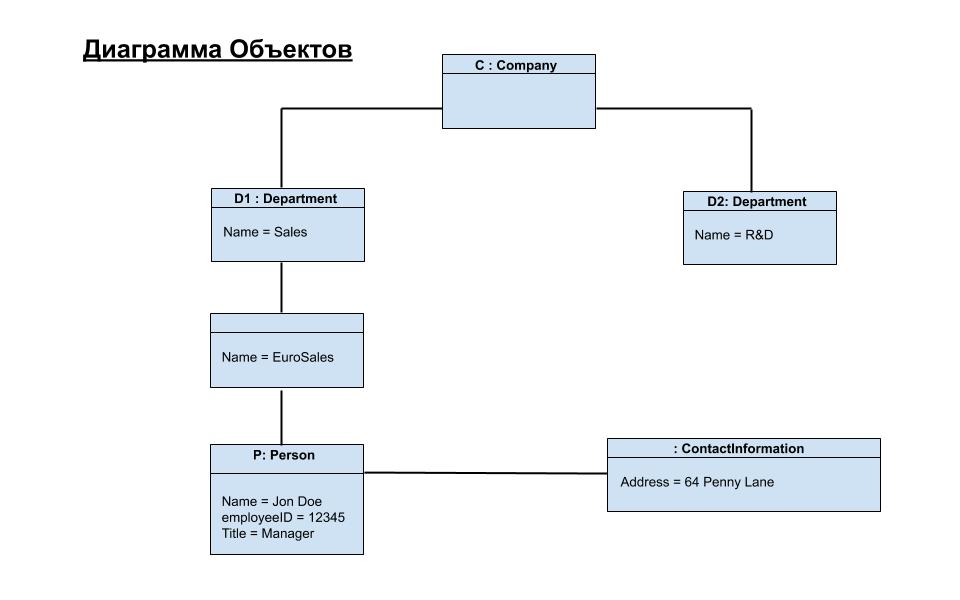


Диаграмма пакетов

Диаграмма пакетов — это структурная схема UML, которая показывает пакеты и зависимости между ними.  
  
Она позволяет отображать различные виды системы, например, легко смоделировать многоуровневое приложение.

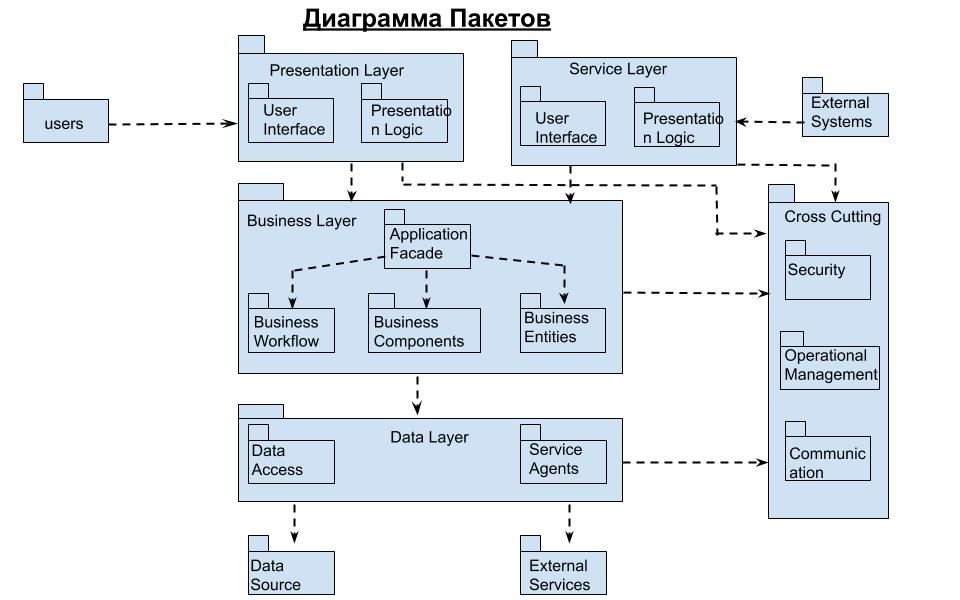


Диаграмма составной структуры

Диаграмма составной структуры аналогична диаграмме классов и является своего рода диаграммой компонентов, используемой в основном при моделировании системы на микроуровне, но она изображает отдельные части вместо целых классов. Это тип статической структурной диаграммы, которая показывает внутреннюю структуру класса и взаимодействия, которые эта структура делает возможными.  
  
Эта диаграмма может включать внутренние части, порты, через которые части взаимодействуют друг с другом или через которые экземпляры класса взаимодействуют с частями и с внешним миром, и соединители между частями или портами. Составная структура — это набор взаимосвязанных элементов, которые взаимодействуют во время выполнения для достижения какой-либо цели. Каждый элемент имеет определенную роль в сотрудничестве.

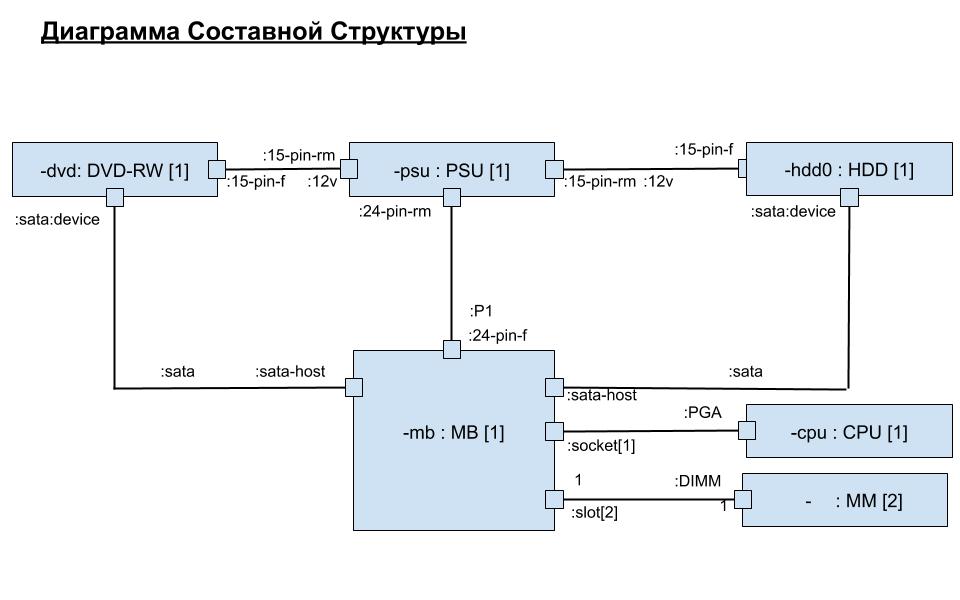


Диаграмма профилей

Диаграмма профилей позволяет нам создавать специфичные для домена и платформы стереотипы и определять отношения между ними. Мы можем создавать стереотипы, рисуя формы стереотипов и связывая их с композицией или обобщением через интерфейс, ориентированный на ресурсы. Мы также можем определять и визуализировать значения стереотипов.

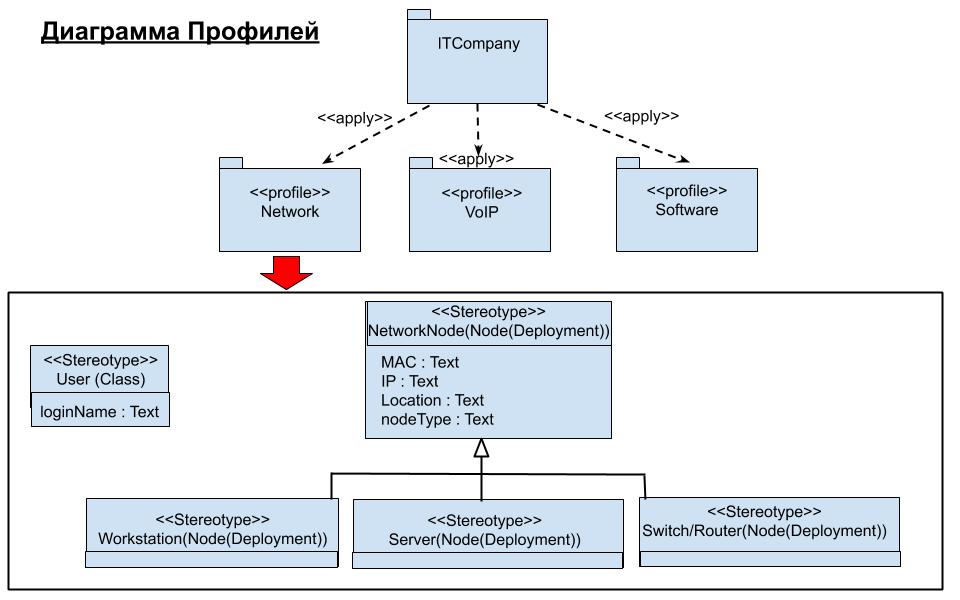


Диаграмма прецедентов

Диаграмма прецедентов описывает функциональные требования системы с точки зрения прецедентов. По сути дела, это модель предполагаемой функциональности системы (прецедентов) и ее среды (актеров).  
  
Прецеденты позволяют связать то, что нам нужно от системы с тем, как система удовлетворяет эти потребности.

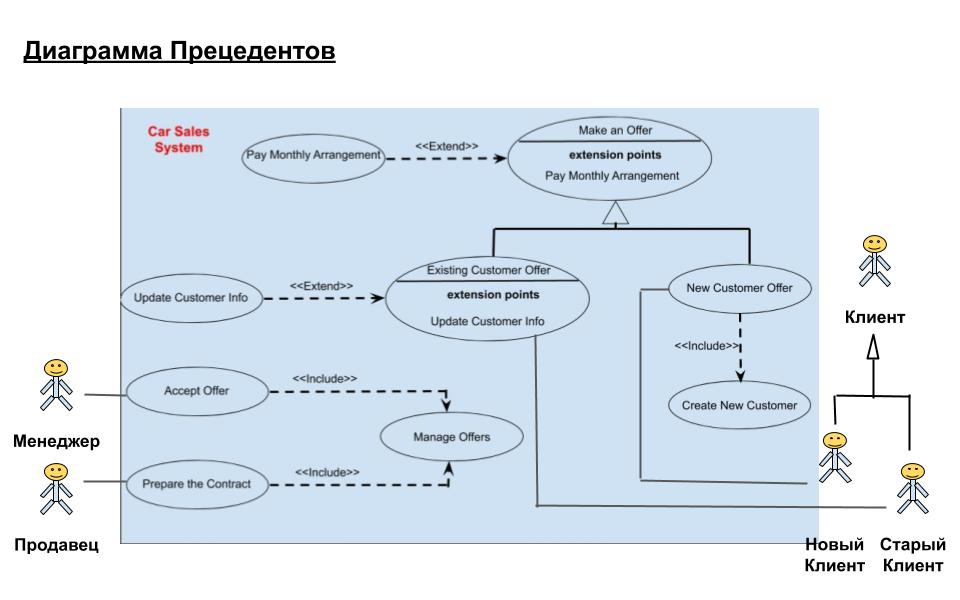


Диаграмма деятельности

Диаграммы деятельности представляют собой графическое представление рабочих процессов поэтапных действий и действий с поддержкой выбора, итерации и параллелизма.  
Они описывают поток управления целевой системой, такой как исследование сложных бизнес-правил и операций, а также описание прецедентов и бизнес-процессов.  
В UML диаграммы деятельности предназначены для моделирования как вычислительных, так и организационных процессов.

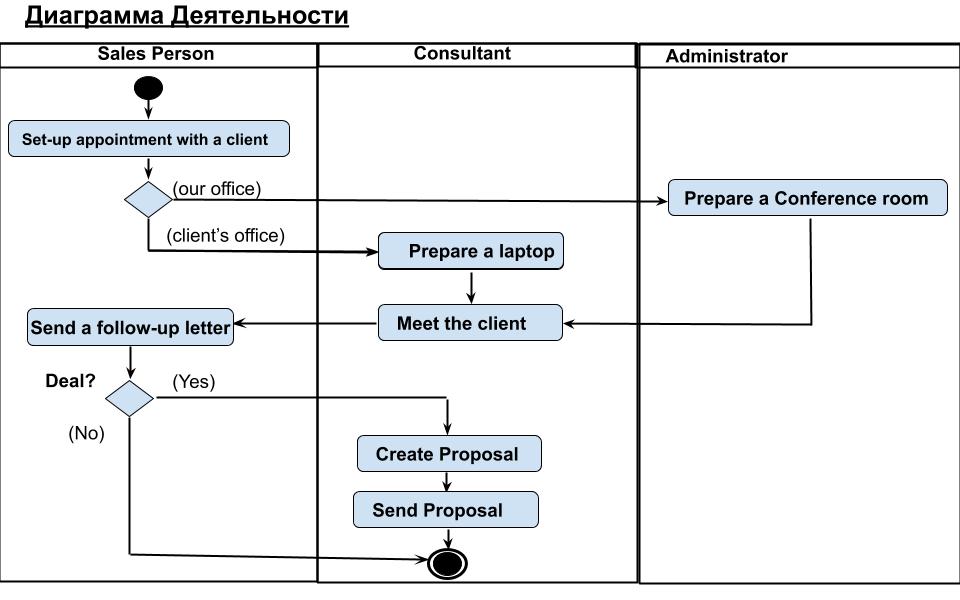


Диаграмма состояний

Диаграмма состояний — это тип диаграммы, используемый в UML для описания поведения систем, который основан на концепции диаграмм состояний Дэвида Харела. Диаграммы состояний отображают разрешенные состояния и переходы, а также события, которые влияют на эти переходы. Она помогает визуализировать весь жизненный цикл объектов и, таким образом, помогает лучше понять системы, основанные на состоянии.

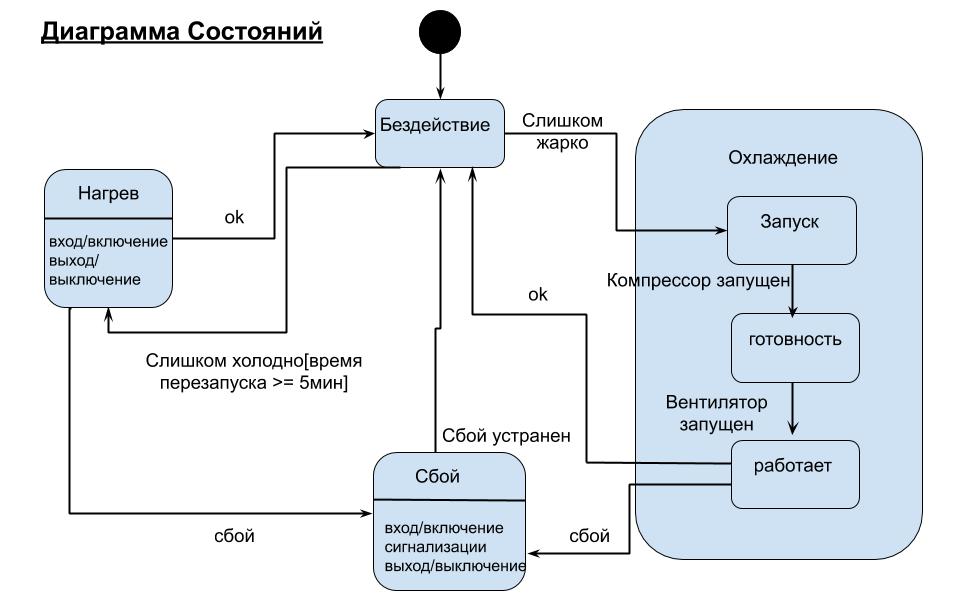


Диаграмма последовательности

Диаграмма последовательности моделирует взаимодействие объектов на основе временной последовательности. Она показывает, как одни объекты взаимодействуют с другими в конкретном прецеденте.

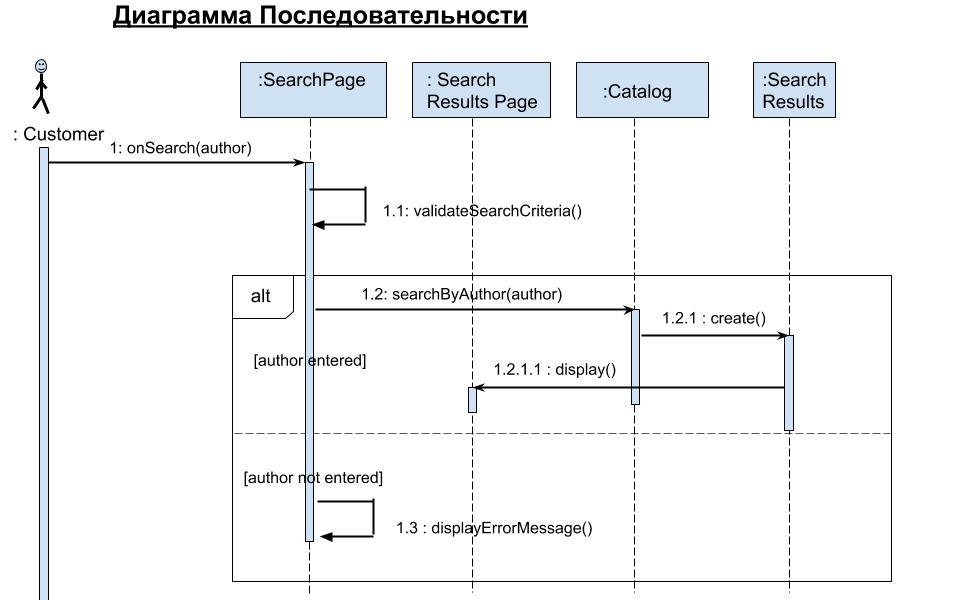


Диаграмма Коммуникации

Как и диаграмма последовательности, диаграмма коммуникации также используется для моделирования динамического поведения прецедента. Если сравнивать с Диаграммой последовательности, Диаграмма коммуникации больше сфокусирована на показе взаимодействия объектов, а не временной последовательности. На самом деле, диаграмма коммуникации и диаграмма последовательности семантически эквивалентны и могут перетекать одна в другую.

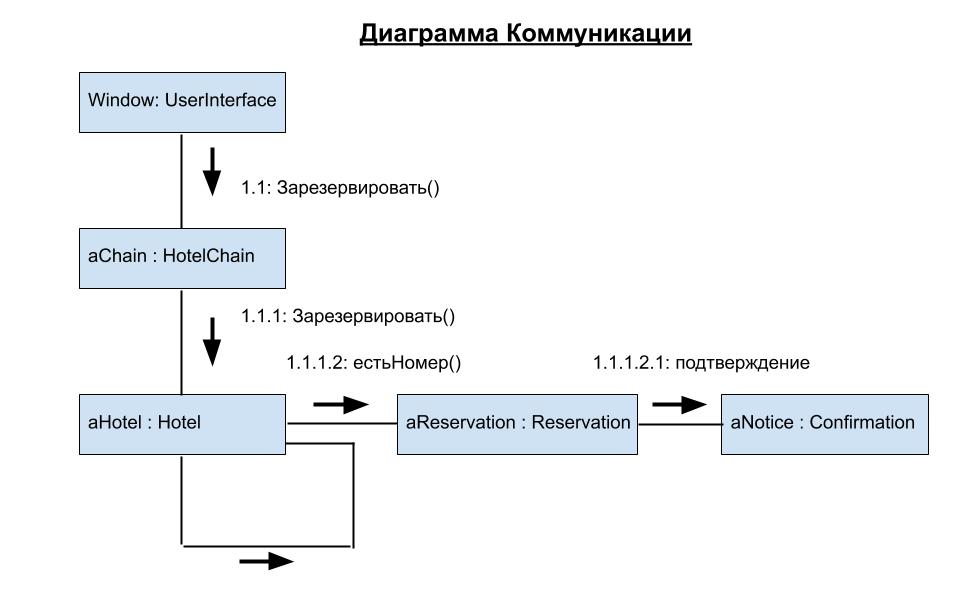
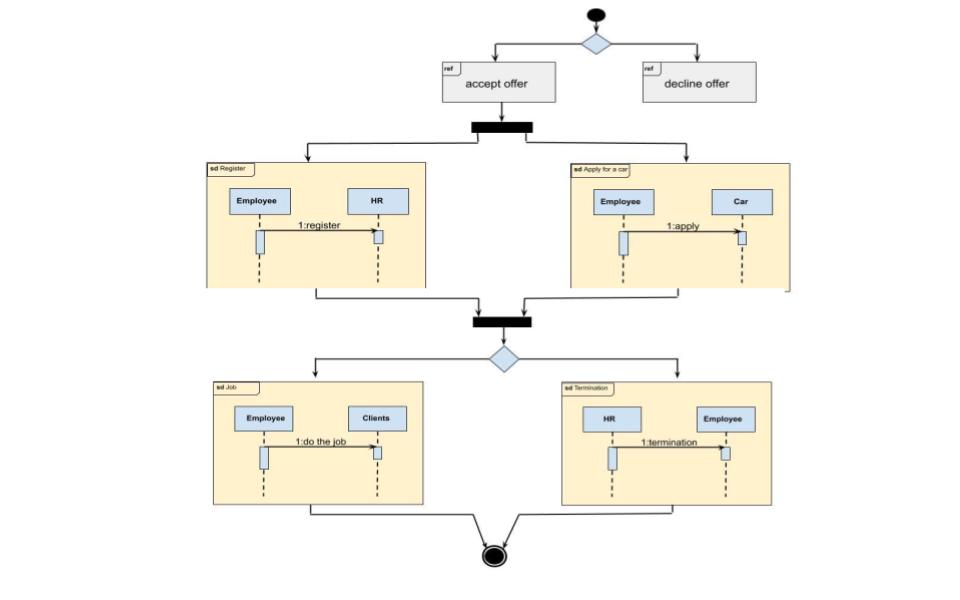


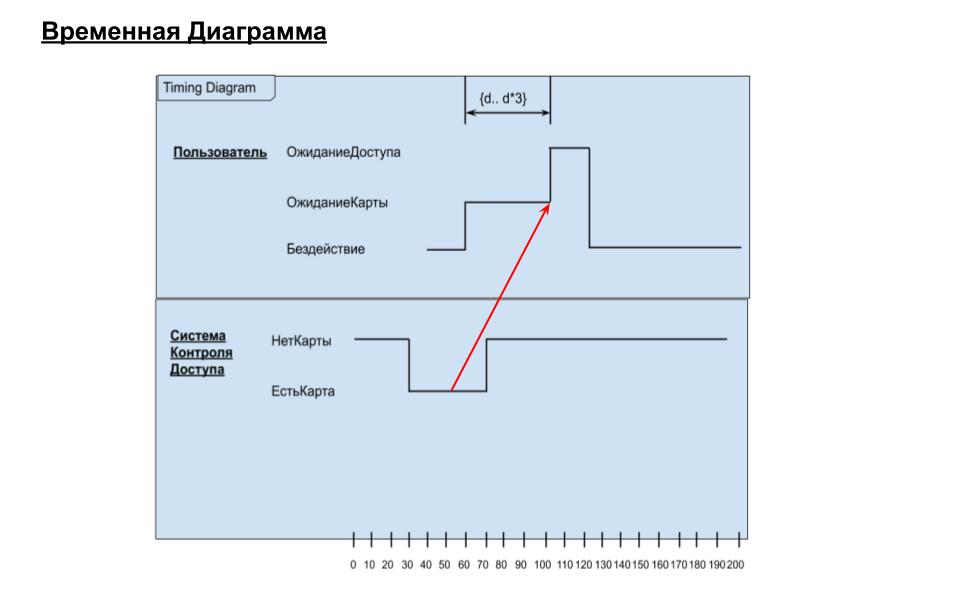
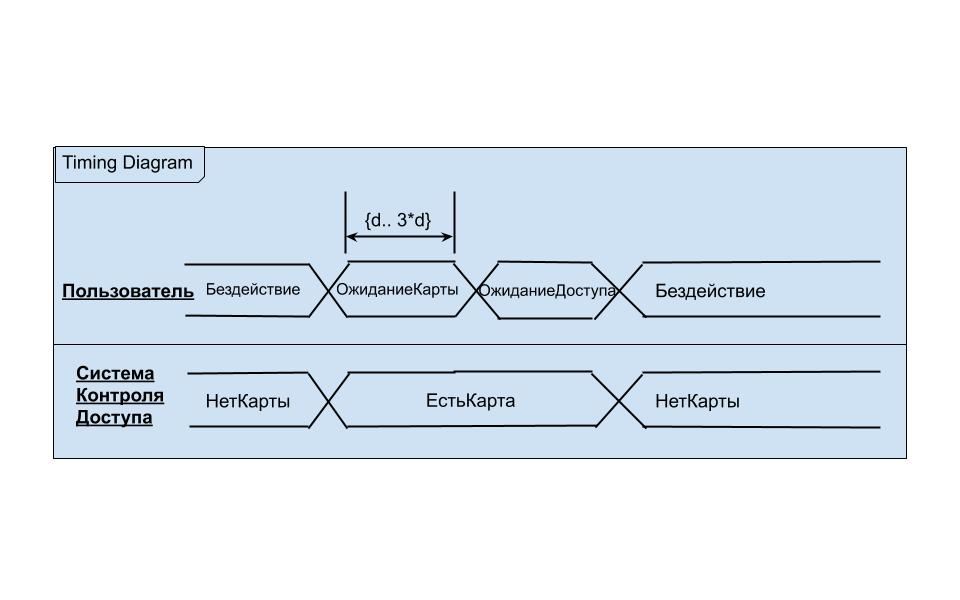
Диаграмма обзора взаимодействия

Диаграмма обзора взаимодействий фокусируется на обзоре потока управления взаимодействиями. Это вариант Диаграммы деятельности, где узлами являются взаимодействия или события взаимодействия. Диаграмма обзора взаимодействий описывает взаимодействия, в которых сообщения и линии жизни скрыты. Мы можем связать «реальные» диаграммы и добиться высокой степени навигации между диаграммами внутри диаграммы обзора взаимодействия.



Временная диаграмма

Временная диаграмма показывает поведение объекта(ов) в данный период времени. По сути — это особая форма диаграммы последовательности и различия между ними состоят в том, что оси меняются местами так, что время увеличивается слева направо, а линии жизни отображаются в отдельных отсеках, расположенных вертикально.

Зачем в UML столько диаграмм?

Причина этого заключается в том, что можно взглянуть на систему с разных точек зрения ведь в разработке программного обеспечения будут участвовать многие заинтересованные стороны, такие как: аналитики, конструкторы, кодеры, тестеры, контроль качества, клиенты, технические авторы.

Все эти люди заинтересованы в различных аспектах системы, и каждый из них требует разного уровня детализации.  
  
Например, кодер должен понимать проект системы и уметь преобразовывать проект в код низкого уровня.  
  
Напротив, технический писатель интересуется поведением системы в целом и должен понимать, как функционирует продукт.  
  
UML пытается предоставить язык настолько выразительным образом, что все заинтересованные стороны могут извлечь выгоду, как минимум из одной диаграммы UML.