Добрый вечер, Михаил Иванович!

1. UML-Какие типы диаграмм существуют?

Существует два основных типа диаграмм UML: структурные диаграммы и поведенческие диаграммы (а внутри этих категорий имеется много других). Эти варианты существуют для представления многочисленных типов сценариев и диаграмм, которые используют разные типы людей. Структурные диаграммы представляют статическую структуру программного обеспечения или системы, они также показывают различные уровни абстракции и реализации. Они используются, чтобы помочь визуализировать различные структуры, составляющие систему, например, базу данных или приложение. Поведенческие диаграммы показывают функциональные возможности системы и демонстрируют, что должно происходить в моделируемой системе.

Структурные диаграммы:

1) **Диаграмма классов**. Эта диаграмма, наиболее распространенная при разработке ПО, используется для изображения логической и физической структуры системы и показывает ее классы. Она похожа на блок-схему, потому что классы представлены в виде блоков. Эта диаграмма предлагает визуальное представление о различных классах и о том, как они взаимосвязаны. У каждого класса есть три секции:
a) Верхняя секция: имя класса
b) Средняя секция: атрибуты класса
c) Нижняя секция: методы или операции класса

2) **Диаграмма объектов**. Часто эта диаграмма используется как способ проверить диаграмму классов на точность. Другими словами, будет ли это работать на практике? Она показывает системные объекты и их взаимосвязи и предлагает лучшее представление о потенциальных недостатках проекта, которые необходимо исправить.

3) **Диаграмма компонентов**. Также известна как блок-схема компонентов, она показывает логические группы элементов и их взаимосвязи. Другими словами, она дает упрощенное представление о сложной системе, разбивая ее на более мелкие компоненты. Каждый из элементов показан в прямоугольной рамке с названием, написанным внутри. Соединители определяют отношения / зависимости между различными компонентами.

4) **Составная структурная диаграмма**. Этот тип редко используется кем-либо за пределами разработки программного обеспечения. Хотя она похожа на диаграмму классов, она требует более глубокого погружения, описывая внутреннюю структуру нескольких классов и показывая взаимодействие между ними.

5) **Диаграмма развертывания**. На этой диаграмме показаны аппаратные (узлы) и программные (артефакты) компоненты и их взаимосвязи. Она предлагает наглядное представление о том, где именно развернут каждый программный компонент.

6) **Диаграмма пакетов**. Этот тип используется, чтобы изобразить зависимости между пакетами, которые составляют модель. Основная цель — показать взаимосвязь между различными крупными компонентами, которые образуют сложную систему.

7) **Диаграмма профиля**. Этот тип меньше похоже на диаграмму и больше — на язык. Диаграмма профиля помогает создавать новые свойства и семантику для диаграмм UML путем определения пользовательских стереотипов, теговых значений и ограничений. Эти профили позволяют настраивать метамодель UML для различных платформ (например, Java Platform, Enterprise Edition (Java EE) или Microsoft .NET Framework) и доменов (например, моделирование бизнес-процессов, сервис-ориентированная архитектура, медицинские приложения и т. д.).

Поведенческие диаграммы:

1) **Диаграмма деятельности**. Этот тип изображает пошаговый процесс с четким началом и концом. Это набор операций, которые должны быть выполнены, чтобы достичь цели. Она показывает, как каждое действие ведет к следующему, и как все они связаны. Помимо разработки программного обеспечения, они могут использоваться практически в любой бизнес-среде. Их также называют картированием или моделированием бизнес-процессов.

2) **Диаграмма вариантов использования**. В этом типе описывается, что делает система, но не то, как она это делает. Вариант использования — это набор событий, которые происходят, когда “оператор” использует систему для завершения процесса. Оператор определяется как кто-либо или что-либо, взаимодействующее с системой (человек, организация или приложение) из-за пределов системы. Таким образом, диаграмма вариантов использования визуально описывает этот набор последовательностей и представляет функциональные требования системы.

3) **Обзорная диаграмма взаимодействия**. Эта зачастую сложная диаграмма похожа на диаграмму деятельности, так как обе показывают пошаговую последовательность действий. Но обзорная диаграмма взаимодействия — это диаграмма деятельности, составленная из разных диаграмм взаимодействия. Они используют те же аннотации, что и диаграмма деятельности (начальная, конечная, решение, слияние, разветвление и соединение узлов) с добавлением таких элементов, как

взаимодействие, использование взаимодействия, ограничение по времени и ограничение продолжительности.

4) **Временная диаграмма**. Когда время имеет критическое значение, используется этот тип диаграмм UML. Известная также как последовательность или диаграмма событий, она не показывает, как объекты взаимодействуют или изменяют друг друга. Функционально она показывает, как объекты и операторы действуют на временной шкале. Основное внимание здесь уделяется тому, сколько времени занимают события и какие изменения происходят в зависимости от ограничений продолжительности. Основные части временной диаграммы включают в себя:
a) Линия жизни: индивидуальный участник
b) Хронология состояний: разные состояния, через которые проходит линия жизни
c) Ограничение продолжительности: время, необходимое для выполнения ограничения
d) Ограничение по времени: время, за которое участник должен выполнить что-то
e) Возникновение разрушения: где заканчивается линия жизни объекта. Никакое другое событие не произойдет после появления разрушения на линии жизни.

5) **Диаграмма конечного автомата**. Эта диаграмма, также называемая диаграммой состояний, применяется, когда поведение объектов является сложным, а детали — существенными. Она помогает описать поведение одного объекта (или иногда оператора) и то, как оно изменяется в зависимости от внутренних и внешних событий.

6) **Диаграмма последовательности**. Эта визуально привлекательная диаграмма, популярная не только в сообществе разработчиков, хорошо показывает все типы бизнес-процессов. Она просто раскрывает структуру системы, показывая последовательность сообщений и взаимодействий между операторами и объектами в хронологическом порядке. Диаграммы последовательности отображают простую итерацию и ветвление. Это имеет преимущества для многозадачности.

7) **Диаграмма связи**. Диаграмма связи или сотрудничества аналогична диаграмме последовательности. Тем не менее, она подчеркивает связь между объектами, показывает организацию объектов, участвующих во взаимодействии, и предлагает более сложные итерации и ветвления.

2. Какие связи между классами можно изобразить в UML с помощью диаграмм классов?

Существует шесть основных типов отношений между классами: наследование, реализация, композиция, агрегация, ассоциация и зависимость.

1) Наследование. Наследование также называется **обобщением** и используется для описания отношений между родительским и дочерним классами. Родительский класс также называется базовым классом, а подкласс также называется производным классом. В отношениях наследования подкласс наследует все функции родительского класса, а родительский класс имеет все атрибуты, методы и подклассы. Подклассы содержат дополнительную информацию в дополнение к той же информации, что и родительский класс. Например: автобусы, такси и автомобили — это автомобили, у них у всех есть имена, и все они могут находиться в дороге.

2) **Реализация. Реализация** в основном используется для указания **связи между интерфейсами и классами реализации**. **Интерфейс-** это набор методов. В отношениях реализации класс реализует интерфейс, а методы в классе реализуют все методы объявления интерфейса. Например: автомобили и корабли — это транспортные средства, а транспортное средство — всего лишь абстрактное понятие мобильного средства, а корабль и транспортное средство реализуют конкретные мобильные функции.

3) Композиция. Композиция: **Отношения между целым и частью, но целое и часть не могут быть разделены**. Комбинированное отношение представляет собой отношение между целым и частью класса, а общее и часть имеют согласованное время жизни. Как только объект в целом перестанет существовать, некоторые объекты не будут существовать, и все они умрут в одной и той же жизни. Например, человек состоит из головы и тела. Они неразделимы и сосуществуют.

4) Агрегация. Агрегация: **отношения между целым и частью, а также целым и частью могут быть разделены.** Агрегатные отношения также представляют отношения между целым и частью класса, объекты-члены являются частью общего объекта, но объект-член может существовать независимо от общего объекта. Например, водители автобусов, рабочая одежда и головные уборы являются частью общих отношений, но их можно разделить. Рабочую одежду и головные уборы можно надевать на других водителей. Водители автобусов также могут носить другую рабочую одежду и головные уборы.

5) Ассоциация. Ассоциация: указывает, что **свойство класса содержит ссылку на экземпляр (или экземпляры) другого класса**. Ассоциация — это **наиболее часто используемая** связь между классом и классом, что означает наличие связи между одним типом объекта и другим типом объекта. **Комбинации и агрегации также относятся к ассоциативным отношениям**, но отношения между классами принадлежности слабее

двух других. Существует четыре вида **ассоциаций**: **двусторонние ассоциации**, **односторонние ассоциации**, **самоассоциация** и **многозначные ассоциации**. Например: машины и водители, одна машина соответствует конкретному водителю, и один водитель может управлять несколькими машинами.

6) Зависимость. Зависимость: предположим, что изменение в классе A вызывает изменение в классе B, тогда скажем, что класс B зависит от класса A. В большинстве случаев **зависимости отражаются в методах класса, использующих в качестве параметра объект другого класса**. Отношение зависимости — это отношение «использования». Изменение в конкретной вещи может повлиять на другие вещи, которые ее используют, и использовать зависимость, когда необходимо указать, что одна вещь использует другую. Например: Автомобиль работает на бензине. Если нет бензина, машина не сможет ехать.

3. Что такое «стереотип» в UML?

Стереотипы являются одним из трех типов механизмов расширяемости в унифицированном языке моделирования (UML). Они позволяют проектировщикам расширять словарь UML для создания новых элементов моделирования, получаемых из существующих, но имеющих определенные свойства, которые подходят для конкретной проблемы предметной области или для другого специализированного использования. Графически стереотип отображается как имя, заключенное в кавычки («», или, если такие кавычки недопустимы, <<>>) и расположенное над именем другого элемента. В дополнение или в качестве альтернативы он может быть обозначен соответствующей иконкой. Значок может даже заменить весь символ UML. Например, стереотипы диаграммы классов могут быть использованы для описания методов поведения, таких как «конструктор» и «геттер».