**1. Назначение и типы Диаграмм Взаимодействия.**

Диаграммы Взаимодействия (Interaction) используются для моделирования взаимодействия между объектами системы. С помощью них проектировщики и разработчики системы могут определить классы, которые нужно создать, связи между ними, а также операции и ответственности (responsibilities) каждого класса.

Существует два типа диаграмм Взаимодействия - диаграммы Последовательности (Sequence) и Кооперативные диаграммы (Collaboration).

Диаграммы последовательности - взаимодействия упорядочены во времени.

Кооперативные диаграммы - связь со временем отсутствует, она больше внимания заостряет на связях между объектами.  
  
**2.** **Перечислите основные элементы Диаграммы Последовательности. Назначение синхронизации и скриптов.**

Объект- действующие лица, участвующие в программе.

Линия жизни (lifeline) - изображается в виде вертикальной штриховой линии под объектом. Сообщение - изображают в виде стрелки, которая проводится между линиями жизни двух объектов (от клиента к серверу) или от линии жизни объекта к самой себе.

Скрипты применяются только на диаграммах Последовательности и пишутся с левой стороны диаграммы напротив соответствующего сообщения. С помощью скрипта можно пояснить назначение сообщения. Например, имеется сообщение «Проверка клиента». В скрипте этого сообщения можно разъяснить его значение следующим образом: «Проверить идентификационный номер, чтобы убедиться, что клиент существует, и что пароль введен правильно». Кроме того, скрипты позволяют указывать на диаграмме некоторые условия. В общем случае нужно стараться избегать слишком большого количества условий, чтобы не усложнять диаграмму.

**3. Перечислите основные элементы Кооперативной диаграммы.**

Кооперативные диаграммы и диаграммы Последовательности содержат одну и ту же информацию, однако представляют ее с различных точек зрения. Если диаграмма Последовательности показывает взаимодействие между действующими лицами и объектами во времени, то на Кооперативной диаграмме связь со временем отсутствует, она больше внимания заостряет на связях между объектами. Кооперативные диаграммы полезны в тех случаях, когда нужно оценить последствия сделанных изменений. При внесении изменений в объект сразу становится ясно, на какие другие объекты это повлияет. Обычно для сценария создают диаграммы обоих типов. На Кооперативной диаграмме связь между объектами или объекта с самим собой указывается в виде обычной линии, возле которой помещается текст сообщения. Сообщения нумеруются, иначе информация о порядке их следования будет потеряна.

**4. Двухэтапный подход при составлении Диаграмм Взаимодействия.**

При разработке диаграмм Взаимодействия часто применяется двухэтапный подход. Прежде всего, отображается информация высокого уровня, которая нужна конечным пользователям проектируемой системы. Сообщения еще не соотносятся с операциями, и объекты могут быть не соотнесены с классами. Эти диаграммы позволяют аналитикам, пользователям и всем, заинтересованным в бизнес-процессах лицам увидеть, как будут развиваться события в системе.

На втором этапе, после того как пользователи придут к согласию по поводу полученной диаграммы, можно углубиться в детали. При этом диаграмма может утратить свою полезность для пользователей, но станет, важна для разработчиков, тестировщиков и остальных участников команды проекта.

В начале второго этапа на диаграмму помещают некоторые новые объекты. Как правило, на каждой диаграмме Взаимодействия имеется управляющий объект, отвечающий за управление последовательностью событий сценария. Управляющий объект не реализует никаких бизнес-процессов, он лишь посылает сообщения другим объектам. Управляющий объект отвечает за координацию действий других объектов и за делегирование ответственности. По этой причине такие объекты называют еще объектами-менеджерами.

Преимущество использования управляющего объекта заключается в отделении бизнес-логики от логики, определяющей последовательность событий. Если надо будет изменить последовательность, это затронет только управляющий объект.

**5. Назначение Диаграммы Классов и этапы ее составления.**

Диаграммы Классов позволяют показать основные характеристики классов системы и связи между классами. Диаграммы Классов можно создавать для отдельного варианта использования, всей системы или подсистемы. С помощью этих диаграмм аналитики могут показать детали системы. С помощью такой диаграммы можно также выявить случаи, когда между сообщающимися классами не определено никаких связей.

Этапы составления диаграммы классов:

1. Добавление классов
2. Добавление связей и их настройка
3. Добавление атрибутов и операций

**6. Назначение Диаграммы Состояний и краткое описание ее элементов (состояния (виды состояний), деятельность, входные действия, выходные действия, переходы, события, ограждающие условия, действия).**

**Диаграмма Состояний** (State Transition) содержит информацию о состояниях, в которых может находиться объект, о том, как он переходит из одного состояния в другое и каким образом он ведет себя в этих состояниях. На диаграмме Состояний отображают жизненный цикл одного объекта, начиная с момента его создания и заканчивая разрушением.

**Состоянием** (state) называется одно из возможных условий, в которых может существовать объект. Находясь в конкретном состоянии, объект может выполнять определенные действия. Для выявления состояний объекта необходимо исследовать две области модели: значения атрибутов объекта и связи с другими объектами.

**Начальным** называется состояние, в котором объект находится сразу после своего создания. Начальное состояние обязательно - читатель должен знать, с чего начинается объект. На диаграмме может быть только одно начальное состояние, его изображают в виде закрашенного кружка. **Конечным** называется состояние, в котором объект находится непосредственно перед уничтожением. Конечные состояния не являются обязательными, их может быть сколько угодно. Конечное состояние изображают в виде закрашенного кружка с ободком («бычий глаз»).

Для уменьшения беспорядка на диаграмме можно вкладывать состояния одно в другое. Вложенные состояния называются подсостояниями, а те, в которые они вложены, - суперсостояниями.

**Деятельностью** называется поведение, реализуемое объектом, когда он находится в определенном состоянии. Деятельность — это прерываемое поведение. Оно может выполняться до своего завершения, если объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должны предшествовать do:.

**Входным действием** называется поведение, которое выполняется, когда объект переходит в определенное состояние. Оно осуществляется не после того, как объект перешел в состояние, а является частью перехода. В отличие от деятельности, данное действие рассматривается как непрерываемое. Входное действие показывают внутри состояния, ему предшествуют entry :.

**Выходное действие** осуществляется как составная часть процесса выхода из состояния. Оно является частью процесса перехода. Как и входное, выходное действие является непрерываемым. Выходное действие изображают внутри состояния, ему предшествуют exit :.

**Переходом** называется перемещение объекта из одного состояния в другое. На диаграмме переходы изображают в виде стрелки, начинающейся в первоначальном и заканчивающейся в последующем состоянии. Переходы могут быть рефлексивными: объект переходит в то же состояние, в котором он в настоящий момент находится. У перехода существует несколько спецификаций: события, аргументы, ограждающие условия, действия и посылаемые события.

**Событие** — это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода. Для отображения события на диаграмме можно использовать как имя операции, так и обычную фразу. У событий могут быть аргументы.

**Ограждающее условие** определяет, когда переход может быть выполнен, а когда нет. На диаграмме ограждающие условия заключают в квадратные скобки и размещают, вдоль линии перехода после имени события. Ограждающие условия задавать необязательно.

**Действием** является непрерываемое поведение, выполняющееся как часть перехода. Входные и выходные действия показывают внутри состояния, поскольку они определяют, что происходит, когда объект входит или выходит из состояния. Другие действия изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

**7. Назначение Диаграммы Компонентов и краткое описание ее элементов (компонент, спецификация и тело подпрограммы, главная программа, спецификация, и тело пакета, файл динамической библиотеки, спецификация и тело задачи). Связи между элементами Диаграмм Компонентов.**

**Диаграммы Компонентов** показывают, как выглядит модель на физическом уровне. На ней изображаются компоненты программного обеспечения системы и связи между ними.

**Компонентом** называется физический модуль кода. Существуют два основных типа компонентов: библиотеки исходного кода и исполняемые компоненты. Например, для языка C++ файлы с расширением «срр» и «h» будут отдельными компонентами.

Ниже обсуждаются различные стереотипы компонентов.

**Спецификация и тело подпрограммы** представляют видимую спецификацию подпрограммы и тело ее реализации. Обычно подпрограмма состоит из стандартных программных компонентов и не содержит определений класса.

**Главная программа** (Main Program) — это файл, содержащий корень программы.

**Спецификация и тело пакета** Пакет в данном случае - это реализация класса. Спецификацией пакета является заголовочный файл со сведениями о прототипах функций для класса. На C++ это файл с расширением «h». Тело пакета содержит код операций класса. На C++ это файл с расширением «срр». Исполняемые компоненты - это исполняемые файлы, файлы DLL и задачи.

**Файл динамической библиотеки** (файл DLL)

**Спецификация и тело задачи** отображают пакеты, имеющие независимые потоки управления. Исполняемый файл обычно представляют как спецификацию задачи с расширением «ехе».

Единственный возможный **тип связей** между компонентами — это зависимость. Он показывает, что один из компонентов должен компилироваться перед началом компиляции другого. Зависимость между компонентами изображают пунктирной линией.

**8. Назначение Диаграммы Размещения и краткое описание ее элементов.**

На **диаграмме Размещения** показывают физическое расположение различных компонентов системы в сети. Для системы может быть создана только одна диаграмма Размещения. Диаграмма Размещения нужна всем участникам проекта.

**Процессором** называется любая машина, имеющая вычислительную мощность, т.е. способная производить обработку данных. В эту категорию попадают серверы, рабочие станции и другие устройства, содержащие физические процессоры.

**Устройством** называется аппаратура, не обладающая вычислительной мощностью. Это, например, принтеры, терминалы ввода/вывода, сканеры т.п.

Процессоры и устройства называются также узлами сети.

**Связью** называется физическая связь между двумя процессорами, двумя устройствами или процессором и устройством. Чаще всего связи отражают физическую сеть соединений между узлами сети. Кроме того, это может быть ссылка Интернета, связывающая два узла.

**Процессом** называется поток обработки информации, выполняющийся на процессоре. Процессом, например, считается исполняемый файл. Процессы отображаются непосредственно под процессором (процессорами), на котором выполняются.

**9. С какой целью на диаграммах UML используются примечания и пакеты.**

На диаграммах UML можно размещать **примечания**, содержащие дополнительную информацию об элементах диаграммы. Хотя примечания не влияют на генерируемый код, они помогают разработчикам и другим участникам проекта лучше понять модель.

Для добавления примечания на диаграмму можно использовать два инструмента. К элементу диаграммы можно прикрепить примечание-комментарий. Если же требуется прокомментировать целую диаграмму, применяется текстовая область. В частности, в такой области указывается заголовок диаграммы.

На языке UML такие элементы, как действующие лица, варианты использования, классы и компоненты, можно сгруппировать в **пакеты**. Это позволяет упорядочить элементы модели.

**10. Модели и ракурсы. Логическая и физическая модели. Статическая и динамическая модели.**

При принятии решений в анализе и проектировании полезно рассмотреть взаимодействие классов и объектов в двух измерениях: логическом/физическом и статическом/динамическом.

В каждом из двух измерений строят несколько диаграмм, которые представляют систему в различных ракурсах. Диаграммы содержат информацию о ключевых абстракциях системы, их связях и поведении. В установившемся состоянии проекта все диаграммы должны быть согласованы между собой и со всей моделью.

1. **Логическая и физическая модели**. Логическое представление описывает перечень и смысл ключевых абстракций и механизмов, которые формируют предметную область или определяют архитектуру системы. Физическая модель определяет конкретную программно- аппаратную платформу, на которой реализована система.

Отображаются при помощи диаграмм: Диаграммы Вариантов Использования; Кооперативные диаграммы; Диаграммы Классов; Диаграммы Компонентов; Диаграммы Размещения.

2. **Статическая и динамическая модели**. Перечисленные в предыдущем пункте типы диаграмм являются в большей части статическими. Но практически во всех системах происходят события: объекты рождаются и уничтожаются, посылают друг другу сообщения (причем в определенном порядке), внешние события вызывают операции объектов. Описание динамических событий на статическом носителе, например, на листе бумаги, будет трудной задачей. В объектно-ориентированном проектировании мы отражаем динамическую семантику двумя дополнительными диаграммами: Диаграммы Последовательности; Диаграммы Состояний. Каждый класс может иметь собственную Диаграмму Состояний, которая показывает, как объект класса переходит из состояния в состояние под воздействием событий. На Диаграмме Последовательности можно показать порядок передачи сообщений.