Основы UML — диаграммы использования (use-case)

Вне зависимости от методологии разработки, которую вы применяете, первым этапом разработки будет являться формулировка требований к продукту. Набор требований к продукту представляет собой *техническое задание*, при этом требования делятся на функциональные (то, что система позволяет сделать, желаемая функциональность) и нефункциональные (требования к оборудованию, операционной системе и т.п.). В языке UML для формализации функциональных требований *применяются диаграммы использования*.

Диаграмму вариантов использования есть смысл строить во время *изучения технического задания*, она состоит из графической диаграммы, описывающей *действующие лица и прецеденты*, а также спецификации, представляющего собой текстовое описание конкретных *последовательностей действий (потока событий)*, которые выполняет пользователь при работе с системой. Спецификация затем станет **основой для тестирования и документации**, а на следующих этапах проектирования она дополняется и оформляется в виде диаграммы (в рамках ICONIX используется диаграмма последовательности, но в UML для этого имеются также диаграммы деятельности). Кроме того, use-case диаграмма достаточно проста, чтобы ее мог понять заказчик, следовательно вы **можете использовать ее для согласования ТЗ** (ведь диаграмма описывает функциональные требования к системе).

На диаграмме использования изображаются:

- акторы — группы лиц или систем, взаимодействующих с нашей системой;
- варианты использования (прецеденты) — сервисы, которые наша система предоставляет акторам;
- комментарии;
- отношения между элементами диаграммы.

На мой взгляд, наиболее правильный порядок построения диаграммы следующий:

1. выделить группы действующих лиц (работающих с системой по-разному, часто из-за различных прав доступа);
2. идентифицировать как можно больше вариантов использования (процессов, которые могут выполнять пользователи). При этом не следует делить процессы слишком мелко, нужно выбирать лишь те, которые дадут пользователю **значимый результат**. Например, кассир может «продать товар» (это будет являться прецедентом), однако «ввод штрих-кода товара для получения цены» самостоятельным прецедентом не является;
3. дополнить прецеденты словесным описанием (сценарием):
    - для каждого прецедента создать разделы: «главная последовательность» и «альтернативные последовательности»;
    - при составлении сценария нужно упорно задавать заказчику вопросы «что происходит?», «что дальше?», «что еще может происходить?» и записывать ответы на них.

Сценарии являются очень важной частью диаграмм использования, хотя их формат и не регламентирован. Ряд авторов предлагает использовать *псевдокод* для представления

сценария и даже сразу строить *диаграммы деятельности* или *взаимодействия*, но на мой взгляд, наиболее предпочтительным вариантом на этапе построения *use-case диаграмм* является текстовый, описывающий систему с точки зрения пользователя (т.к. именно этот формат будет наиболее понятен заказчику, с которым вам предстоит *согласовывать техническое задание*).

Рассмотрим *разработку диаграмм вариантов использования на примере* — пусть заказчик дал нам следующее техническое задание:

Цель — развитие у детей математических навыков.
Платформа: Linux, Windows, Android.
Функциональность:

- для учеников:
  - выбор подготовленного учителем блока заданий;
  - выполнение заданий;
- для учителя:
  - подготовка для учеников блоков заданий;
  - добавление в систему ученика;
  - просмотр отчетов.

При первом запуске система должна позволять ввести пароль учителя. Задания представляют собой математические задачи на сложение, вычитание, умножение и деление. В блоке задач могут быть задачи различных типов (указывается количество). Помимо ввода типа выполняемой в примере операции необходимо указывать допустимые диапазоны чисел (или даже отдельные числа, т.к. при изучении таблицы умножения часто сначала учат умножение на 2, затем на 5, а только потом все остальное). Кроме того, для операции вычитания необходимо иметь возможность установить вычитаемое меньше уменьшаемого (т.к. в противном случае результат будет отрицательным, а отрицательные числа в школе проходят гораздо позже).

Очевидно, несмотря на то, что заказчик очень подробно описал некоторые детали, мы не можем не только приступить к реализации задачи, но даже приблизительно оценить стоимость и сроки выполнения. Из такого задания не понятно, например, что должны содержать отчеты. Однако, мы сразу можем выделить две группы пользователей и несколько видов их деятельности.

Пример диаграммы использования

Сплошные линии на диаграмме представляют собой отношения ассоциации, отражающие возможность использования актором прецедента. После того, как определен набор вариантов использования, можно приступать к составлению сценариев. Сценарии должны описываться с точки зрения пользователя, при этом важно описывать взаимодействие пользователя с элементами интерфейса. Так например сценарий прецедента регистрации ученика мог бы выглядеть следующим образом:

Название прецедента: регистрация ученика

Действующее лицо: учитель

Цель: добавить ученика в систему, получив его пароль

Предусловия: учитель осуществил вход в систему

Главная последовательность:

1. учитель выбирает в *главном меню* пункт «*добавить ученика*«;
2. система показывает учителю *окно добавления ученика,* содержащее *поля для ввода логина и пароля*, а также *кнопки «далее» и «назад»*;
3. учитель вводит желаемый логин и пароль ученика, нажимает кнопку *«далее»*;
4. система добавляет ученика;
5. учителю открывается *главное меню* и в течении 5 секунд выводится уведомление о том, что ученик был добавлен успешно.

Альтернативная последовательность (возврат в главное меню без добавления ученика):

1. учитель выбирает в *главном меню* пункт «*добавить ученика*«;

2. система показывает учителю *окно добавления ученика,* содержащее *поля для ввода логина и парол*я, а также *кнопки «далее» и «назад»*;
3. учитель нажимает кнопку *«назад»*;
4. учителю открывается *главное меню (*при этом данные, введенные в формы *окна добавления ученика* не сохраняются).

Альтернативная последовательность (добавление ученика, уже имеющегося в системе):

1. учитель выбирает в *главном меню* пункт «*добавить ученика*«;
2. система показывает учителю *окно добавления ученика,* содержащее *поля для ввода логина и парол*я, а также *кнопки «далее» и «назад»*;
3. учитель вводит желаемый логин и пароль ученика, нажимает кнопку *«далее»*;
4. учителю в течении 5 секунд отображается уведомление о том, что запрашиваемый логин занят.

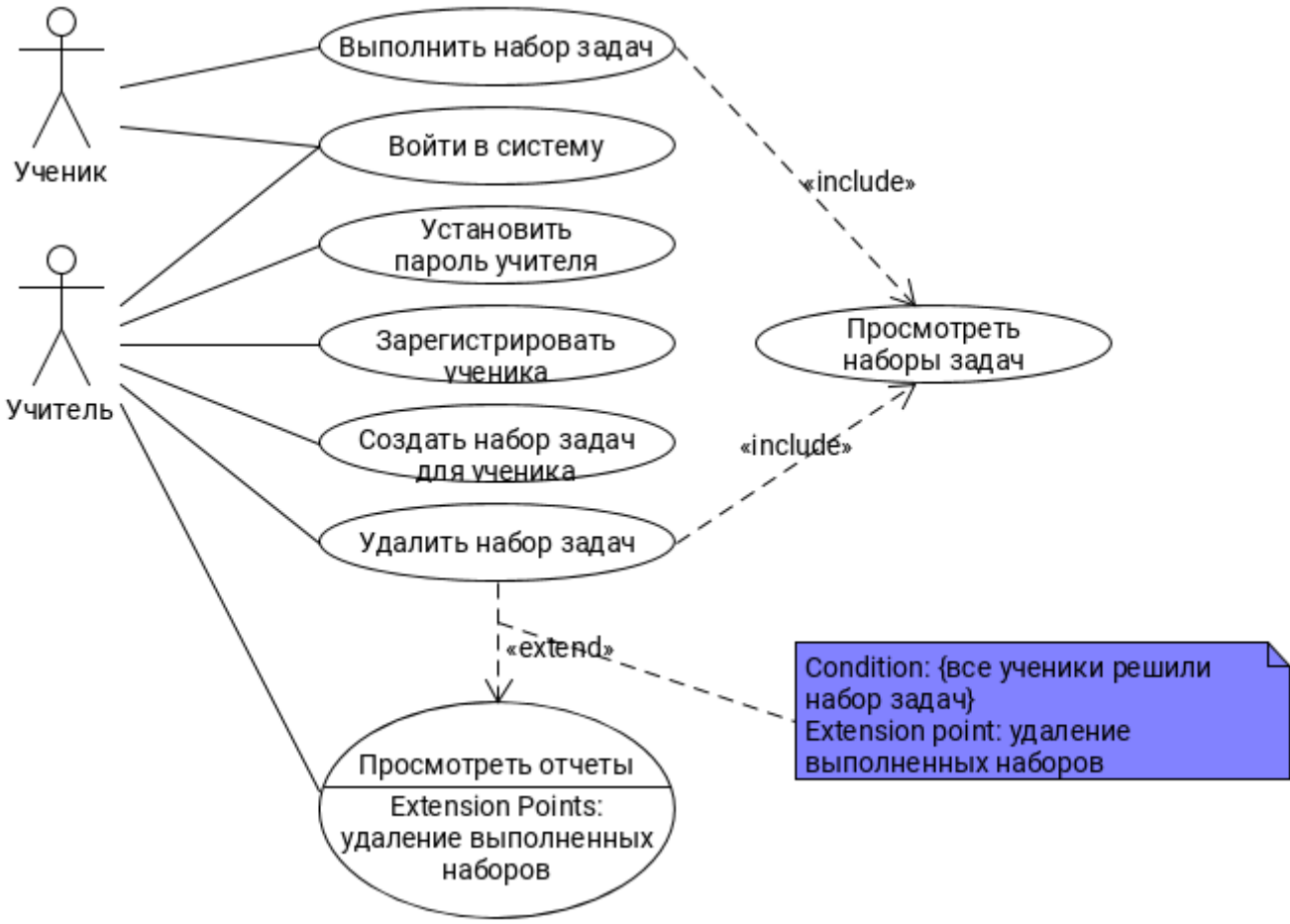
Аналогичным образом должны быть прописаны все прецеденты, изображенные на диаграмме. Составлять сценарии нужно достаточно упорно чтобы описать все возможные варианты действий пользователя в системе. Заказчик может делать это с большим удовольствием, а программист за счет этого раньше узнает возможные пожелания заказчика (так из приведенного сценария он мог бы выяснить, что программа должна отображать всплывающие уведомления).

Несмотря на простоту приведенного сценария, в его последовательностях можно найти дублирование, если оно имеет место в ваших сценариях — вы можете выделить некоторые фрагменты описания в отдельные прецеденты (которые могут быть как самостоятельными, так и являться лишь частью других вариантов использования). При этом между прецедентами появится либо отношение **расширения (extend),** либо **включения (include)**, которые отображаются на диаграммах (в *UML* также существует отношение обобщения, а в *OML* — вызова и предшествования).

Отношение включения указывает на то, что поведение одного прецедента включается в некоторой точке в другой прецедент в качестве составного компонента. Особенности включения заключаются в том, что включаемый прецедент должен быть обязательным для дополняемого (включение должно быть безусловным, а дополняемый вариант использования без включения не сможет выполняться), т.е. это это отношение задает очень сильную связь. Например, если мы хотим изобразить на диаграмме тот факт, что удаление набора задач учителем и выполнение задач учеником не должно происходить без ***обязательного*** просмотра всех наборов задач — то нам нужно использовать отношение включения:

Отношение включения на диаграмме использования

Отношение расширения отражает ***возможное*** присоединение одного варианта использования к другому в некоторой точке (*точке расширения*). При этом подчеркивается то, что расширяющий вариант использования выполняется лишь при определенных условиях и не является обязательным для выполнения основного прецедента. На диаграмме такой вид отношения изображается стрелкой, направленной к расширяемому прецеденту, в отдельном разделе которого *может быть описана* ***точка расширения***, а ***условия расширения*** *могут быть приведены* в комментарии с *ключевым словом* ***Condition***. Таким образом, расширение позволяет моделировать *необязательное поведение системы*, которое является *условным* и *не изменяет поведение* основного прецедента. Например отношение расширения нужно применить, если по техническому заданию *требуется возможность* удаления набора задач в *прецеденте просмотра отчетов* при условии, что все ученики решили этот набор.



Отношение расширения на диаграмме использования

Таким образом можно показать, что у учителя появляется возможность (но не обязанность) удалить набор задач при просмотре отчетов если все ученики выполнили этот набор.

На последней диаграмме используется символ комментария для задания условий расширения, при этом комментарий связывается пунктирной линией с отношением расширения, т.к. относится к нему. В ряде публикаций по UML и ICONIX предлагается описывать с помощью комментариев на диаграмме прецедентов:

- нефункциональные требования к системе (при этом используется *стереотип* <<*requirement*>>);
- сценарии вариантов использования (связывая комментарий с соответствующим прецедентом);
- детали реализации и другие выводы, к которым разработчики пришли в процессе обсуждения задачи (не все с этим согласны, т.к. use-case диаграмма показывается заказчику, которому не нужны детали).

Наиболее **типичными ошибками** при построении этого вида диаграмм являются:

- неправильное использование отношений расширения и включения, в том числе попытки использовать диаграммы для функциональной декомпозиции системы**.** Возникает из-за непонимания различий между этими двумя видами отношений и

того, что use-case диаграмма должна выражать лишь требования к системе, а не детали ее реализации;

- разработка диаграммы с точки зрения программиста, а не пользователя. В сценариях должны использоваться названия элементов управления (видимые пользователю), но нежелательно изображать детали реализации (такие как менеджер событий), не понятные заказчику;
- не достаточная проработка сценариев:
    - отсутствие или недостаточное количество альтернативных последовательностей, в которых должен быть учтен, в том числе, ввод некорректных данных в систему;
    - описание действий пользователя без указания конкретных элементов интерфейса системы и отсутствие описаний реакции системы в сценариях.

Важно, что *процесс ICONIX* является итеративным, поэтому если вы допустите неточности на этапе разработке диаграмм использования — их можно будет найти и исправить на следующих этапах (в частности, пропущенные объекты могут быть выделены при работе над диаграммами робастности, а сценарии детально проработаны при построении диаграмм последовательности).

Стоит отметить, что нет единого мнения по поводу использования в текстах сценария условных операторов или циклов. Ряд аналитиков считают, что наличие слов типа «если» в сценарии является ошибкой, которая исправляется добавлением альтернативной последовательности. Другие — допускают использование 1-2 ветвлений в сценарии, при этом отмечают, что такой подход улучшает читабельность сценариев.

Если следовать всем приведенным правилам составления диаграмм вариантов использования, с их помощью можно достаточно подробно ***проработать техническое задание*** чтобы ***оценить сроки*** и стоимость его выполнения, описать конкретные сценарии взаимодействия с системой, которые лягут в ***основу тестов и документации***, и ***согласовать*** всё это с заказчиком.