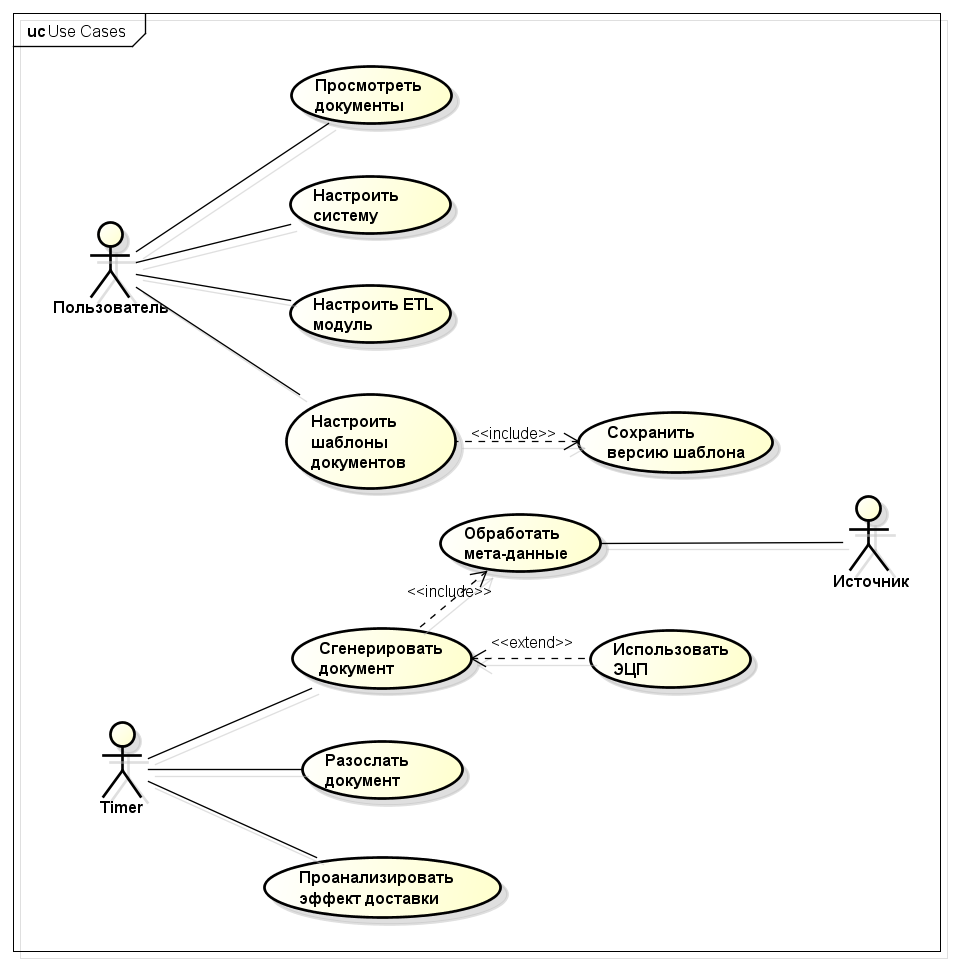
Диаграмма сценариев использования в процессе разработки ПО



Уже несколько лет я, как аналитик, довольно широко использую в своей работе сценарии использования (СИ) и диаграммы СИ для документирования требований. Вообще, у сценариев использования есть разные названия. Их называют use cases, варианты использования и даже прецеденты. Помню, как в середине 2000-х, на некоторых аналитических ресурсах шло жаркое обсуждение того, как же перевести термин use case на русский язык. Вот тогда это страшное слово «прецедент» и появилось, даже более того, некоторые товарищи утверждали, что русский язык ущербен и не позволяет передать все тонкости понятия use case. Но как показало время, понятие сценарий использования (или вариант использования) вполне себе подходит и довольно приятно на слух.  
  
Имея только положительный опыт работы с СИ, я был очень удивлен, когда при общении с коллегами из других компаний узнал, что они их не используют и, более того, даже пытались переубедить меня, что рисование человечков, овалов и палочек между ними – это пустая трата времени. Поэтому я и решил поделиться своим опытом и продемонстрировать, как диаграммы СИ помогают мне в работе.  
  
Во-первых, что такое сценарий использования? Сценарий использования – это связный рассказ о поведении системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды. Кто-то или что-то может быть пользователем системы, это может быть какая-то информационная система или устройство. И этот кто-то или что-то называется Действующим лицом (ДЛ). А сам сценарий использования представляет собой последовательность шагов, которые описывают действия ДЛ и реакцию системы на них.  
  
Например, сценарий регистрации на каком-нибудь сайте может выглядеть вот так:

1. Пользователь дает команду на регистрацию.
2. Сайт отображает форму регистрации.
3. Пользователь заполняет поля формы и подтверждает регистрацию.
4. Сайт подтверждает правильность заполнения формы.
5. Сайт отправляет на указанный e-mail письмо с запросом для подтверждения регистрации.
6. Пользователь подтверждает регистрацию.
7. Сайт регистрирует пользователя.

Как можно заметить, действия пользователя и системы чередуются между собой, они будто играют в мяч, делая пасы друг другу. Подробнее про СИ можно прочитать здесь: [ru.wikipedia.org/wiki/Сценарий\_использования](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B9_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)  
  
Но въедливый читатель может справедливо возразить, что в разработку такой сценарий отдавать нельзя. И он будет прав! Мы ведь сейчас посмотрели на систему с высоты, наверно, 5-этажного дома, когда общее поведение уже понятно, но детальных требований ещё нет. Поэтому сценарий использования должен быть, как минимум, дополнен альтернативными потоками выполнения (то что представлено выше, называется основной поток и описывает действия системы, когда все идет как надо). Альтернативные потоки – это потоки, в которых описывается реакция системы на ошибочные действия пользователя или исключительные ситуации, либо же случаи альтернативных действий пользователя, если он захотел, например, зарегистрироваться с помощью аккаунта в социальной сети. Также документ со сценарием использования можно дополнить деталями пользовательского интерфейса, правилами валидации данных, различными бизнес-правилами и сообщениями об ошибках. А вот нефункциональные требования обычно описываются в отдельном документе, так как они обычно применимы ко всей системе целиком, а не к конкретным СИ.  
  
Если же подняться ещё выше и посмотреть на систему с большей «высоты», то она будет выглядеть как набор услуг, предоставляемых системой пользователю (сценарии использования также можно рассматривать как высокоуровневые требования к системе). Тут мы приходим к тому, что неплохо бы было визуализировать эти требования, и для этого отлично подходит соответствующая диаграмма UML: диаграмма сценариев использования. Кусочек диаграммы показан ниже. Она состоит из действующих лиц, сценариев использования и различных связей между ними.



Для разработчиков и тестировщиков она, на первый взгляд, несет ещё меньше пользы, чем сами сценарии использования. Но ведь и предназначена она для другого!  
  
Диаграмма СИ рисуется обычно аналитиком в самом начале проекта для того, чтобы широкими мазками охватить весь планируемый функционал проекта и согласовать его с заказчиком. И после этого каждый СИ уже детализируется с помощью шагов и детальных требований. СИ помогают смотреть на систему с точки зрения предоставляемых системой услуг и не закапываться в частности.  
  
Я обычно использую диаграмму СИ для решения следующих задач:

* Оценка трудоемкости проекта;
* Планирование графика работ;
* Выявление пропущенных требований;
* «Оглавление» для проектных документов.

Давайте посмотрим, как СИ помогают решать эти задачи.

Оценка трудоемкости проекта

Как показывает опыт и практика, оценка получается тем точнее, чем детальнее выделен список предстоящих работ. Ведь действительно, задачу «написать статью» оценить гораздо сложнее, чем задачу «найти 5 картинок для иллюстраций». Так вот, сценарии использования являются первой итерацией разбивки системы на отдельные элементы. Более того, эти элементы обладают хорошей связностью (в данном случае – функциональная) и могут быть оценены отдельно друг от друга. Если этого недостаточно, то уже каждый СИ можно декомпозировать на основной/альтернативные потоки или даже задачи, которые необходимо выполнить программисту для реализации отдельного сценария.  
  
Здесь довольно часто также проявляется возможность повторного использования ранее созданных артефактов. Например, если для системы требуется функционал регистрации пользователей, то плюс/минус он будет работать одинаково во многих системах, что при оценке сразу же снижает степень неопределенности.  
  
Даже существует специальный способ оценки, основанный на СИ – [Use Case Points](https://en.wikipedia.org/wiki/Use_Case_Points). В этом случае выделяются действующие лица и сценарии, у них по определенным правилам определяется сложность, задаются некоторые поправочные коэффициенты, учитывающие профессионализм команды и сложность предметной области, и на выходе появляется готовая оценка. То есть, к оценке, в идеале, даже не надо привлекать разработчиков и тестировщиков.

Планирование графика работ

В проектах с фиксированном бюджетом обычно явно выделяются последовательные стадии проектирования, разработки и тестирования (как в водопадной модели). Но в реальности тестирование уже частично начинается даже до завершения стадии проектирования, так как тестировщики подключатся к ревью разработанных требований и начинают писать тест-дизайн. И если для аналитика и, скажем, архитектора, составить график работ не составляет большого труда, так как активности этих ролей начинаются вместе со стартом проекта, то определить, когда и на какую задачу стоит подключать тестировщиков с разработчиками, уже становится сложнее. Действительно, ведь входными данными для них являются подготовленные и согласованные документы с требованиями. И для решения этой проблемы опять хорошо подходят СИ. Так как каждый сценарий содержит в себе завершенное описание какого-то процесса, то их проще согласовывать с заказчиками и, соответственно, можно довольно точно установить в план-графике ключевые точки согласования документов требований. А уже от этих точек планировать работу остальной команды.

Выявление пропущенных требований

Не удивительно, что когда заказчик каким-то образом озвучивает требования к будущей системе, то он в своем описании сконцентрирован именно на полезном функционале и никак не упоминает, например, функции настройки системы или функции управления учетными записями пользователей, хотя без них система не будет полноценно работать. Очень часто после получения таких требований команда начинает выполнять оценку. У меня на практике были случаи, когда команда сразу бросалась в бой и оценивала то, что описал заказчик, оставляя за бортом неозвученные, но, тем не менее, обязательные функции системы.  
  
Также бывает и другая ситуация, когда в описании системы присутствуют требования о том, что в процессе работы пользователи должны создавать какие-то сущности. В подавляющем большинстве случаев, к таким сущностям применимы все CRUD (Create, Read (или также встречается расшифровка Retrieve), Update, Delete) операции, про которые тоже иногда забывают. В чем же здесь польза модели СИ? А как раз в графическом представлении требований. Когда глаз охватывает всю картинку, то гораздо проще заметить недостающие элементы, чем когда требования сформулированы обычным текстом.

«Оглавление» для проектных документов

Очень часто заказчик просит вместе с системой передавать комплект проектной документации. У себя в компании мы пишем проектную документацию в виде сценариев использования. Если же заказчик требует документы по ГОСТу, то все равно сперва пишутся сценарии использования, а затем уже на их основе формируются ГОСТовские документы.  
  
У нас используется, как минимум, два подхода к документированию: когда каждый СИ описывается в отдельном документе, и когда в одном документе описываются сразу несколько СИ, относящихся к одной подсистеме. Часто тот или иной подход определяется, исходя из принятого процесса или привычек заказчика.  
  
Выше уже было сказано, что первая версия списка сценариев использования уже формируется на этапе оценки и планирования графика работ. В дальнейшем этот список может дополняться и корректироваться, но уже с определенной степенью точности можно договариваться с заказчиком о том, какие именно документы будет предоставлять проектная команда.

Заключение

После этих пояснений, надеюсь, стала понятна польза, которую дает использование диаграммы СИ. Несмотря на её простоту и даже избыточность непосредственно для разработчиков, она дает довольно хороший импульс к старту любого проекта. И рассматривать её надо прежде всего в этом ключе. Алистер Коберн в своей книге «Современные методы описания функциональных требований к системам» тоже пишет о том, что не видит смысла в рисовании палочек и овалов, так как они не приближают читателя к пониманию того, как система должна работать. Но и он, как раз, говорит уже про этап детальной проработки требований, который следует после определения набора сценариев.