Новиков Александр 631 группа

Домашнее задание на 19 ноября 2021 года

Юзкейсы

**Определение**

В юзкейсах первичны не действия, которые необходимо сделать, а зачем пользователь подходит к системе, взаимодействует с ней, то есть желает получить некую услугу или пользу (value – англ.) от системы. Вот это value, которое хочет получить пользователь – это и есть название юзкейса.

Юзкейс нельзя правильно описать, если не сказать, что есть черный ящик, и что есть система. Мы не можем сказать, что есть юзкейс, что есть actor, пока не определили, что есть система. Итак, юзкейс – услуга системы или автоматизированная бизнес-работа. Actor – кто или что, который получает или принимает, то есть обменивается информацией с нашим чёрным ящиков (системой). Это могут быть пользователи (главное или основное действующее лицо), но система может взаимодействовать с несколькими внешними системами для получения-передачи информации.

Если есть actor, то есть требования на интерфейс с этим actor-ом. Это может быть графический интерфейс GUI (если это человек), может быть API.

Очень показательным является определение юзкейса из книги Коберна: «Вариант использования фиксирует соглашение между участниками системы о её поведении. Вариант использования описывает поведение системы при её ответах на запрос одного из участников, называемого основным действующим лицом, в различных условиях».

Юзкейсы являются инструментом планирования, проектирования, программирования, тестирования и развёртывания. Юзкейс модель – это юзкейс диаграмма плюс тексты. Actor всегда вне черного ящика по определению и работает, как руль взаимодействия с системой (не бухгалтер Иванов, бухгалтер Петров, а просто бухгалтер).

Юзкейс определяет последовательность действий, выполняемых системой, которая создаёт наблюдаемый результат пользы actor-а.

Каждый юзкейс:

- Описывает действия системы, которые все вместе по цепочки поставляют что-то полезное эктору;
- Показывает системную функциональность, нанизанную на сценарий;
- Моделирует диалог системы и эктора;
- Является полным и значимым потоком событий с перспективой конкретного эктора;

Преимущества юзкейсов:

- Для каждой функции мы даём контекст (сценарий, в котором она используется);
- Функциональные требования прописываются в логической последовательности, польза от которых понятна заказчику и пользователю;
- Иллюстрирует пользу от системы, проверяем, что все требования выполнены;

- Мы говорим на языке заказчика;
- В наших примерах мы строим ИС и говорим, что с её помощью теперь будут выполняться те или иные задачи (регистрация на курсы и тд);
- Говорим конкретную историю использования системы;
- Проверяем, что заинтересованные лица всё поняли;
- В результате улучается коммуникация с заказчиком и появляется соглашение, что система будет делать;

Итого юзкейс – это специальная форма требований к системе, которая формируется на двух уровнях:

1. Value (польза) – понятна заказчику и пользователю;
2. Пошаговый сценарий, понятный программистам, тестировщика, архитекторам.

Юзкейсы – это услуга системы, которую мы заранее выделили.

**Требования к юзкейсам**

1. Определить границы системы (КИС в нашем случае). Что-то есть внутри системы, но нам это не интересно (будем заниматься на шаге проектирования и дизайна, а мы сейчас занимаемся системным анализом). Нам интересно то, что есть вне системы. Нам интересно, как система, как чёрный ящик, обменивается и с кем или чем (это actor-ы)

2. Actor-ы. Люди (запускают приложения в системе), другие информационные системы (legacy – унаследованные системы, с которым наша система должна по постановке задачи взаимодействовать). При формировании actor-ов будут требования на интерфейсы (дырка в чёрном ящике, через которую можно получать/принимать информацию – поименованный набор операций).

3. Для каждого основного пользователя мы формируем список (услуги КИС). Эти услуги называются юзкейсы (варианты/прецеденты использования). С другой стороны за каждым юзкейсом стоит сценарий или последовательность действий, которая выглядит: actor что-то делает, система ему отвечает, actor снова что-то делает, система ему отвечает и т.д.). Что делает система на своей стороне в сценарии – функции ИС. Мы требует, чтобы наша ИС для выполнения данной услуги в данном сценарии умела делать эти функции – это функциональные требования нашей системы, нанизанные на сценарий. Такие сценарии называются usecase specification.

**Работа с картотеками**

У нас есть картотеки, мы можем с ними работать. Работа с картотекой называется CRUD (Create Read Update Delete) или поддержка информации. Если в нашем чёрном ящике внутри системы предполагается хранение картотек.

**Тестирование приложения**

Тестирование по юзкейсам проводится, чтобы детектировать дополнительные логические баги в приложении или программе, которые бывает очень сложно найти в тестировании привычными способами либо отдельных частей приложения.

Тестирование с помощью юзкейсов может проводится как составляющая приёмочного тестирования. Для удобства визуального восприятия Use Case часто рисуют в виде диаграмм с переходами.

Перед тем, как начать тестировать приложение необходимо поинтересоваться у заказчика, чтобы он предоставил хотя бы несколько Юзкейсов пользователей, так как нужно знать, какой набор сценариев используется чаще всего и, отталкиваясь от этого, нужно составить тест кейсы, чтобы все сценарии были покрыты. Наибольший авторитет имели те, которые стоит покрыть автоматизированными тестами.

Если такой информации нет, необходимо провести анализ приложения самостоятельно для выявления юзкейсов, сценарии по которым предположительно зачастую будут использовать будущие пользователи тестируемой программы или приложения.

**Что лучше текст или диаграмма?**

Выбор всегда остаётся за командой разработчиков. В принципе можно использовать и диаграммы, и текстовое описание к ним.

Не смотря на кажущуюся простоту и универсальность в своей наглядности диаграмм, многие предпочитают текстовое описание сценарием по следующим причинам:

- Такой формат более понятен некоторым заказчикам;
- Текст проще и быстрее создать для мало функциональных продуктов;
- Текст проще и быстрее отредактировать при малом объёме информации;

При этом данные преимущества не отменяют того факта, что нужно пользоваться только текстовыми описаниями. У диаграмм тоже есть свои преимущества:

- Диаграмма гораздо легче воспринимается при презентации;
- Диаграмма вмещает в себя больше сценариев использования;
- Диаграмма наглядно демонстрирует сценарии использования;

Подытоживая, следует отметить, что самым лучшим вариантом безусловно является наличие и текстового и диаграммного описания сценариев взаимодействия с пользователями. Однако не стоит забывать, что на создание и того и другого требуется время и ресурсы, поэтому всегда необходимо взвешивать и оценивать необходимость того или иного действия команды.

**Кому нужны юзкейсы?**

Давайте разберёмся, кому в принципе нужны сценарии взаимодействия продукта с пользователями.

Разработчикам. Очевидно, что без понимания и, что самое важное, одинакового понимания у всей команды разработчиков, сценариев использования создаваемого продукта будущими пользователями создать что-то потенциально интересное для рынка не получится. При этом всегда необходимо указывать не только основной поток (basic flow), но и альтернативные. Также должно быть указано, кто, когда и что вызывает, чем пользуется и что получается в результате;

Заказчикам. Если не предоставить заказчику сценарии взаимодействия создаваемого продукта с будущими пользователями данного продукта, может случится так, что придётся переделывать чуть ли не весь продукт. Поэтому необходимо выяснить, что конкретно ожидает от продукта заказчик, и как он планирует взаимодействия данного продукта с его будущими пользователями. Только потом можно передавать согласованные с заказчиком сценарии разработчикам. Конечно, никто не отменяет вероятность, что после создание продукта заказчику что-то не понравится, но в случае, если проблема возникнет не в рамках оговорённых и согласованных сценариев, данная проблема будет выступать в роли доработки (скорее всего за дополнительную плату), нежели в роли исправления (что делается за бесплатно). То есть, чем подробнее будет юзкейс от заказчика (согласованный с заказчиком), тем меньше вероятность исправлять что-то за бесплатно;

Тестировщикам. Как уже было отмечено выше, тестирование продукта без юзкейса практически неэффективно. Тестирование – это репетиция использования продукта пользователями на специальных сотрудниках, которые знакомы с внутренним устройством продукта и могут дать разработчикам полезные и детальные оценки и указания. Этот вид тестирования называется альфа-тестированием. Также существует и бета-тестирование продукта, обычно оно проводится после альфа-тестирования. Отличается бета-тестирование тем, что вместо специальных сотрудников в роли тестировщиков выступают реальные пользователи (конечно, чаще всего ограниченных круг пользователей, которые лучше разбираются в продукте или внесли плату).

Из вышеописанного следует, что юзкейсы нужны практически всей проектной команде и даже некоторому кругу пользователей, которые будут проводить бета-тестирования созданного продукта.

**В каких случаях нужны юзкейсы?**

Подытоживая описательную часть следует отметить, что юзкейсы нужны:

- При необходимости получить полную и качественную спецификацию требований к использованию продукта;
- Для поддержания работоспособности и корректного функционирования продукта;
- Для оперативного исправления багов, ошибок и прочих недочётов в работе продукта;
- Для описания функционального использования продукта и создания инструкции по его использованию для пользователей;

**Как создавать юзкейсы?**

Рассмотрим пару примеров того, каким образом можно создать юзкейс.

Пример 1. Разблокировать учётную запись пользователя:

| | |
|---|---|
| Действующие лица: | Администратор, Система |
| **Цель** | Изменить статус учётной записи пользователя на «активный». |
| **Предусловие** | Учётная запись пользователя не активна. |

| | |
|---|---|
| **Успешный сценарий:**<br>1. Администратор выбирает пользователя и активирует «Разблокировать».<br>2. Система переключает учётную запись пользователя в статус «активный», и посылает сообщение пользователю на email, если поле «User Account → email» не пусто). | |
| **Результат** | Учётная запись пользователя была переведена в статус «активный». |

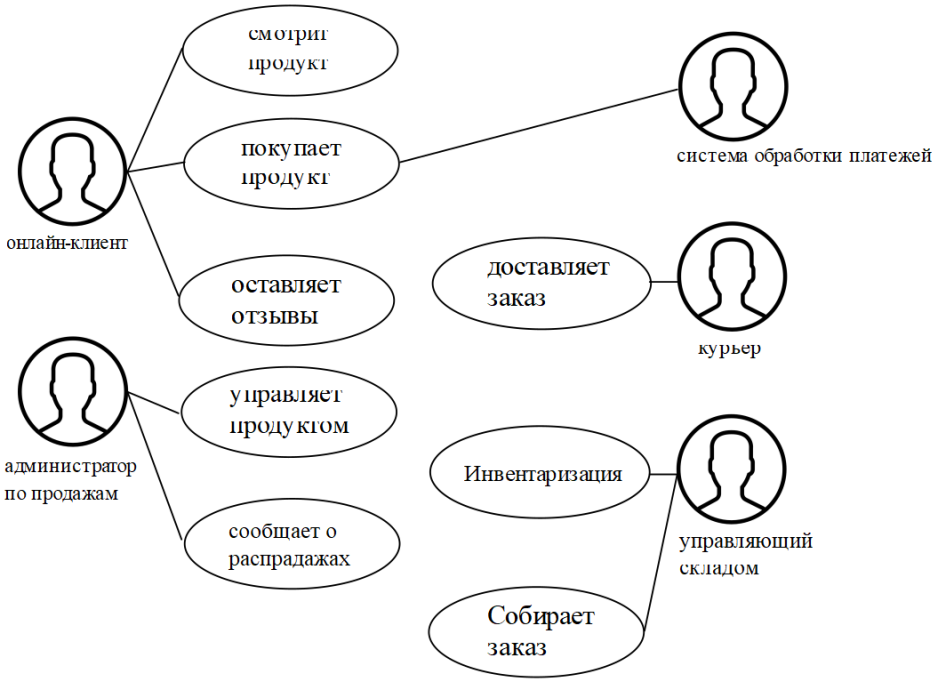
Пример 2. Авторизация пользователя:

| | |
|---|---|
| **Действующие лица** | Пользователь, Система |
| **Цели** | Пользователь: авторизоваться в системе и начать работать; Система: идентифицировать пользователя и его права. |
| **Успешный сценарий:**<br>1. Пользователь запускает систему. Система открывает сессию пользователя, предлагает ввести логин и пароль.<br>2. Пользователь вводит логин и пароль.<br>3. Система проверяет логин и пароль.<br>4. Система создаёт запись в истории авторизаций (IP адрес пользователя, логин, дата, рабочая станция).<br>5. Система выдаёт пользователю сообщение по поводу успешной авторизации. | |
| **Результат** | Пользователь успешно авторизирован и может работать с системой. |
| **Расширения:** | |
| ***а** | Нет доступа к БД. Система выдаёт сообщение об ошибке. Результат: пользователь не может войти. |
| **1а** | В настройках безопасности для данного IP адреса существует запрет на вход в систему. Результат: форма логина не предоставляется, система выдаёт сообщение пользователю. |
| **2а** | Пользователь выбирает: «Напомнить пароль». Вызывается сценарий «Напомнить пароль». |
| **3а** | Пользователь с введёнными логином и паролем не найден. Результат: отказ в авторизации. Система выдаёт сообщение об этом. Переход на шаг 2. |
| **3б** | Количество неудачных попыток авторизоваться достигло максимального, установленного в настройках. Результат: пользователь не может войти. Выдаётся сообщение об этом. Вход с IP адреса пользователя заблокирован на время, установленное в настройках. |

Пример 3. Как мы расписывали сценарий взаимодействия клиента кафе с кафе, как с бизнес-системой:

| Альтернативные потоки | Процесс (работа) | Результат | Исполнитель |
|---|---|---|---|
| -//- | -//- | -//- | -//- |

Пример 4. В виде диаграммы




Пример 5. Текстом. Регистрация в системе.

«Пользователь нажимает кнопку «Зарегистрироваться» на сайте, переходит на страницу регистрации, заполняет форму для регистрации (вводит логин и email, придумывает пароль). Затем нажимает «Зарегистрироваться». Система проверяет, зарегистрирован ли пользователь с таким логином или email. Если да, система выдаёт сообщение об этом и просит пользователя либо войти в аккаунт, либо придумать другой логин. Если пользователь с таким логином или email не найден, система отправляет на email ссылку для подтверждения электронной почты и показывает пользователю сообщение, что на его email отправлена ссылка для подтверждения электронной почты…»

Как можно заметить, текстовый вариант расписывания сценариев чаще всего очень и очень громоздкий. Невозможно объять все альтернативные потоки, не запутав читателя. Однако, как было отмечено выше, в некоторых случаях можно воспользоваться и текстовым описанием.

**Важные моменты**

Следует отметить несколько основных моментов:

- Единый стиль спецификации. Не стоит что-то описывать текстом, что-то таблицей, а что-то диаграммой;
- Минимальное количество слов и пунктов, необходимых для однозначного понимания сценария. Если юзкейс получается слишком длинный, лучше разбить его на несколько, так как с очень длинными сценариями и большим количеством расширений очень неудобно работать;
- Если в двух и более сценариях повторяется одинаковый набор шагов, есть смысл вынести эти шаги в отдельный сценарий, и ссылаться на них;

- Список сообщений, которые система выдаёт пользователю, стандартные тексты электронных писем и т.п. удобно расположить в едином месте в документе, и ссылаться на нужный пункт из разных юзкейсов;
- Разница между валидацией (проверка системы на ту пользу, которая она делает пользователям) и верификацией (проверка приложения на удовлетворения требованиям)