Contents

Что такое User Story?

Тезис, цель которого *кратко* описать *ценность* для какого-то участника в системе.

Как понять, является ли тезис пользовательской историей

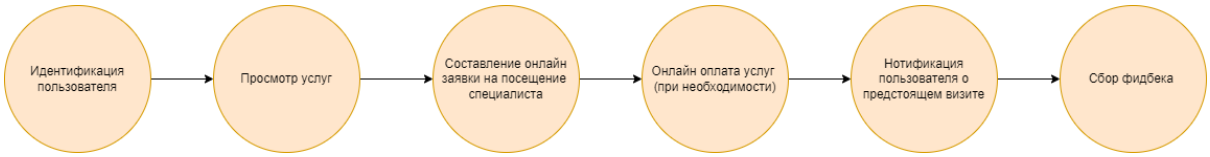
- Тезис достаточно краток, так что оценка сложности реализации описываемой ценности не вызывает затруднения. Если с оценкой возникают сложности, то такой тезис не является пользовательской историей и нуждается в декомпозиции либо в уточнении минимального функционала. Также трудности с оценкой могут возникнуть из-за недостаточной квалификации ответственного лица(лиц), что необходимо учитывать.
- Тезис несёт ценность для участника системы. Ценностью, в данном случае, выступает такой артефакт, который может использоваться *хоть каким-то* участником системы, для закрытия своих потребностей. Пример участников системы: конечный пользователь (тот, кто заинтересован в использовании профильной услуги системы), администратор системы, разработчик системы, stakeholder системы и другие.
- Тезис не должен содержать детали имплементации. Детали уточняются у PO на момент планирования спринта или позже.

Возможные методы при составлении User Stories.

На основании этапов какого-то сценария системы.

При составлении User Story можно пользоваться следующим приемом: берем в рассмотрение один из сценариев для заданного пользователя. Далее, из этого сценария выделяем основные этапы, на их основе могут строиться User Stories.

Пример. Рассмотрим сценарий “Вызов сантехника на дом” на сайте крупной ЖКХ компании со стороны конечного пользователя (непосредственного пользователя услугами ЖКХ).

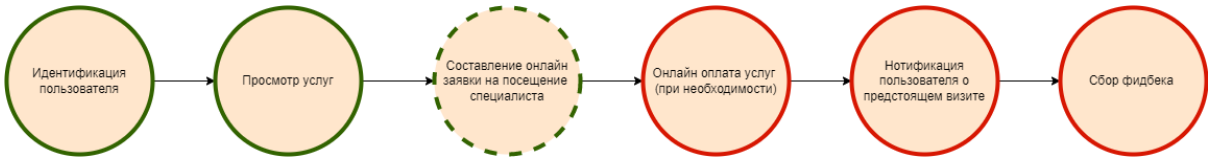


В данном примере сценарий разбит на несколько основных этапов, каждый из которых обозначен кружком на схеме. Теперь, на основании выделенных этапов нетрудно составить User Story. Так, этап “Просмотр услуг” может трансформироваться в тезис “Я, как пользователь услуг ЖКХ, хочу просматривать список предоставляемых услуг, чтобы выбрать необходимую”.

Дальнейшая работа с User Stories. Планирование спринта.

При планировании спринта важно отобрать те User Stories, которые позволяют максимально полно закрывать пользовательский сценарий, но при этом важно чтобы эти истории были *выполнимы* в течении спринта.

Рассмотрим приведённый ранее пример. Сценарий “Вызов сантехника на дом” включает в себя несколько этапов, каждый из которых является отдельной User Story. Однако, взятие всех User Stories в спринт может оказаться *ошибочной* стратегией. Причин тут может быть несколько: 1) у команды может быть недостаточно ресурсов 2) заинтересованное лицо (например, PO) хочет проверить гипотезу. Поэтому, стоит выделить те этапы в сценарии, которые не обойдутся без автоматизации и те, которые можно автоматизировать частично или вовсе исключить.



Если взглянуть на наш процесс, то можно заметить, что первые два этапа являются неотъемлемой частью нашего процесса (по мнению заинтересованных лиц) и исключить их не получится. Этап “Составление онлайн заявки на посещение специалиста” поддается упрощению, например, вместо формы на сайте с требованием ввода полной информации об услуге, фото проблемы и прочих деталей можно оставить форму, в которой достаточно ввести номер телефона, после чего оператор совершит обратный звонок и зарегистрирует заявку в системе. Последние три этапа можно вовсе исключить или обойтись ручным исполнением: специалист примет оплату на месте; пользователю позвонит оператор за несколько часов до прихода; специалист попросит поставить роспись в квитанции после работы.

Таким образом, в спринт попадут 3 User Story, что позволит команде донести ценность до конечного пользователя всего за один спринт.

Декомпозиция User Story

После того, как User Stories были отобраны, происходит их декомпозиция. Декомпозиция может происходить во время планирования спринта (с участием всей команды) или перед планированием спринта наиболее компетентными участниками команды. Таким образом, User Story делится на задачи (tasks).
Задача (task, та́ска) – это техническое задание, которое помогает достичь ценности, объявленной в User Story.

На данном этапе для корректной декомпозиции User Story требуется уточнить у ответственных лиц (PO или заказчик) в каком виде будет предоставляться данная ценность, т.е. внести детали. Поясним последний тезис на примере одной из User Story в сценарии “Вызов сантехника на дом” на сайте ЖКХ, который мы разбивали на User Stories ранее. Рассмотрим следующую User Story - “Я, как пользователь услуг ЖКХ, хочу просматривать список предоставляемых услуг, чтобы выбрать необходимую”. Декомпозировать данную историю невозможно без внесения конкретики, поэтому со стороны ответственных лиц для данной истории должны быть добавлены такие детали, как:

- Сайт будут открывать только на десктопе в браузере. Не ожидаем пользователей с телефона.
- Описание об слуге может быть до 20 предложений, в описании могут быть видео и картинки
- …

Таким образом, после уточнения деталей по User Story можно приступить к её декомпозиции.

При декомпозиции User Story на задачи можно пользоваться ролевым разбиением на задачи. В нашем примере, история может быть разбита на следующие технические задачи:

- Дизайн окна с описанием услуги
- Верстка окна с описанием услуги
- Отрисовка окна с описанием услуги по данным из бекенда
- Реализация API для получения услуг и их описание
- Развертывание фронтенда и бекенда в production среде
- …

Итого, при декомпозиции User Story у нас должны получиться задачи, при достижении которых мы принесем ценность участнику нашей системы.

Работа со Scrum / Kanban доской

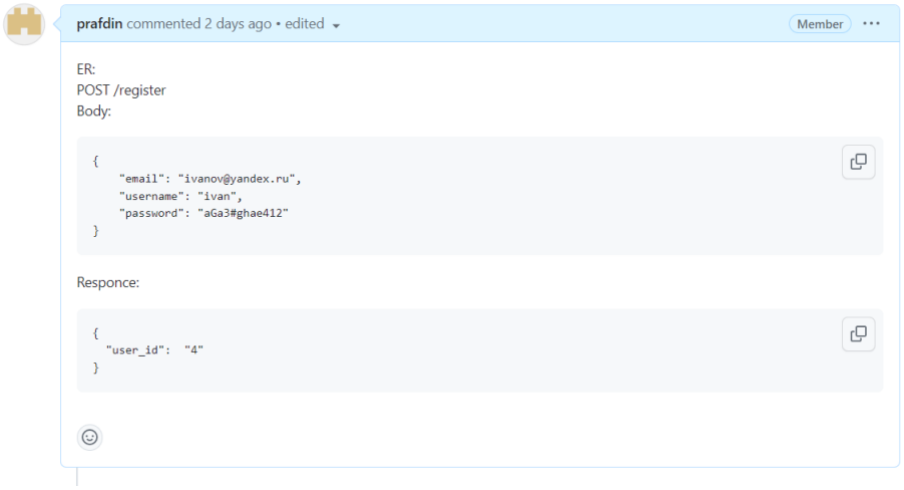
При планировании и проведении спринта удобно пользоваться Scrum / Kanban доской.

Существуют различные сервисы, которые предоставляют такие доски в электронном виде: Jira, Trello, GitHub Projects, Kaiten и другие. Представленные сервисы отличаются по интерфейсу, интеграциями с другими решениями, количеству предустановленных колонок и другими незначительными деталями. При этом, основной принцип ведения

досок остается неизменным, а значит команда может выбрать любую доску, исходя из личных предпочтений.

При планировании спринта технические задачи – tasks, которые были получены при декомпозиции User Story, заносятся на доску в виде краткого тезиса, который позволяет всем заинтересованным лицам понять сущность задачи. При этом, организация карточки может быть разная, однако, в сообществе существуют некоторые полезные практики:

1. Карточка закрепляется за конкретным членом команды т.н. assignee (асайни) – человеком, ответственным за выполнение поставленной задачи. Такое закрепление может быть полезно при проведении различных мероприятий в рамках спринта (daily standup, demo и др.). Также важно чтобы у карточки был один асайни. Если задача требует вовлечение разных участников команды, например, фронт + бек, то такую задачу необходимо декомпозировать.
2. В карточке указывается член команды, который будет ответственный за принятие задачи и её закрытия т.н. Reviewer (ревьюер).
3. В карточке указывается лицо, у которого можно получить дополнительные детали по задаче т.н. Point Of Contact (POC). Часто бывает, что POC и ревьюер это один человек.
4. Карточка дополняется техническим описанием поставленной задачи. В отличии от названия карточки, в описании детально представлены моменты, на которые стоит обратить внимание при реализации.
5. Если при составлении задачи возможно формально описать поведение системы после решения задачи, то в описание задачи добавляется пункт “Expected Result” или “ER”. Например, для задачи “Реализация API для регистрации нового пользователя” ER может быть описан в следующем виде:



Далее, у каждой карточки есть свой workflow – как она двигается по доске, т.е. по каким колонкам и в какой очередности. Некоторые сервисы могут строго закреплять workflow за карточкой (например, Jira), другие же оставляют это на выбор команды и строго не регламентируют workflow. В сообществе приняты некоторые стандартные workflow для задач: *Backlog, In progress, In review, In QA, Done* или *New, Backlog, Ready, In progress, In review, Done* и тд. Каждая карточка помещается в

соответствующую колонку при выполнении некоторых условий, заранее определенных на уровне команды. Пример таких условий:

- New. Сюда помещается все задачи, которые не находятся в текущем спринте. Обычно, при создании новой карточки, в первую очередь, она помещается в эту колонку.
- Backlog. Сюда помещаются все задачи, которые находятся в текущем спринте. Карточки в эту колонку, обычно, попадают из колонки New при планировании спринта.
  Хорошей практикой является введение карточек в эту колонку только во время планирования спринта, но не после его начала. Однако, не всегда получается всё учесть на момент планирования спринта и в эту колонку задачи могут попадать уже после начала спринта.
- Ready. Сюда помещаются все задачи, находящиеся в спринте, но, в отличии от колонки Backlog, задачи в этой колонке *полностью* понятны для асайни и готовы к взятию в работу. При необходимости (члены команды плохо представляют границы задачи, т.е. не имеют представление какие моменты нужно уточнять для решения задачи; ограниченное взаимодействие с POC; сокращение временных затрат;) команда вводит Definitions of Ready (см. ниже) для задачи.
- In progress. Сюда помещаются все задачи в спринте, которые выполняются на данный момент.
- In review. Сюда помещаются все задачи в спринте, которые, по мнению асайни, готовы к закрытию. На данном этапе workflow ревьюер проверяет, что Definitions of Done (см. ниже) соблюдены и задача может быть закрыта, а её результат может быть доставлен в production среду.
- Done. Сюда помещаются все задачи, которые прошли ревью и были доставлены до конечно пользователя.

DoR задачи

Definition of Ready (DoR) – это список условий, который формируется для упрощения процесса передачи задачи в команду разработки. Данный список можно считать некоторым чек-листом, который позволяет дать четкий, формальный ответ со стороны команды: “да, задача понятна” или “нет, задача не понятна” в случае если все пункты этого листа выполнены или не выполнены. Данный набор условий задается не для конкретной задачи, а для всех задач или какого-то конкретного класса задач, например, для задач frontend. Данный артефакт не является обязательным, поэтому команда сама может оценить его необходимость. Например, если на сессиях ретроспективы часто поднимается проблема о том что задачи слабо формализованы и приходится тратить много времени на выяснение деталей, то стоит попробовать ввести практику использования DoR в команде.

DoD задачи

Definition of Done (DoD) – это список условий, который формируется для упрощения процесса передачи задачи из статуса “В разработке”, в статус “Готово”. По аналогии с DoR, данный список можно считать чек-листом, который вносит конкретики в фразу “Задача готова”. При наличии данных условий поддерживать качество выпускаемой продукции легче: каждый участник команды знает (при наличии DoD), что готовая задача это не только написанный код, но и написанные и запущенные Unit тесты и соблюдение других критерий, которые команда выставляет перед собой. Также введение DoD положительно влияет на time to market кода, потому что фаза доделывания/дотестирования сокращается, если у асайни есть способ проверить себя самостоятельно по набору условий. Как и для DoR, definition of done обычно задаются не для конкретной задачи, но для некоторого класса задач: DoD для бекенда, DoD для фронтенда и т.д. “Жесткость” требований, предъявляемых в DoD, также решает для себя команда, например, варьируя качество и количество условий мы можем изменять скорость доставки кода на production (при этом, как правило, мы теряем в качестве) или наоборот увеличивать качество продукта, за счет требований покрытия тестами 100% кода, проведения интеграционного тестирования и т.д. В отличии от DoR, использовать данный артефакт стоит практически всегда, потому что без DoD ввод кода в эксплуатацию может быть затруднен.

Примеры DoD для IT команды:

Условия завершения задачи для DevOps

1. Необходимое реализовано. В issue предоставлены нужные ссылки, ip, мануалы
2. В issue описаны шаги для выполнения поставленной цели (если это уместно)
3. В issue прикреплена Диаграмма/Схема (если это уместно)
4. Прошло Review

Условия завершения задачи для FrontEnd

Дизайн:

1. Дизайн соответствует требованиям
2. В issue прикреплена ссылка на Page в которой задача была реализована
3. Дизайн одобрил Frontend-разработчик
4. Прошло Review

FronEnd:

1. Вёрстка соответствует дизайну
2. Функционал соответствует требованиям
3. Создан Pull Request в develop ветку
4. PR одобрен

Условия завершения задачи для BackEnd

1. Функционал соответствует требованиям
2. В issue прикреплён скриншот dev теста.
3. Создан Pull Request в develop ветку
4. PR одобрен

DoD истории

Definition of Done для User Story формируется по аналогии с DoD для задачи, за тем исключением, что требования, предъявляемые в DoD для User Story, могут носить более менее технический характер: все задачи были показаны на демо; получен апрув для доставки изменения на production и др. User Story не имеют привязки к конкретной сфера, поэтому DoD могут быть сформированы для всех User Story сразу. Пример:

Условия завершения User Story

1. Все задачи находятся в статусе Done
2. Product Owner принял User Story