Работа со Scrum

JSP & Servlets
15 уровень, 4 лекция

ОТКРЫТА

User Story

Пользовательские истории (User Story) — эффективный способ изложить требования к программному обеспечению, находящемуся в разработке. Такие истории содержат краткие советы от лица пользователя ПО.

Поскольку в методике Scrum постановка задач — это обычно прерогатива заказчика или владельца ПО, именно они считаются главным способом влияния на процесс разработки. Каждая User Story имеет ограничение в объеме текста и сложности изложения. Историю чаще всего пишут на небольшом листе, что само по себе ограничивает объем.

Благодаря пользовательским историям можно документировать пожелания клиента и оперативно реагировать на запросы рынка.

User Story следует считать упрощенным показателем требований, поскольку в ней нет процедуры приемочного тестирования. Составление пользовательской истории должно соответствовать приемной процедуре. Это даст гарантию, что User Story достигла своей цели.

Структура истории имеет примерно такой вид: “Будучи пользователем <тип пользователя>, я хочу выполнить <действие> для получения <результат>” (As a product owner I want …). Подобная структура не только проста, но и понятна каждому.

Преимущества применения User Stories:

- Истории невелики по объему и их легко создавать.
- Помогают всем заинтересованным лицам обсуждать работу над проектом и его поддержку.
- Не требуют постоянного обслуживания.
- Актуальны лишь при использовании.
- Улучшают взаимодействие с клиентом.
- Благодаря им можно разделить проект на мелкие этапы.
- Облегчают работу над проектами с плохо понятными требованиями.
- Упрощают оценку задач.

Недостатки User Stories:

- Без предварительного согласования процедуры могут усложнить использование в виде базы для соглашения.
- Их использование нуждается в тесном контакте с клиентом на протяжении всей работы над проектом, что иногда затрудняет рабочий процесс.
- Имеют недостатки при масштабировании на крупных проектах.
- Напрямую связаны с профессиональным уровнем разработчиков.
- Используются для старта обсуждения, но могут не завершать обсуждение и не применяются для документации системы.

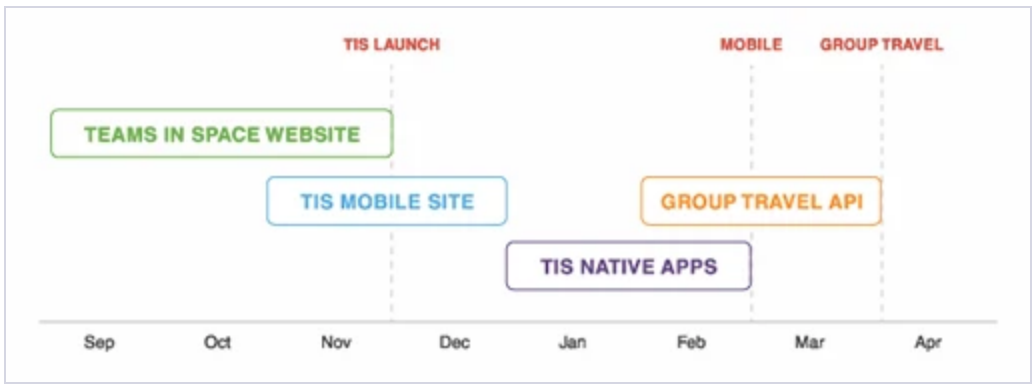
Backlog

Бэклог продукта — это текущие задачи в виде списка, составленного в порядке приоритета. Список формируют на базе дорожной карты (roadmap) проекта и изложенных в ней пунктов. Самые важные задания обычно находятся в начале списка. Это необходимо для понимания, какая работа должна быть сделана первой.

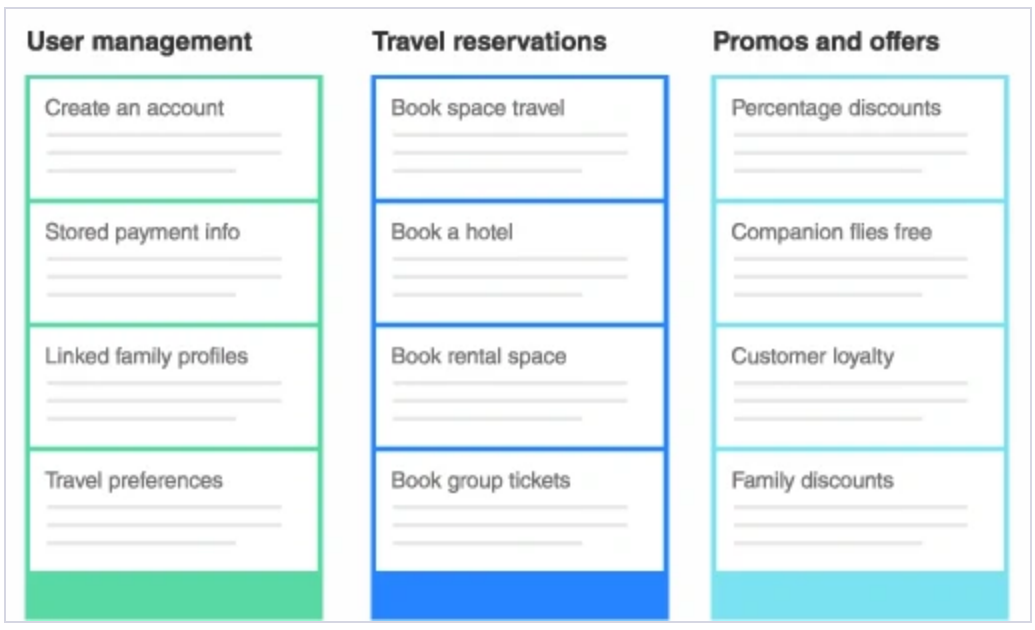
Команда разработчиков выбирает скорость выполнения задач бэклога независимо от пожеланий заказчика, а исходя из своей квалификации и опыта прошлых спринтов. “Подгонять” программистов крайне нежелательно. Команда сама выбирает задания из бэклога по собственным соображениям и возможностям. Выполнение проходит без перерыва (Kanban) или несколькими итерациями (Scrum).

Два важных условия бэклога

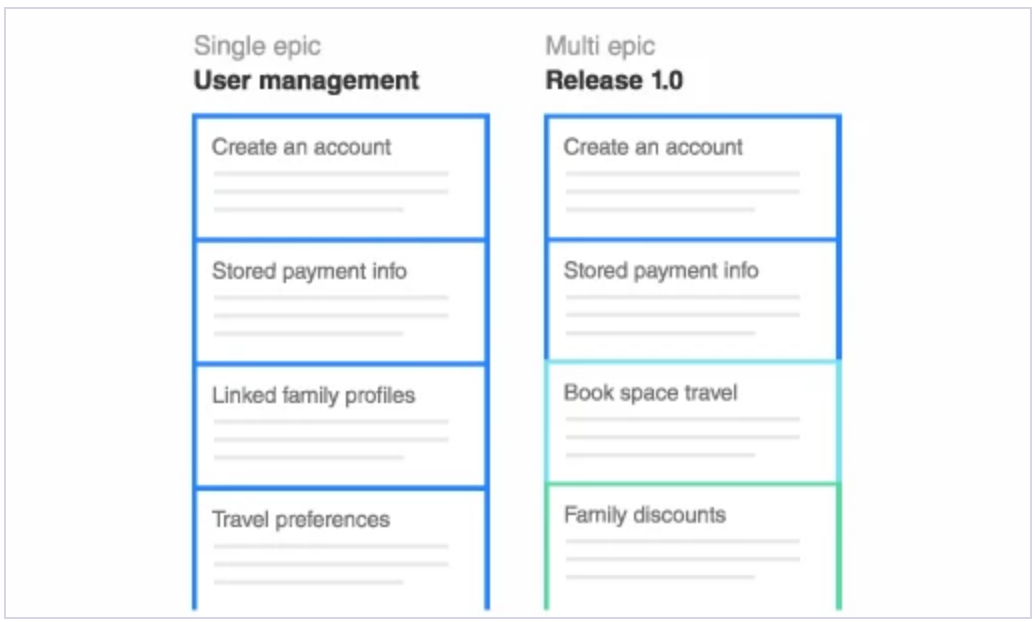
Основа бэклога продукта состоит из дорожной карты, предложений и условий выполнения. В эпиках содержатся условия и User Story. Давайте внимательно изучим пример типичный дорожной карты.



Создание сайта “Команды в космосе” — первое предложение из дорожной карты. Его требуется разделить на эпики (на рисунке они показаны зеленым, синим и бирюзовым цветами) и User Story для каждого эпика.



Заказчик ПО формирует из нескольких User Story один список. При необходимости он может изменить порядок выполнения историй, чтобы разработчики сначала занялись одним из наиболее важных эпиков (слева) или проверили, как работает льготное бронирование билетов. Чтобы это выполнить, требуется реализовать истории из эпиков (справа). Оба варианта можно увидеть ниже.



На основе каких факторов заказчик должен расставлять приоритеты?

- Актуальность для пользователей.
- Наличие обратной связи.
- Сложность разработки.
- Взаимосвязь между заданиями (чтобы выполнить “Б”, для начала нужно сделать “А”).

Приоритеты в работе определяет заказчик, но свое мнение об этом могут высказать и другие стороны. Успех бэклога зависит, в том числе, от мнения клиентов и программистов. Коллективными усилиями они могут добиться улучшения результатов и обеспечить своевременную поставку готового продукта.

Как вести бэклог

Если бэклог уже создан, то после этого нужно периодически менять его по ходу дальнейшей работы. Заказчику ПО стоит удостовериться в правильном составлении бэклога перед каждым новым планированием итерации. Это поможет уточнить приоритеты или что-то изменить после анализа последней итерации. Корректирование бэклога в Agile иногда называют “грумингом” или “уточнением” или “ведением бэклога”.

Если бэклог уже относительно большой, то заказчику нужно сгруппировать задачи по краткосрочности и долгосрочности выполнения. Краткосрочные задания нужно тщательно изучить, прежде чем дать им этот статус. Придется составить User Story, выяснить все нюансы внутри команды.

Что касается долгосрочных задач, то крайне желательно чтобы разработчики дали им свою оценку. Это упростит расстановку приоритетов. Возможно, что-то изменится, но команда улучшит понимание задач и быстрее продолжит работу.

Бэклог — это важная составляющая между заказчиком и командой программистов. Заказчик может всегда изменить приоритеты, изучив отзывы клиентов, прогнозы или получив новые требования.

Рекомендуется избегать внесения изменений непосредственно во время работы. Это плохо влияет на рабочий процесс и эмоциональное состояние программистов.

Sprint

Спринт — это небольшой промежуток, во время которого нужно выполнить ранее оговоренный объем работы. Спринты основаны на методологиях Scrum и Agile. Верный выбор спринтов помогает agile-команде разрабатывать качественное ПО.

“Применяя Scrum, можно разработать продукт за несколько итераций с четкой продолжительностью — спринтами. Это помогает разбивать крупные проекты на мелкие задачи”, — утверждает Меган Кук, руководитель направления Jira в компании Atlassian.

Как происходит планирование и выполнение спринтов в Scrum?

По мнению авторов методологии Scrum, для планирования будущего спринта нужно всем встретиться на отдельном собрании. На этом мероприятии участники команды должны выяснить ответы на два главных вопроса: что требуется сделать в этом спринте и как это лучше всего выполнить?

В определении списка рабочих задач участвуют заказчик ПО, Scrum-мастер и программисты. Заказчик объясняет цель спринта и задания из бэклога.

Затем команда разрабатывает план, по которому будут происходить выполнение задач на спринте. Этот план вместе с выбранными рабочими заданиями называется бэклогом спринта. После совещания по планированию команда приступает к работе. Разработчики выбирают задания из бэклога, по мере завершения работы статус каждого задания меняется с “В работе” на “Готово”.

Во время спринта команда проводит ежедневные Scrum-совещания (стендапы), на которых обсуждается текущие проблемы и ход работы. Такие встречи нужны для выявления трудностей, способных повлиять на завершение спринта.

Если спринт завершен, то команда показывает результаты своей работы на обзоре итогов (demo). С результатами может ознакомиться каждый участник проекта. Ознакомление следует проводить до того, как готовый код будет смерджен в рабочую среду.

Завершает цикл спринтов ретроспектива. На ней команда определяет области, которые нужно улучшить в будущем спринте.

На что нужно обратить внимание, а чего делать не стоит

Большинство молодых команд сталкиваются с трудностями, впервые внедряя спринты в свой рабочий процесс. Чтобы избежать проблем, рекомендуем изучить список действий, которые нуждаются в первоочередном внимании.

Что нужно делать:

- Проверьте, что команда осознает цель спринта и способ достижения успеха. Это необходимо, чтобы все вместе двигались к успешному результату.
- У вас должен быть четкий и понятный бэклог. Если бэклог велся неверно, это может стать проблемой, способной повредить рабочему процессу.
- Убедитесь, что ваша оценка темпов работы верна, с учетом летних отпусков и других факторов.
- Активно принимайте участие в планировании спринта. Поощряйте членов команды расширять план для историй, багов и заданий.
- Отказывайтесь от заданий, во время которых разработчикам не удастся решить вопросы с зависимостями.
- После утверждения плана назначьте сотрудника, на которого будет возложено внесение данных в программу для управления проектами (карточки Jira или др.).

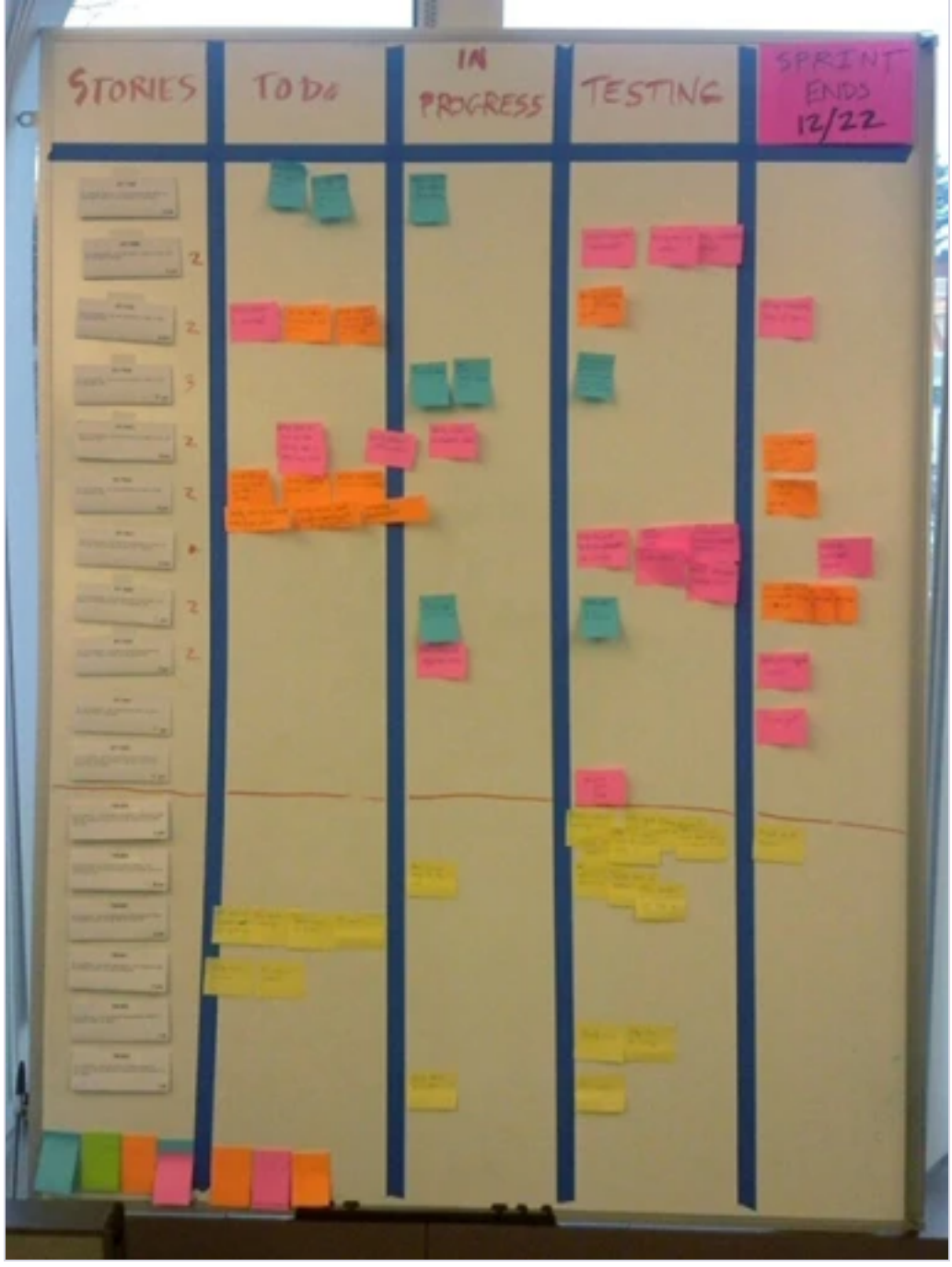
Чего стоит избегать:

- Не злоупотребляйте большим количеством историй, трезво оценивайте темпы работы и не назначайте задания, которые будет трудно выполнить на спринте.
- Помните о качестве работы. Проверьте, есть ли у вас достаточно времени для контроля качества и исправления ошибок в коде.
- Позаботьтесь, чтобы все члены команды четко понимали содержимое спринта. Не гонитесь за скоростью. Вся команда должна двигаться вместе.
- Не взваливайте на разработчиков лишний объем работы. Скоро будет еще один спринт.
- Если команда высказывает волнение по поводу нагрузки или дедлайна, вам стоит учесть ее мнение. Разберитесь с проблемами и откорректируйте их при необходимости.

Scrum board

Scrum-доска — это инструмент, который демонстрирует, как выполняется работа Scrum-команды. Отображать информацию на такой доске можно на бумаге, на стене или в электронном виде (JIRA, Trello).

Scrum-доска состоит не менее чем из трех столбцов: “сделать”, “в работе” и “готово”. Перед вами пример доски:



На Scrum-доске размещена вся информация из бэклога, ранее утвержденного на планировании. Как правило, карточки бизнес-задач закреплены на доске по приоритету сверху вниз. Можно разделить их на конкретные типы работ (работа над кодом, дизайн и другие).

После того, как часть работы завершена, карточку передвигают по доске в соседнюю колонку. Показать видимость прогресса работы команды помогает “оставшаяся работа” по дням на Burndown Chart.

Можно также использовать доску с флипчартами. На ней названия работ пишут на бумажных наклейках и прикрепляют их на доску. Как только работа закончена, стикеры перемещаются в другую колонку.

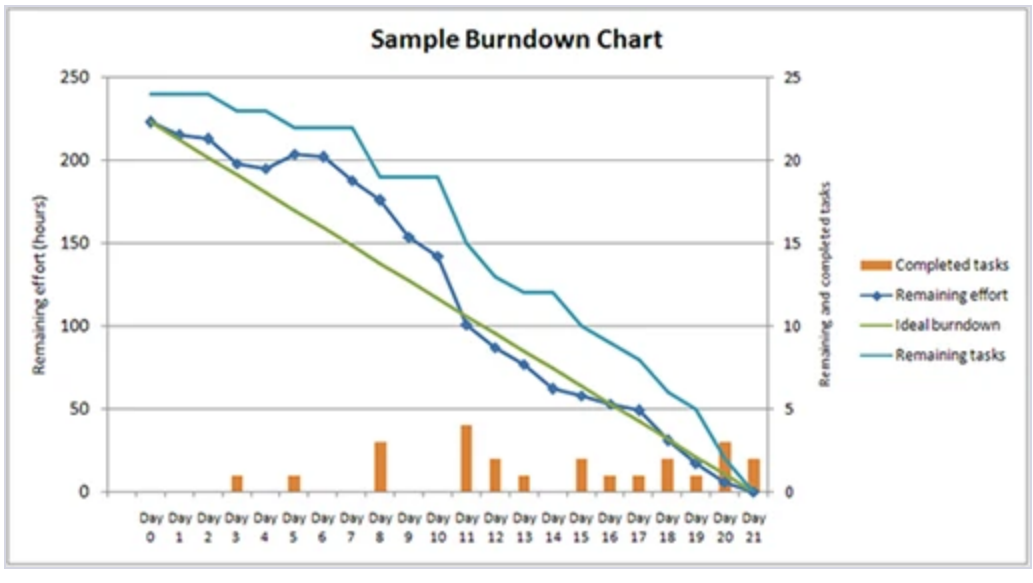
Burndown chart

Диаграмма сгорания задач (Burndown chart) показывает количество выполненной и оставшейся работы. Она обновляется каждый день и доступна всем заинтересованным лицам. График нужен для отображения прогресса в работе над спринтом.

Есть два вида диаграмм:

- Burndown chart с отображением прогресса работы в спринте.
- Burndown chart с отображением прогресса работы до выпуска продукта (данные суммируются из нескольких спринтов).

Пример диаграммы:



В этом примере используется психология: диаграмма показывает не число сделанных задач, а количество оставшихся (несделанных).

То есть, если команда сделали 90 задач из 100, то может возникнуть ложное ощущение, что уже все готово. Ведь прогресс с 90 до 100 задач особо ничего не меняет.

Если же отображать число оставшихся задач, то нельзя не заметить, как с каждым разом их становиться все меньше. Это подсознательно подстегивает участников проекта быстрее достичь цели — на доске не должно остаться недоделанных заданий.

‹ Предыдущая лекция

Следующая лекция ›

Комментарии (1) популярные **новые** старые

JavaCoder

Введите текст комментария

Сергей Уровень 79 8 ноября 2022, 16:08

Про scrum на ютубе

Ответить 0

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store