Техники тест-дизайна. Планирование, оценка трудозатрат. Отчетность

www.andersenlab.com

План

- Техники тест дизайна
- Планирование спринта
- Цель спринта
- Оценка трудозатрат
- Ретроспектива спринта, Sprint Review Meeting
- Отчётность
- Definition of Done
- Матрица соответствия требований

Тест дизайн

Тест-дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест-кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

Роли в тест дизайне:

- тест-аналитик - определяет "ЧТО тестировать?";
- тест-дизайнер - определяет "КАК тестировать?".

Техники тест дизайна

Техники черного ящика:

- Эквивалентное Разделение (Equivalence Partitioning - EP).
- Анализ Граничных Значений (Boundary Value Analysis - BVA).
- Тестирование: Таблицы Альтернатив (Decision Table Testing - DTT).
- Тестирование: Диаграмма Состояний и Переходов (State Transition Testing - STT).
- Тестирование: Сценарии использования (Use Case Testing - UCT).

Техники белого ящика:

- Тестирование и покрытие операторов (Statement Testing and Coverage - ST&C).
- Тестирование и покрытие условий (Decision Testing and Coverage - DT&C).

Техники, основанные на опыте:

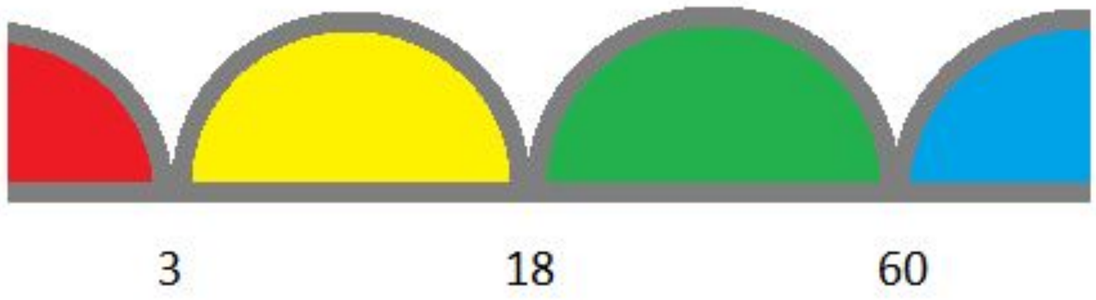
- Предположение об ошибках (Error Guessing - EG).
- Исследовательское тестирование (Exploratory Testing - ET).
- Тестирование на основе чек-листов (Checklist-based Testing).

Другие техники :

- Причина / Следствие (Cause/Effect - CE).
- Исчерпывающее тестирование (Exhaustive Testing – ET).
- Попарное / Парное тестирование (Pairwise Testing – PT).
- Инструмент: Mind Map.

Эквивалентное Разделение

Эквивалентное разбиение делит данные на группы (классы эквивалентности), которые, обрабатываются схожим образом. Области эквивалентности могут быть как для правильных, или позитивных, так и неправильных, или негативных, значений.



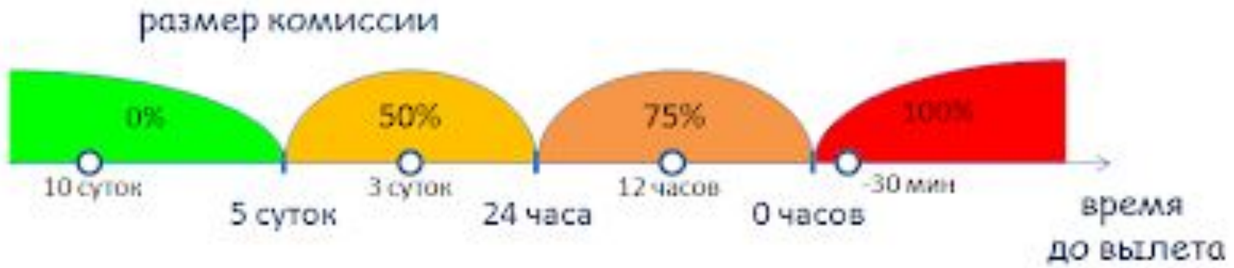
Эквивалентное разделение, алгоритм использования техники:

1. Необходимо определить класс эквивалентности. Это главный шаг техники. От него во многом зависит эффективность её применения.
2. Затем нужно выбрать одного представителя от каждого класса. На этом шаге из каждого эквивалентного набора тестов мы выбираем один тест.
3. Нужно выполнить тесты. На этом шаге мы выполняем тесты от каждого класса эквивалентности.

Пример:

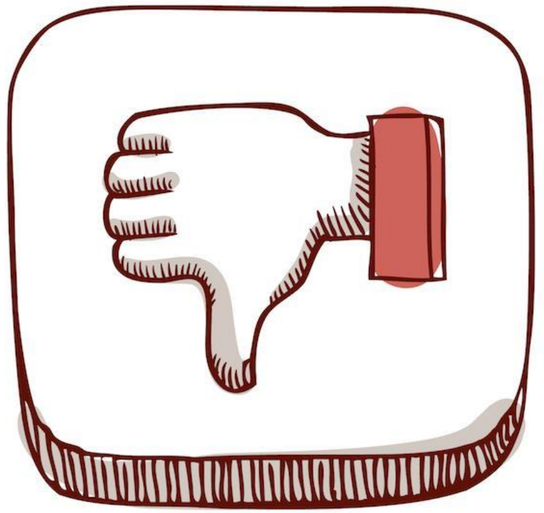
Функцию подсчета комиссии при отмене бронирования авиабилетов. Размер комиссии зависит от времени до вылета, когда совершена отмена:

- За 5 суток до вылета комиссия составляет 0%.
- Меньше 5 суток, но больше 24 часов – 50%.
- Меньше 24 часов, но до вылета – 75%.
- После вылета – 100%.



Плюсы и минусы техники анализа классов эквивалентности:

- К плюсам можно отнести заметное сокращение времени и улучшение структурированности тестирования.
- К минусам можно отнести то, что, при неправильном использовании техники, мы рискуем потерять баги.

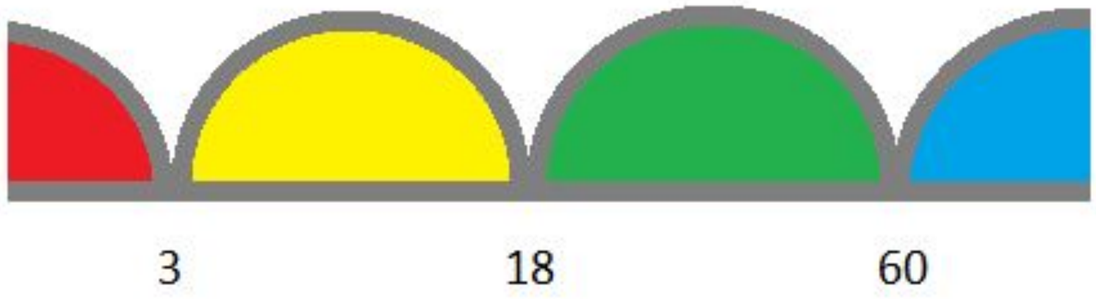


Задача: Определить классы эквивалентности

Существует некоторая система - которая принимает на вход численное значение и после анализа введенных данных, выдает ответ. Если ввести числа меньше 3, ответ системы - Отказано в правах доступа. Если ввести числа меньше 18 но больше 3, ответ системы - Права с ограниченным доступом. Если ввести числа меньше 60 но больше 18, ответ системы - Права доступа предоставлены. Если ввести числа больше 60, ответ системы - Права доступа предоставлены, супер пользователь!

Анализ Граничных Значений

Техника анализа граничных значений является продолжением метода эквивалентного разбиения, но может быть применима, только если классы состоят из упорядоченных числовых значений. Максимальное и минимальное значение класса являются его границами



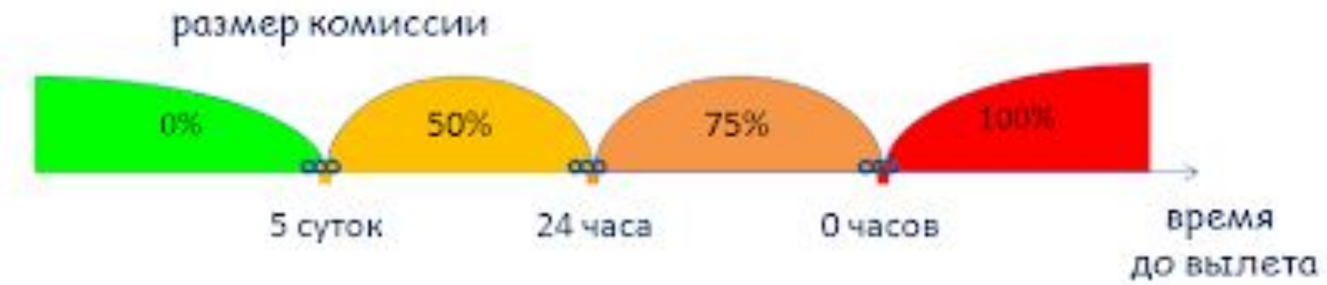
Примерный алгоритм использования техники анализа граничных значений:

- Во-первых, нужно выделить классы эквивалентности. Опять же, это очень важный шаг и от правильности разбиения на классы эквивалентности зависит эффективность тестов граничных значений.
- Далее нужно определить граничные значения этих классов.
- Нам нужно понять, к какому классу будет относиться каждая граница.
- Для каждой границы нам нужно провести тесты по проверке значения до границы, на границе, и сразу после границы.

Пример:

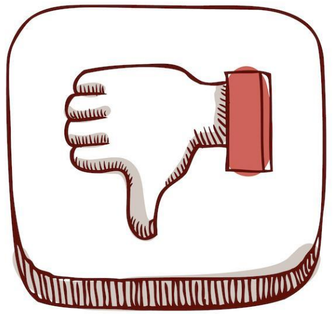
Функцию подсчета комиссии при отмене бронирования авиабилетов. Размер комиссии зависит от времени до вылета, когда совершена отмена:

- За 5 суток до вылета комиссия составляет 0%.
- Меньше 5 суток, но больше 24 часов – 50%.
- Меньше 24 часов, но до вылета – 75%..
- После вылета – 100%



Плюсы и минусы техники анализа граничных значений:

- Эта техника добавляет в технику анализа классов эквивалентности ориентированность на конкретный тип ошибок.
- То есть, техника анализа классов эквивалентности просто говорит нам о том, что нужно разбить все тесты на классы и провести тестирование всех классов. А техника граничных значений ориентирована на обнаружение конкретной проблемы – возникновения ошибок на границах классов эквивалентности.
- Но, как и для техники анализа классов эквивалентности, эффективность техники анализа граничных значений зависит от правильности ее использования. Мы должны приложить усилия, чтобы правильно определить классы эквивалентности и их границы. Если мы отнесемся к этому поверхностно, то рискуем пропустить ошибки.



Задача: Определить граничные значения

Существует некоторая система - которая принимает на вход численное значение и после анализа введенных данных, выдает ответ. Если ввести числа меньше 3, ответ системы - Отказано в правах доступа. Если ввести числа меньше 18 но больше 3 включительно, ответ системы - Права с ограниченным доступом. Если ввести числа меньше 60 но больше 18 включительно, ответ системы - Права доступа предоставлены. Если ввести числа 60 и больше, ответ системы - Права доступа предоставлены, супер пользователь!

Тестирование с Помощью Таблицы Альтернатив

Таблицы альтернатив – способ записи сложных бизнес-правил, которые должны быть реализованы в системе. В процессе создания таблицы, тестировщик определяет условия/сущности (входы) и результирующие действия системы (выходы). Пары условий/сущностей и действий образуют строки таблицы, при этом условия указываются сверху, а действия – снизу. Каждый столбец представляет собой бизнес-правило с уникальной комбинацией условий и действий, связанных с этим правилом.

Техники тест дизайна

	Правило 1	Правило 2	…	Правило p
Сущность				
Свойство-1				
Свойство-2				
…				
Свойство-m				
Действия				
Действие-1				
Действие-2				
…				
Действие-n				

Сущность (conditions) от 1 до m - это разные свойства системы, они представляют в таблице входные данные, которые можно ввести в систему. Действия (actions) от 1 до n - это действия которые могут произойти с указанной комбинацией сущностей, в зависимости от комбинации всех входных данных сущностей, действия принимают нужные значения.

Пример:

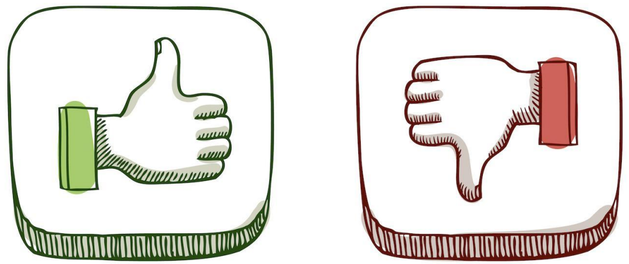
Представим, что тестируем приложение для страховой компании. Это приложение вычисляет скидку на страхование автомобилей, в зависимости от того, был ли водитель хорошим студентом и состоит ли он в браке. Как вычисляется скидка с помощью decision table:

	Правило 1	Правило 2	Правило 3	Правило 4
Сущность				
Состоит в браке?	Yes	Yes	No	No
Хороший студент?	Yes	No	Yes	No
Действия				
Скидка ($)	60	25	50	0

Эта таблица содержит все возможные комбинации значений сущностей

Плюсы и минусы техники:

- К плюсам можно отнести то, что она выявляет комбинации условий, которые могли быть не проверены при тестировании. Помогает определить несоответствия в требованиях, может быть применена во всех ситуациях и на любом уровне, где поведение программного обеспечения зависит от комбинации условий.
- К минусам можно отнести то, что, при неправильном использовании техники, мы рискуем потерять баги.



Задача: Оформить решение в виде таблицы решений

Существует некоторая система - которая вычисляет скидку в кино, в зависимости от того, человек в очках и/или с попкорном. Как вычисляется скидка с помощью decision table:

	Правило 1	Правило 2	Правило 3	Правило 4
Сущность				
Действия				
Скидка				

Тестирование с Помощью Диаграммы Состояний

Диаграмма состояний и переходов показывает начальное и конечное состояния системы, а также описывает переходы между состояниями. Каждый переход вызывается событием (например, вводом данных пользователем). Если одно и то же событие может привести к разным переходам, выбор перехода может задаваться контрольным условием. Смена состояния может завершаться выполнением какого-либо действия (вывод результатов или сообщения об ошибке и т.д.).

Техники тест дизайна

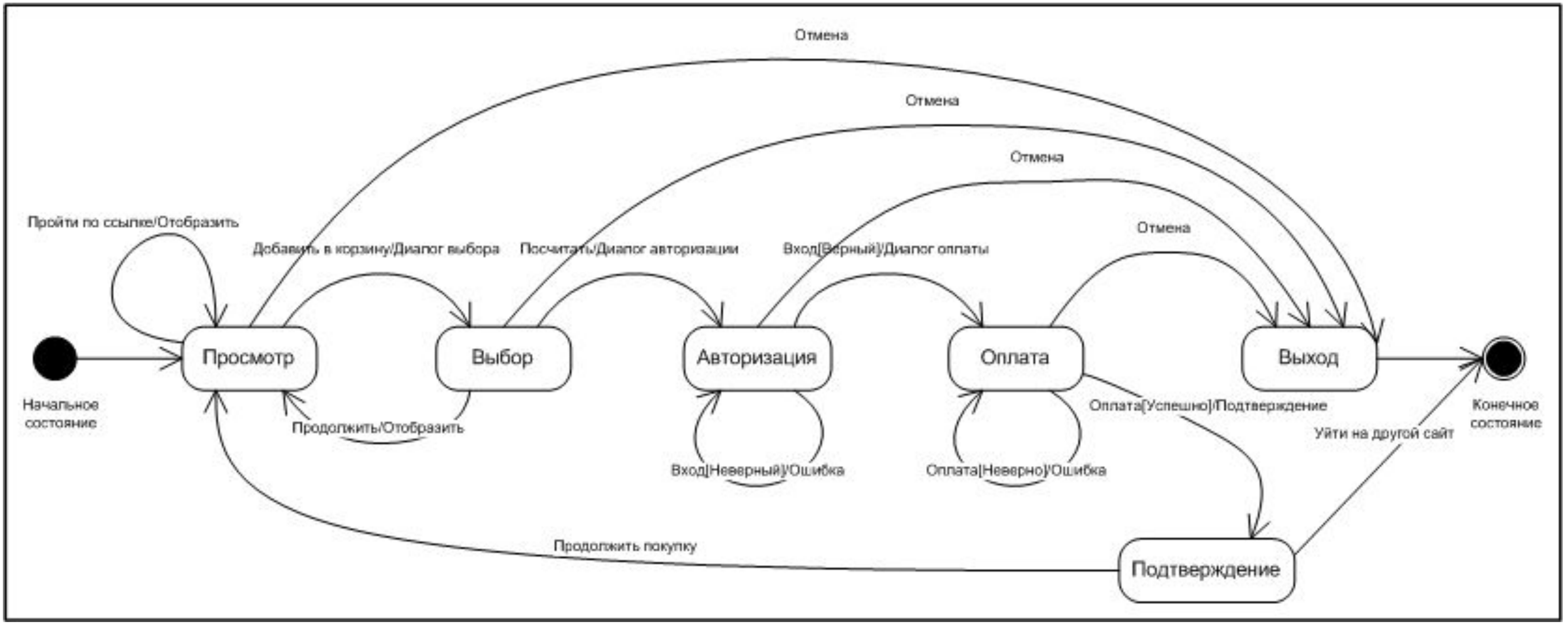
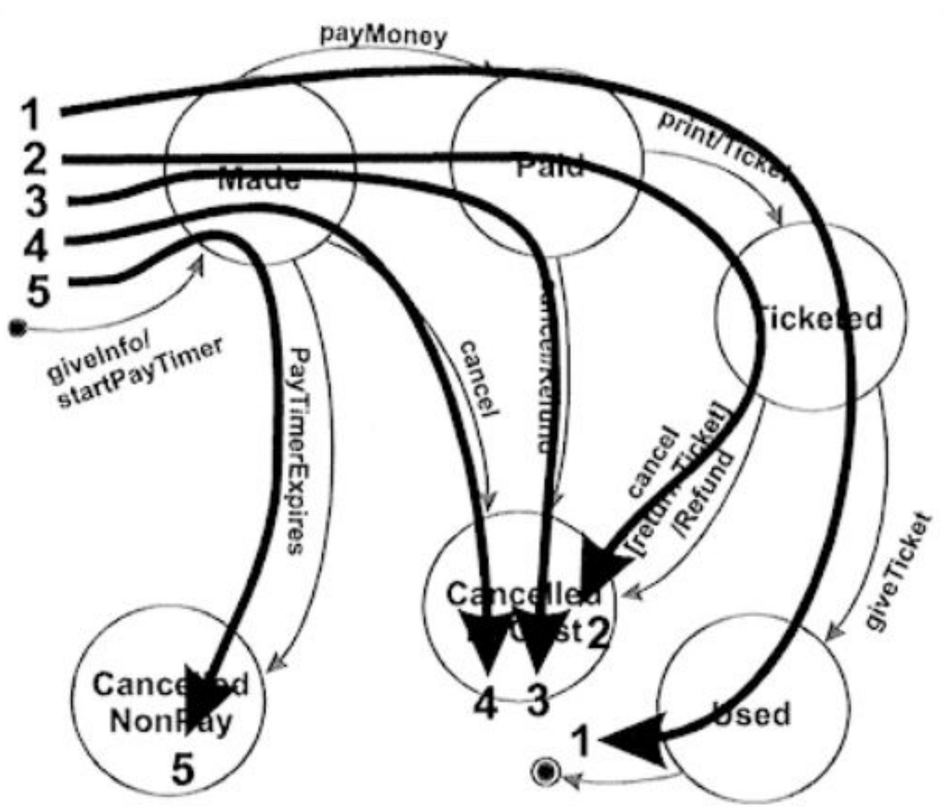


Таблица переходов представляет собой все возможные комбинации начальных и конечных состояний, включая действительные и недействительные переходы, инициирующие события, защитные условия и результирующие действия.

Диаграммы состояний и переходов обычно, показывают только действительные переходы и исключают недействительные переходы. Тесты создаются для покрытия типичной последовательности состояний, покрытия каждого возможного состояния, покрытия каждого возможного перехода, проверки специфических последовательностей переходов, или для проверки недействительных переходов.

State-Transition Diagrams могут быть легко использованы для создания тест кейсов. Необходимо создать набор тест-кейсов, который должен пройти по всем переходам хотя бы раз.



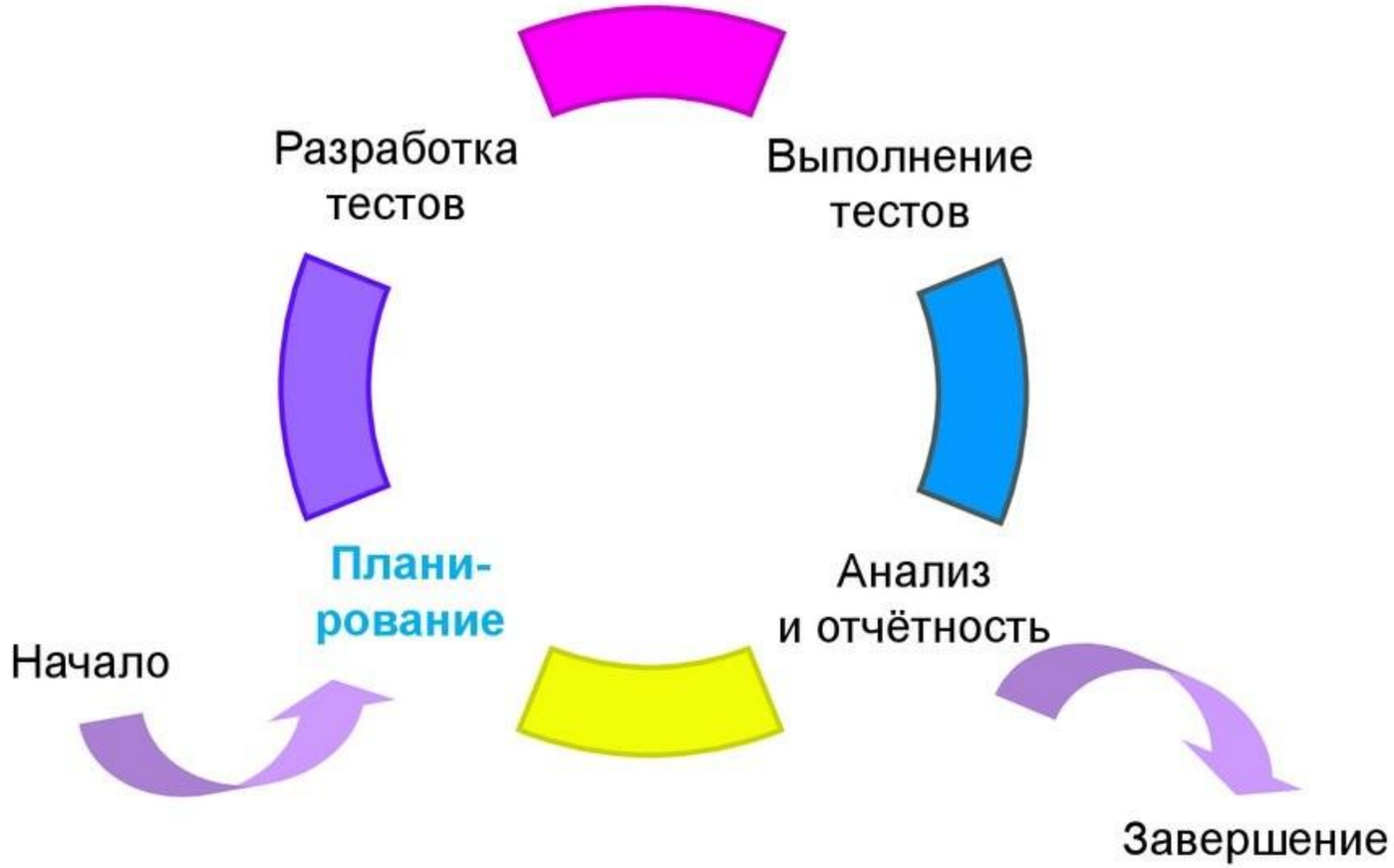
Подходы к оценке и измерению тестового покрытия:

Покрытие требований (Requirements Coverage) - оценка покрытия тестами функциональных и нефункциональных требований к продукту, путем построения матриц трассировки (traceability matrix).

Покрытие кода (Code Coverage) - оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.

Тестовое покрытие на базе анализа потока управления - оценка покрытия основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.

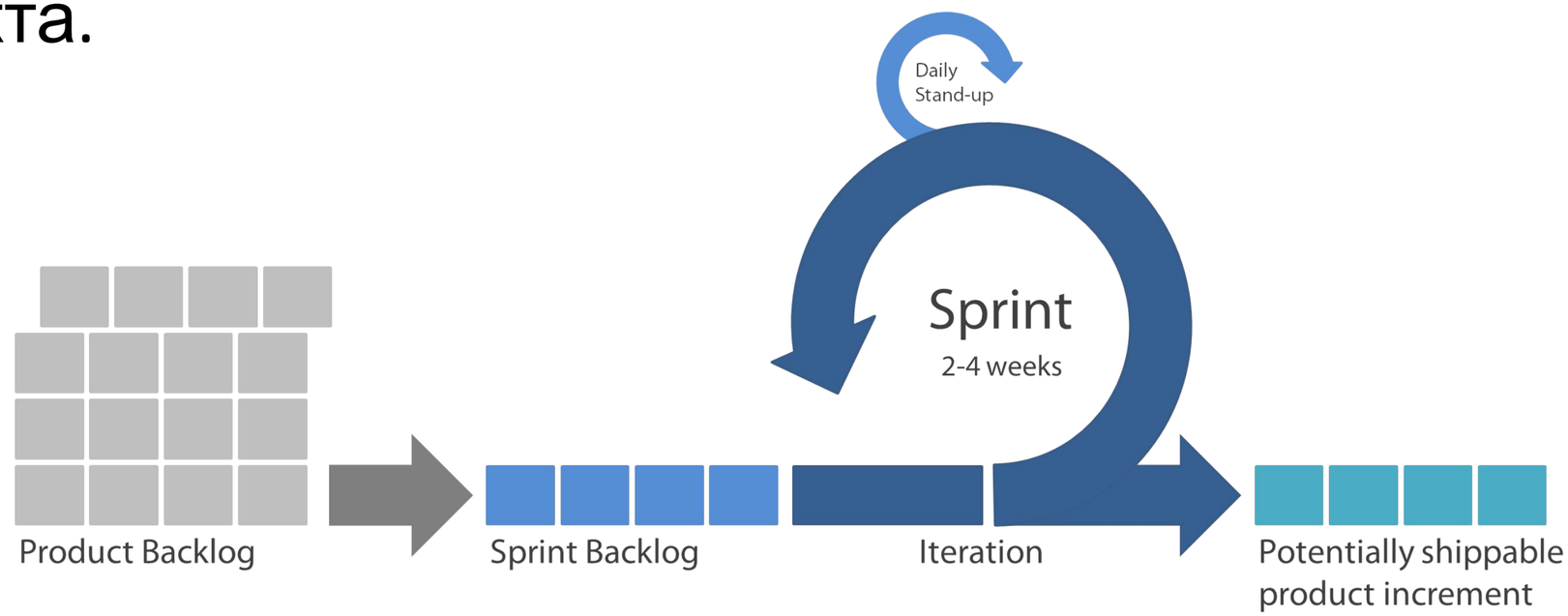
Жизненный цикл тестирования



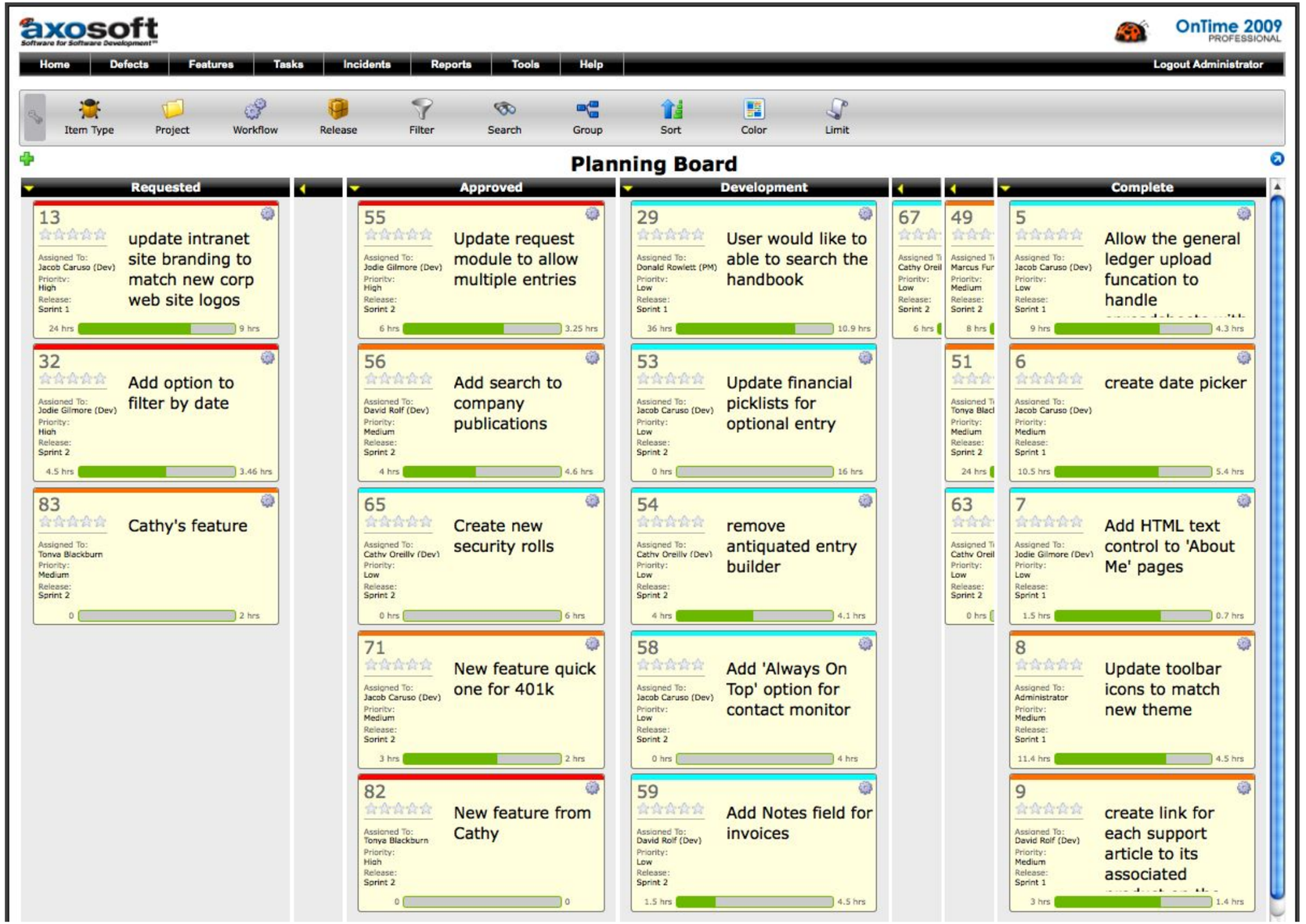
Планирование Спринта

По результатам планирования спринта скрам-команда решает:

- каким будет инкремент в конце спринта;
- как организовать работу, чтобы получить готовый инкремент продукта.

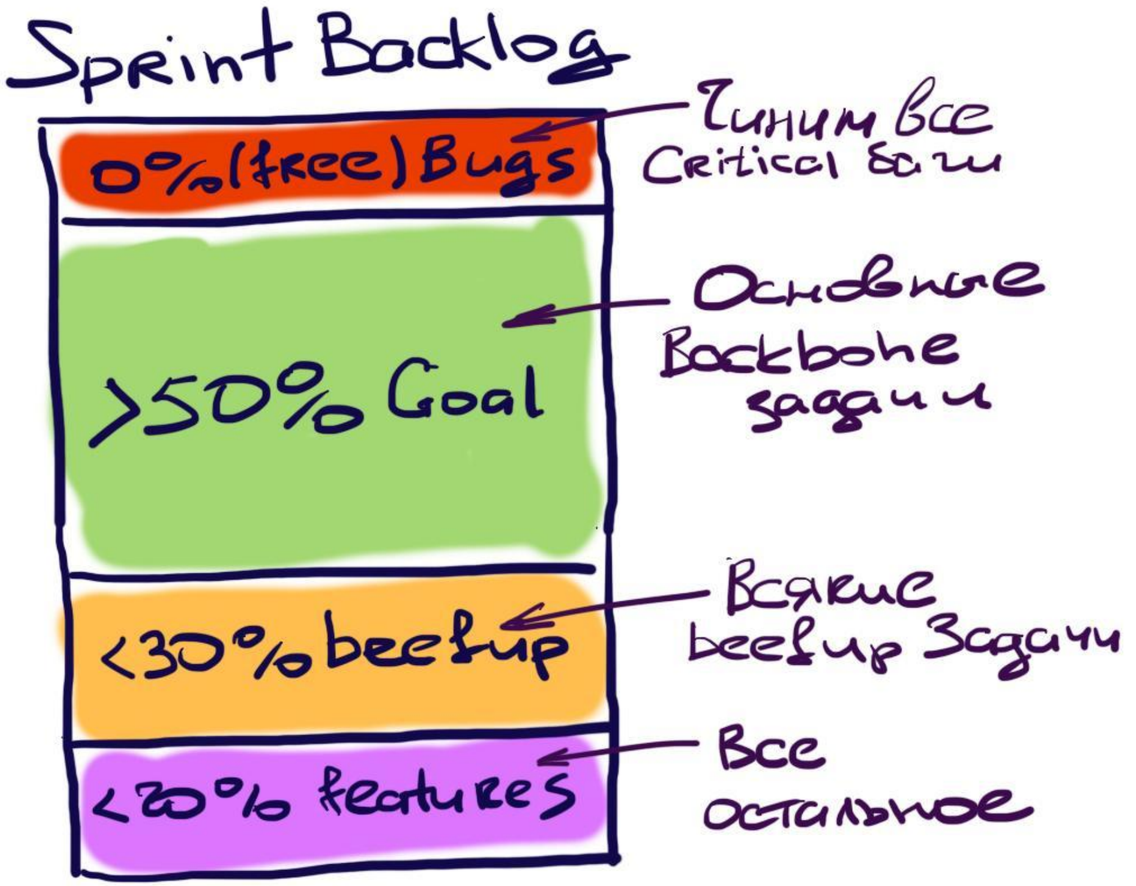


Планирование



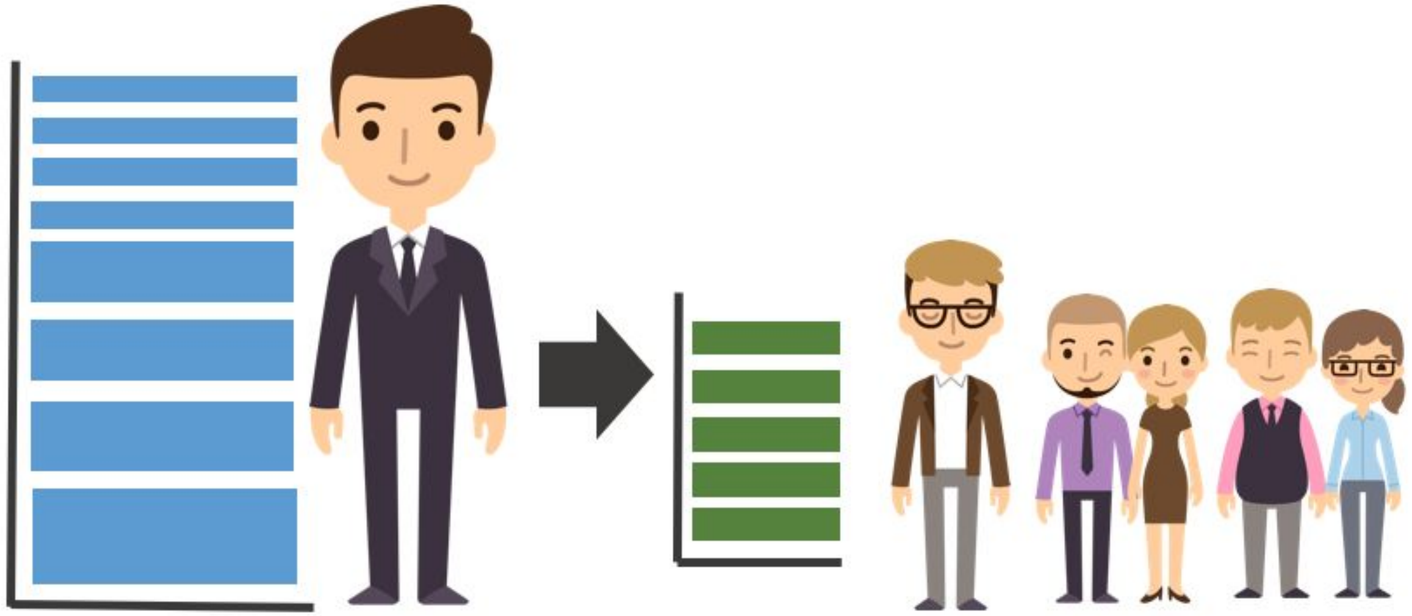
Цель Спринта

Цель Спринта – это установленный для спринта ориентир, который достигается через выполнение части бэклога продукта. Цель спринта формируется во время его планирования и объясняет команде разработки, для чего создается инкремент.



Планирование

Планирование спринта - это ограниченная по времени встреча в начале спринта, на которой команда и владелец продукта (ВП) обсуждают и принимают решение о том, какая работа будет завершена в спринте.



ЧАСТЫЕ ПРОБЛЕМЫ:

- Владелец продукта сам определяет и решает, какая работа будет завершена.
- Беклог продукта не актуален, не приоритезирован или не готов к обсуждению.
- В конце планирования все слишком детализировано и вся работа уже распределена по исполнителям (эту проблему трудно преодолеть).
- Никто не понимает, что означает статус "Готово".
- Встреча слишком длинная.
- Встреча не включает участников в процесс.
- Некоторым людям сложно проявляться.
- Неподходящая среда, команда не чувствует поддержки или безопасности.
- Нет доверия или уважения с обеих сторон.
- Команда не понимает, для чего нужна эта встреча.

Длительность встречи зависит от длины спринта, чем дольше спринт, тем больше времени нужно для его планирования. Для ориентира:

- Однонедельный спринт - 2 часа.
- Двухнедельный спринт - 4 часа.
- Спринт длиной в 1 месяц - 8 часов.

Длительность также очень зависит от зрелости и эффективности команды и владельца продукта, от объема предварительной подготовки.

Планирование

Задача встречи - сформулировать цель спринта. Ее можно представить в форме беклога спринта. Беклог спринта - список приоритезированных задач, которые команда берется завершить до конца спринта. Здесь важно помнить о командных критериях готовности (Definition of Done).

Производительность команды:

- Достаточно взять среднее последних 3 спринтов, как руководство.
- Обсудите часы доступности команды, отпуска, режим работы членов команды.
- Помните, что спринты не бывают одинаковыми!
- Не пытайтесь быть слишком детальными - это бесполезная трата времени, т.к. количество неизвестных слишком велико.
- Команда все равно согласовывает объем работы.
- Оставьте некоторое время для решения пока неизвестных вопросов и проблем. Так команда получает больше свободы действия.
- Проще добавить работу в спринт, если у вас хорошо проработан беклог, чем убрать ее.

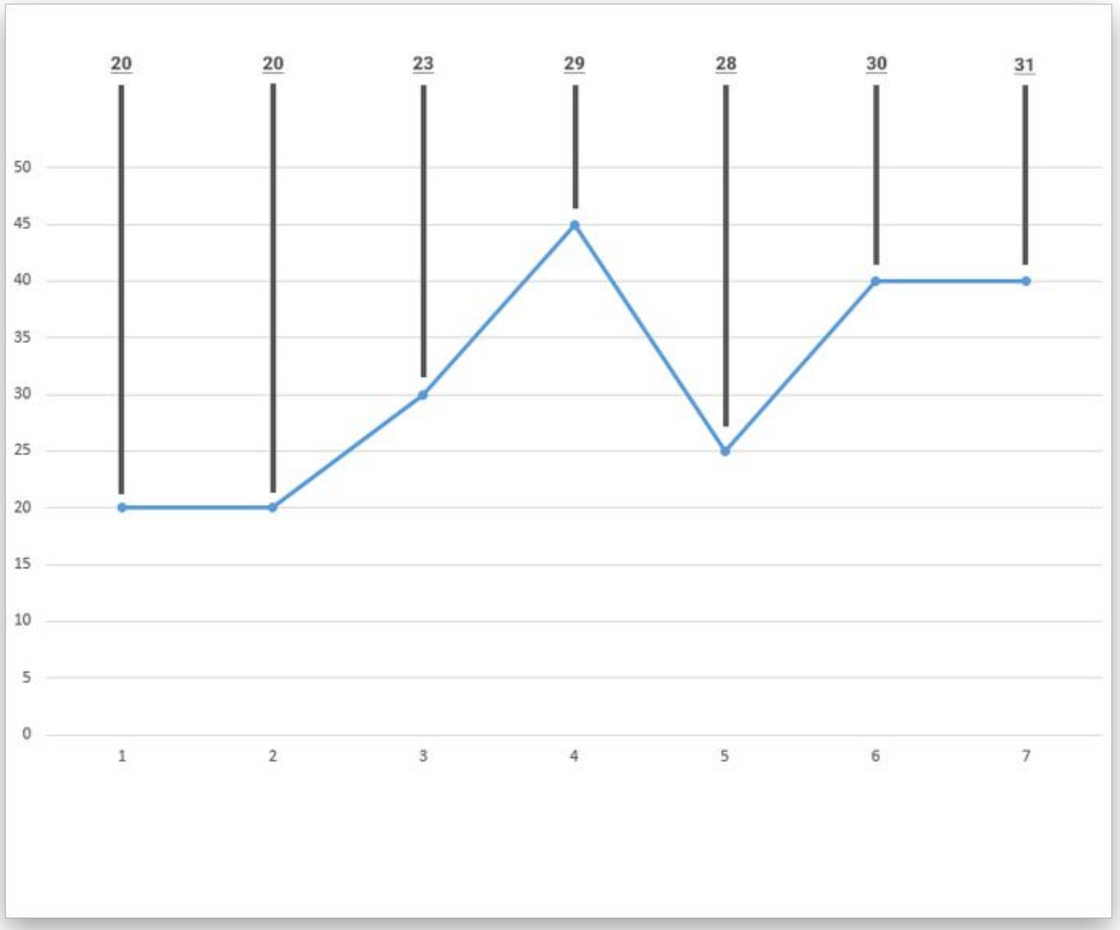
Capacity(Ёмкость):

- Capacity прогноз - количество идеальных часов, доступное в следующем спринте.
- Понимание, сколько часов у нас есть на работу: на написание кода, тестирование, т.д.
- Как правило, участник проекта работает не более пяти часов в день.
- Эффективное распределение задач.
- Нет смысла планировать задачи на тех, кто будет в отпуске или занят другими активностями.
- Мало пользы принесет технический анализ задачи, выполненный участником проекта, который в следующем спринте будет отсутствовать.
- Аккуратное и точное планирование.
- Мы оцениваем задачи в часах и берем в спринт столько, сколько соответствует нашей capacity.

Velocity **Scrum**

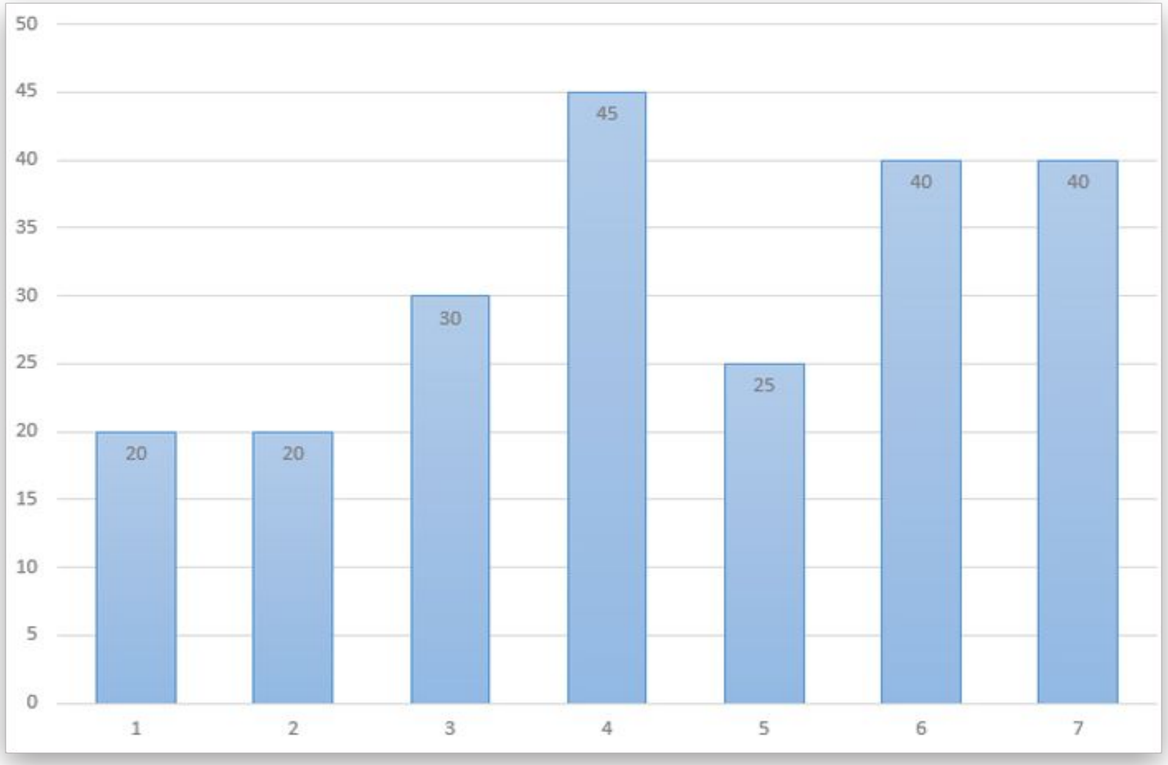
Как и в движении на автомобиле, скорость можно измерять и в Scrum, и называется это Velocity (скорость). Расчет Scrum Velocity очень простой и также состоит из поставленных отметок, как через каждые 60 минут было какое-то количество километров.

График Velocity, отображающий по горизонтальной оси количество Sprints, а по вертикальной - Story Points.



Планирование

В таком графике, по сути, изображено Story Points и на основе этих показателей выстраивается среднее значение скорости. Однако, график Velocity может быть и иной, с простым отображением Story Points, на основе которых визуально видна тенденция.



Трудозатраты - количество рабочего времени, необходимого для выполнения работы (выражается в человеко-часах).

Перед выполнением каждого задания, возникают следующие вопросы:

- Как много времени понадобится на выполнение работы?
- Когда всё будет готово?
- Можно ли гарантированно выполнить работу к такому-то сроку?
- Каковы наиболее оптимистичный и пессимистичный прогнозы по времени?

Основные принципы оценки:

- Любая оценка лучше её отсутствия.
- Оптимизм губителен.
- Оценка должна быть аргументирована.
- Простой способ научиться оценивать — оценивать.

Алгоритм обучения формированию оценки:

- Сформируйте оценку.
- Запишите полученную оценку.
- Выполните работу.
- Сверьте реальные результаты с ранее сформированной оценкой.
- Учтите ошибки при формировании новых оценок.
- Повторяйте этот алгоритм как можно чаще для самых различных областей жизни.

Полезные идеи по формированию оценки трудозатрат:

- добавляйте небольшой «буфер» (по времени, бюджету или иным критическим ресурсам) на непредвиденные обстоятельства;
- выясните свой «коэффициент искажения»;
- принимайте во внимание не зависящие от Вас обстоятельства;
- задумывайтесь заранее о необходимых ресурсах;
- ищите способы организовать параллельное выполнение задач;
- периодически сверяйтесь с планом, вносите корректировки в оценку и уведомляйте заинтересованных лиц о внесённых изменениях заблаговременно;

Оценка трудозатрат

Структурная декомпозиция - иерархическая декомпозиция объёмных задач на всё более и более малые подзадачи с целью упрощения оценки, планирования и мониторинга выполнения работы.

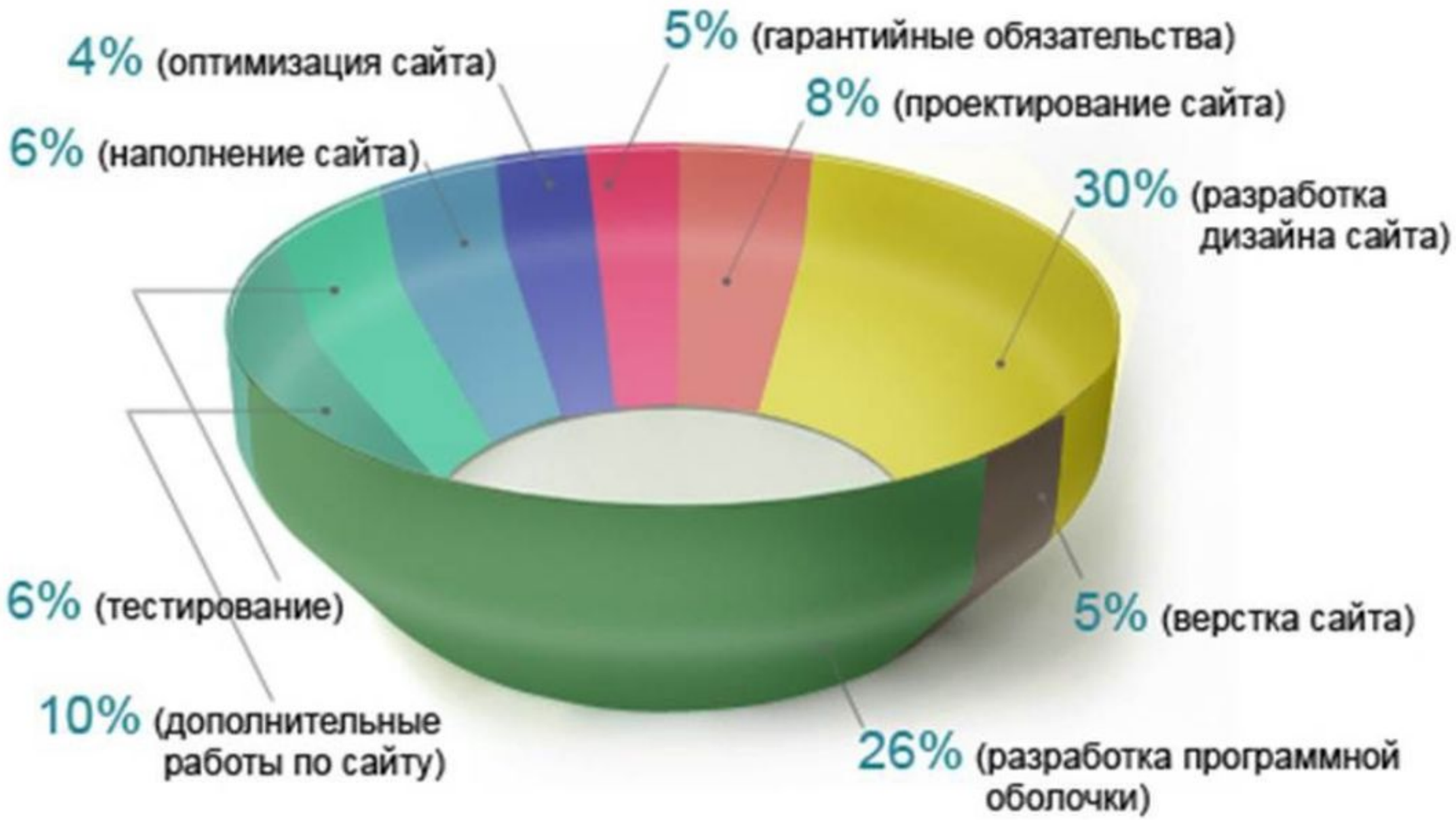
Использование структурной декомпозиции позволяет:

- описать весь объём работ с точностью, достаточной для чёткого понимания сути задач, формирования достаточно точной оценки трудозатрат и выработки показателей достижения результатов.
- определить весь объём трудозатрат как сумму трудозатрат по отдельным задачам (с учётом необходимых поправок).
- от интуитивного представления перейти к конкретному перечню отдельных действий, что упрощает построение плана, принятие решений о распараллеливании работ и т.д.

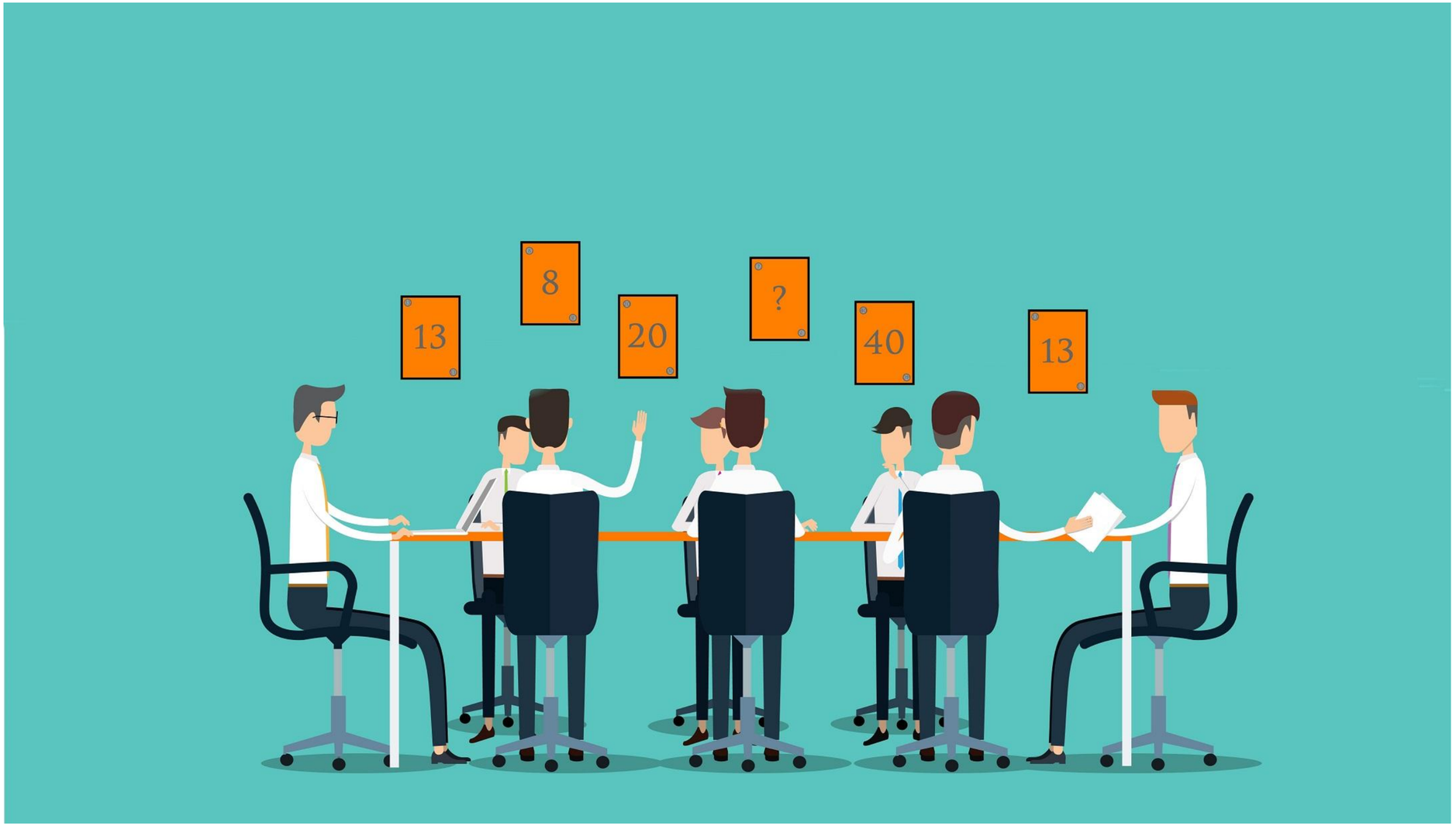
Другие методы:

- Метод «пальцем в небо»;
- Экспертная оценка;
- Специальный метод;
- Структура декомпозиции работ;
- Метод Дельфи;
- Метод определения трудозатрат в процентном отношении к разработке;
- Метод процентного распределения.

Оценка трудозатрат



Planning Poker (Scrum Poker)



Story Points

Одна из самых важных сторон методологии Scrum – так называемые Story Points. Эта сторона очень плотно интегрирована в Scrum совместно с технологией Planning Poker.

Название	Длительность	Story Points
Добавить веб-форму на сайт	4 часа	5
Оптимизация картинок для сайта	4 часа	5
Добавить ссылки социальных сетей на главную	2 часа	3
Добавить ALT для картинок согласно запросам	2 часа	3
Добавить Google Map на сайт	1 час	2

Задача: Оценить время на выполнение тестирования, задачу про ответы системы в зависимости от возраста

Метод маек:

xs - 0,5 дня

s - 1 день

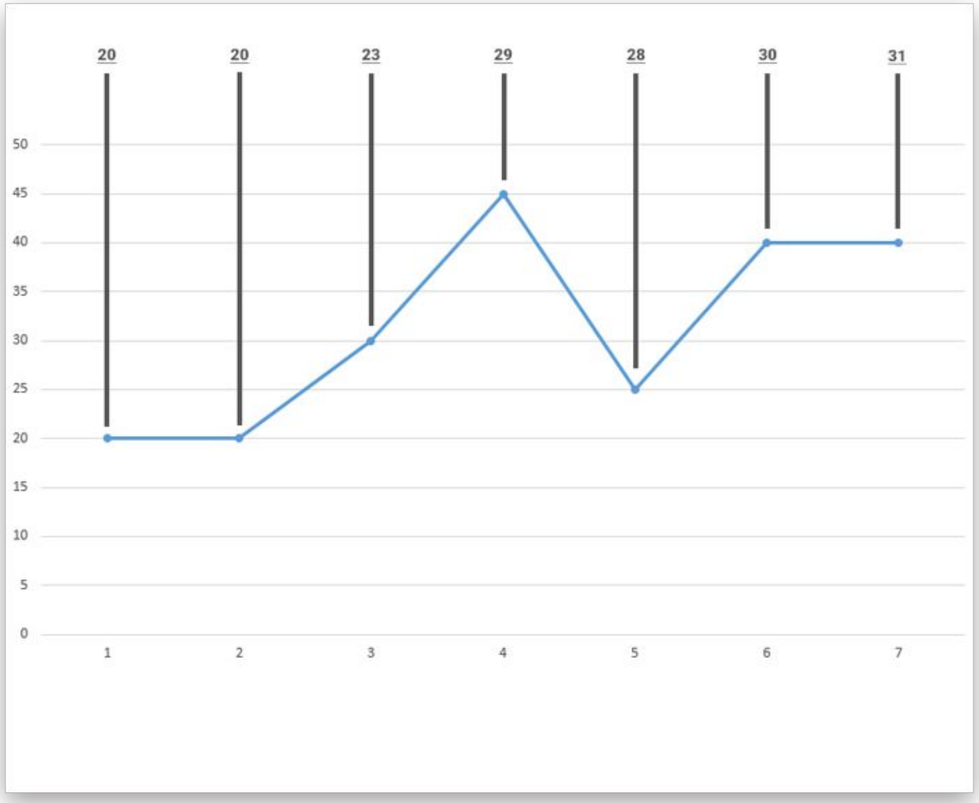
m - 1,5 дня

l - 2 дня

xl - 2,5 дня

Оценка трудозатрат

Когда Scrum Team способна оценивать свою работу, ведет график Velocity, следит за Диаграммой сгорания задач, рано или поздно (или с самого начала работы) абсолютно все оценки будут вестись в Story Points.



Ретроспектива спринта

A Typical Sprint Retrospective Model

What worked well?

What could be improved?

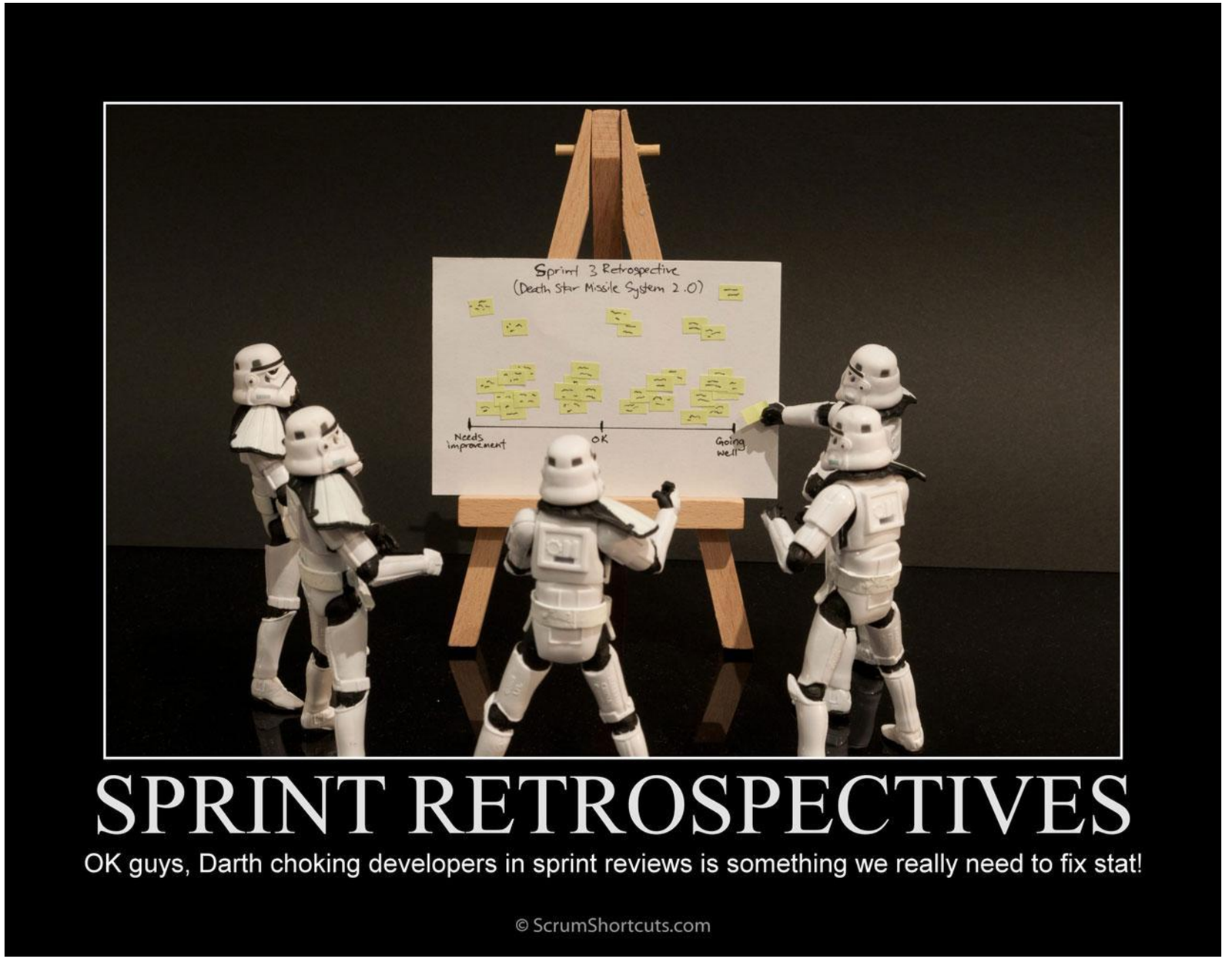
What will we commit to doing in the next Sprint?

Scrum Team members make actionable commitments

Цели проведения ретроспективы спринта:

- инспекция прошедшего спринта применительно к людям, отношениям, процессам и инструментам. Обнаружение и упорядочение того, что прошло хорошо и того, что нуждается в улучшении;
- создание плана внедрения улучшений в процесс работы скрам-команды.

Ретроспектива спринта



Sprint Review Meeting

В ходе **Sprint Review Meeting** проект оценивается в отношении цели спринта, которая была определена во время планирования. В идеале, команда выполнила все задачи из Product Backlog, помещенные в Sprint, но это не самое важное, а важно то, что была достигнута цель спринта.

На что смотрит заказчик на Sprint Review Meeting:

- Команда передала законченный продукт.
- Работа команды завершена.
- Показатели проекта (завершенность кода и тд.).
- Работоспособность выполненных задач.
- Обзор приоритетов (для следующих итераций / спринтов).

Отчётность

Отчётность - сбор и распространение информации о результатах работы (включая текущий статус, оценку прогресса и прогноз развития ситуации).

К высокоуровневым задачам отчётности относятся:

- Сбор, агрегация и предоставление в удобной для восприятия форме объективной информации о результатах работы.
- Формирование оценки текущего статуса и прогресса (в сравнении с планом).
- Обозначение существующих и возможных проблем (если такие есть).
- Формирование прогноза развития ситуации и фиксация рекомендаций по устранению проблем и повышению эффективности работы.

Отчётность

Отчёт о результатах тестирования включает следующие разделы:

- Краткое описание.
- Команда тестировщиков.
- Описание процесса тестирования.
- Расписание.
- Статистика по новым дефектам.
- Список новых дефектов.
- Статистика по всем дефектам.
- Рекомендации.
- Приложения. Фактические данные (как правило, значения метрик и графическое представление их изменения во времени).

Definition of Done

Definition of Done — это набор критериев, которые позволяют понять, сделано ли то, что было целью разработки. Формат Definition of Done может быть любым, но чаще всего это простой список с перечнем активностей, которые должны быть успешно завершены, чтобы функционал мог считаться готовым.

Например:

- код написан;
- юнит-тесты написаны и успешно выполнены;
- код прошел ревью;
- документация обновлена;
- функциональное тестирование успешно завершено;
- регрессионное тестирование успешно завершено.

Чем больше и объемнее Definition of Done, тем более строгим оно считается. И тем больше нужно времени и усилий, чтобы наш функционал добрался до желаемого состояния «сделано». И тем выше вероятность, что функционал будет работать в соответствии с ожиданиями бизнеса.

Баланс между потерями на первое и вероятностью второго — это беспощадное поле брани, на котором было сломано много копий, команд, проектов, продуктов и сервисов.

Диаграмма сгорания

В большинстве своем мы привыкли к графикам идущими вверх, что означает положительную динамику, однако они могут идти и вниз, и также показывать положительную динамику.

Одним из таких ярких примеров является Диаграмма сгорания задач (Burndown chart). Само сочетание Burn Down дословно переводится как «гореть вниз» и действительно это так.

Данный график является основным средством для отслеживания выполненных задач в спринте или во всем проекте. Хотя по сути он может использоваться как угодно, но мы его рассматриваем внутри методологии Scrum.

Диаграмма сгорания

Синим на диаграмме сгорания отмечена идеальная линия выполнения задач, на которую и следует опираться.

Красным отмечена реальная история выполнения задач.

- **По шкале Y** отмечают количество запланированных баллов (в данном случае), идеальные часы, количество задач и так далее.
- **По шкале X** отмечают количество дней до окончания Sprint.

Как может показаться на первый взгляд – данная Диаграмма сгорания задач / Burndown chart служит всего лишь для самоконтроля и самоотчета, однако ее использование может рассказать об очень многом.

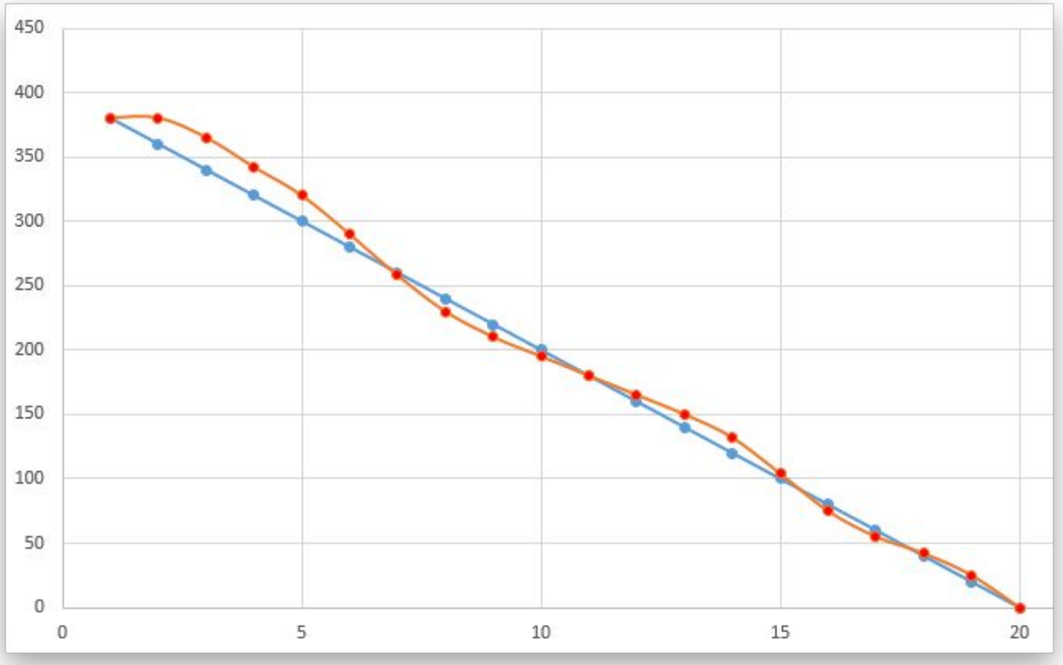
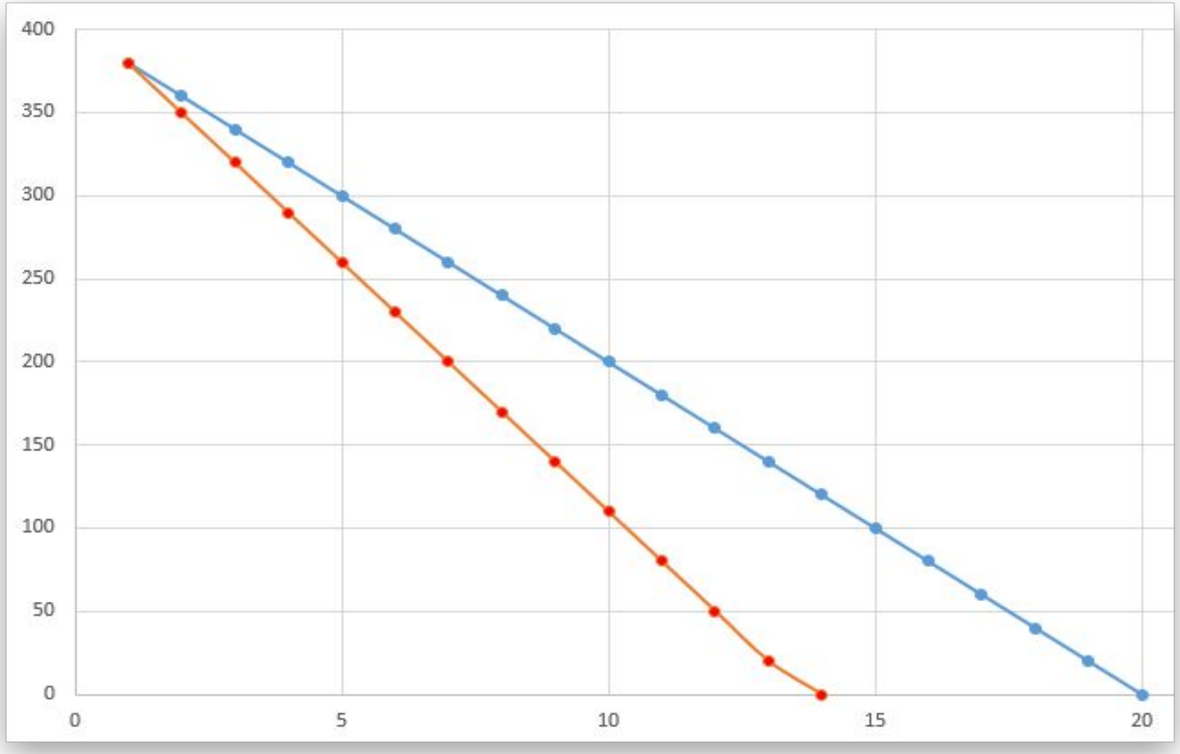


Диаграмма сгорания

Burndown Chart: Слишком рано

По Диаграмме сгорания задач / Burndown chart отчетливо видно, что команда все задачи выполнила раньше срока. Такая ситуация тоже не является позитивной, так как это означает ряд совершенных проблем:

- Команда сделала неправильную оценку предстоящей работы.
- В случае быстрого выполнения задач, разработчики не добавляли задачи из следующего спринта.
- Команда сильно перестраховалась, включив изначально дополнительный срок.
- В случае такой проблемы, чаще всего Scrum Master спрашивает команду о возможности добавления дополнительных задач из Product Backlog.



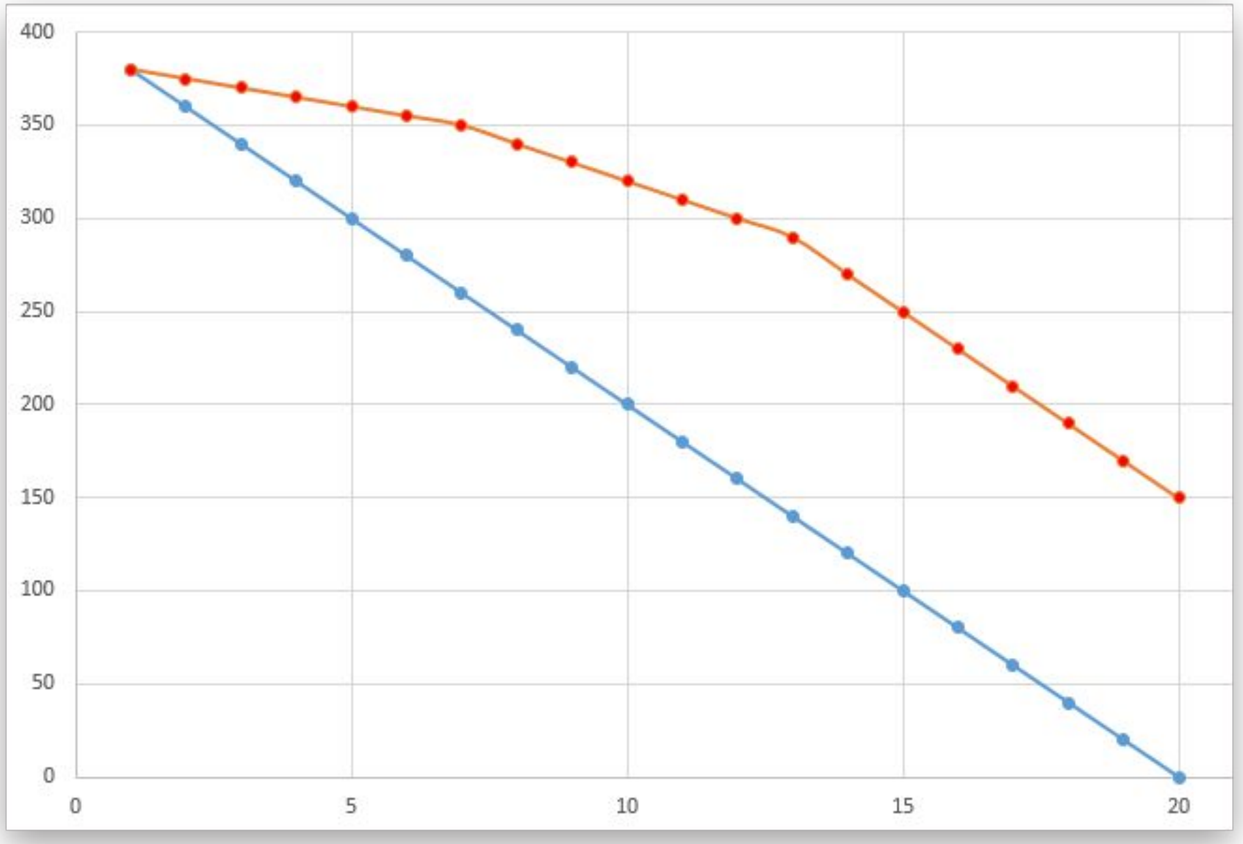
Burndown Chart: Опоздали

Также один из видов негативных диаграмм сгорания задач.

Одной из возможных причин здесь может быть постоянное добавление новых задач во время спринта, что увеличило нагрузку.

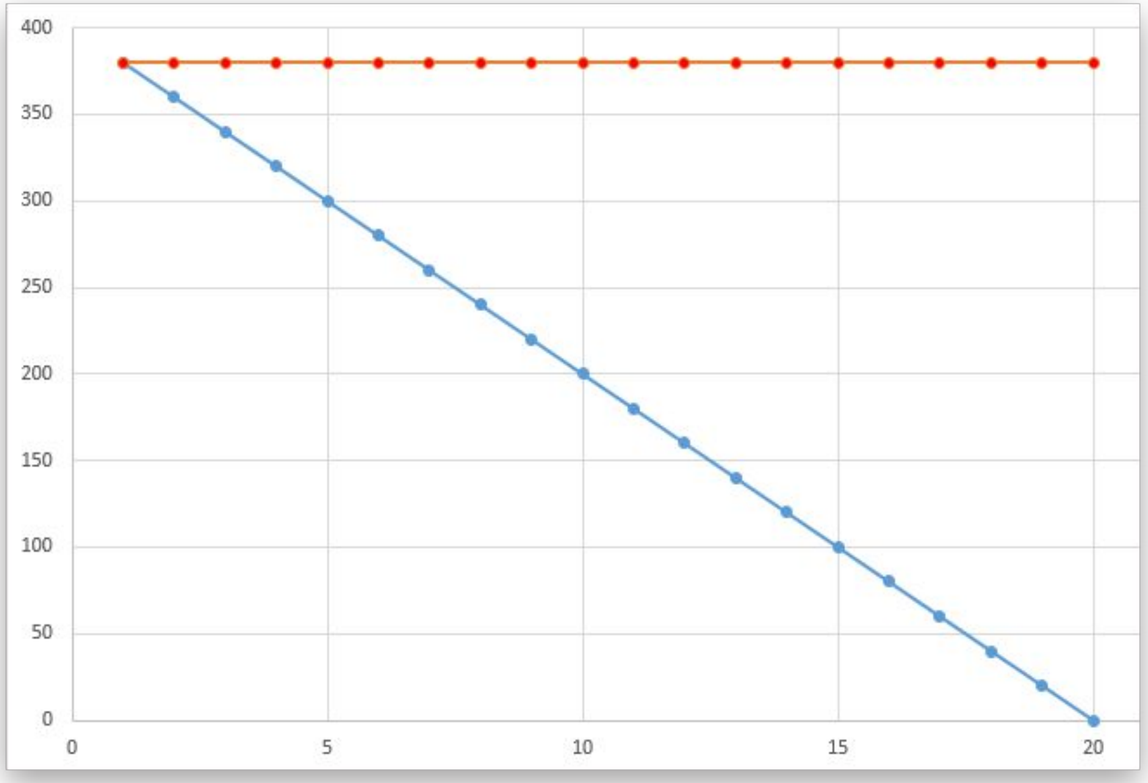
Второй частой проблемой является недоделанность задач, когда задачи сделаны наполовину. Такие задачи, как выразился Джефф Сазерленд – являются хламом.

В такой ситуации, на Daily Scrum Meeting обязательно нужно говорить о проблемах, мешающих идти к цели ровной дорогой. Как только линия реальных задач пошла выше, сразу надо решать проблему – это также один из постулатов методологии Scrum.



Burndown Chart: Без оценок

Может быть даже команда и работала, только забыла или не захотела использовать Диаграмму сжигания задач, что является прямо сказать дурным тоном и противоречит эффективной работе.

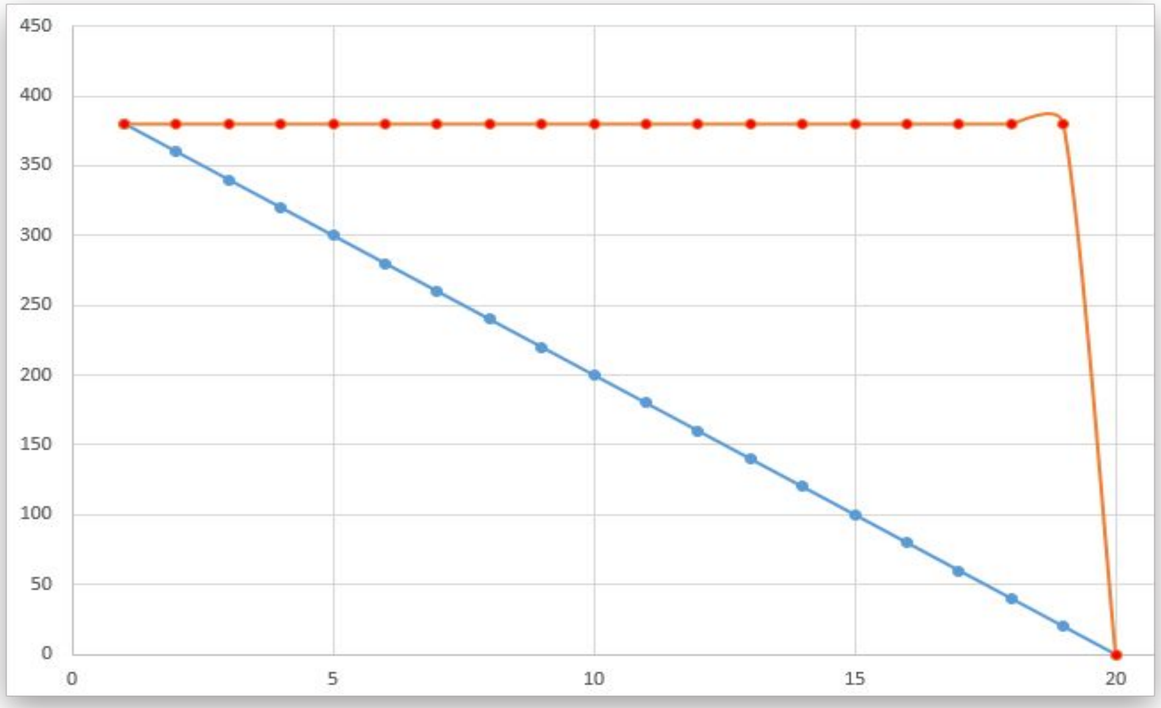


Команда не может контролировать себя, не может совершенствоваться и так далее.

Burndown Chart: Конечная оценка

Собственно, ситуация равна предыдущей. Не смотря на законченный Sprint, все итоговые оценки были внесены в диаграмму сгорания в самый последний день после завершения работы. Это равносильно тому, когда вообще законченные задачи не вносятся.

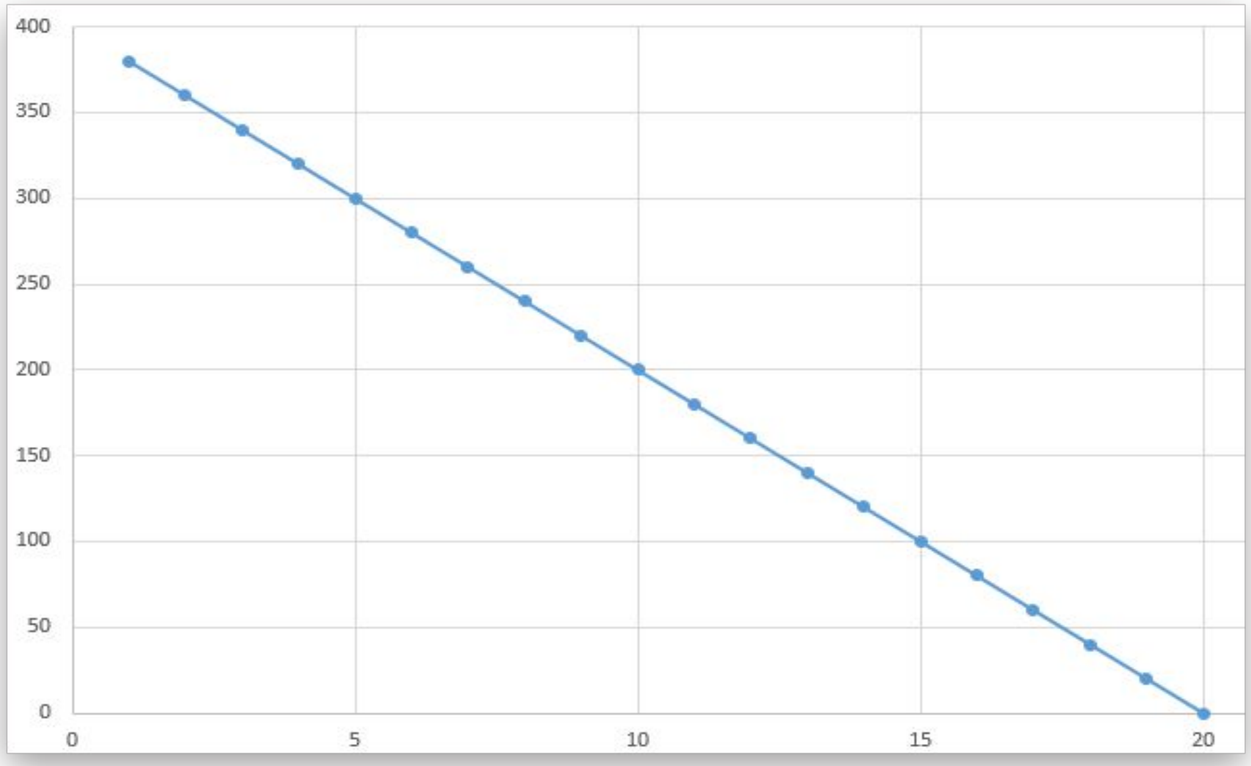
По данному графику невозможно сделать вывода о правильности работы команды, и даже более того, можно предположить, что команда не стремится к развитию.



Burndown Chart: Zero

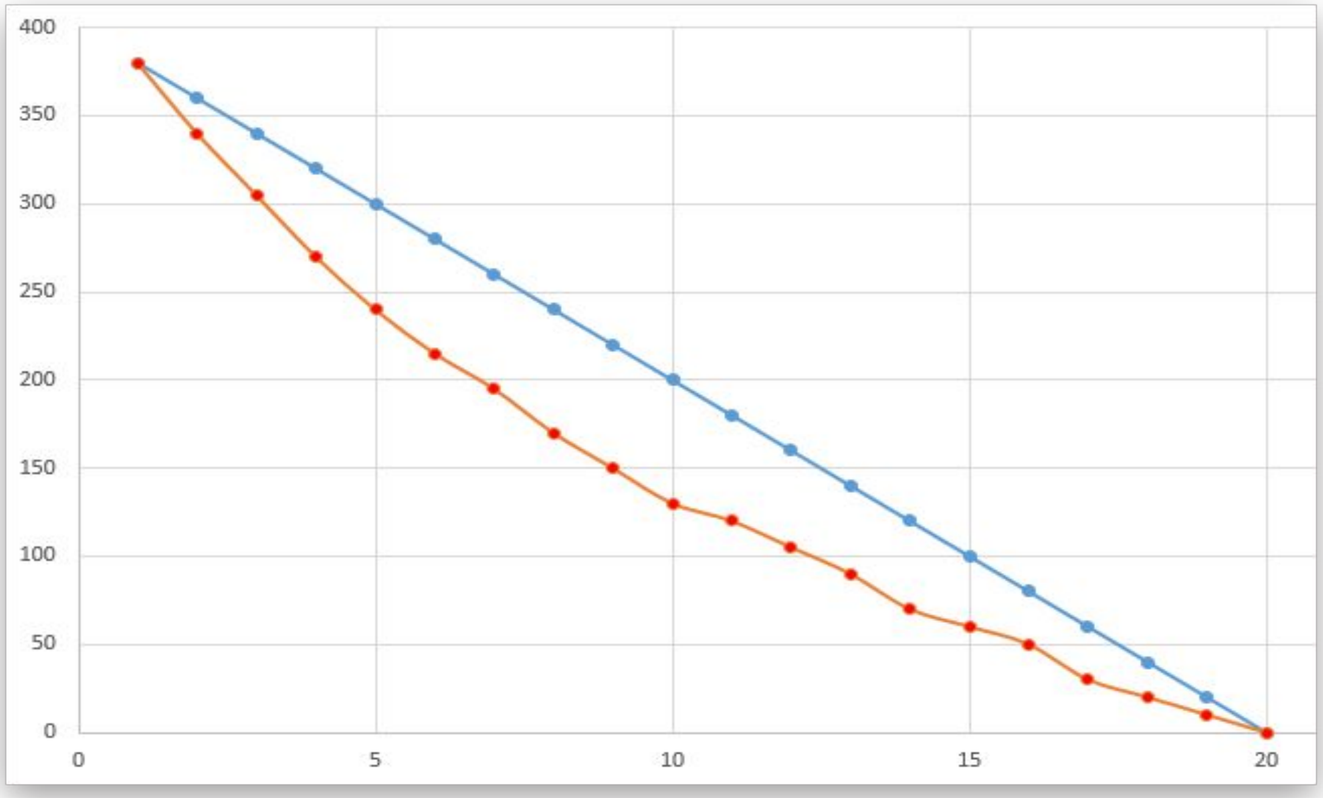
Отсутствие показателя реальных задач в диаграмме не является поводом считать, что работа не производилась, ведь она могла быть просто не оценена.

Как и в предыдущих пунктах, такая позиция не позволяет контролировать работу собственной команды и совершенствоваться.



Burndown Chart:
Релаксирующая команда

Этот пример диаграммы сгорания задач уже значительно лучше нежели другие, ведь в нем можно увидеть, как усовершенствовать команду. Возможные проблемы здесь такие же, как и в пункте «Слишком рано», но Scrum Team решили не заканчивать Sprint раньше, а более расслаблено продолжить работу, что также является ошибкой.

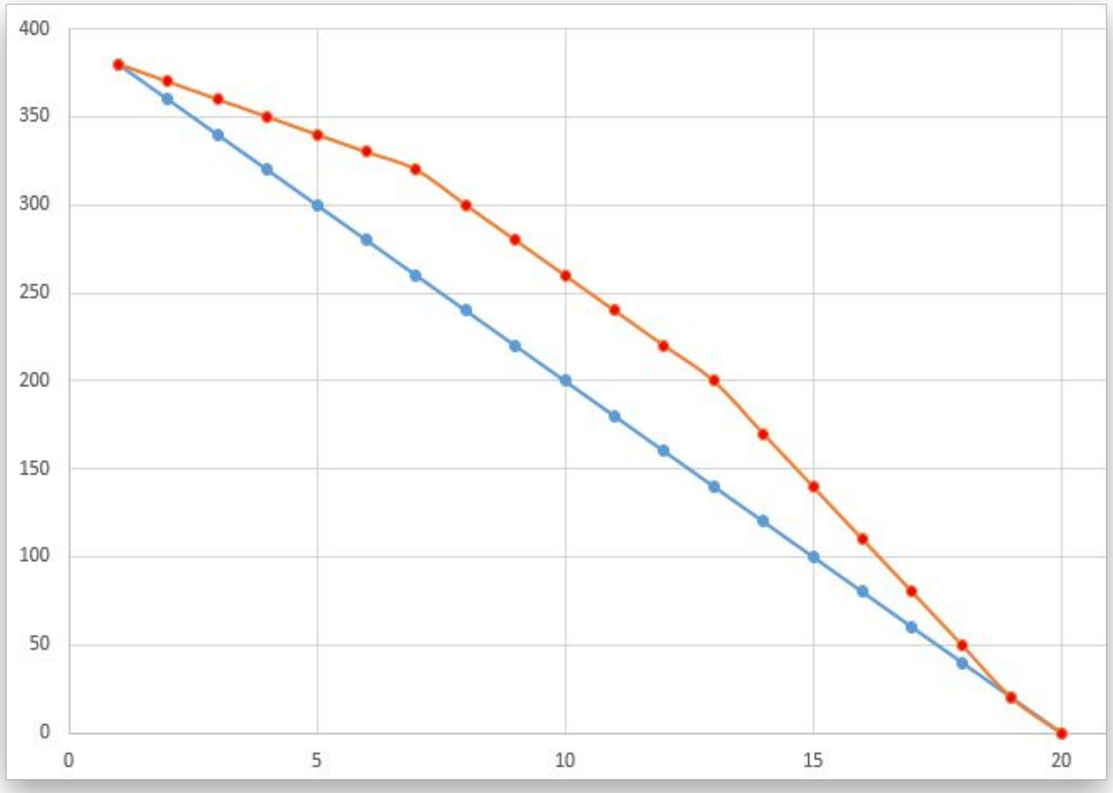


Burndown Chart: Совершенствование

Scrum Team на текущих показателях выглядит достаточно хорошо. По линиям видно, что в самом начале были трудности, но вовремя Daily Scrum Meeting все вопросы вскрывались и Scrum Master исправлял работу ведя команду к цели.

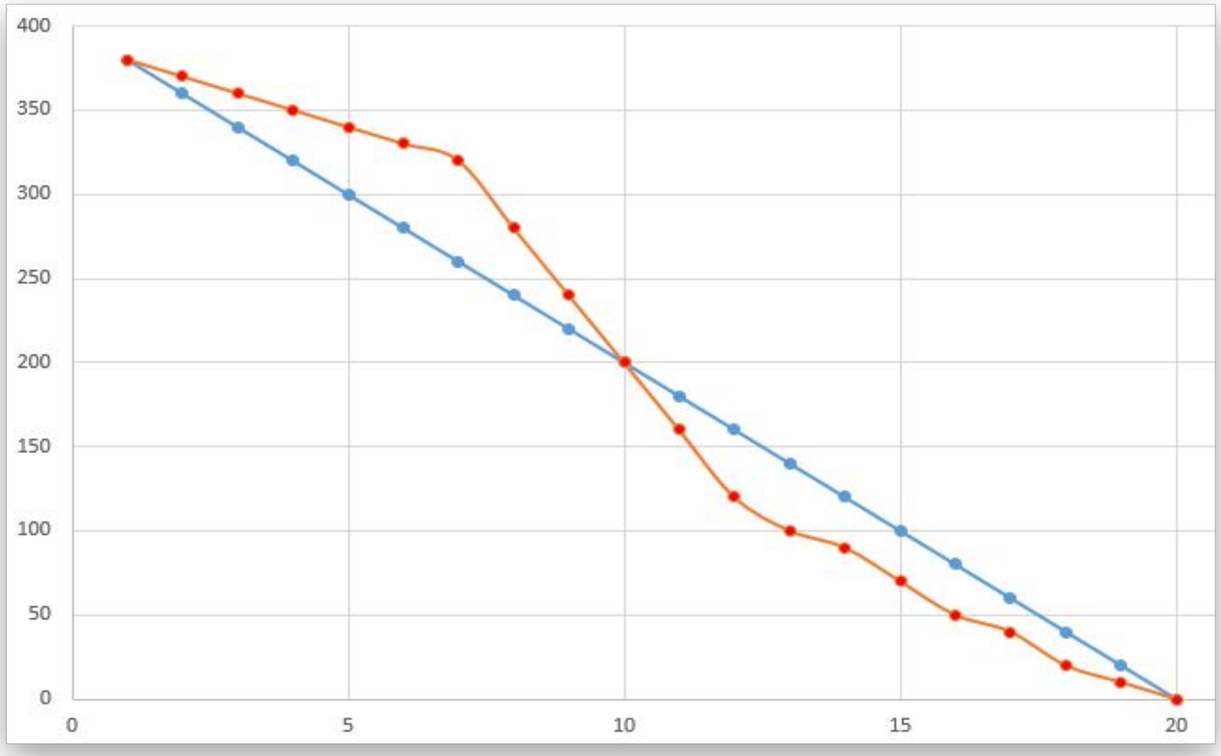
Также возможно группа делала принципиальное ускорение, для достижения цели.

Еще одной причиной, к примеру, может быть то, что команда брала дополнительные задачи.



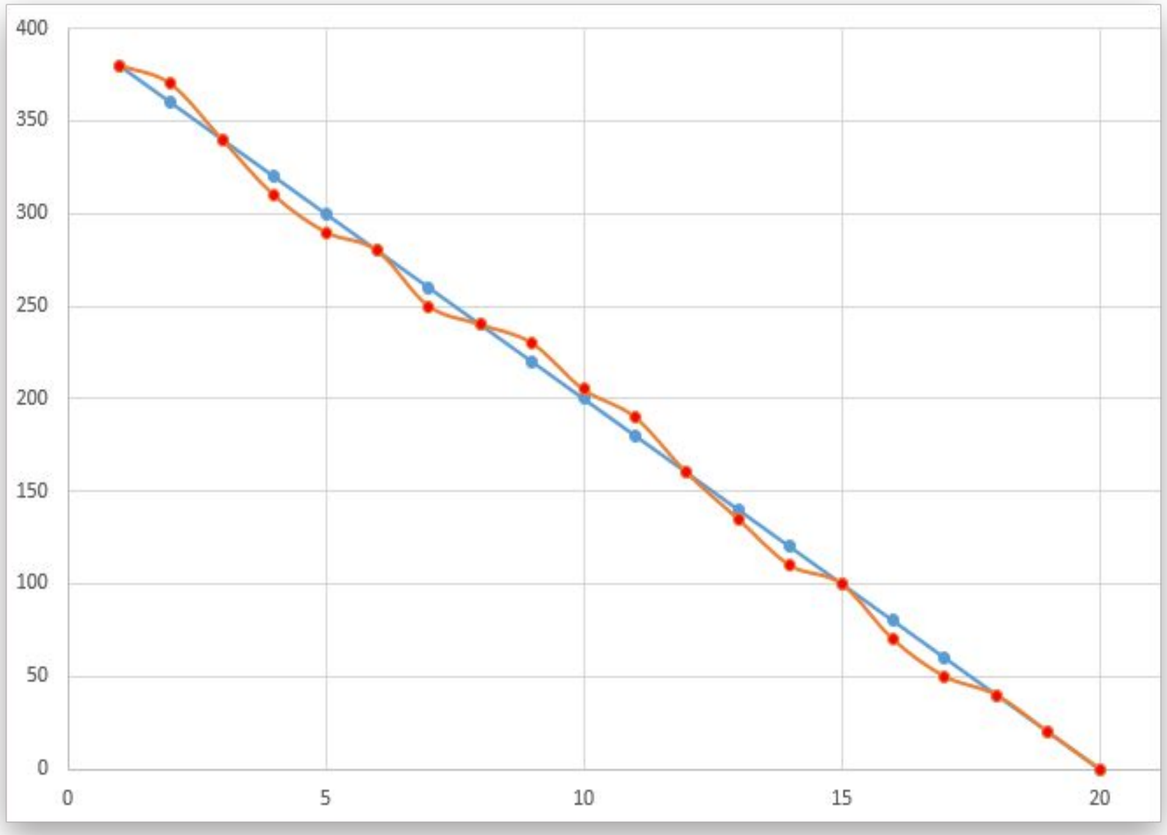
Burndown Chart: Опыт

На лицо опытная группа, которая после начала работы, сразу исправляет все возникающие трудности и совершенствуется так, что резко переходит к активному сжиганию.



Burndown Chart: A++

Бесконечно можно смотреть на три вещи: как горит огонь, как течет вода и как строится идеальный график =).



Матрица соответствия требований

Матрица соответствия требований (Requirements Traceability Matrix) – это двумерная таблица, содержащая соответствие функциональных требований (**functional requirements**) продукта и подготовленных тестовых сценариев (**test cases**).

В заголовках колонок таблицы расположены требования, а в заголовках строк — тестовые сценарии. На пересечении — отметка, означающая, что требование текущей колонки покрыто тестовым сценарием текущей строки. Матрица обычно хранится в виде электронной таблицы.

Матрица соответствия требований используется **QA**-инженерами для валидации покрытия требований по продукту тестами. Цель «Traceability Matrix» состоит в том, чтобы выяснить:

- какие требования «покрыты» тестами, а какие нет.
- избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

Данный тестовый артефакт является неотъемлемой частью тестирования.

Матрица соответствия требований

Пример «Traceability Matrix»:

Requirements Traceability Matrix (Authorisation/Registration)									
Requirements	Authorisation by Email	Authorisation by Google+	Authorisation by Twitter	Authorisation by Facebook	Forget Password	Registration by Email	Registration by Google+	Registration by Twitter	Registration by Facebook
Test-Suite/Amount of Test-Cases	2	3	3	3	2	2	3	3	3
1.1	x								
1.2		x							
1.3			x						
1.4				x					
2.0					x				
3.1						x			
3.2							x		
3.3								x	
3.4									x

Спасибо за внимание!

www.andersenlab.com