
ITMAE
General
S(crum)-Light
ПОНЯТНЫЙ ПУТЬ
УПРАВЛЕНИЯ ПРОЕКТАМИ
ЮРИЙ НЕДРЕ
АЛЕКСАНДР БАЗАНОВ

Александр Базанов
Юрий Недре

S(crum)-Light – Понятный путь управления проектами

http://www.litres.ru/pages/biblio_book/?art=70074073
SelfPub; 2023

Аннотация

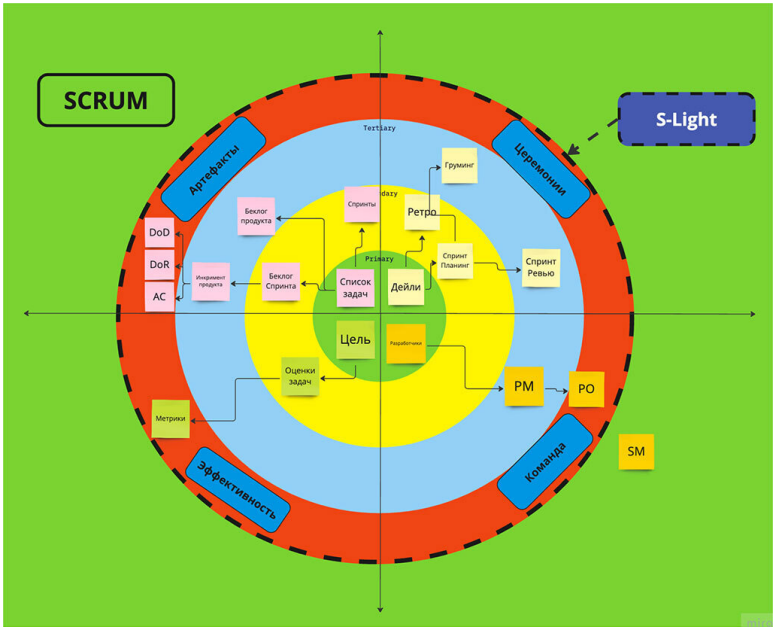
Зачем нужен S(crum)-Light?Чтобы облегчить жизнь. S(crum)-Light не подразумевает сложной (да и вообще, какой-либо) философии. S-Light – это набор простых правил, выполнения большинство из которых не является обязательным. Любая команда, может взять минимальные элементы S-Light (список задач, разработчиков и дейли) и у них уже будет S-Light.Дальше, они могут, как конструктов, добавлять новые элементы, которые им подходят. Так же они могут регулярно проводить простую самооценку, чтобы понимать, как глубоко они внедрили S-Light и что еще они могут еще добавить, чтобы попробовать увеличить свою производительность и повысить качество разработки. А может быть, им это и не нужно.

Содержание

Юрий Недре, Александр Базанов S(crum)-Light – Понятный путь управления проектами

Что такое S-Light?

S-Ligth – это фреймворк/метод/инструкция/набор правил (называйте как хотите) по процессам разработки программного обеспечения.


SCRUM
S-Light
Tertiary
Артефакты
Церемонии
Эффективность
Команда
DoD
DoR
AC
Инкремент продукта
Беклог продукта
Беклог Спринта
Спринты
Ретро
Груминг
Secondary
Primary
Спринт Планинг
Спринт Ревью
Список задач
Дейли
Цель
Разработчики
Оценки задач
Метрики
PM
PO
SM

Зачем это нужно

На сегодняшний день, 2023 год, самый распространенный метод ведения проектов по разработке программного обеспечения (SDLC), является фреймворк Scrum.

Он применяется, как в больших Enterprise компаниях, так и в стартапах. Его популярность обусловлена простотой инструментов и краткостью описания. Это вам не PMBoK.

Казалось бы, собирай команду каждый день на небольшие летучки (и прочие собрания), назначь Product Owner'а (который будет принимать решения), Scrum Master'а (который будет отвечать за процесс), ну а команда developers есть всегда по умолчанию. И все счастливы. Scrum внедрен, а нам осталось наслаждаться регулярной поставкой ценности клиентам.

На этом останавливается бóльшая часть команд. Вроде всё по Scrum, но команды спрашивают: "А что изменилось? Мы всё так же перетаскиваем карточки по доске. От того что наш Project Manager стал называться Scrum Master ничего не поменялось. Наша производительность не изменилась. Зачем это было нужно?"

Это и есть главная проблема и недостаток Scrum. Его нельзя просто понять, а главное внедрить, прочитав Scrum Guide. Его надо постичь. Scrum это не 5 ритуалов, 3 роли и 3 артефакта. Его бóльшую часть занимает философия. Фи-

лософию Scrum многие не могут понять и за несколько лет работы как Scrum Master.

Вот два примера вопроса из экзамена по Скрам-гайду:

Что представляет собой выбранный Элемент Бэклога Продукта во время Планирования Спринта?

1) Цель

2) Прогноз

3) План

4) Обязательство

или

Разработчики хотят разделиться на две группы: разработчиков и тестировщиков. Что должен делать скрам-мастер в этой ситуации?

1) Поощряйте этот подход, поскольку Скрам-команда обладает самоуправлением и лучше знает, как выполнять работу.

2) Предупредите, что это не лучшая идея и научите Скрам-команду брать на себя ответственность за продукт, который они создают как команда.

3) Поощряйте подобный подход, так как он делает роли и обязанности в Скрам-команде более четкими.

Как вы видите, ответы далеко не очевидны и их не найти напрямую в Скрам-гайде, Но без этих знаний, компании, которые владеют монополией на сертификацию в области Скрама считают, что вы не знаете Скрам и работаете не по нему. Кстати, в обоих случаях, правильным будет ответ но-

мер 2.

Многие ошибочно думают, что Scrum это гибкий фреймворк.

"А разве не так?", спросят многие: *"Он ведь относится к Agile. А это про гибкость."*

Скрам, на самом деле, это жесткий набор правил и настоятельных рекомендаций.

• Дейлики растягиваются до 30 минут – нет, это не Scrum.

• Решили скипнуть ретро, так как команда вымотана в конце спринта – нет, это не Scrum.

• PO решил сам погрумить задачи без команды – нет, это не Scrum.

• Нет Scrum master – нет, это не Scrum.

• **Сделали шаг влево от гайда – нет, это не Скрам.**

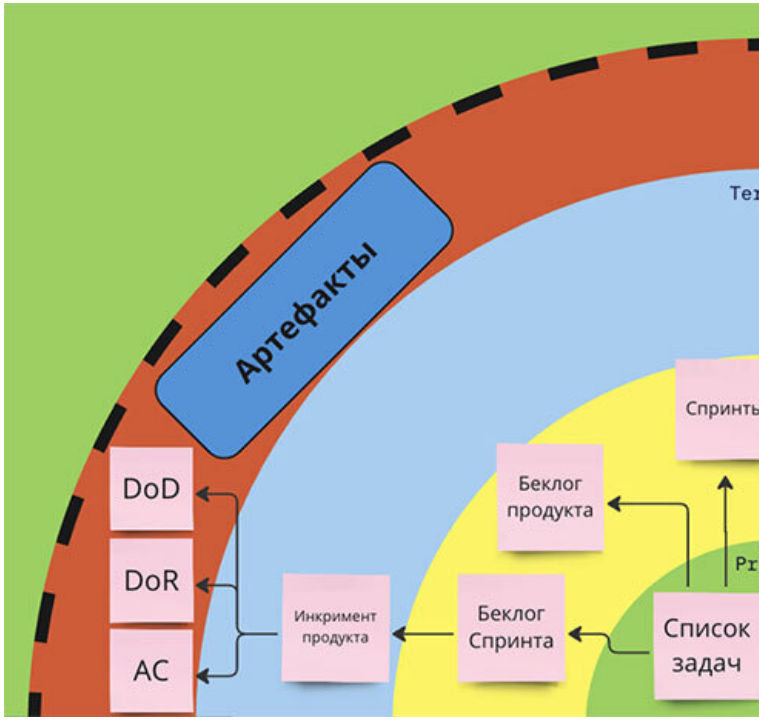
Соблюдать все обязательные пункты и следовать рекомендациям Scrum Guide 99 % командам сложно (а иногда просто и невозможно), но без этого Scrum Guide говорит: «У вас не Scrum».

Зачем нужен S-Light?

Чтобы облегчить жизнь. S-Light не подразумевает сложной (да и какой-либо) философии. S-Light – это набор простых правил, выполнения большинство из которых не является обязательным. Любая команда, может взять минимальные элементы S-Light (цель, список задач, разработчиков и дейли) и у них уже будет S(crum)-Light.

Дальше они могут, как конструктов добавлять новые элементы, которые им подходят. Так же они могут регулярно проводить простую самооценку, чтобы понимать, как глубоко они внедрили S-Light и что еще они могут еще добавить, чтобы попробовать увеличить свою производительность и повысить качество разработки. А может быть, им это и не нужно.

Артефакты



Бэклог

Бэклог – список задач, которые необходимо выполнить

для достижении цели продукта или проекта. Все задачи, которые выполняют разработчики, должны быть положены в бэклог.

Все элементы бэклога необходимо иметь пять элементов: описание, приоритет, оценка (относительная или абсолютная), ответственный и критерии приемки.

За один элемент бэклога ответственный один участник команды разработки.

Оценку элементов бэклога делают сотрудники, которые причастны к его реализации и/или понимают специфику разработки этого элемента. Рекомендуется, что Front-End задачи оценивались Front-End разработчиками, Back-End разработчики оценивали свои задачи, дизайнеры – свои и так далее.

Если команда дошла до стадии, когда работа планируется спринтами, то бэклог разделяется на две части:

- бэклог продукта или проекта
- бэклог спринта

Бэклог спринта – набор задач, которые необходимо выполнить команде разработки за спринт, чтобы достичь цели спринта.

Инкремент

Минимальная полезная функциональность или новая версия продукта, которая была доведена до пользователя путем поставки этой функциональности на продакшен.

Инкремент может быть создан после выполнения и доставки на продакшен:

1) Новой Feature;

2) Завершенной User Story;

3) Устранение бага (когда баг блокировал работу какой-то функциональности);

4) Завершение задачи (task), которая не подразумевает создание новой функциональности (например подключение новой библиотеки или новой версии базы данных).

Каждый поставленный Инкремент должен быть шагом к достижению цели продукта или проекта.

Рекомендуется определить командой Definition of Done для инкремента: набор признаков, когда элементы(ы) бэклога превращаются в инкремент доставленный на продакшен.

1. **Definition of Ready (DoR) –** критерий готовности задачи к тому, чтобы взять ее в работу. DoR будет одинаковым для всех задач. Для этого на старте надо договориться, что именно должно быть в задаче, чтобы ее можно было реализовать. К примеру: задача имеет описание, критерии приемки (AC), ответственного, оценку, связь с другими задачами,

ссылка на документацию, макет UI.

2. **Definition of Done (DoD) –** критерий выполнения задачи. DoR одинаков для всех задач. Для этого нужно договориться, в каком случае мы считаем, что задача полностью выполнена. К примеру: разработка завершена, проведены все виды тестирования, выполнена поставка на продакшен, описание дополнено в документации.

3. **Acceptance Criteria (AC) –** критерий проверки того, что задача работает так, как планировалась. AC будут уникальными для каждой задачи, написанными специально для этой задачи. Обычно составляются в виде чек-листа к задаче. AC должны быть прописаны так, чтобы были понятны, как разработчикам, так и тестировщикам и чтобы не было их двойного толкования.

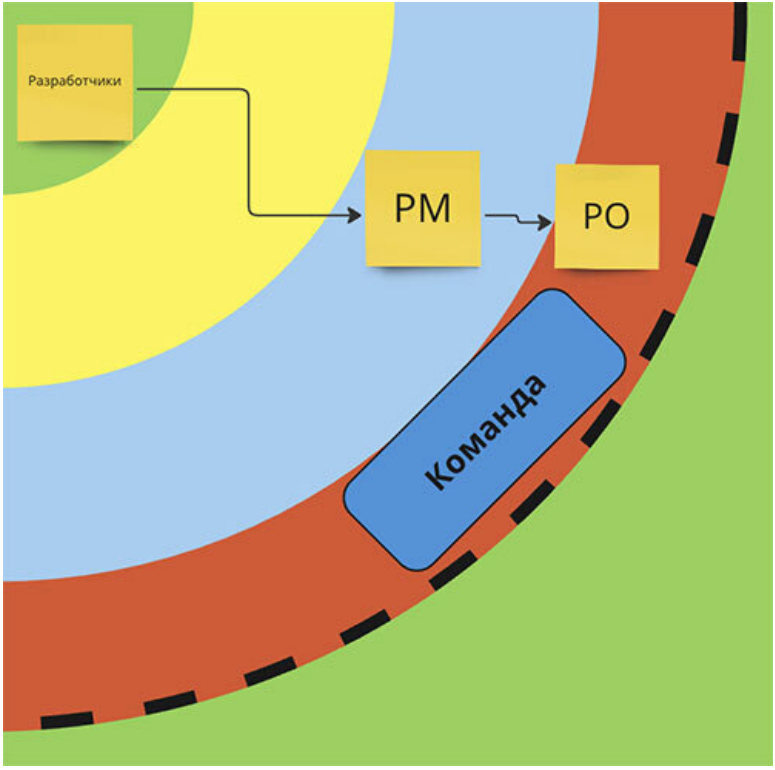
DoR и DoD могут быть свои в каждой команде, (их можно устанавливать единые критерии на всю компанию, если это применимо)

Спринты

Спринты – это повторяющийся временной интервал для выполнения запланированного объема работы. У спринта должна быть цель. Цель формирует и разъясняет команде владелец продукта, менеджер проекта или старший разработчик (тимлид) или другой сотрудник, выполняющий обязанности руководителя продукта или проекта. Все спринты имеют одинаковую длину. Новый спринт стартует сразу по завершению предыдущего спринта. Все события происходят внутри спринта.

Рекомендуется иметь длину спринтов в две недели. На бóльшей дистанции у команды может размываться фокус цели спринта и чем дольше спринт, тем сложнее его планировать. Спринты короче чем две недели, приведут к частому повторению всех церемоний спринта. Спринт может быть отменен руководителем команды (PO/PM/TL или другой сотрудник, выполняющий обязанности руководителя продукта/проекта), но только при существенном изменении планов (к примеру, со стороны бизнеса) или после совещания с командой.

Команда



Разработчики

Разработчики – минимальная необходимая роль для старта работы в S-Light. Работа может начаться, даже если в команде будут только одни разработчики. Они сами устанавливают себе приоритеты задач, сами определяют формат разработки, встречи (кроме Daily – они обязательны), артефакты (кроме Списка задач – они обязательны).

Под разработчиками понимается кросс-функциональная команда, которая занимается аналитикой, архитектурой, разработкой, тестированием и выкаткой на продакшен окружение.

Команда может быть, как полностью кросс-функциональной (каждый участник может выполнить весь цикл разработки (от описания до выкатки) самостоятельно), так и полностью распределенной (каждый участник выполняет свою часть работы).

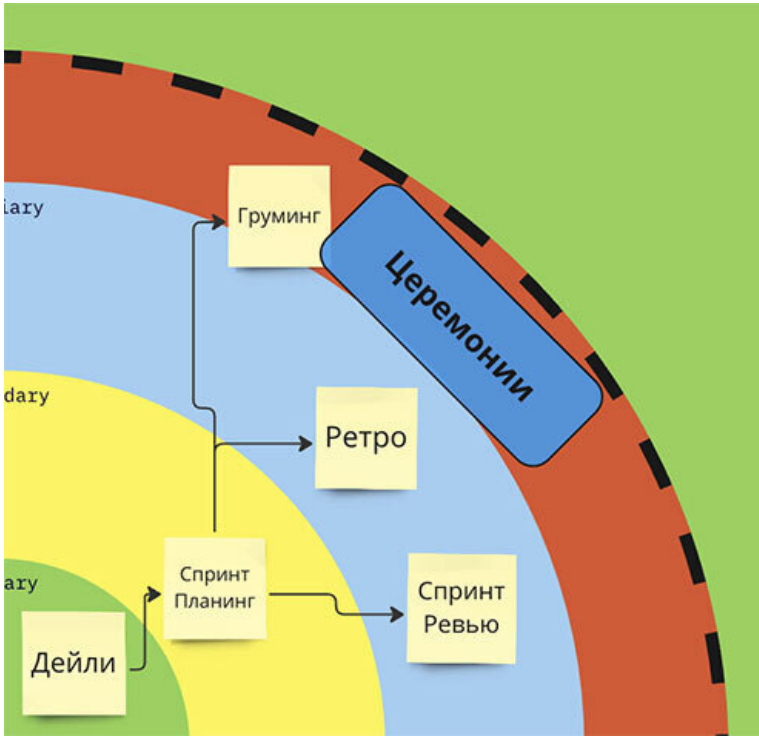
PM

Менеджер проекта – участник команды, который осуществляет координацию ведения продукта. Его задачей является помощь команде в достижении целей и устранении препятствий на пути к ней. Он является входным звеном потока информации извне (требования заказчиков, изменения обстоятельств) к команде разработки. Он может принимать ключевые решения в отсутствии других руководителей. Главная цель работы – выстроить процессы внутри команды разработки, чтобы сотрудники могли работать автономно.

PO

Является ответственным за обеспечение достижения целей продукта. Он приоритезирует список задач и доносит их смысл до команды разработки. Он говорит команде ЧТО делать, но не говорит КАК делать.

Церемонии



Дейли

Единственная обязательная церемония в S-Light.

Цель – понять что блокирует команду, обновить данные которые появились за вчера.

На ней собирается вся команда (разработчики), PO/PM по возможности (но очень желательно)

Формат – выбрать можно любой который захочет команда, можно начать с нашего примера.

Продолжительность – 15 минут. 2 минуты на выступление.

Изначально каждый участник отвечает на 3 вопроса:

Что сделал за вчера?

Что будет делать сегодня?

Какие есть проблемы?

Если в команде уже настроена доска задач и участники хорошо с ней работают (всегда актуальные статусы и приоритеты), то можно оставить только последний вопрос. Так же данный митинг можно использовать для рассказа о прошедших событиях, которые не отображены на доске:

- встречи и что на них обсудили
- новые договоренности или вводные по продукту которые меняют изначальные задачи

Рекомендации:

1. Митинг всегда в одно и тоже время;
2. Минимум через 30 минут после начала рабочего дня;
3. Если не успеваете в тайминг, то можно проводить стоя;
4. Не перебиваем участников, поднимаем руки;
5. Не дольше 15 минут весь дейлик.

Планирование спринта

Разделим планирование на 2 части – **Груминг** и **Планирование спринта**

Груминг бэклога

Груминг – это собрание представителей команды, во время которого обсуждаются детали бэклога продукта и готовится очередное планирование спринта.

Этот процесс необходим для того, чтобы задачи, представленные в бэклогебыли актуальными, а те, которые представлены в верхней части списка, были готовы к планированию в спринте, реализации и релизу.

Груминг бэклога часто называют предварительным планированием. Обычно собственник продукта и представители команды организуют его в середине спринта.

Что мы делаем во время груминга?

• Написание новых пользовательских историй и задач;

• Удаление неактуальных пользовательских историй и задач;

• Переоценка приоритетов для пользовательских историй и задач;

• Добавление новых функций, определение приоритетов;

• Усовершенствование и изменение приоритетов ранее описанных пользовательских историй и задач;

• Разбивка некоторых пользовательских историй и задач на более мелкие;

• Пересмотр критериев тестирования.

Результат хорошего груминга

• Когда задач вверху бэклога достаточно для 2–3 спринтов;

• Пользовательские истории понятны всем членам команды;

• Пользовательские истории и задачи имеют размер, позволяющий реализовать несколько из них за один спринт.

Важно помнить, что груминг должен стать постоянным событием в управлении продуктом, которым не стоит пренебрегать. Этот процесс – норма для качественного развития продукта. Самое главное в нем – оптимизировать задачи бэклога для последующей работы с ними.

Груминг бэклога помогает прояснять релевантность задач, представленных в бэклоге, анализировать существующие вопросы и получать дополнительную информацию о задачах, которые пока не до конца ясны.

Подытоживая, отметим основные преимущества груминга:

• Устраняет неопределенность и неизвестные факты в задачах что несомненно повышают эффективность продукта;

• Помогает избежать “переделок” в разработке и тестировании;

• Идентифицирует зависимости в команде и помогает

прогнозировать риски;

• Постоянные проведение экономит время команды разработчиков для дальнейшего обсуждения во время цикла спринта, потому что обеспечивает ясность для разработчиков и тестировщиков;

• Успешный груминг приводит к эффективному планированию спринта.

Планирование спринта

Планирование инициирует спринт, планируя работу, которую необходимо выполнить в этом спринте. Результатом события становится план, созданный совместными усилиями всей командой.

PO (или другой руководитель) обеспечивает готовность всех участников к обсуждению наиболее важных элементов бэклога и их связи с целью продукта/проекта. Команда может приглашать на планирование для консультаций и других людей.

В ходе планирование спринта рассматриваются следующие темы:

Почему этот спринт ценен? PO (или другой руководитель) предлагает, как можно повысить ценность и практичность продукта в текущем спринте. Затем вся команда совместно определяет цель спринта, которая объясняет, почему спринт ценен. Цель спринта должна быть сформулирована до окончания планирования.

Результат хорошего планирования

- Все задачи в бэклог спринта оценены;
- Поставлена цель спринта и все участники с ней согласны;

• В спринт взяли оптимальное кол-во задач с учетом скорости команды.

Спринт Ревью (Демо)

Цель Демо – показать стейкхолдерам, реализованный за спринт функционал, собрать обратную связь и свериться с ожиданиями.

Придать команде дополнительную мотивацию, после благодарности от стейкхолдеров или вместе принять доработки.

Sprint Retrospective

Цель Sprint Retrospective – запланировать повышение качества и эффективности.

Зачем нужна ретроспектива?

Это не праздный вопрос, его часто задают начальники, когда им предлагают провести ретроспективу. Они спрашивают: «Зачем? Мы можем сами все решить». Почему же нельзя сделать так, чтобы какой-то начальник или эксперт пришел, посмотрел и сказал, что команде надо делать, а что в рабочем процессе стоит изменить?

Основных причин две. Во-первых, если мы приходим к команде с готовыми решениями, возникает феномен, который называется «not invented here». Даже если члены команды понимают, что это правильное решение и его нужно выполнять, у них нет чувства собственности по отношению к нему. Такие решения, не «выстраданные» самим коллективом, а «навязанные» или предложенные сверху, имеют меньше шансов на реализацию.

Во-вторых, сейчас разработка ПО настолько сложная и запутанная вещь, что вряд ли найдется специалист, который сможет, не зная контекста, расписать, как на самом деле должны работать процессы в конкретной команде при реше-

нии определенной задачи. Чтобы это выяснить, надо что-то пробовать, проводить эксперименты, смотреть, к чему приводят те или иные решения. Только попробовав, можно понять, хороша или не очень та или иная практика в контексте данной команды.

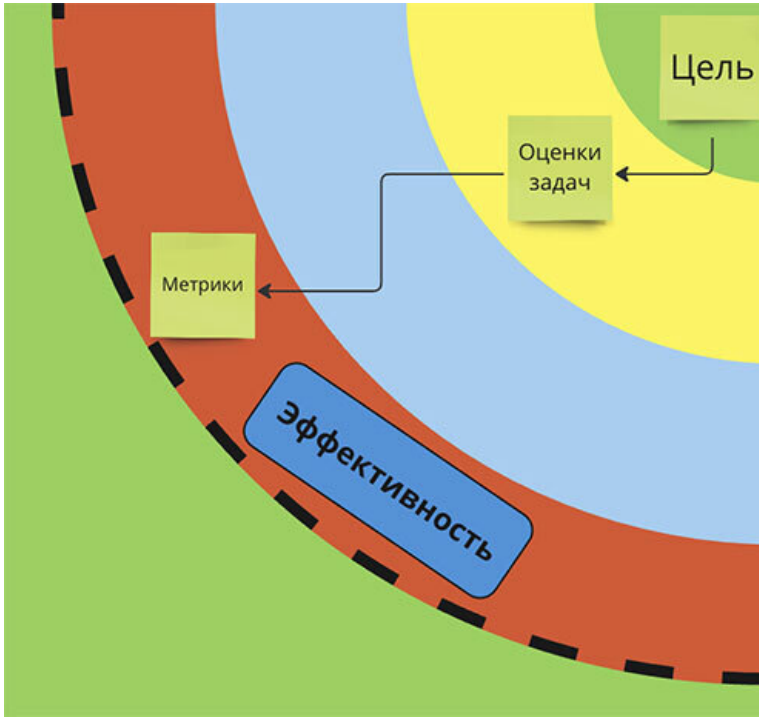
В основе ретроспективы лежит концепция цикла Деминга, PDCA (англ. Plan-Do-Check-Act). Цель ретроспективы – к ее окончанию получить план изменений. Но важно понимать, что это не план окончательных изменений в процессе – это план эксперимента на ближайший период. Мы что-то пробуем, а потом смотрим, что из этого вышло, и на основании этого принимаем решение.

Цикл Деминга: Plan – запланируй, Do – выполни, Check – посмотри, что получилось, Act – прими какие-то дальнейшие решения, реши, что дальше делать. Ретроспектива должна проходить именно по этому циклу. Собственно, сама ретроспектива – это стадия Plan.

Короткий план:

- Плюсы. Что шло хорошо в прошлой итерации?
- Минусы. Какие проблемы были в прошлой итерации?
- Идеи. Какие идеи появились по ходу ретроспективы?
- План. Какие улучшения мы запланируем на следующую итерацию?

Эффективность



Анализ эффективности является важным процессом повышения производительности команды.

Цель

Команда должны понять, для чего они реализуют данный проект/продукт. Цель должна быть простой и понятной всем членам команды. Цель должны быть зафиксирована и доведена до всех членов команды. Цельпо мере развития продукта может меняться, но при любых изменениях в цели они должны быть снова объяснены и доведены до команды.

Оценка задач

Первый шаг к пониманию эффективности команды является оценка задач. Задачи рекомендуется оценивать в относительных величинах, сравнивая их друг с другом. Проще и легче всего внедрить оценку в Story points, когда задачи измеряются в абстрактных величинах. Команде нужно начать замерять, сколько Story points она «съедает» в конце спринта (если разработка идет без спринтов, то за неделю, две недели, месяц) и, на основе этих данных, может прогнозировать свою эффективность в следующем отрезке (спринте) разработки.

Метрики

Метрики являются следующим этапом развития анализа эффективности и прогнозирования времени достижения целей. Они помогают более точно планировать сроки и бюджеты работ, а также информировать пользователей о поставке им готового функционала.

Команды могут использовать различные метрики, но наиболее простые и эффективные на начальном этапе являются:

Velocity

Скорость работы команды. Средняя величина выполнения задач. Может измеряться, как в количестве задач, так и в Story points. Рассчитывается из среднего значения выполненных задач / Story points за последние 4–6 спринтов. Рекомендуется определять данную метрику, как по команде в целом, так и по каждому разработчику (тут имеется ввиду именно разработчик = программист). Постоянно обновляя данную метрику, можно более точно провести анализ достижения различных целей.

Code destiny

Чистота кода. Измеряется в процентном соотношении ко-

личества багов к количеству задач во всем бэклоге продукта. Индикатор отслеживается на регулярной основе, помогая определить на сколько чистый код пишет команда и не является ли количество багов в продукте критичной проблемой. Рекомендуется держать данный показатель не выше 30 %.

Дополнительные метрики

• **Cycle Time** – время, которое задача находилась в разработке от момента, когда ей начали заниматься, до момента, когда она прошла фазу конечной поставки (команда должна установить, что является точкой старта и окончания работы над задачей).

• **Lead Time** – время от появления задачи до ее конечной поставки. Включает Cycle Time и время ожидания в очереди на реализацию (команда должна установить, что является точкой старта и окончания работы над задачей).

• **WIP** – количество задач одновременно находящихся в работе. Разделяется по разным стадиям работы над задачей.

• **Wasted Time** – время, которое задача проводит в различных очередях, а не непосредственно в работе.

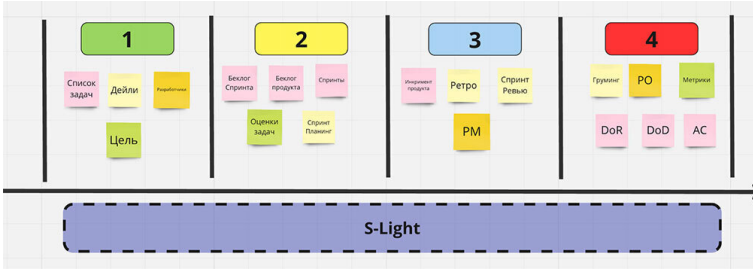
• **Effectiveness** – процент времени, которое тратится непосредственно на работу с задачей, а не на ожидания в различных очередях.

Команды могут внедрять много разных метрик, главное помнить, что метрики внедряются не ради самих метрик, а

для того, чтобы понимать, где в процессе возникают проблемы. Если метрика не дает этого понять, то ее следует исключить и не тратить на нее время.

Метрики должны служить главной цели – повышение точности прогнозирования. За метрики, обычно, отвечает менеджер проекта (PM).

Оценка + чек лист



В рамках разработки S-Light мы постарались устранить основную проблему Scrum – точно понять, работает ли команда в рамках фреймворка или нет.

Дальше команда может оценить, как глубоко внедрен S-Light и на сколько близко команда подошла к переходу на полноценный Scrum.

Но надо помнить, что целью внедрения S-Light не является переход в Scrum или повышения глубины внедрения самого S-Light.

С помощью данного чек листа команда может легко и быстро понять, как глубоко она внедрила S-Light. Также команда может определить, какие еще действия она можете сделать, чтобы внедрить больше элементов S-Light. Но перед планированием внедрения дополнительных элементов,

команда должна для себя ответить на вопрос: а надо ли их внедрить? /

Чек-лист внедрения S-Light

Церемонии	Балл	Артефакты		Команда		Эффективность	
Дейли	0	Список задач	0	Разработчики	0	Цель	0
Спринт планинг	1	Беклог продукта	1	PM	1	Оценка задач	1
Ретро	1	Беклог спринта	1	PO	1	Метрики	1
Спринт ревью	1	Спринты	1				
Груминг	1	Инкримент продукта	1				
		DoR	1				
		DoD	1				
		AC	1				

Используя данный чек-лист вы легко можете оценить, на сколько глубоко вы внедрили наш метод.

При его использовании есть два простых правила:

1) Основные пункты из зеленой зоны должны быть внедрены обязательно. Без них начинать работу нельзя.

2) Зачитывать балл за внедрение пункта можно, если все пункты из предыдущего этапа были внедрены. Например, если вы не оцениваете эффективность и внедрили Метрики,

но не внедрили Оценку задач, то баллы за Метрики засчитывать нельзя. Внедрение S-Light надо проводить от центра к краю, двигаясь в область Scrum.