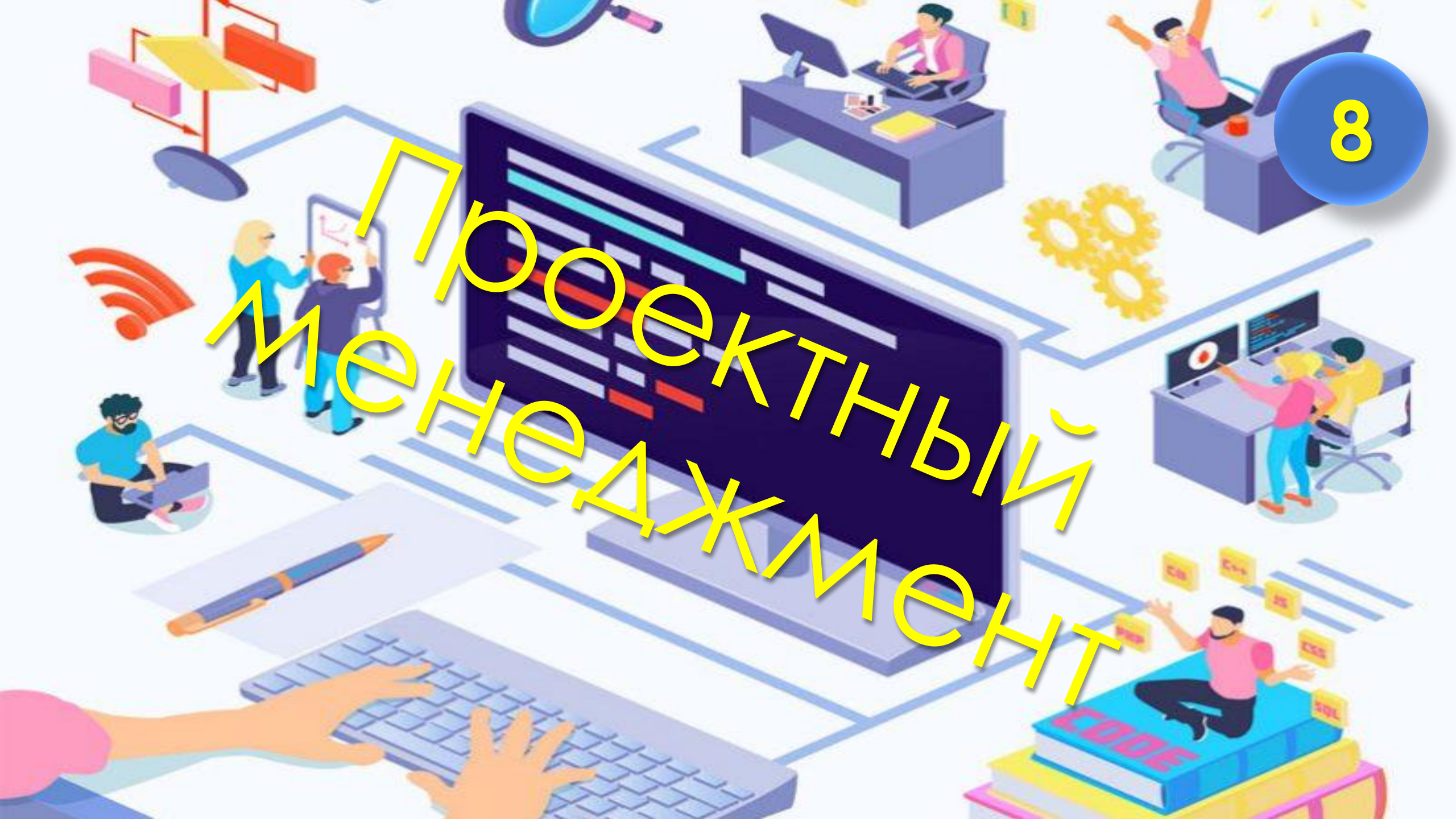


Проектный менеджмент



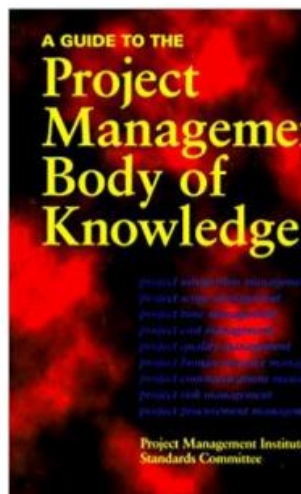
ПРОЕКТ

- Проект – это уникальный набор процессов, состоящих из скоординированных и управляемых задач с начальной и конечной датами, предпринятых для достижения цели.

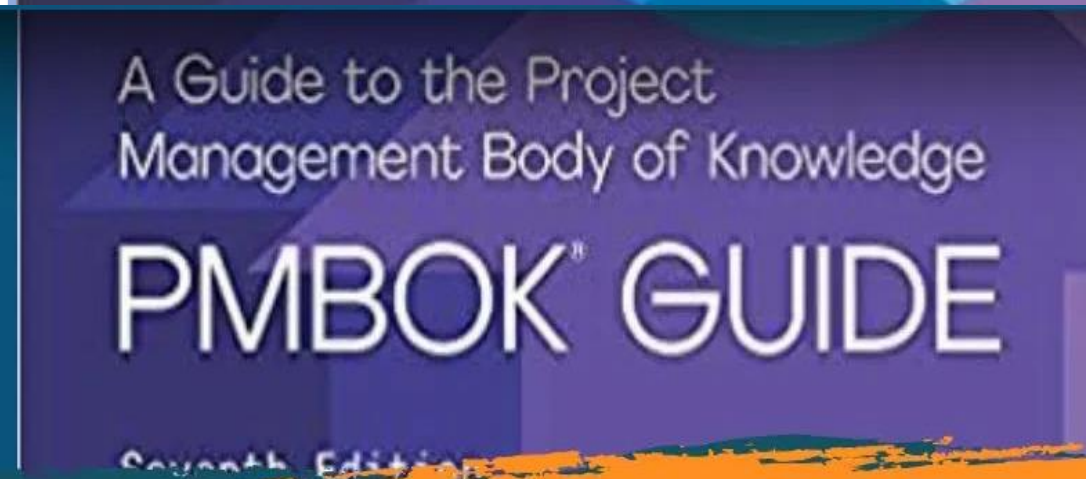
Достижение цели проекта требует получения результатов, соответствующих определенным заранее требованиям, в том числе ограничениям на получения результатов, таких как время, деньги и ресурсы.

PMI: разви

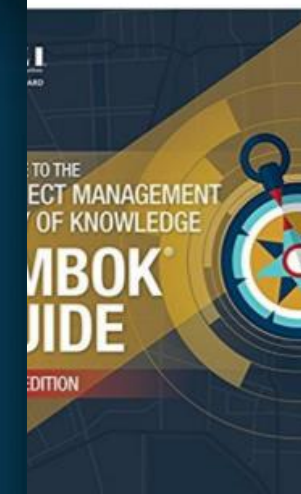
management



1996
1-е издание
176 страниц
9 областей
знаний
37 процессов



**PMBOK® Guide - 7th
Edition - coming on 1
August 2021**



2017
1-е издание
756 стр.
10 областей
знаний
процессов

Развитие методов ведения проектов

- «Waterfall Model» (каскадная модель или «водопад»)
- «V Model»
- «Incremental Model»
- «RAD Model»
- «Iterative Model»
- «Spiral Model»
- «Agile Model»

«Waterfall Model» (каскадная модель или «водопад»)

- Когда требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
- Нет проблем с доступностью людей нужной квалификации.
- В относительно небольших проектах.



«Waterfall Model»

подразумевает последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей. В модели Waterfall легко управлять проектом.

(+) Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены.

(-) Но каскадная модель будет давать отличный результат только в проектах с четко заранее определенными требованиями и способами их реализации.

(-) Нет возможности сделать шаг назад, тестирование начинается только после того, как разработка завершена или почти завершена.

(-) Продукты, разработанные по данной модели без обоснованного ее выбора, могут иметь недочеты (список требований нельзя скорректировать в любой момент), о которых становится известно лишь в конце из-за строгой последовательности действий.

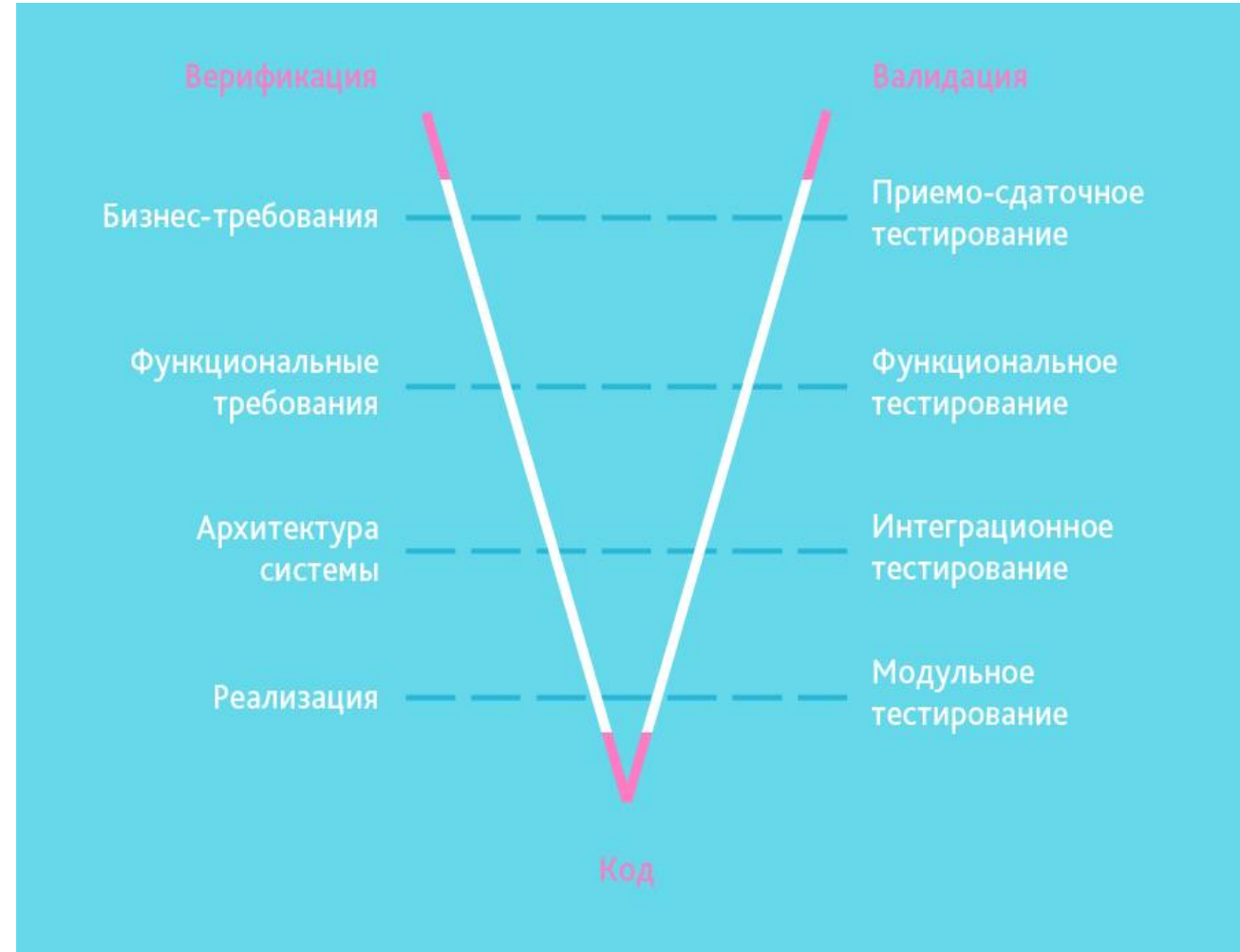
(-) Стоимость внесения изменений высока, так как для ее инициализации приходится ждать завершения всего проекта.

(+) Но фиксированная стоимость часто перевешивает минусы подхода.

Исправление осознанных в процессе создания недостатков возможно, требует дополнительных соглашений к контракту с небольшим ТЗ.

«V Model»

- Если требуется тщательное тестирование продукта, то V-модель оправдывает заложенную в себя идею: validation and verification (Проверка и подтверждение).
- Для малых и средних проектов, где требования четко определены и фиксированы.
- В условиях доступности людей необходимой квалификации, особенно тестировщиков.



«V Model»

- V-модель— это улучшенная версия классической каскадной модели. Здесь на каждом этапе происходит контроль текущего процесса, для того чтобы убедиться в возможности перехода на следующий уровень. В этой модели тестирование начинается еще со стадии написания требований, причем для каждого последующего этапа предусмотрен свой уровень тестового покрытия. В V-модели каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования..

«V Model»

Достоинства V-образной модели

- модель проста в использовании (относительно проекта, для которого она является приемлемой)
- в модели особое значение придается планированию, направленному на верификацию и аттестацию разрабатываемого продукта на ранних стадиях его разработки
- в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта
- в V-образной модели определение требований выполняется перед разработкой проекта системы, а проектирование ПО — перед разработкой компонентов
- модель определяет продукты, которые должны быть получены в результате процесса разработки, причем каждые полученные данные должны подвергаться тестированию
- благодаря модели менеджеры проекта может отслеживать ход процесса разработки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы является контрольной точкой

«V Model»

Недостатки V-образной модели

- не позволяет справиться с параллельными событиями
- в ней не учтены итерации между фазами
- в модели не предусмотрено внесение динамических изменений на разных этапах жизненного цикла
- тестирование требований в жизненном цикле происходит слишком поздно, вследствие чего невозможно внести изменения, не повлияв при этом на график выполнения проекта
- в модель не входят действия, направленные на анализ рисков

«V Model»

Область применения V-модели (+ примеры):

- когда вся информация о требованиях доступна заранее
- при разработке систем высокой надежности:
- *прикладные программы для наблюдения за пациентами в клиниках*
- *встроенное ПО для устройств управления аварийными подушками безопасности в автомобилях*

«Incremental Model» (инкрементная модель)

- Когда основные требования к проекту четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени.
- Требуется быстрое исполнение проекта.
- Могут появиться новые возможности (фичи).



«Incremental Model» (инкрементная модель)

В инкрементной модели полные требования к системе делятся на различные сборки. Терминология часто используется для описания поэтапной сборки ПО. Имеют место несколько циклов разработки, и вместе они составляют жизненный цикл «мульти-водопад». Цикл разделен на более мелкие легко создаваемые модули. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования. Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов». Процесс продолжается до тех пор, пока не будет создана полная система.

«RAD Model»

(rapid application development model или быстрая разработка приложений)

- Может использоваться только при наличии высококвалифицированных и узкоспециализированных специалистов.
- Бюджет проекта большой, чтобы оплатить этих специалистов вместе со стоимостью готовых инструментов автоматизированной сборки.
- Может быть выбрана при уверенном знании конечной цели и необходимости срочного исполнения проекта в течение 2-3 месяцев.



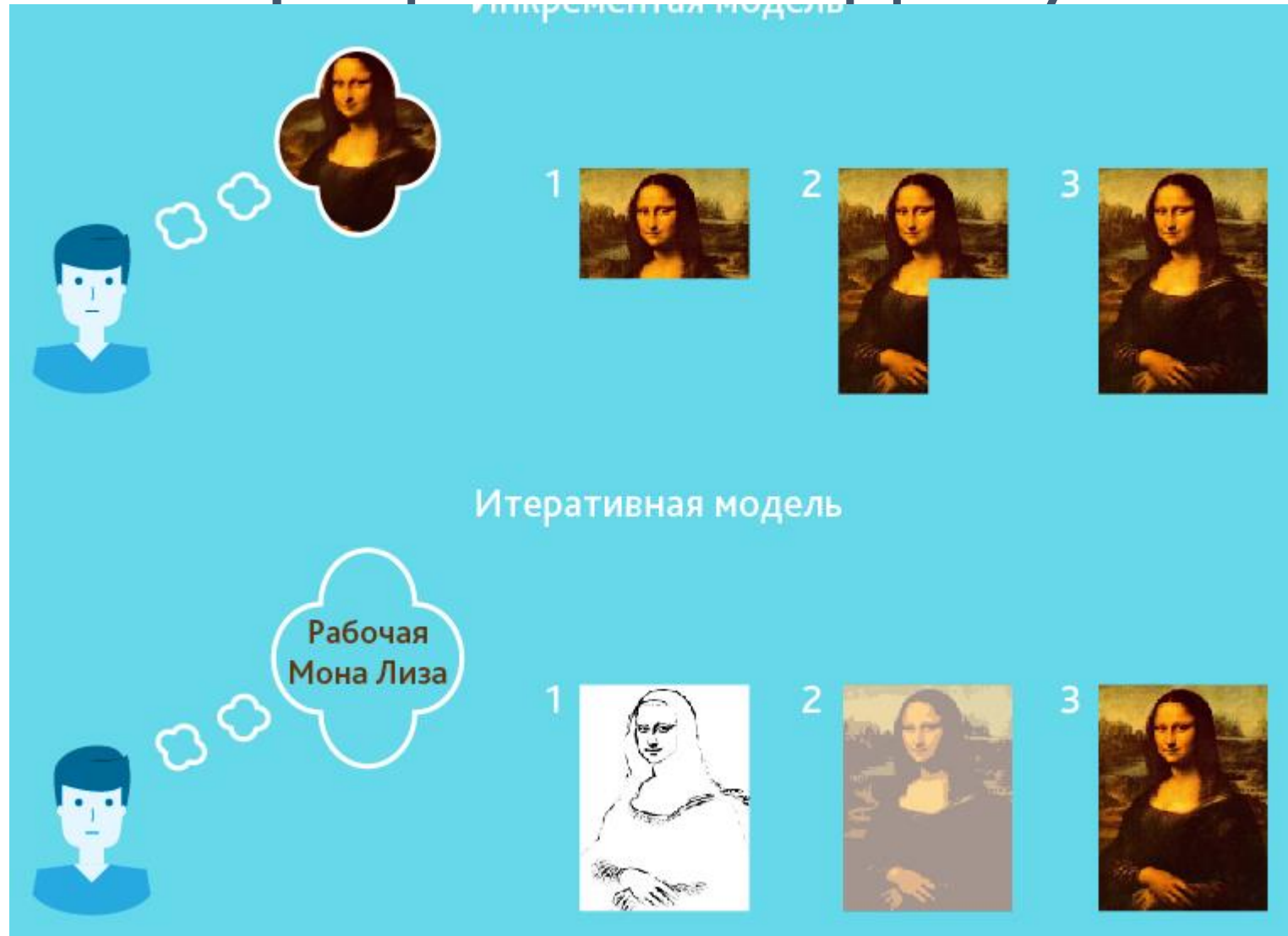
«RAD Model» (rapid application development model или быстрая разработка приложений)

RAD-модель — разновидность инкрементной модели. В RAD-модели компоненты или функции разрабатываются несколькими высококвалифицированными командами параллельно, будто несколько мини-проектов. Временные рамки одного цикла жестко ограничены. Созданные модули затем интегрируются в один рабочий прототип. Синергия позволяет очень быстро предоставить клиенту для обозрения что-то рабочее с целью получения обратной связи и внесения изменений.

«Iterative Model»

(итеративная или итерационная модель)

- Требования к конечному результату проекта заранее четко определены и понятны.
- Проект большой или очень большой.
- Основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.



«Iterative Model»

(итеративная или итерационная модель)

Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, мы стремимся к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна.

«Spiral Model» (спиральная модель)

- Для сложных и дорогих проектов (государственных).
- На каждом витке 4 этапа:
 - ☐ планирование;
 - ☐ анализ рисков;
 - ☐ конструирование;
 - ☐ оценка результата и при удовлетворительном качестве переход к новому витку



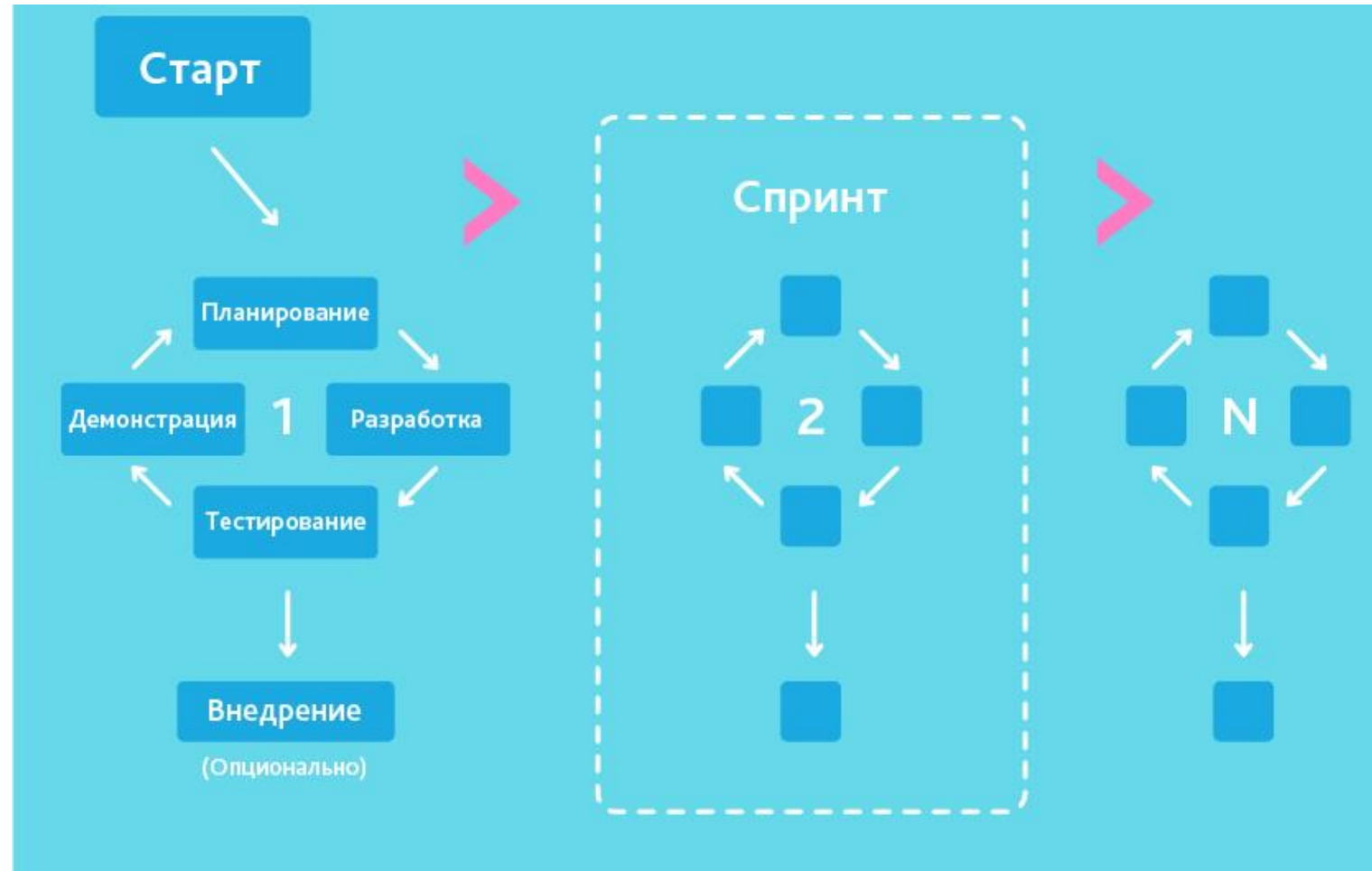
«Spiral Model» (спиральная модель)

«Спиральная модель» похожа на инкрементную, но с акцентом на анализ рисков. Она хорошо работает для решения критически важных бизнес-задач, когда неудача несовместима с деятельностью компании, в условиях выпуска новых продуктовых линеек, при необходимости научных исследований и практической апробации.

«Agile Model»

(гибкая модель)

- Когда потребности пользователей постоянно меняются в динамическом бизнесе.
- Изменения на Agile реализуются за меньшую цену из-за частых инкрементов (операций увеличения переменной).
- В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.



«Agile Model»

(гибкая модель)

В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из преимуществ гибкой модели. К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку. Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

В основе такого типа — непродолжительные ежедневные встречи — «Scrum» и регулярно повторяющиеся собрания (раз в неделю, раз в две недели или раз в месяц), которые называются «Sprint». На ежедневных совещаниях участники команды обсуждают:

- отчёт о проделанной работе с момента последнего Scrum'a;
- список задач, которые сотрудник должен выполнить до следующего собрания;
- затруднения, возникшие в ходе работы.

«Agile Model»

(гибкая модель)

Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка. Соответственно, в процессе реализации требования изменяются. Стоит вспомнить класс творческих людей, которым свойственно генерировать, выдавать и опробовать новые идеи еженедельно или даже ежедневно. Гибкая разработка лучше всего подходит для этого психотипа руководителей

«КЛАССИКА»

(КАСКАДНАЯ МЕТОДОЛОГИЯ «WATERFALL»)

| ПЛЮСЫ | МИНУСЫ |
|---|---|
| Подробная документация. | Медленный запуск. |
| Согласованные и утвержденные требования. | Трудноизменяемые жесткие требования. |
| Выпускаются менее квалифицированными разработчиками. | Клиент не видит продукт до завершения разработки. |
| Сниженное число дефектов благодаря тщательному планированию структуры. | Малая гибкость затрудняет изменение направления. |
| Заданная начальная и конечная точка для каждой фазы, что позволяет легко измерять прогресс. | Клиенты в начале не имеют ясного представления о своих требованиях. |

Когда использовать методологию «Waterfall»?

- Требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
- Нет проблем с доступностью человеческих ресурсов нужной квалификации.
- В относительно небольших проектах.

AGILE

| ПЛЮСЫ | МИНУСЫ |
|---|---|
| Быстрый запуск, пошаговый выпуск и регулярная критика и отзывы от клиента. | Может быть неверно расценен как отсутствие плана или расхлябанность. |
| Изменение требований со временем. | Требует высококвалифицированной, ориентированной на клиента группы разработчиков. |
| Способность быстро реагировать на изменения. | Требует высокого уровня вовлеченности клиента. |
| Меньше доработок благодаря непрерывному тестированию и вовлеченности клиента. | Отсутствие долгосрочных подробных планов. |
| Связь в реальном времени между группой разработчиков и клиентом. | Производит меньше отчетной документации. |

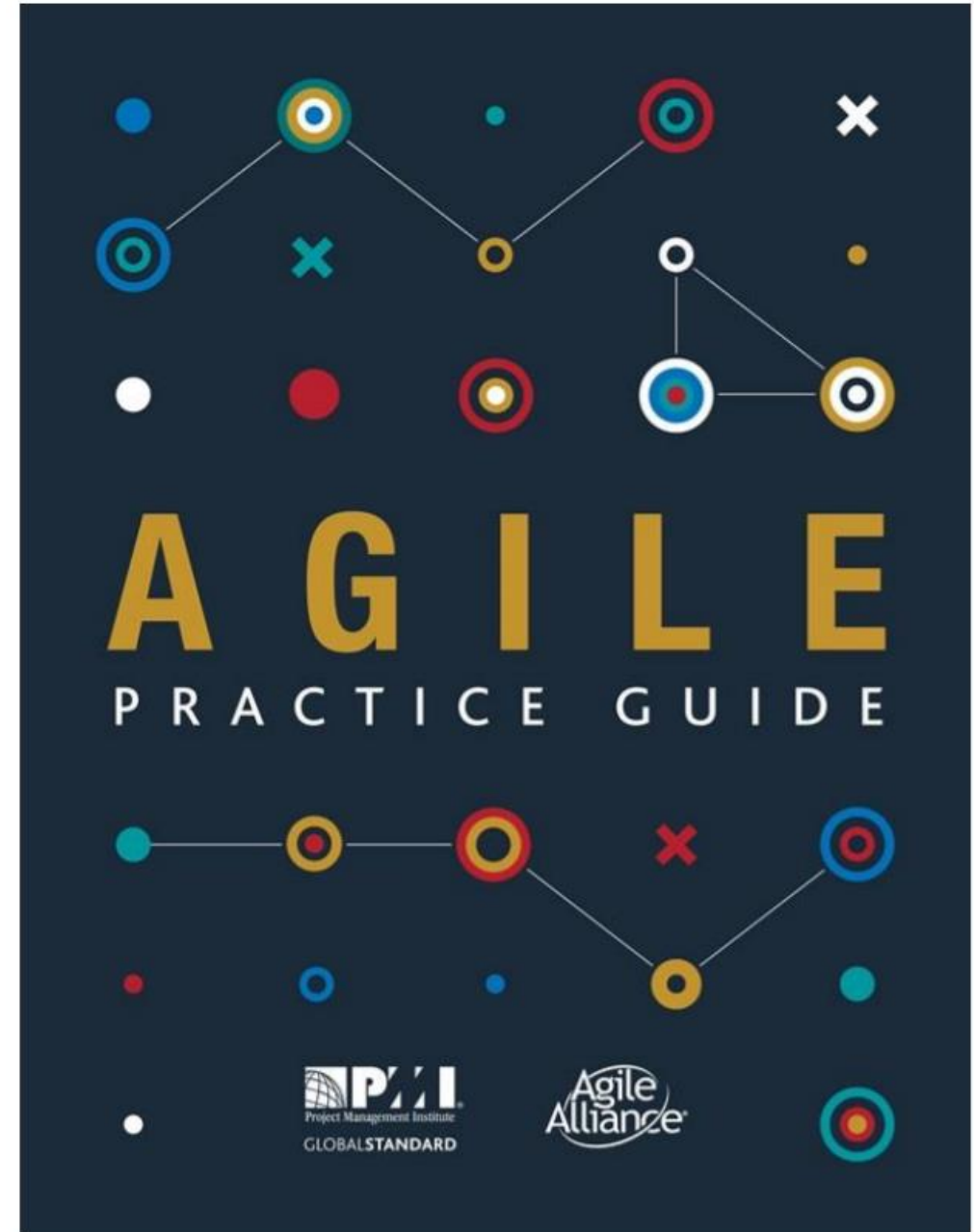
- **Когда использовать Agile:**

- Когда потребности пользователей постоянно меняются в динамическом бизнесе.
- Изменения на Agile реализуются за меньшую цену из-за частых инкрементов.
- В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

Agile - это строго говоря не гибкая методология разработки, это система ценностей, философия или даже образ мышления.

Agile - это собирательный бренд, любых подходов, принципов, методов или методологии, позволяющих действовать согласно ценностям и принципов Agile manifest.

Agile - это процессы, продукты и инструменты.



Agile-манифест разработки программного обеспечения

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

Люди и взаимодействие важнее процессов и инструментов

Работающий продукт важнее исчерпывающей документации

Сотрудничество с заказчиком важнее согласования условий контракта

Готовность к изменениям важнее следования первоначальному плану

То есть, не отрицая важности того, что справа,
мы всё-таки больше ценим то, что слева.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Основополагающие принципы Agile-манифеста

Мы следуем таким принципам:

Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.

Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

Работающий продукт — основной показатель прогресса.

Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.

Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.

Простота — искусство минимизации лишней работы — крайне необходима.

Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Agile-манифест разработки программного обеспечения

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и в этом другим. Благодаря проделанной работе мы смогли осознать,

Люди и взаимодействие важнее процессов и инструментов
Работающий продукт важнее исчерпывающей документации
Сотрудничество с заказчиком важнее согласования условий
Готовность к изменениям важнее следования первоначальному

То есть, не отрицая важности того, что справа,
мы всё-таки больше ценим то, что слева.

| | | |
|-------------------|----------------|------------------|
| Kent Beck | James Grenning | Robert C. Martin |
| Mike Beedle | Jim Highsmith | Steve Mellor |
| Arie van Bennekum | Andrew Hunt | Ken Schwaber |
| Alistair Cockburn | Ron Jeffries | Jeff Sutherland |
| Ward Cunningham | Jon Kern | Dave Thomas |
| Martin Fowler | Brian Marick | |

Самоорганизующаяся команда

- Над проектом работают мотивированные люди.
- Создаются все условия, поддержка и полное доверие.
- Самые лучшие архитектуры, требования и дизайны систем создаются самоорганизующимися командами.
- Команда сама организует оптимальный процесс.



Самоорганизующаяся команда — это команда, которая берет на себя ответственность за все решения и ее совместная работа является наилучшим средством достижения поставленных целей.

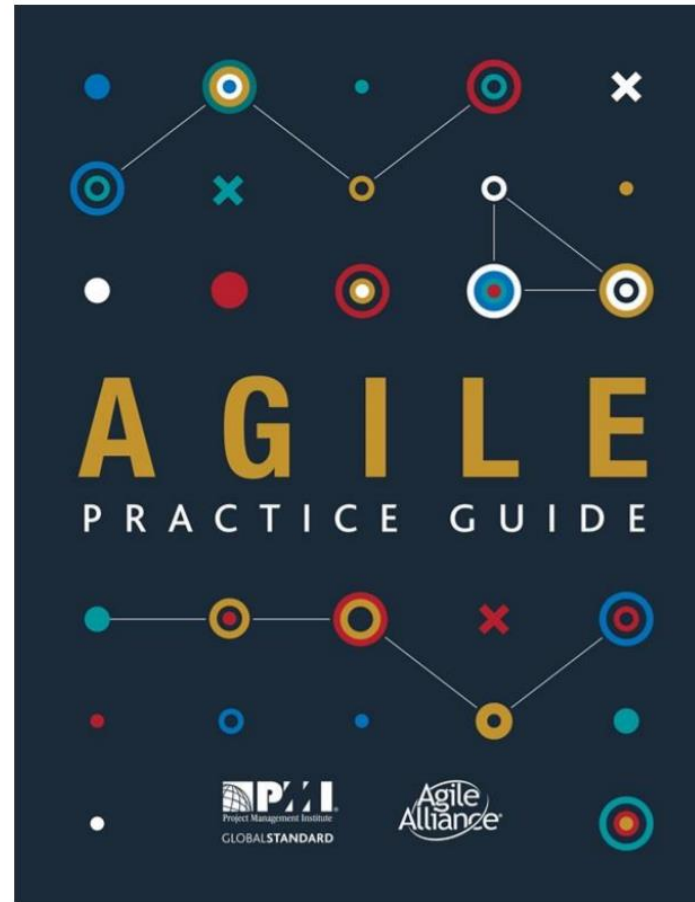
Больше общения

- Потенциальные пользователи системы и разработчики должны работать **вместе** на протяжении всего проекта.
- Самый действенный и эффективный способ обмена информацией как внутри команды разработчиков, так и с внешним миром - **непосредственное общение.**



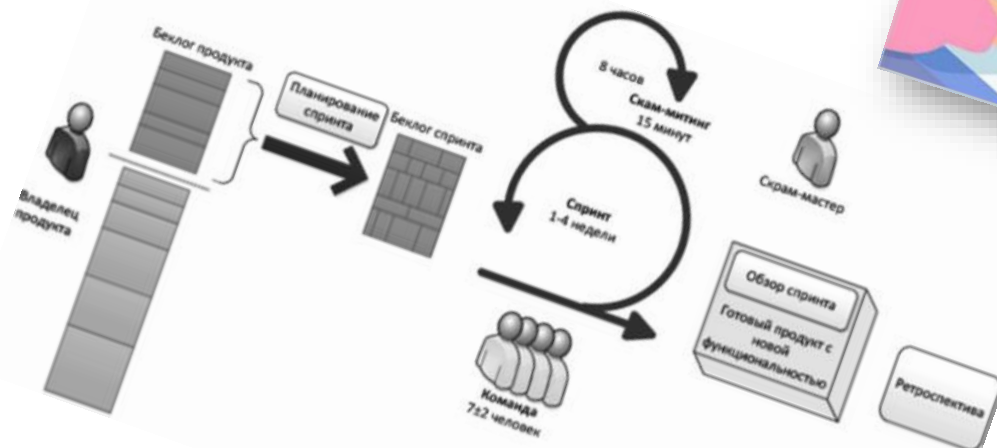
Самые популярные AGILE-подобные методы ведения проектов (сейчас)

- **SCRUM**
- **Kanban**



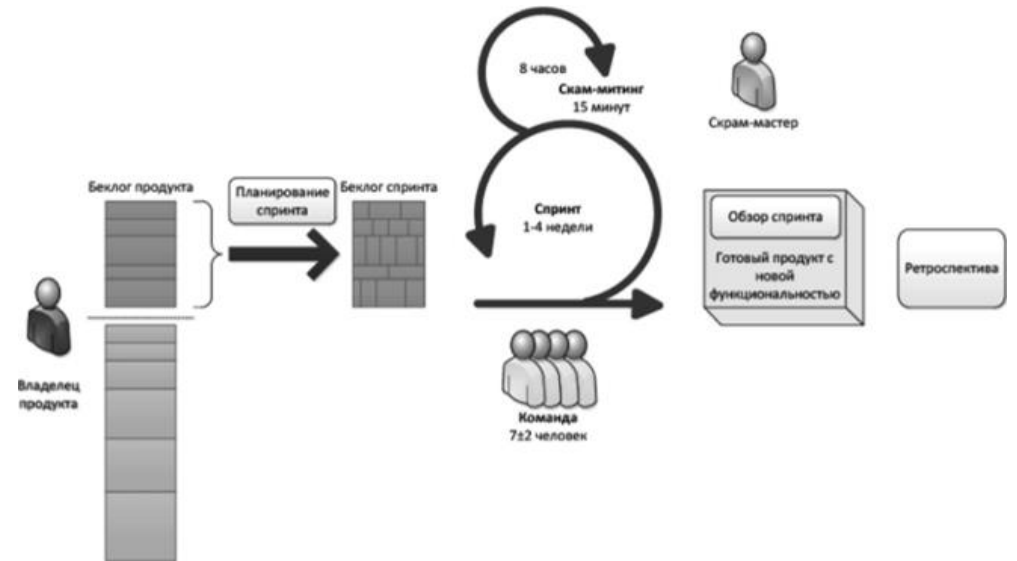
(Из Лекции 7)

- SCRUM

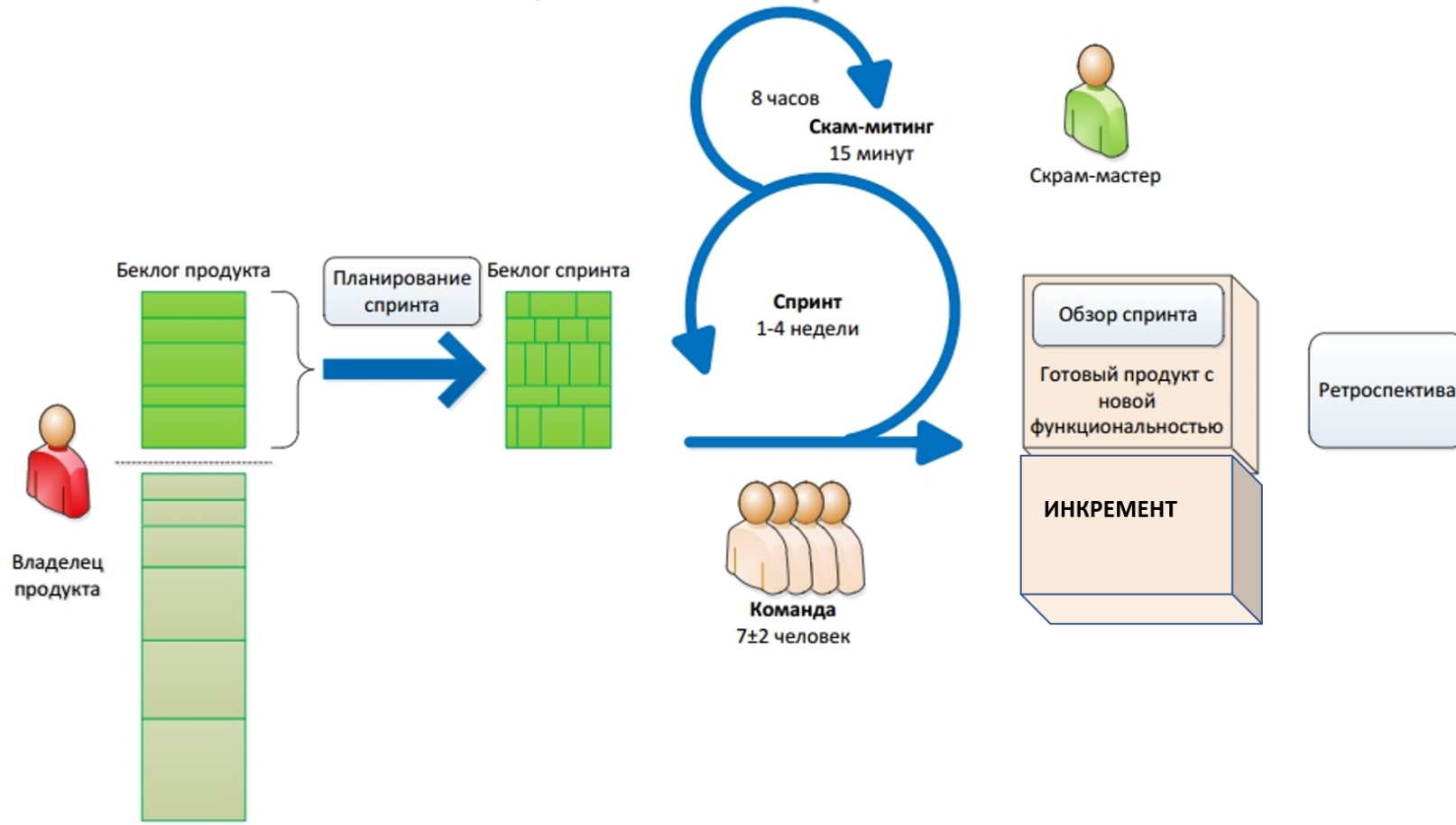


Scrum

- Scrum устанавливает правила управления процессом разработки и позволяет использовать уже существующие практики кодирования, корректируя требования или внося тактические изменения. Использование этой методологии дает возможность выявлять и устранять отклонения от желаемого результата на более ранних этапах разработки программного продукта.

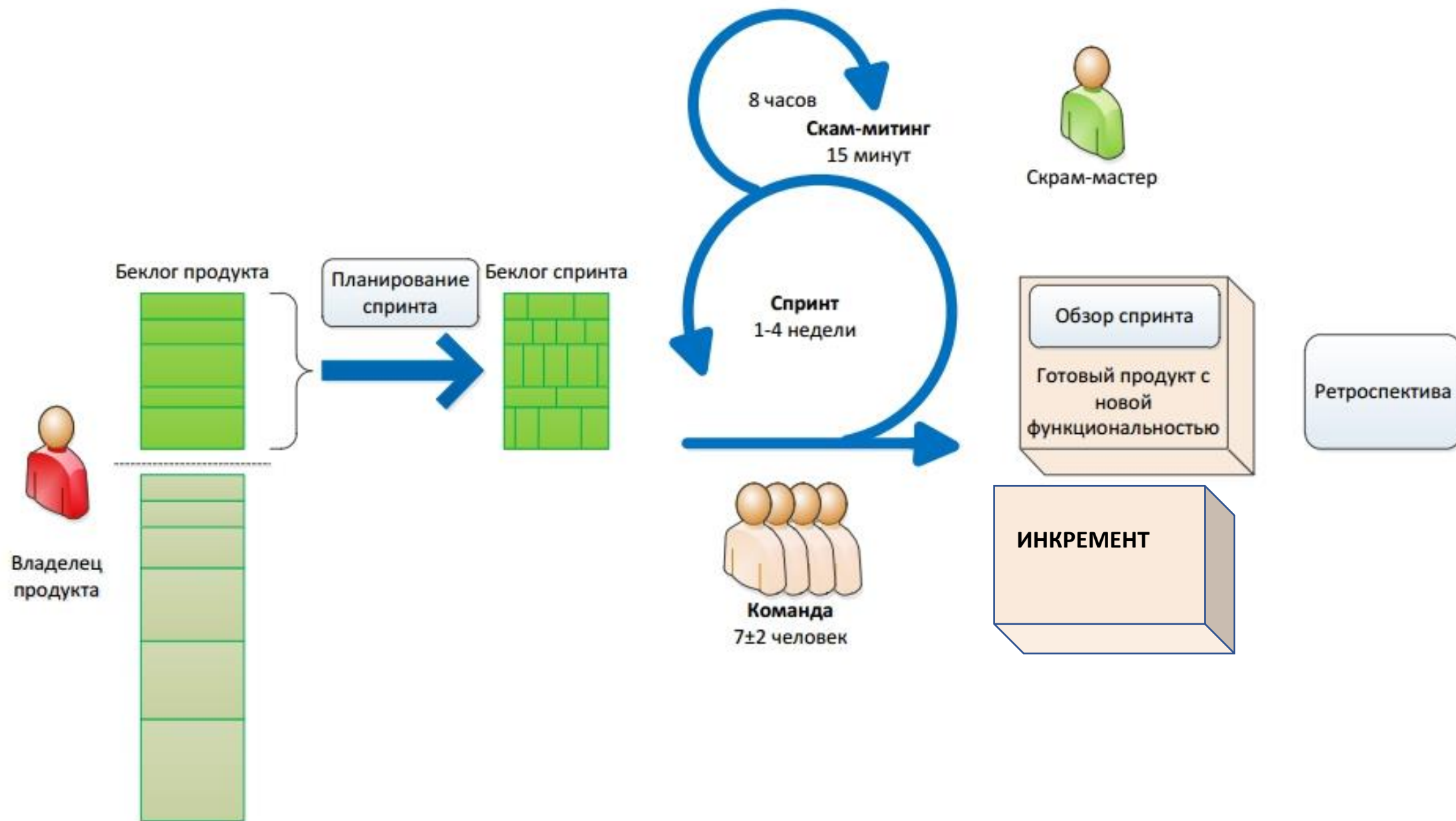


- По своей сути Scrum — это набор **принципов** разработки, которые позволяют представлять конечному пользователю (и заказчику) действующий продукт или прототип, обладающий новыми функциями и возможностями с наивысшим приоритетом. Работа в этом случае проводится в четко фиксированные недлительные (в среднем от 1 до 6 недель, чаще от 2 до 4) итерации (**спринты**)



- Работу над проектом ведут небольшие команды с пересекающимися функциями, причем каждая команда включает всех нужных специалистов. Скрам-мастер (руководитель проекта) отвечает за создание продуктивной атмосферы и соблюдение процессов. Обязанность скрам-мастера — постоянное нахождение с проектной командой для понимания текущей ситуации.
- Требования заказчиков разбиваются на небольшие функциональные части, которые как можно меньше зависят друг от друга и ориентируются на конечных пользователей. Результатом становится получение списка задач для продукта, элементы которого ранжируют в зависимости от важности и оценивают их объем.
- За требования и приоритеты отвечает владелец продукта. Чаще всего им является представитель заказчика, который и замыкает на себе всех заинтересованных в проекте лиц.
- Короткие и фиксированные итерации длительностью от 1 до 4 недель — основа работы проекта. Такие итерации называют спринтами. В конце каждого спринта разработчики предоставляют законченную функциональность — инкремент продукта. При желании его даже можно вывести на рынок.

- У каждой команды проекта есть своя скорость. В зависимости от скорости **команда выбирает** себе задачи на итерацию, которая не изменяется по времени. На **ежедневном скрам-митинге** синхронизируется работа команд и обсуждаются возникшие вопросы/проблемы. По мере работы каждая **команда берет в работу элементы списка** задач в зависимости от приоритета и собственных возможностей.
- **Обзор спринта** — логическое завершение итерации. Во время обзора спринта получают **отклик от владельца** продукта и проводят **ретроспективу спринта** для оптимизации собственных процессов. Также на обзоре спринта владелец продукта может изменить требования и приоритеты, а также запустить новый спринт.



Три принципа Scrum:

- **Прозрачность.** Важные документы и элементы должны быть доступны тем, кто за них отвечает. Информация должна быть структурирована таким образом, чтобы обеспечить свободный обмен внутри команды проекта и между внешними участниками проекта. Прозрачность обеспечивается с целью построения эффективных коммуникаций, оперативного выявления проблем и реализации изменений.
- **Рефлексия.** Участники проекта должны на постоянной основе собирать информацию о продукте, состоянии проекта и окружении с целью выявления отклонений, рисков и потенциальных возможностей. В этот процесс должны быть вовлечены все участники проекта.
- **Непрерывное улучшение.** Команда постоянно ищет возможности для улучшения своего продукта и рабочего процесса. Целью улучшений в первую очередь должна являться удовлетворённость заказчика.

В основе метода лежат:

3 роли

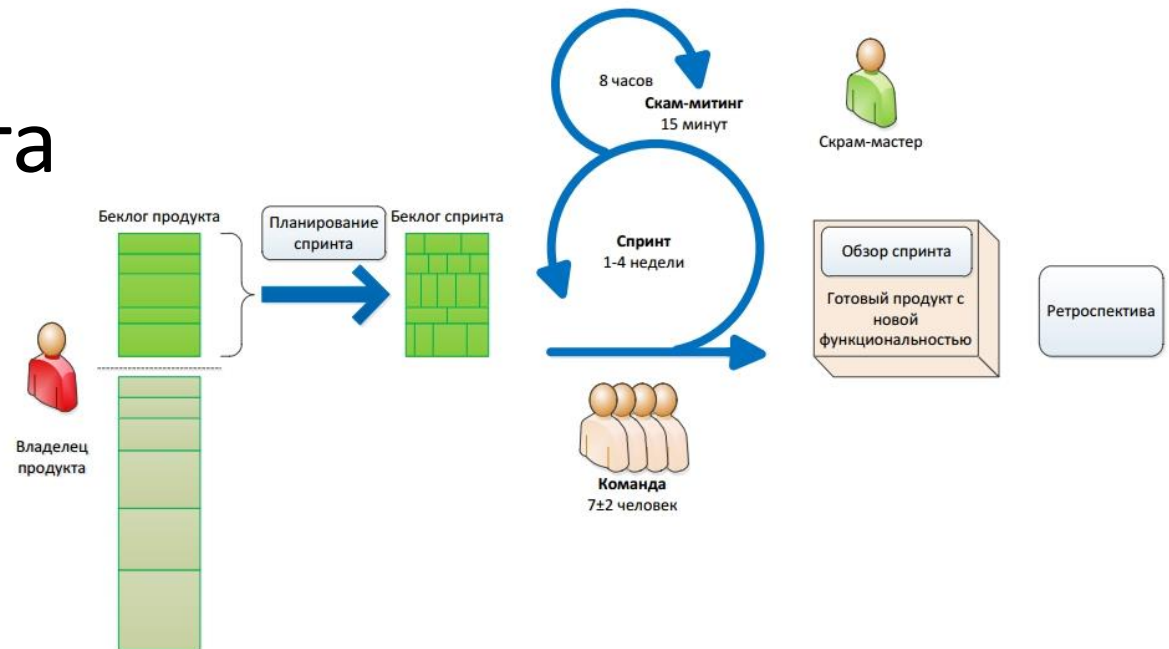
3 артефакта

5 событий



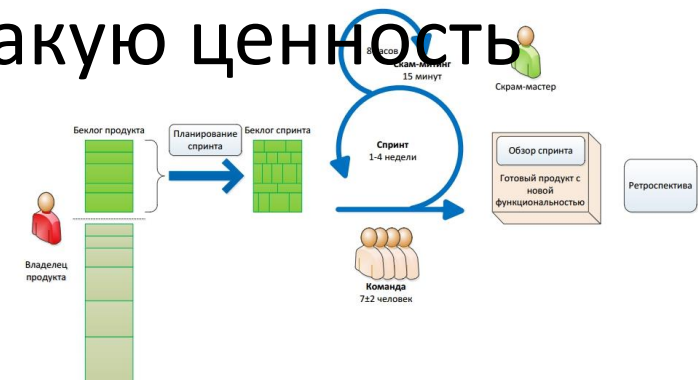
Основная структура процессов Scrum вращается вокруг 5 основных встреч:

- упорядочивание бэклога,
- планирование Спринта,
- ежедневные летучки,
- подведение итогов Спринта
- ретроспектива Спринта.



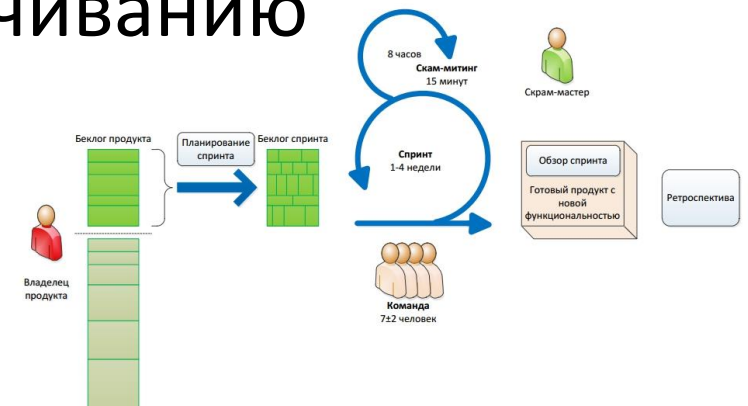
Встреча по упорядочиванию беклога (Backlog Refinement Meeting, «Backlog Grooming»):

Эта встреча аналогична фазе планирования в классическом проектном управлении, и проводится в первый день каждого Спринта. На ней рассматривается – что уже было сделано по проекту в целом, что ещё осталось сделать и принимается решение о том, что же делать дальше. Владелец продукта определяет, какие задачи на данном этапе являются наиболее приоритетными. Данный процесс определяет эффективность Спринта, ведь именно от него зависит, какую ценность получит Заказчик по итогам спринта.



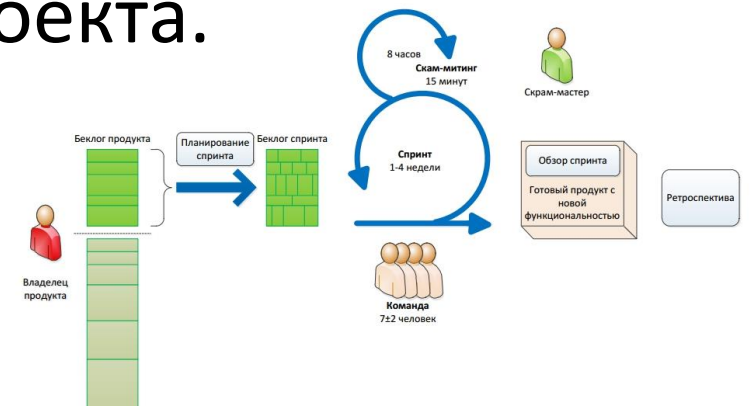
Планирование Спринта:

После того, как Владелец продукта определил приоритеты, команда совместно решает, что же конкретно они будут делать во время грядущей итерации, как достигнуть поставленной на предыдущей встрече цели. Команды могут применять различные инструменты планирования и оценки на данном этапе, лишь бы они не противоречили принципам и логике Scrum. Планирование Спринта проводится в самом начале итерации, после Встречи по упорядочиванию продукта.



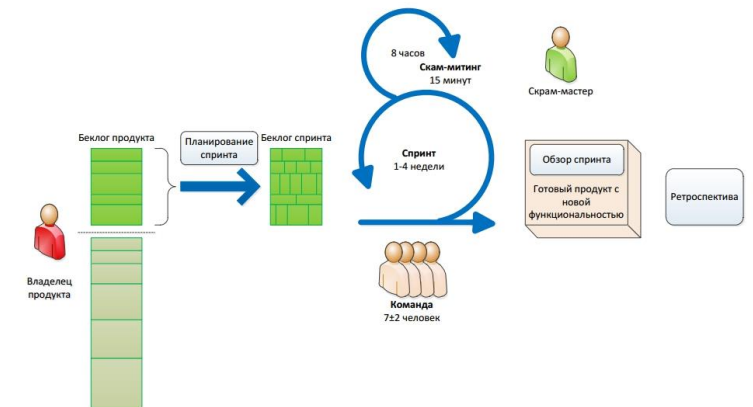
Ежедневные летучки:

Каждый день спринта, в идеале, в одно и то же время, члены команды тратят 15 минут на то, чтобы поделиться информацией о статусе задач и состоянии проекта. На ней не происходит обсуждений проблем или принятия решений – если после встречи возникают вопросы и конфликты, Scrum Мастер и вовлечённые участники обсуждают их отдельно. Летучка же нужна для обмена информации и поддержания всех членов команды в курсе состояния проекта.



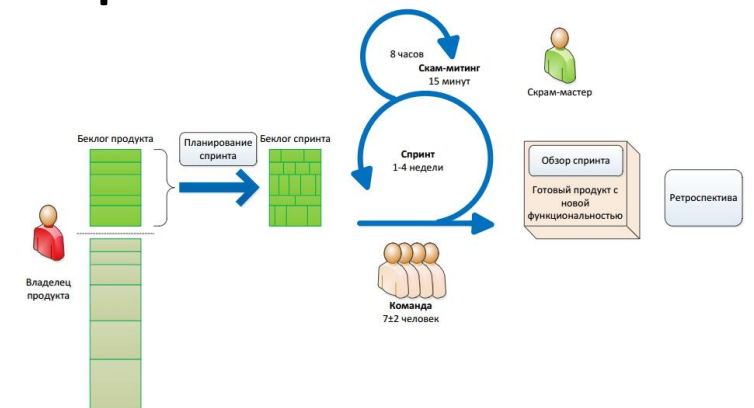
Подведение итогов Спринта:

Цель этапа – обследование и адаптация создаваемого продукта. Команда представляет результаты деятельности всем заинтересованным лицам. Основная задача – убедиться, что продукт этапа соответствует ожиданиям участников и согласуется с целями проекта.



Ретроспектива Спринта:

Проводится сразу после Подведения итогов спринта и до планирования следующего спринта. На нём команда выясняет, насколько чётко и слаженно проходил процесс реализации этапа. Обследованию подвергаются возникшие проблемы в работе, методологии и взаимодействии. Именно этот этап позволяет команде провести рефлекссию и следующий Спринт провести эффективнее.



В основе метода лежат:

3 роли

3 артефакта

5 событий (процессы)



Scrum Master

Scrum-мастер



24 H

Ежедневный

Роли

- Владелец продукта
- Скрам-мастер
- Команда

Артефакты

- Беклог продукта
- Беклог спринта
- Инкремент продукта

Процессы

- Планирование спринта
- Обзор спринта
- Ретроспектива
- Скрам-митинг
- Спринт



владе
проду



Бэкло
продук

Спринта

продукта)

Владелец продукта (Product owner)

- Представитель бизнеса (это может быть другой отдел или целая компания).
- Знает, как должен работать или выглядеть конечный продукт и заинтересован в его качестве, отвечает за максимальную его ценность.



Scrum-мастер (Scrum Master)

- Участник команды, обучает команду и компанию фреймворку.
- Согласовывает с представителем бизнеса изменения или внедрение какого-либо решения
- Проводится каждый день в фиксированное время
- Рекомендуется проводить (стоя) в течение 10-15 минут
- Если что-то нужно обсудить, назначается время после Scrum'a

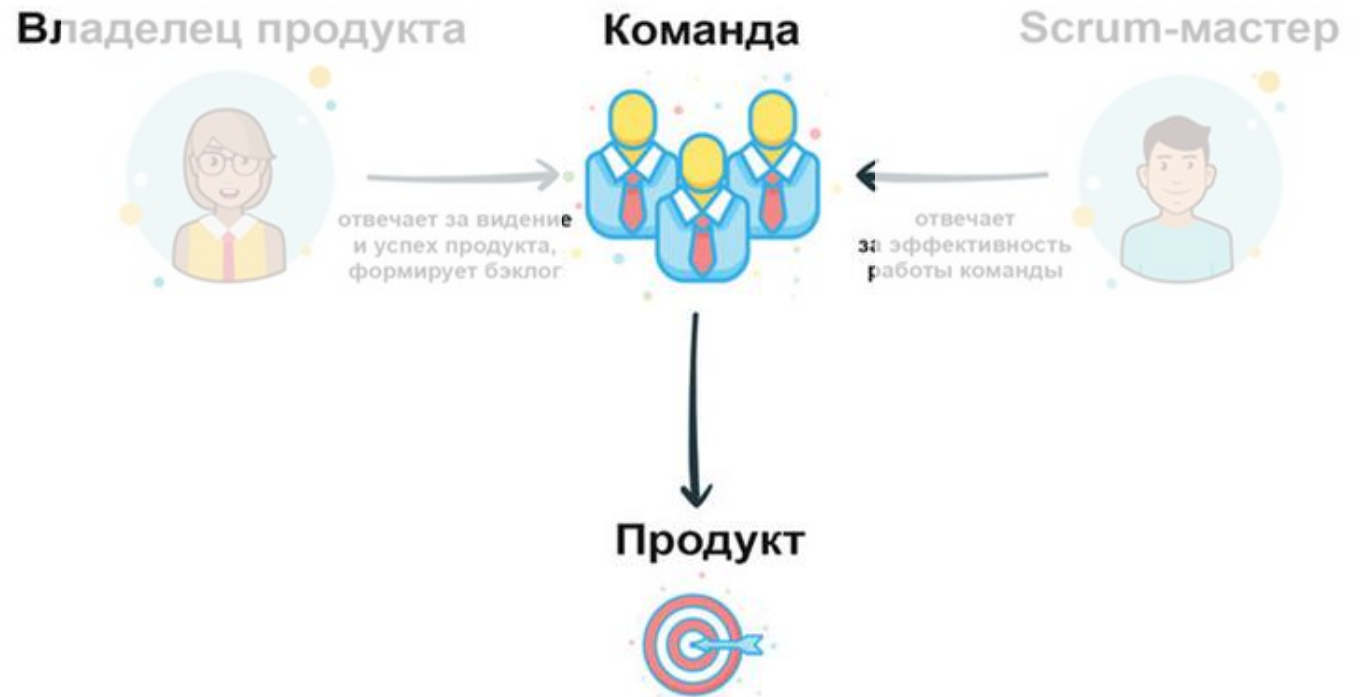


Scrum Master спрашивает каждого участника Scrum'a:

- *Что ты делал?*
- *Что ты собираешься делать?*
- *Какие были проблемы?*

Команда (Team)

- Профессионалы,
выполняющие конкретный
объем работы для создания
продукта, который должен
соответствовать
требованиям.
от 3 до 9 человек



Product Backlog

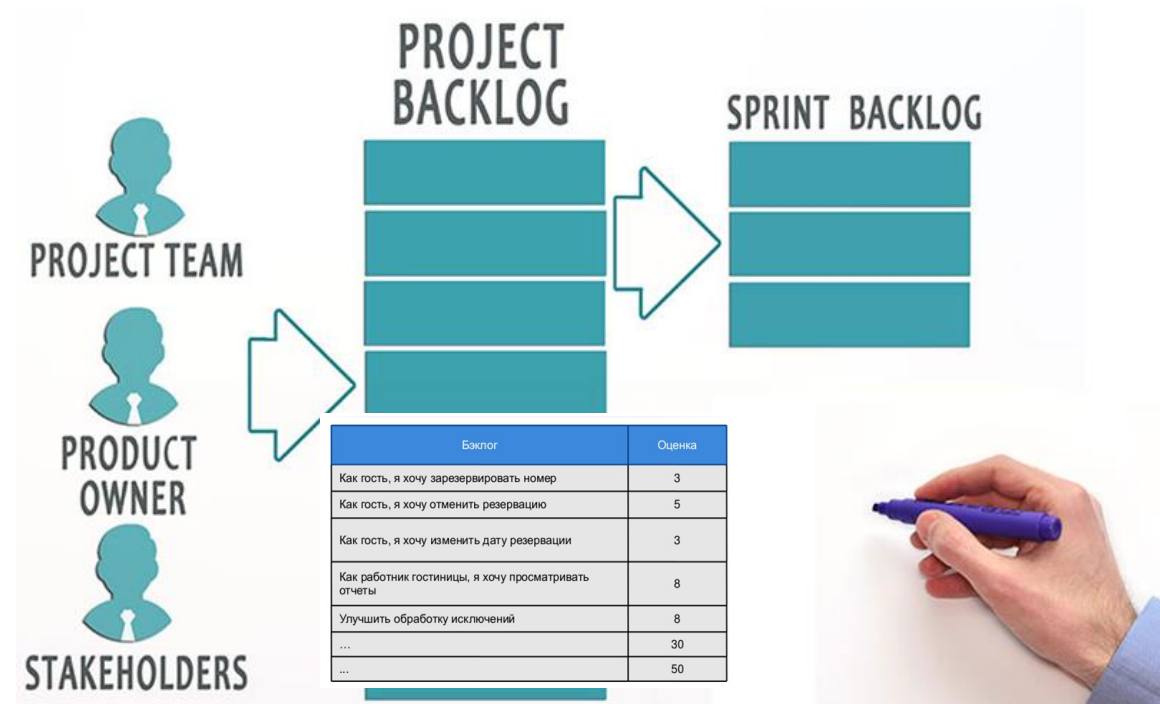
(описание продукта)

- **Полный список задач и понятные требования** для создания конечного продукта
- Product backlog **один** на весь релиз
- Им владеет менеджер продукта (“**product owner**”)
- **Он не статичен** – записи можно добавлять, удалять, менять им приоритет - живой документ
- **Общедоступен**, но поддерживается одним человеком

| Бэклог | Оценка |
|---|--------|
| Как гость, я хочу зарезервировать номер | 3 |
| Как гость, я хочу отменить резервацию | 5 |
| Как гость, я хочу изменить дату резервации | 3 |
| Как работник гостиницы, я хочу просматривать отчеты | 8 |
| Улучшить обработку исключений | 8 |
| ... | 30 |
| ... | 50 |

Sprint Backlog

- Задачи, которые команда берет в работу из Product Backlog на время спринта
- Описывает задачи, запланированные командой на спринт
- Задачи – действия, необходимые для реализации запланированной на спринт функциональности
- В описание задачи входит ее оценка
- Члены команды заносят в свой бэклог работы на свой выбор
- Задачи никогда не назначаются принудительно



Ретроспектива спринта

- Периодический пересмотр того, что работает, а что нет
- После каждого спринта
- Участвуют все члены команды
- Цель - осознать:
 - Что было хорошо?
 - Что могло бы быть лучше
- 15-30 минут



Спринт (Sprint)

- Фаза разработки состоит из нескольких итераций – **спринтов.**
- Обычно спринт длится **1-2-4 недели.**
- Этапы:**
 - Планирование
 - Разработка
 - Демонстрация
 - Ретроспектива



Ежедневный Scrum (Daily Scrum)

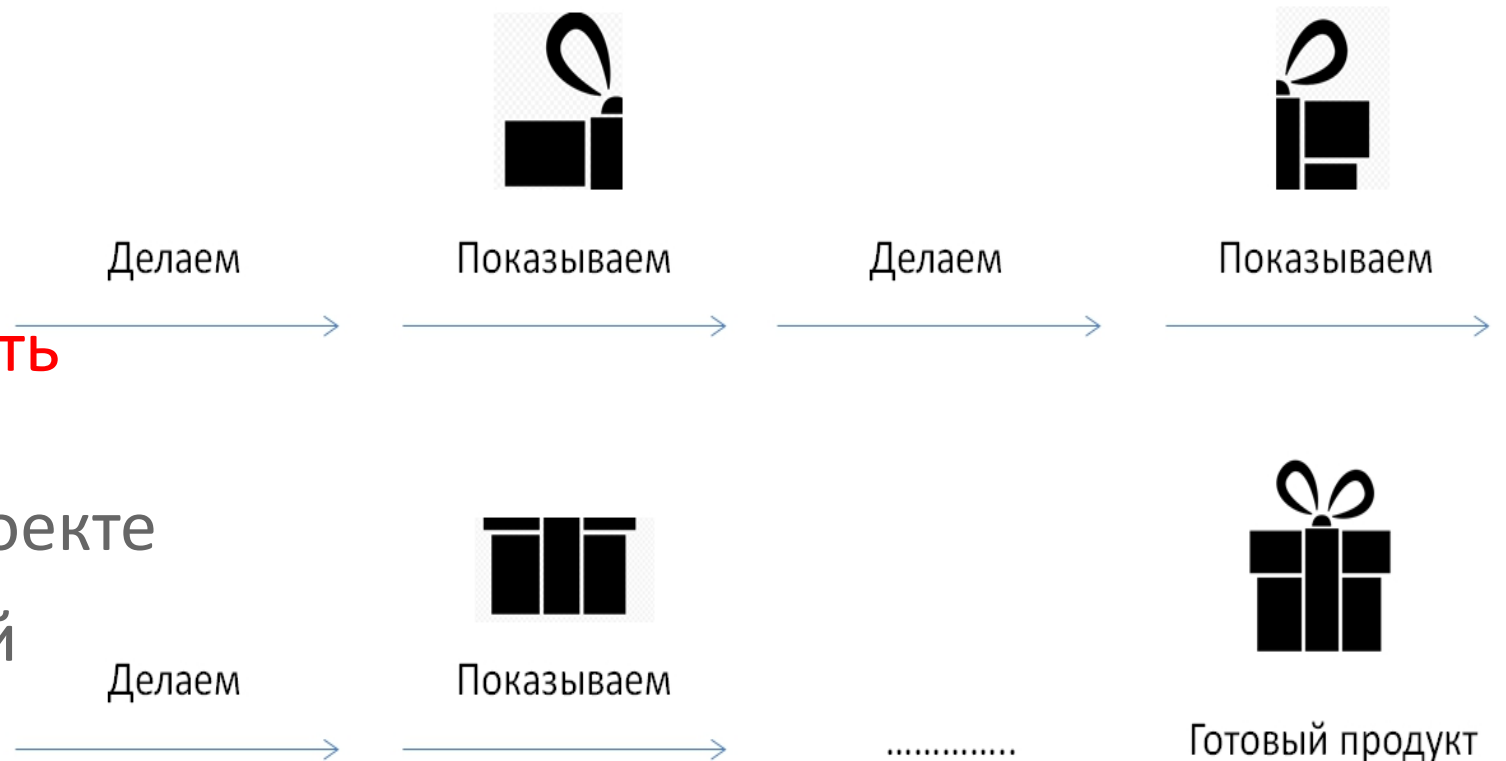
- Проводится каждый день в фиксированное время
- Рекомендуется проводить **стоя** в течение **10-15 минут**
- Если что-то **нужно обсудить**, назначается время **после Scrum'a**



Демонстрация (ревью)

- В конце каждого спринта проводится ревью
- Это демонстрация реализованной функциональности
- В ней может участвовать любой человек, задействованный в проекте
- В идеале после каждой демонстрации можно отправлять продукт заказчику

Просто Scrum

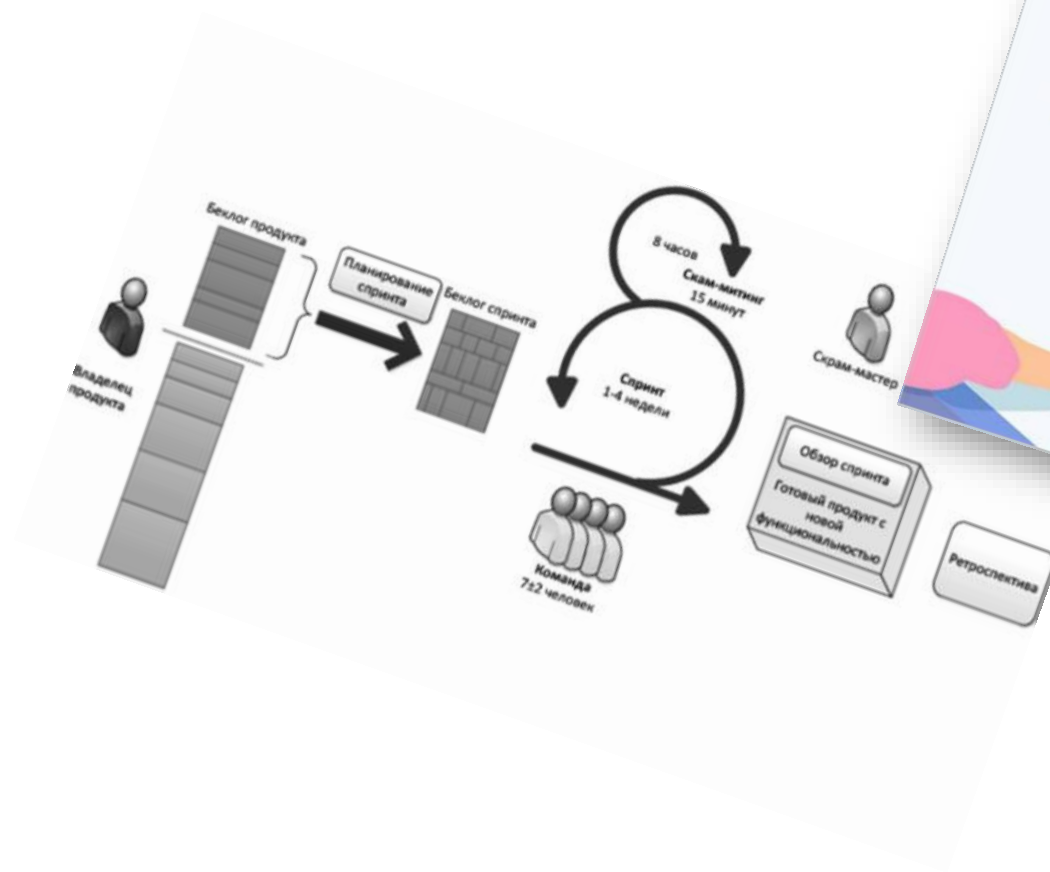


Вывод по Scrum:

- **Scrum – это «подход структуры».** Над каждым проектом работает универсальная команда специалистов, к которой присоединяется еще два человека: владелец продукта и scrum-мастер. Первый соединяет команду с заказчиком и следит за развитием проекта; это не формальный руководитель команды, а скорее куратор. Второй помогает первому организовать бизнес-процесс: проводит общие собрания, решает бытовые проблемы, мотивирует команду и следит за соблюдением scrum-подхода.
- **Scrum-подход** делит рабочий процесс на **равные спринты – обычно это периоды от недели до месяца, в зависимости от проекта и команды.** Перед спринтом формулируются задачи на данный спринт, в конце – обсуждаются результаты, а команда начинает новый спринт. Спринты очень удобно сравнивать между собой, что позволяет управлять эффективностью работы.

(должно было быть в Лекции 7)

- SCRUM
- Канбан



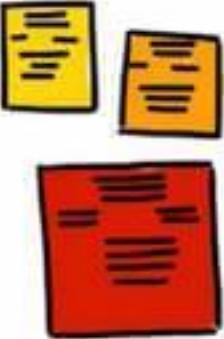






Канбан (канбан, яп. カンバン) «табличка» (копирайтн Тайичи Оно (Taiichi Ono), 1953)

- Канбан намного менее строгий, нежели Scrum – он не ограничивает время спринтов, нет ролей, за исключением владельца продукта. Канбан даже позволяет члену команды вести несколько задач одновременно, чего не позволяет Scrum. Также никак не регламентированы встречи по статусу проекта – можно делать это как Вам удобно, а можно не делать вообще.
- Для работы с Канбан необходимо определить **этапы потока операций** (workflow). В Канбан они изображаются как **столбцы**, а **задачи** обозначают специальные **карточки**. Карточка перемещается по этапам, подобно детали на заводе, переходящей от станка к станку, и на каждом этапе процент завершения становится выше. На выходе мы получаем готовый к поставке заказчику элемент продукта. Доска со столбцами и карточками может быть как настоящей, так и электронной – Канбан не накладывает никаких ограничений на пользователей.

4 базовых принципа Канбан

- **Карточки:** Для каждой задачи создаётся индивидуальная карточка, в которую заносится вся необходимая информация о задаче. Таким образом, вся нужная информация о задаче всегда под рукой.
- **Ограничение на количество задач на этапе:** Количество карточек на одном этапе строго регламентировано. Благодаря этому сразу становится видно, когда в потоке операций возникает «затор», который оперативно устраняется.
- **Непрерывный поток:** Задачи из бэклога попадают в поток в порядке приоритета. Таким образом, работа никогда не прекращается.
- **Постоянное улучшение («кайзен» (kaizen)):** Концепция постоянного улучшения появилась в Японии в конце XX века. Её суть в постоянном анализе производственного процесса и поиске путей повышения производительности.

КАНБАН

| Новые Запросы | Готово к работе | В работе | | Готово к тестированию | Тестирование | | Завершено |
|---|---|---|--|---|---|-------------------|---|
|  |  | В ПРОЦЕССЕ | ГОТОВО |  | ПРОВЕРКА | СОГЛА- СОВАНИЕ |  |
| | |  |  | |  | | |

1-1 Задача

15.03

29.03

3

≡

▶

≡

≡

≡

★

Критерии
качества

Критерии
качества

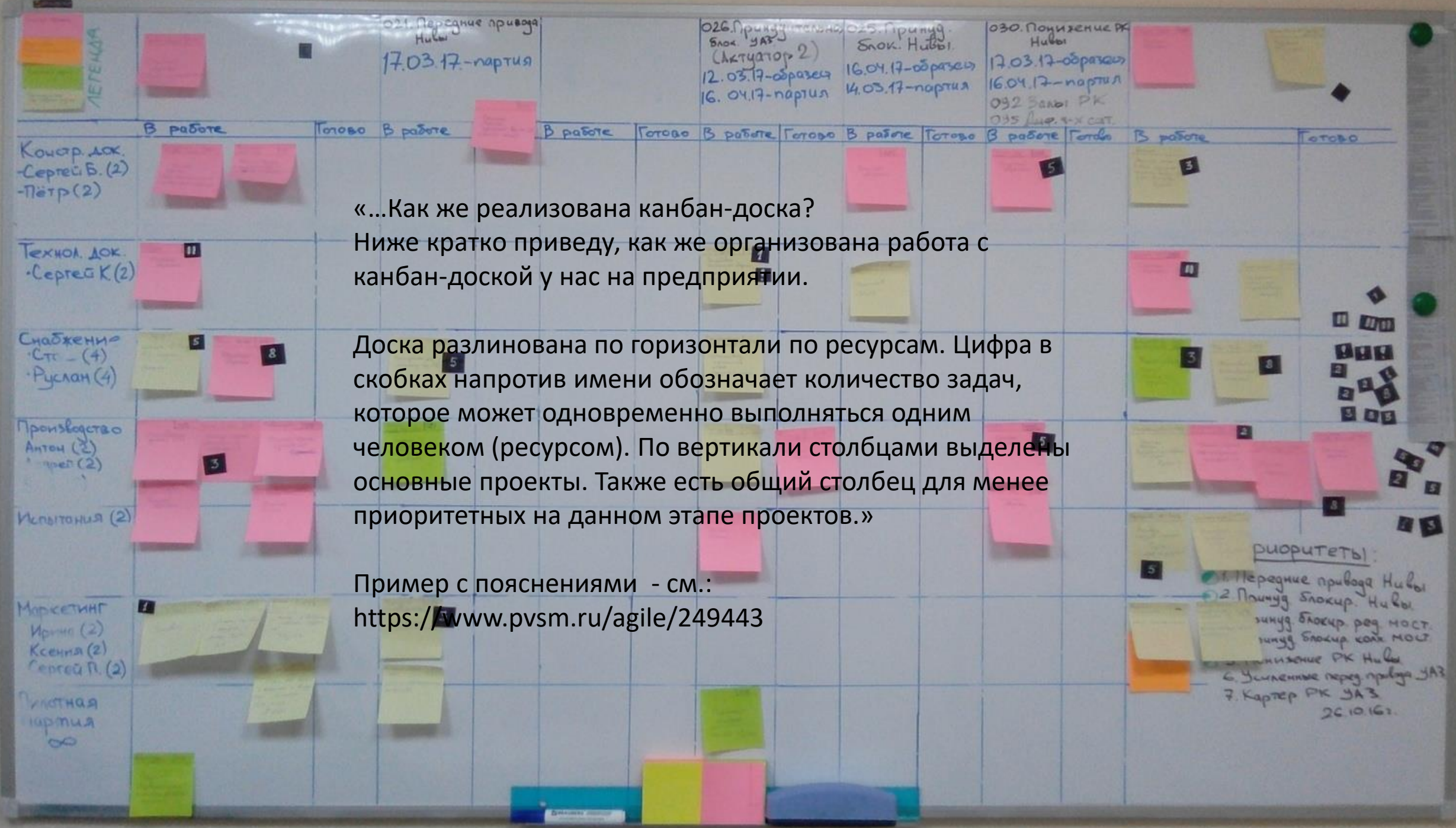


| ОЧЕРЕДЬ | В РАБОТЕ (3) | ЗАКОНЧЕНО |
|--|---|--|
|     |    |   |

**В работе -
не более 5
карточек**

Разные цвета карточек - разные типы работ





«...Как же реализована канбан-доска?

Ниже кратко приведу, как же организована работа с канбан-доской у нас на предприятии.

Доска разлинована по горизонтали по ресурсам. Цифра в скобках напротив имени обозначает количество задач, которое может одновременно выполняться одним человеком (ресурсом). По вертикали столбцами выделены основные проекты. Также есть общий столбец для менее приоритетных на данном этапе проектов.»

Пример с пояснениями - см.:
<https://www.pvsm.ru/agile/249443>

изготовить УАЗ / Цанговый патрон

Отливки
корпусов и крыш
Рисунки
Поиск обр.

Изготовить
цанговый
патрон
Сибирский

5

Рисунки
кратчайшими

БЮДЖЕТ ДОСТАТКА И ПЕЧАТЬ

БЮДЖЕТ ДОСТАТКА
ДОСТАТКА ПОС. 2
САТЕЛЛИТА

103

ВЕРСИЯ ПЛАТФОРМЫ
ИНСТРУКЦИИ

(Решение не дано)

Приоритеты:

1. Передние приводы Нивы
 2. Принуд. блокир. Нивы.
 3. Принуд. блокир. ред. мост.
 4. Принуд. блокир. колх. мост.
 5. Снижение РК Нивы.
 6. Усиленные перед. приводы УАЗ.
 7. Картер РК УАЗ.
- 26.10.16г.

Опытный образец

Подготовка
выпуска пилотной
партии

Пилотная партия

Промежуточные,
параллельные задачи

ЛЕГЕНДА

Грибы стандарт / 012

Опытный
образец
(Обработка термичка →
→ испытания)

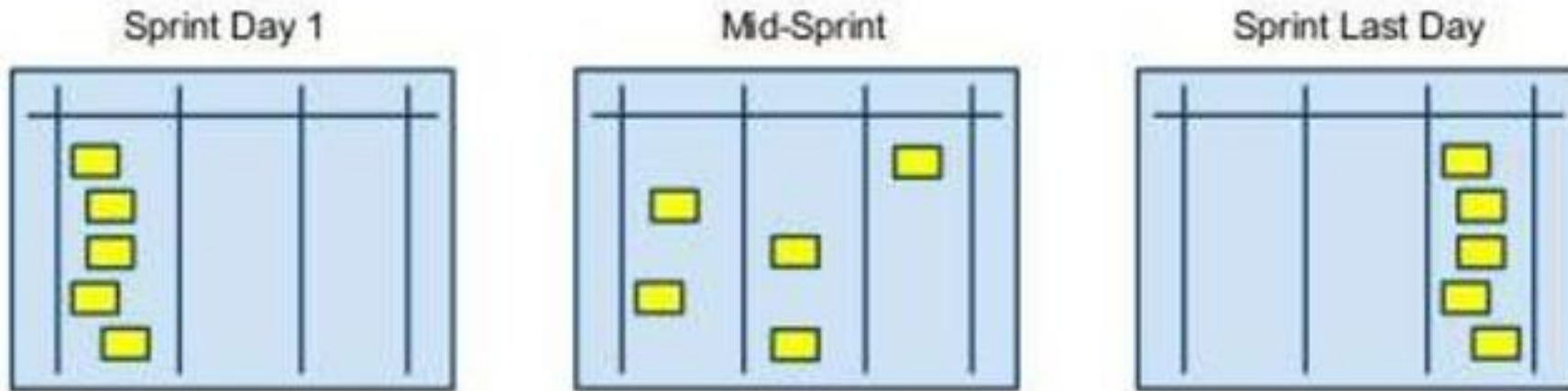
В работе

Вывод по Kanban

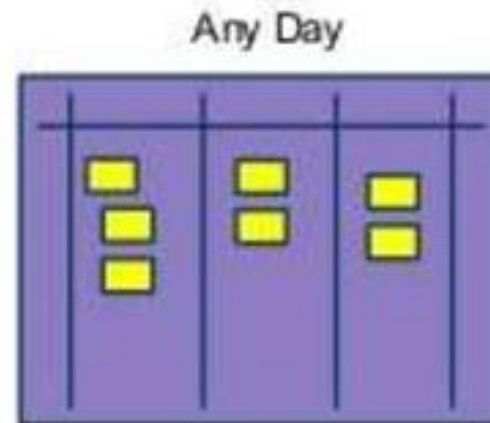
- **Kanban – это «подход баланса».** Его задача – сбалансировать разных специалистов внутри команды и избежать ситуации, когда дизайнеры работают сутками, а разработчики жалуются на отсутствие новых задач.
- Вся команда едина – в kanban нет ролей владельца продукта и scrum-мастера. Бизнес-процесс делится не на универсальные спринты, а на стадии выполнения конкретных задач: «Планируется», «Разрабатывается», «Тестируется», «Завершено» и др.
- Главный показатель эффективности в kanban – это среднее время прохождения задачи по доске. Задача прошла быстро – команда работала продуктивно и слаженно. Задача затянулась – надо думать, на каком этапе и почему возникли задержки и чью работу надо оптимизировать.
- Для визуализации agile-подходов используют доски: физические и электронные. Они позволяют сделать рабочий процесс открытым и понятным для всех специалистов, что важно, когда у команды нет одного формального руководителя.

SCRUM vs KANBAN

Scrum



Kanban



Отличия

| Scrum | Kanban |
|---|--------------------------|
| Время итерации ограничено | Ограничений нет |
| Команда обязана выполнить объем | Опционально |
| Метрика - производительность | Время прохождения задачи |
| Кроссфункциональные команды | Опционально |
| Обязательная декомпозиция для завершения в спринт | Ограничений нет |
| Наличие burndown-диаграммы | Не нужно диаграмм |
| Ограничение НЗР за спринт | Прямое по операциям |
| Оценки задач обязательны | Оценки задач опциональны |
| Предписаны 3 роли | Нет предписанных ролей. |
| Приоритезированный Product Backlog | Опционально |

Отличия: SCRUM

- В Scrum вы разделяете свою организацию на небольшие, кросс-функциональные, самоорганизующиеся команды. В Канбане нет жесткого требования про кросс-функциональные команды
- В Scrum есть особенные роли, а для Канбана подойдут обычные
- В Scrum ежедневное стендап-собрание — это сердечный ритм проекта. В Канбане планерки проводить жестко не требуются
- В Scrum вы должны разделить свою работу на список небольших конкретных результатов. В Канбане достаточно разделить работу на части, написать каждую часть на стикер и повесить его на стену. Они не должны быть результатами.
- В Scrum вы должны разбить время на короткие итерации фиксированной длины (1-4 недели), с разными версиями минимально-работоспособного продукта (MVP), которые вы будете демонстрировать после каждой итерации. В Канбане же работа непрерывная, а не итеративная.
- В Scrum вы должны отсортировать список поставляемых результатов по приоритету и оценить необходимые усилия для каждого элемента. В Канбане нет жесткого требования оценивать работу.
- В Scrum, основываясь на информации, полученной на проверке релиза после каждой итерации, оптимизируется план выпуска и обновляются приоритеты вместе с заказчиком. В Канбане этого не происходит.
- В Scrum у вас есть фиксированные события, такие как начало, планирование, обзор и ретроспектива. В Канбане же есть задающие ритм каденции.
- В Scrum рабочий процесс всегда должен оптимизироваться после каждой проведенной ретроспективы. В Канбане же это не обязательно, но можно провести в каденциях.
- В Scrum должен быть список невыполненных работ по продукту и график сгорания задач, чего в Канбане нет и в помине.

Отличия: KANBAN

- Канбан фокусируется на представлении рабочего процесса команды, давая им возможность визуализировать его и улучшить как можно скорее. Scrum имеет фиксированный процесс и церемонии.
- Канбан позволяет использовать любые именованные столбцы в вашей доске, чтобы проиллюстрировать, где находится каждый элемент, продукт или услуга в рабочем процессе. Scrum фокусируется на результатах с конкретными столбцами: “бэклог”, “бэклог спринта”, “работа в процессе” и “выполненная работа”.
- Канбан ограничивает “работу в процессе» WIP-лимитом. В Канбане необходимо установить ограничения на количество работ, которые могут выполняться в каждом столбце рабочего процесса. В Scrum нет никаких правил на этот счет.
- Одна из самых важных вещей в Канбане — это измерение среднего времени выполнения одного элемента, называемое “временем цикла”. Это очень важно, потому что это дает вам возможность оптимизировать процесс, чтобы сделать работу как можно короче и предсказуемее.
- В Канбан можно вносить изменения по мере необходимости. В Scrum изменения не должны прерывать спринт.

SCRUM vs KANBAN

Scrum лучше подходит для проектов по разработке продуктов. По сути, вы заранее определяете работу для следующего спринта. Затем вы блокируете спринт, выполняете всю работу, и после пары спринтов ваша очередь должна быть пуста.

Канбан лучше подходит для поддержки производства. Лимит здесь не определяется спринтом, а размером очереди каждого столбца доски — ограничением по незавершенным работам (WIP-лимит). Это значит, что вы можете изменить элементы в очередях в любое время, и что у работы нет конца — она идет сплошным потоком.

ВОПРОСЫ?

