# Project Manager Manifest

## Содержание

# Основная терминология

**Проект -** это ограниченное во времени предприятие, направленное на создание уникальных продуктов и услуг или получение принципиально новых результатов.

Проекты:

- имеют набор заданий с четко определенным конечным результатом и сроком выполнения;
- связаны с созданием, обновлением или корректировкой конкретного документа, процесса, результата или другой отдельной единицей работы;
- имеют заранее определенную область, ограниченную конкретным результатом;
- повышают качество, эффективность, улучшают управление затратами или повышают удовлетворенность клиентов конкретным, заранее определенным образом.

**Домен** - это имя сайта. (доменное имя, доменный адрес) — это, простыми словами, «название» сайта. Понятия «домен» и «сайт» часто путают, но это не одно и то же. Сайт — это веб-страницы, которые отображаются в интернете, т. е. контент. А домен сайта — это его уникальный «адрес». Если у вашего сайта не будет домена, пользователи просто не найдут к нему дорогу и не увидят содержимое.

**CSS –** это формальный язык, служащий для описания оформления внешнего вида документа, созданного с использованием языка разметки (HTML, XHTML, XML). Название происходит от английского Cascading Style Sheets, что означает «каскадные таблицы стилей».

**Продукт** — товар или услуга, которую можно предложить для рынка, и которая будет удовлетворять потребности потребителей.

**SLA** - Service Level Agreement соглашение об уровне оказываемых услуг – это документ, который составляется между двумя сторонами (потребитель – исполнитель) и отражает условия предоставляемых услуг

**Digital** - интернет-пространства для распространения или обмена информацией различными способами.

**Seo** - Это большой комплекс работ по увеличению видимости сайта по поисковым запросам в выдаче поисковых систем.

**CMS** - приложение для управления сайтом и/или его содержимым. Позволяет пользователю, который не владеет языками программирования или разметкой HTML, редактировать содержимое сайта. CMS расшифровывается как Content Management System, переводится — система управления контентом. Как правило, CMS реализовано в виде веб-приложения, однако, это может быть и программа, которая устанавливается на операционную систему.
Примеры наиболее популярных систем управления:
➔ WordPress
➔ 1С-Битрикс
➔ Joomla
➔ Drupal
➔ OpenCart
➔ MODx

➔ Magento
➔ TYPO3
➔ Tilda

**Фича** — это дополнительная функция или особенность продукта. Различают несколько видов:
- "базовая фича", которая является неотъемлемой частью продукта — например, возможность ответить на сообщение в мессенджере или поставить лайк в инстаграме;
- "киллер-фича" — это то чем мы значительно отличаемся от конкурентов;
- "вау-фича" - то, чего наш клиент не ждет, но такая функция может его покорить и сделать нашим клиентов навсегда.
Фича описывает сам функционал и как с ним будет взаимодействовать пользователь, какую ценность он получит.

**Backlog (Бэклог)**
Это полный список задач, которые надо выполнить в рамках проекта. Задачи в бэклоге четко организованы и выполняются в приоритетном порядке.

**Тикет -** задача в таск трекере.

**Спринт** — это заранее определенное количество времени для завершения одного цикла, итерации или задачи в рамках проекта.

**Итерация** - повторение. В итерационных циклах на каждом шаге вычислений происходит последовательное приближение и проверка условия достижения искомого результата.

**CJM (Client journey map) -** Схема клиентского пути.

**MVP (Minimal Viable Product) -** минимально жизнеспособный продукт

**Baseline (Базовый план)**
Это исходный план или оценка сроков, бюджета, масштаба и целей проекта. Используя его, можно измерять прогресс и вносить коррективы.

**Bottleneck (Узкое место)**
Оно сужает и ограничивает поток, темп и мощность проекта. Это могут быть заинтересованные стороны, у которых слишком много задач для утверждения, или процессы, которые могут быть перегружены.

**Contingency Plan (План на случай непредвиденных обстоятельств)**
Он содержит действия на случай чрезвычайных ситуаций, позволяет эффективно управлять любыми краткосрочными и долгосрочными сбоями или проблемами, которые могут возникнуть на протяжении всего проекта.

**Critical Path Method, CPM (Метод критического пути)**
Это метод моделирования для пошагового планирования проекта. Это алгоритм, основанный на том, сколько времени займет выполнение каждой задачи, и помогающий определить порядок задач и сроки для всего проекта.

**Deliverables (Практические результаты)**
У всех проектов есть один или несколько практических результатов, которые получают на протяжении всего пути или в конце проекта.

**Kickoff Meeting (Стартовая встреча)**
Это первая встреча в начале проекта с заинтересованными сторонами и членами команды.

**Meeting Minutes (Протокол собрания)**
Это заметки, сделанные во время встречи, которые включают ключевые выводы и следующие шаги.

**Milestone (Вехи)**
Это ключевые моменты на временной шкале проекта, которые можно использовать в качестве маркеров для оценки прогресса.

**PERT Chart (Диаграмма PERT)** Program Evaluation Review Technique
Техника оценки и анализа проектов. Она представляет собой инструмент управления проектами, используемый для простой организации задач и сроков.

**RACI Chart (Диаграмма RACI)**
RACI означает Responsible, Accountable, Consulted, Informed (ответственный, подотчетный, консультируемый, информированный). Эта диаграмма позволяет распределить роли и обязанности между заинтересованными сторонами и членами команды в отношении деятельности по проекту.

**Work Breakdown Structure, WBS** (Иерархическая структура работ)
Это легко усваиваемая и организованная разбивка проекта на иерархические разделы и подзадачи.

**RC-версия** (Релиз-кандидат, release candidate, пре-релиз, Pre) – это стадия кандидат на то, чтобы стать финальной. Программы на данной стадии прошли все комплексное тестирование, благодаря чему исправлены все критические ошибки, однако вероятность нахождения некоторых ошибок остается.

**SOLID** - принцип программирования
- 1 объект=1 функция
- сущности расширяются, но не меняются
- разделение интерфейсов
- инверсия зависимостей

**REST** - Интерфейс управления информацией для API.

**API** - программный интерфейс для взаимодействия систем

**Json** - текстовый формат обмена данными

**XML** - Структура данных

**Postman** — это HTTP-клиент для тестирования API. HTTP-клиенты тестируют отправку запросов с клиента на сервер и получение ответа от сервера. API (Application Programming Interface) — это интерфейс для обмена данными с сервера между двумя приложениями или компонентами ПО. Тестировщикам Postman помогает в проектировании дизайна API и создании mock-серверов (имитаторов работы приложения).

**Swagger** – это язык описания интерфейса для описания RESTful API, выраженных с помощью JSON. Swagger используется вместе с набором программных средств с открытым исходным кодом для проектирования, создания, документирования и

использования веб-служб RESTful.

**Сокет** - IP адрес + N порта

**Git** - система управления версиями

**SQL** - язык запросов к базе данных

**http/https** - протоколы шифрования

**Хо́стинг** (англ. hosting) — услуга по предоставлению ресурсов для размещения информации на сервере

**IP-адрес** – это уникальный адрес, идентифицирующий устройство в интернете или локальной сети.

**IP (intellectual property) -** интеллектуальная собсвенность
https://nris.ru/blog/intellektualnaya-sobstvennost-v-sfere-it/

# IT команды

— Дизайнер
— Разработчик
— Системный администратор
— DevOps
— Тестировщик
— Системный аналитик
— Бизнес аналитик
— Аналитик
— Маркетолог
— Саппорт
— Менеджер продукта
— Менеджер проекта

“Классическая” орг структура:

| | **CEO** | | | |
|---|---|---|---|---|
| | **Директор по продуктам** | **Директор по проектам** | **Технический директор** | **Арт-директор** |
| **Продукт 1** | Продакт | Проджект | Разработка | Дизайн |
| **Продукт 2** | Продакт | Проджект | Разработка | Дизайн |
| **Продукт 3** | Продакт | Проджект | Разработка | Дизайн |

## Agile

Манифест - https://agilemanifesto.org/

Гибкий подход к разработке программного обеспечения, который часто применяют в небольших командах и больших организациях.

Управление проектами по методике agile — это итеративный подход к управлению разработкой ПО, ключевую роль в котором играют непрерывные релизы и обратная связь от клиентов при каждой итерации.

Agile флоу: Требования > План > Работа > Ревью > Повтор

Эджайл это про то, как разбить ваш огромный проект на ряд маленьких задач (обычно их называют пользовательскими историями) и определить наиболее приоритетные.

Определение приоритетов очень важно, по сути это и есть самое главное в гибкой методологии разработки. Вам нужно убедиться, что команда сфокусирована на одном спринте или на наиболее важном на данный момент результате. Это будет гарантией того, что вы достигнете ваших бизнес-целей. Так ваша команда не потеряется в потоке запросов и требований и будет знать, что каждая подзадача, над которой она работает в данный момент, важна для прогресса всего проекта.

Agile хорошо подходит для проектов, в которых некоторые детали неясны с самого начала. Это делает его подходящим для отраслей, имеющих дело с постоянными или непредсказуемыми изменениями, или для команд, создающих новый продукт. Более традиционные стили управления лучше подойдут для бизнес-идей, имеющих строгие ограничения по времени или фиксированный бюджет.

Преимущества управления проектами по методике agile

- Более быстрые циклы обратной связи.
- Выявление проблем на ранней стадии.
- Больше возможностей повысить удовлетворенность клиентов.
- Значительное ускорение выхода на рынок.
- Повышение прозрачности и подотчетности.
- Повышение производительности выделенных команд с течением времени.
- Гибкая расстановка приоритетов, ориентированная на поставку ценности.

Недостатки методики agile

- Критический путь и зависимости между проектами могут быть определены не так четко, как в каскадной модели.
- Организации требуется время на освоение.
- Полноценное внедрение методики agile с конвейером непрерывного развертывания требует затрат и определения множества технических зависимостей.

### Ключевые принципы Agile

- Люди и их взаимодействие важнее процессов и инструментов.
- Работающий продукт важнее документации и отчётности.
- Сотрудничество с заказчиком важнее соблюдения формальных условий.
- Готовность к изменениям важнее, чем следование плану.

### Принципы agile разработки

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

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

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

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

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

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

7. Работающий продукт — основной показатель прогресса.

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

9. Постоянное внимание к техническому совершенству и качеству

10. проектирования повышает гибкость проекта.

11. Простота — искусство минимизации лишней работы — крайне необходима.

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

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

**LeSS (Large-Scale Scrum)**
в переводе на русский означает «скрам на больших масштабах». Это фреймворк, позволяющий применить принципы скрама в больших проектах. В зависимости от количества команд используется либо LeSS (2–8 команд), либо LeSS Huge (больше 8 команд).

Переход от Scrum к Less и Safe
https://simpleone.ru/blog/perehod-ot-scrum-k-less-i-safe/#:~:text=LeSS%20(Large%2DScale%20Scrum),Huge%20(%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B5%208%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4)

https://www.atlassian.com/ru/agile/agile-at-scale/less

**Scaled Agile Framework® (SAFe®)**
Scaled Agile Framework® (SAFe®) — это набор организационных шаблонов и шаблонов рабочих процессов для реализации agile-методик в масштабе всей компании.

В ее основу легли три основных блока знаний: гибкая (agile) разработка программного обеспечения, «бережливая» (lean) разработка продукции и системное мышление.

Согласно SAFe, существует пять основных показателей встроенного качества:
- процесс
- качество архитектуры и дизайна
- качество кода
- качество системы
- качество релиза.

## Принципы SAFe

*Принцип № 1. Смотрите с точки зрения экономики*
Определение последовательности работ для получения максимальной выгоды, понимание экономических компромиссов и работу в рамках «бережливых» бюджетов.
Необходимо чтобы каждый человек в цепочке принятия решений понимал экономические последствия задержек.

*Принцип № 2. Применяйте системное мышление*

При решении вопросов, которые касаются компании, занимающейся построением системы, платформа SAF рекомендует учитывать людей, менеджмент и процессы организации.

*Принцип № 3. Допускайте вариативность; сохраняйте варианты*
Концепция вариативного проектирования.

*Принцип № 4. Выполняйте инкрементальные сборки с быстрыми циклами обучения, интегрированными в процесс*
Цикл Уолтера Э. Шухарта планируй — делай — проверяй — корректируй, который является схемой для постоянного улучшения качества и механизмов контроля вариативности разработки.

*Принцип № 5. Контрольные точки должны основываться на объективной оценке работающих систем*
Демонстрация действующей рабочей системы предоставляет более надежные основания для принятия решений, чем документ с требованиями или другая поверхностная оценка успеха. Заинтересованные стороны, с ранних этапов участвующие в принятии решений о технической реализации, способствуют построению доверительных отношений и поддерживают системное мышление.

*Принцип № 6. Визуализируйте и ограничивайте незавершенные работы (WIP), уменьшайте объем работ и управляйте длиной очередей*
Применительно к разработке программного обеспечения это означает ограничение количества параллельных работ, сложности каждого элемента работы и общего объема работы, выполняемой в данный момент времени.

*Принцип № 7. Применяйте каденции, выполняйте синхронизацию с помощью кросс-доменного планирования*
Agile-команды применяют каденции с помощью спринтов или итераций. Создание каденции для всех возможных работ позволяет уменьшить сложность, устранить неопределенность, выработать автоматизм, обеспечить качество и постепенно прививать сотрудничество.

*Принцип № 8. Раскройте внутреннюю мотивацию работников умственного труда*
Речь идет о раскрытии потенциала команд и замене командно-административного мышления руководства на обучающий и помогающий подход к работе с командами.

*Принцип № 9. Децентрализуйте принятие решений*
Руководители должны сохранять свои полномочия по принятию решений по вопросам стратегической важности и позволять командам решать все остальные вопросы.

# Scrum

https://scrumguides.org/

## Обзор

Scrum — это методология Agile, предназначенная для разработки продуктов в среде, подверженной изменениям.

Методику Scrum чаще всего применяют команды разработчиков, но принципы и опыт ее использования можно применить к командной работе любого рода. Scrum часто представляют как платформу для управления проектами по методике Agile. Участники команды Scrum проводят собрания, используют специальные инструменты и принимают на себя особые роли, чтобы организовать работу и управлять ею.

Понятия Scrum и Agile часто путают, потому что Scrum строится вокруг идеи о постоянном совершенствовании, которое является главным принципом Agile. И все же Scrum — это методика работы, а Agile — это образ мышления.

Согласно Scrum, команда не знает всего в начале проекта, но будет развиваться, извлекая уроки из опыта. В структуре Scrum заложена та свобода, с которой команды приспосабливаются к меняющимся условиям и требованиям пользователей. Рабочий процесс предусматривает изменение приоритетов и короткие циклы релиза, что способствует постоянному обучению и совершенствованию команды.

В основе методологии лежат три принципа:

1. Прозрачность. Все вовлеченные в процесс игроки имеют полный доступ ко всей информации.
2. Адаптация. Работа по проекту в любой момент может поменять вектор без потери производительности и времени.
3. Обновление. Команда стремится постоянно улучшать продукт и сам процесс разработки.

Ключевые принципы Scrum:

- Деление работы на части, которые называются спринтами (обычно в спринте две недели);
- Спринты планируются исходя из требований для этого конкретного момента времени;
- Относительная оценка времени выполнения работ;
- Ревью каждого спринта, чтобы понять, как он прошел и что можно было бы улучшить;
- Фидбек по поставляемому продукту;
- Ежедневные собрания (очень короткие).

Состав scrum-команды предполагает три отдельные роли: владелец продукта, scrum-мастер и команда разработчиков.

Поскольку scrum-команды сочетают в себе множество функций, в команду разработчиков также входят тестировщики, дизайнеры, UX-специалисты и инженеры по операциям.

*Владелец продукта Scrum*

Их задача — понимать требования бизнеса, клиента и рынка.

- Они составляют бэклог продукта и управляют им.
- Они тесно сотрудничают с руководством компании и командой, сообщая каждому участнику значение рабочих задач в бэклоге продукта.
- Они дают команде понятные указания, чтобы ее участники знали, какие возможности поставить следующими.
- Они решают, когда поставить продукт, стремясь делать это как можно чаще.


Владелец продукта несет ответственность за обеспечение максимальной ценности. Владелец продукта представляет бизнес и говорит разработчикам, чего требуется достичь. Доверие между этими двумя ролями имеет принципиальное значение.

Владелец продукта должен не только понимать клиента, но и иметь концепцию ценности, которую Scrum-команда поставляет клиенту. Кроме того, владелец продукта определяет баланс между потребностями других заинтересованных сторон в организации.

Таким образом, владелец продукта должен учесть все эти входные данные и расставить приоритеты в работе. Пожалуй, это его самая значимая обязанность.

*Scrum-мастер*
Scrum-мастера следят за применением Scrum в своих командах.

Scrum-мастер фокусируется на следующем.

Прозрачность. Для проведения эффективного анализа и адаптации важно, чтобы нужные люди могли видеть происходящее. Но добиться этого гораздо сложнее, чем кажется. В задачи Scrum-мастера входит обеспечение прозрачности работы Scrum-команды. Для этого, например, создаются карты историй или вносятся в Confluence идеи по результатам ретроспективы.

Эмпирический подход. Основная идея подходов Scrum и Agile состоит в том, что лучший способ планирования — работать и учиться на результатах. Эмпирический процесс непрост и требует, чтобы Scrum-мастер обучил Scrum-команду разбивать работу на составные части, четко описывать и оценивать конечные результаты.

Самоорганизация. Это означает, что команда разработчиков может самостоятельно наладить свой рабочий процесс. Самоорганизация приходит со временем, и для ее реализации требуется помощь и поддержка. Scrum-мастер стимулирует участников выходить из зоны комфорта и пробовать различные приемы и практики, такие как «покер делегирования». Этот прием помогает выявлять и оспаривать традиционные идеи о границах ролей и обязанностей.

Ценности. В Scrum выделяют пять ценностей (смелость, сосредоточенность, ответственность, уважение и открытость) не потому, что они хороши, а по той причине, что они создают атмосферу безопасности и доверия. Эта атмосфера необходима для успешного развития гибкости. Каждый участник Scrum-команды обязан следовать этим ценностям, а Scrum-мастер активно поддерживает такие начинания и напоминает всем об их важности.


Необходимо четко разделять управление процессом разработки(срам-мастер/менеджер проекта) и развитием продукта(владелец продукта/менеджер продукта).

И традиционный менеджер проекта, и scrum-мастер помогают команде в выполнении работы, но их подходы очень разные. Менеджер проекта устанавливает и отслеживает временные рамки и контрольные точки, сообщает о прогрессе и координирует взаимодействие в команде.

Вместе с тем он делает это с позиции контроля, играя более традиционную управленческую роль.

Scrum-мастер помогает команде улучшать и оптимизировать процессы, с помощью которых она достигает своих целей. Он действует как один из команды, как участник общего дела и в идеале не руководит в традиционном смысле слова. Лучшие scrum-команды практикуют самоорганизацию и потому плохо реагируют на управление «сверху вниз».

*Команда разработчиков Scrum*
На команды Scrum ложится вся основная работа.
Согласно scrum это небольшие команды, 5-7 человек.


## Артефакты Scrum


- Бэклог продукта

Владелец продукта постоянно обращается к бэклогу продукта, меняет в нем приоритеты и поддерживает его актуальность.

- Бэклог спринта

Список рабочих задач, пользовательских историй или исправлений багов, отобранных командой разработчиков для реализации в текущем цикле спринта, который может меняться по ходу спринта.
Бэклоги спринтов составляются следующим образом: задание из бэклога продукта делится на рабочие задачи меньшего размера, которые можно выполнить за спринт. Возьмем в качестве примера задание «создать страницу корзины для покупок», предполагающее множество подзадач по дизайну и разработке. Основное задание будет находиться в бэклоге продукта, а вспомогательные, такие как «создать макет визуального дизайна корзины для покупок» или «написать код для функционала корзины для покупок», помещаются в бэклог спринта.

- Инкремент с критериями готовности (цель спринта)

Готовый к использованию конечный продукт по итогам спринта. Желательно, тот, что можно продемонстрировать.
Каждому спринту всегда соответствует один инкремент, и он определяется на этапе планирования спринта. Инкремент существует вне зависимости от того, согласится ли команда выпустить его для клиента. Инкременты продукта имеют важнейшее значение. Они дополняют процессы CI/CD в рамках контроля версий и, когда это необходимо, отката версий.

- Диаграмма сгорания задач

Диаграмма Burndown (или Burnup) спринта не является официальным Scrum-артефактом, но используется многими командами для обмена информацией о продвижении к цели и ее отслеживания во время спринта. На диаграммах Burndown показаны задания, выполненные в течение спринта. Диаграммы Burndown помогают измерить скорость активного выполнения работы командой, чтобы ее участники понимали, смогут ли они завершить запланированную работу в срок либо нужно менять приоритеты заданий в спринте.

## Мероприятия Scrum

### Спринт

Спринты повышают управляемость проектов, позволяют командам поставлять высококачественные продукты быстрее и чаще, а также обеспечивают большую гибкость при адаптации к изменениям.

Основные этапы спринта Scrum:

- Планирование работы и будущих целей
- Создание заданий, выполнение которых поможет достичь эти цели
- Распределение заданий по спринтам с учетом их зависимостей и важности
- Выполнение заданий
- Обзор и анализ результатов для сравнения с целями
- Повторение этих шагов

*Цикл спринта*


1. Владелец продукта определяет цель спринта и задачи из бэклога продукта, при выполнении которых она будет достигнута.
   Затем команда создает план, согласно которому будут выполняться задачи бэклога, чтобы к окончанию спринта вся работа была завершена.

2. В течение спринта команда собирается на ежедневные Scrum-совещания (стендапы), чтобы обсудить ход работы.

3. По окончании спринта команда показывает выполненную работу на обзоре итогов спринта.

4. На ретроспективе команда может определить области, требующие улучшения в следующем спринте.

*Три правила автоматизации, которые часто используются в спринтах Jira.*

1. Еженедельно отправлять сообщение Slack со списком всех задач, еще открытых в спринте.
2. После завершения спринта назначить нерешенные задачи следующему спринту.
3. Если состояние задачи изменилось на In Progress (В работе) и спринт стал пустым, переместить задачу в следующий активный спринт.

### Организация бэклога

За это мероприятие, также известное как ведение бэклога, несет ответственность владелец продукта. Для этого владелец продукта и ведет список, изменяя в нем приоритеты и поддерживая его в актуальном виде на основании информации от пользователей и команды разработчиков, чтобы в любое время можно было приступить к работе над внесенными в него задачами.

Задачи должны быть оценены в часах или стори поинтах.

### Планирование спринта

*Состав участников:* команда разработчиков, Scrum-мастер, владелец продукта.

*Когда проходит:* в начале спринта.

*Продолжительность:* обычно около одного часа на каждую неделю итерации. Например, двухнедельный спринт нужно начинать с двухчасового собрания по планированию.

*Назначение:* на собрании по планированию закладывается основа для успешного спринта. На это собрание владелец продукта приносит бэклог продукта с расставленными приоритетами. Вместе с командой разработчиков он обсуждает каждую рабочую задачу, после чего группа совместными усилиями оценивает необходимые трудозатраты. Затем команда разработчиков дает прогноз на спринт, в котором указывает, какой объем работы из бэклога продукта она может выполнить. Этот объем работы становится бэклогом спринта.

Посвятите первую часть собрания по планированию спринта определению цели спринта, а не наполнению бэклога. Когда акцент делается на цели, а не работе, можно продумать альтернативные варианты достижения этой цели.

В конце встречи по планированию каждый участник должен четко понимать, что должно быть сделано в спринте и как можно добиться назначенных инкрементов.

Итого по результатам встречи у вас должны быть:

- цели спринта
- бэклог спринта
- оценка и распределение задач по ответственным
- всем понятны инкременты и что будет ожидаться на демо.

На собрании по планированию спринта проработайте все мельчайшие детали работы, которую предстоит сделать. Пусть участники команды набросают примерные задания по всем историям, багам и задачам, которые входят в спринт. Поощряйте обсуждения и придите к единому взгляду на план действий.

Чтобы собрание по планированию спринта прошло успешно, владелец продукта должен подготовиться: освежить в памяти выводы, сделанные при прошлом обзоре итогов спринта, ознакомиться с мнениями заинтересованных сторон и их концепцией продукта. Можно сказать, что они подготавливают почву для спринта. Чтобы текущая ситуация была прозрачна и понятна, бэклог продукта следует привести в соответствие с современными реалиями и скорректировать.

Шаблон собрания
https://www.atlassian.com/ru/software/confluence/templates/sprint-planning-meeting

### DSM, Ежедневное Scrum-совещание, или стендап.

*Состав участников:* команда разработчиков, Scrum-мастер, владелец продукта

*Когда проходит:* раз в день, как правило, утром.

*Продолжительность:* не более 15 минут. Не занимайте конференц-зал и не давайте участникам стендапа садиться. Если все будут стоять, собрание не займет много времени.

*Назначение:* стендап нужен для того, чтобы быстро сообщить всем о ситуации в команде. Это не полноценная планерка. Атмосфера должна быть легкой и непринужденной, но не бессодержательной. Пусть каждый участник команды ответит на следующие вопросы.

- «Что мне удалось завершить вчера?»
- «Над чем я буду работать сегодня?»
- «Есть ли препятствия в моей работе?»

Когда отчитываешься о прогрессе за вчерашний день перед коллегами, проявляется личная ответственность. Никто не хочет оказаться участником команды, который постоянно делает одно и то же и не движется вперед.

Стендап — подходящее время сообщить обо всем, что мешает вам достичь цели спринта, в том числе о блокерах.

Важно, чтобы каждый участник ежедневного стендапа до начала совещания знал, что он или она собирается сказать. Тогда стендап пройдет динамично, и никто не успеет заскучать. Сотрудники используют доски Jira и быстрые фильтры, чтобы ориентироваться в своих проектах. Для подготовки к стендапу отлично подходит сочетание двух фильтров: «только мои задачи» и «недавно обновлено». Применив два этих фильтра одновременно, вы увидите задачи, назначенные вам и обновленные за последний день.

## Спринт ревью, обзор итогов спринта, Демо

*Состав участников:*
Обязательно: команда разработчиков, Scrum-мастер, владелец продукта.
Необязательно: заинтересованные в проекте стороны.

*Когда проходит:* в конце спринта или по достижении контрольной точки.

*Продолжительность:* обычно 60 минут на каждую неделю итерации. То есть за двухнедельным спринтом должен следовать двухчасовой обзор итогов.

*Назначение:* на собрании по обзору итогов итерации команда представляет результаты своей работы. Собрание может проходить как в свободной обстановке (по типу «пятничных демонстраций»), так и в более формальной. На нем команда отмечает свои успехи, демонстрирует результаты работы, выполненной за итерацию, и выслушивает мнение заинтересованных в проекте сторон из первых уст. Команда разработчиков представляет заинтересованным сторонам и коллегам завершенные рабочие задачи из бэклога, чтобы собрать отзывы. Владелец продукта решает, стоит ли выпускать инкремент, хотя в большинстве случаев команда получает зеленый свет. На собрании по обзору итогов владелец продукта также изменяет бэклог продукта на основании результатов последнего завершенного спринта. Этот процесс может перейти в планирование следующего спринта. Если спринт длится один месяц, отводите под собрание для обзора итогов не более четырех часов.

Помните: результат работы можно считать окончательным и готовым к демонстрации в ходе обзора итогов, только если он может быть представлен полностью и отвечает критериям качества команды.

Шаг к эффективному обзору спринта - Определите критерии готовности.
Четко сформулированные критерии готовности помогают командам сосредоточиться на конечной цели по каждой рабочей задаче. Когда владелец продукта добавляет работу в бэклог команды, в его обязанности входит и определение критериев приемки. Какой должна быть завершенная пользовательская история?
Команда прописывает критерии приемки и примечания к тестированию там же, где находится описание пользовательской истории в Jira. Так все участники команды получают полное представление о ходе выполнения каждой задачи.

- Критерии приемки — это метрики, по которым владелец продукта может определить, удовлетворяет ли результат работы над пользовательской историей его требованиям.

- Примечания к тестированию — это краткие, предметные инструкции от команды по контролю качества, следуя которым специалист по разработке может написать более качественный функциональный код и автоматические тесты.

### Ретроспектива спринта

*Состав участников:* команда разработчиков, Scrum-мастер, владелец продукта.

*Когда проходит:* в конце итерации.

*Продолжительность:* обычно 45 минут на каждую неделю итерации. То есть после двухнедельного спринта нужно провести 90-минутную ретроспективу.

Цели ретроспективного совещания:

- Оценить, как прошел последний спринт, итерация или иная рабочая единица (это особенно важно в контексте динамики, процессов и инструментов команды).
- Сформулировать удачные и неудачные моменты и определить их приоритет.
- Составить и осуществить план по улучшению работы команды.

Ретроспективы — это не только время для накопленных жалоб. Ретроспективы помогают определить, что работает (и пусть команда продолжит сосредоточиваться на этих аспектах), а что нет (уделите время поиску нестандартных решений и разработке плана действий).

#### *Как провести ретроспективу*

На ретроспективе должен присутствовать каждый участник команды. Руководит дискуссией организатор совещания: это может быть scrum-мастер, владелец продукта, или же можно передавать эту обязанность по всей команде. Смело привлекайте к обсуждению дизайнеров, маркетологов или других сотрудников, которые участвовали в текущем спринте или итерации.

Стандартный шаблон встречи выглядит следующим образом.

1. Составьте короткий список того, что работало хорошо, и того, что можно улучшить. Этот список можно создать на магнитной доске, на странице Atlassian Confluence или даже на стене с помощью стикеров. Как бы вы ни зафиксировали этот первичный отзыв, обязательно сохраните его после собрания, чтобы на него можно было ссылаться в будущем.
2. Всей командой расставьте элементы этого списка по важности. Вы можете обнаружить общие темы, которые удобнее сгруппировать.

3. Обсудите способы и тактики для улучшения двух верхних элементов списка «Области для улучшения». Сосредоточьтесь на конечных результатах, а не на действиях, людях или прошлом.
4. Составьте план действий. К концу сеанса команда должна подготовить несколько практических идей по улучшению выбранных областей, для которых будут четко определены владельцы и сроки выполнения.

Подсказки:
В конце концов, ретроспектива призвана выявить, что работает, а что нет. Можете использовать для подсказки следующие варианты.

- Начать/прекратить/продолжить: какие действия команде нужно начать выполнять, а какие — прекратить или продолжить. Сосредоточьтесь на том, как перестать выполнять элементы из столбца «Прекратить».

- Больше/меньше: что команда должна делать больше, а что — меньше. Создайте план по блокировке главных элементов из списка «Делать меньше».

- Радость/грусть/гнев: что заставляет участников команды радоваться, грустить или испытывать гнев. Как вы уже догадались, нужно сосредоточиться на списках «Грусть» и «Гнев» и том, как улучшить ситуацию, чтобы в следующий раз был заполнен только список «Радость».

Шаблон
https://www.atlassian.com/ru/software/confluence/templates/retrospective

**Kanban**

Kanban — это популярный подход к реализации принципов agile и DevOps при разработке ПО. Методика предполагает обсуждение производительности в режиме реального времени и полную прозрачность рабочих процессов. Рабочие задачи визуально представлены на доске Kanban, что позволяет участникам команды видеть состояние каждой задачи в любой момент времени.

По ходу процесса разработки, стикер или карточка, представляющая проект, перемещаются от одной фазы к другой, пока все задачи не будут выполнены. Это отличный способ вести рабочий процесс и выявлять узкие места. В отличие от Scrum, Kanban уделяет меньше внимания фиксированным срокам, и работа происходит в непрерывном потоке. Методология проповедует принципы бережливого управления, при котором исключается возможность перепроизводства, то есть изготовление продукта в запас. Это экономит ресурсы и время.

### Артефакты

- Еженедельные собрания;
- Непрерывная разработка;
- Визуализация процесса на доске;
- Решение сначала самых важных задач;
- Поэтапные улучшения

### Канбан доски

Пять составляющих досок Kanban:

1. видимые сигнал (карточки с задачами)
2. столбцы
3. лимиты незавершенной работы

Ограничения WIP — это максимальное количество карточек, которое может находиться в одном столбце одновременно.

4. точка принятия обязательств (на доске как правило присутствует беклог)
5. точка поставки продукта (завершение рабочего процесса)

Пример KanBan доски:

| To do | Design | In progress | Code review | Testing | Done |
|---|---|---|---|---|---|
| | | | | | |

Для удобства разделите работу на отдельные активности, из которых будет состоять рабочий процесс (столбцы). После этого вы можете решить, как и когда добавлять на доску новые задания (карточки). Будет ли у вас служба техподдержки, через которую клиенты будут передавать идеи, или команда будет проводить совещания для составления и размещения новых карточек?

Кроме того, стоит определить размер карточки и объем работы, который она покрывает. Выберите способ оценки продолжительности или сложности работы для всех карточек. Если какое-то задание слишком объемное или сложное, разбейте его на несколько карточек.

Когда точка принятия обязательств и точка поставки продукта определены, можно начинать работу. Со временем процесс будет совершенствоваться на основании замечаний команды. Kanban требует от участников всех уровней постоянно проявлять инициативу. Эта философия называется «кайдзен». Уважение к людям и постоянное совершенствование в Kanban превыше всего. Следуя этим ценностям, вы очень быстро овладеете этой методологией.

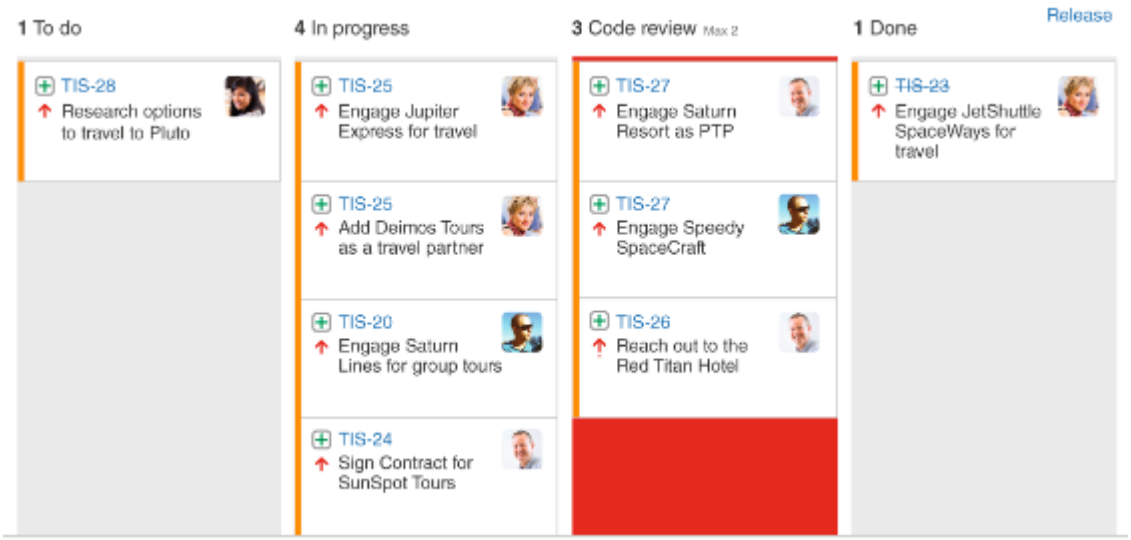
## Лимиты незавершенной работы (WIP)

Лимиты незавершенной работы (WIP) применяются в процессе agile-разработки, чтобы ограничить максимальное количество задач на каждом этапе рабочего процесса.

Ограничения WIP повышают производительность команд и уменьшают количество «почти выполненных» задач, заставляя команды сосредоточивать усилия на меньших объемах работы. По сути, ограничения WIP вырабатывают у команды привычку выполнять работу до конца. Что еще важнее, они проливают свет на препятствия и проблемные места. Когда имеется некий индикатор, который однозначно указывает на проблемные места, команда может дружно штурмовать задачи, препятствующие дальнейшему продвижению работы, чтобы разобраться в них, выполнить и разрешить проблемы.

Если команда раньше никогда не использовала лимиты незавершенной работы, они могут показаться неудобными. Поднимите вопрос о лимитах WIP в первых нескольких итерациях. Научитесь понимать, когда и почему команда достигает лимита незавершенной работы. На первых порах не поддавайтесь соблазну самовольно менять эти лимиты. Если лимит достигается на постоянной основе, значит, установлен слишком строгий лимит WIP или вы обнаружили слабое место в рабочем процессе команды.

При развертывании нового рабочего процесса обсудите всей командой, какое ограничение WIP нужно установить для каждого статуса. Прежде чем устанавливать ограничения, рекомендуем понаблюдать за несколькими спринтами и определить, сколько рабочих задач в среднем находится в каждом статусе.




В примере выше для столбца Code review (Проверка кода) установлено ограничение незавершенной работы. Это ограничение превышено, поэтому фон столбца окрашен в красный цвет.

Поскольку задачи, перемещенные в столбец Done (Завершено), не требуют дополнительных действий, для него ограничение WIP не устанавливается. На доске выше статус To do (К выполнению) получают пользовательские истории, которые были тщательно проверены владельцем продукта и командой. Когда команда разработчиков приступает к выполнению рабочих задач, она переносит их из столбца To do (К выполнению) в столбец In progress (В работе). Важно, чтобы в статусе To do (К выполнению) всегда находилось достаточно задач. Так участники команды разработчиков будут задействованы максимально эффективно. При этом на этапе To do (К выполнению) должно быть не слишком много историй, иначе владелец продукта может уйти слишком далеко вперед к тому моменту, когда появятся новые требования, и программу будет сложнее адаптировать к изменениям.

В статусе In progress находятся задачи, над которыми идет активная работа. Здесь ограничения WIP нужны, чтобы никто не сидел без работы и чтобы никто не занимался несколькими задачами одновременно. На доске, показанной выше, для задач в статусе In progress установлено ограничение в 4 задачи, и в этом статусе в настоящий момент находятся 3 задачи. Значит, команда может взяться за дополнительную задачу. В некоторых командах целесообразно выбирать такое ограничение незавершенной работы, которое было бы меньше числа участников команды. Это поможет подготовиться к внедрению надлежащих Agile-методик. Когда разработчик заканчивает работу над задачей и видит, что команда уже достигла ограничения WIP, он понимает, что пора выполнить несколько проверок кода или присоединиться к другому разработчику, чтобы заняться парным программированием.

В статусе Code review находятся истории, для которых код уже полностью написан, однако его требуется проверить, прежде чем выполнять слияние с базой кода. Своевременные проверки кода — лучшее

средство для обеспечения высокого качества, ускорения вывода инноваций на рынок, упрощения слияний (открытых веток становится меньше) и обмена знаниями между участниками команды разработчиков. Приступать к работе на этой стадии следует как можно скорее по нескольким причинам.

- Код не пострадает, когда участники команды будут загружать новые порции кода.
- Разработчик не потеряет контекст, который он использовал при написании исходного кода.
- Проверенную функциональную возможность можно будет объединить с главной веткой для релиза.
- Благодаря ограничениям незавершенной работы непроверенный код не будет накапливаться бесконечно.

**Цели WIP**

- Цель 1. Научиться делить работу на отдельные задачи примерно равного объема. Разбивая на части требования и пользовательские истории, важно следить, чтобы на выполнение отдельной задачи уходило не более 16 часов. В итоге команда будет увереннее подходить к оценке сложности работы, и проблемных мест станет меньше.
- Цель 2. Подбирать ограничения WIP в соответствии с навыками команды.
- Цель 3. Сократить простои. Если у какого-либо участника команды появилось свободное время, посоветуйте ему помочь коллегам на других этапах работы.
- Цель 4. Сохранить здоровую культуру разработки.

## Сравнение Scrum и Kanban

Scrum и Kanban - это два разных подхода к управлению проектами.

Scrum хорошо подходит для разработки новых продуктов, когда требуется гибкость и способность быстро реагировать на изменения. Он также хорошо подходит для сложных и неопределенных проектов, когда требуется часто пересматривать планы и принимать решения в ходе работы.

Kanban хорошо подходит для управления потоком работы, когда требуется обеспечить постоянный поток работы и улучшить эффективность процесса. Он также хорошо подходит для управления рутинными процессами и проектами с постоянными запросами.

| | Scrum | Kanban |
|---|---|---|
| Источник | Разработка программного обеспечения | Бережливое производство |
| Основная идея | Учиться на собственном опыте, самоорганизовываться и расставлять приоритеты, анализировать свои победы и поражения, чтобы постоянно совершенствоваться. | Повышать качество выполняемой работы с помощью наглядных материалов |
| График | Регулярные спринты фиксированной продолжительности (например, 2 недели) | Непрерывный процесс |
| Методы | Планирование спринтов, спринт, ежедневное Scrum-совещание, обзор спринта, ретроспектива спринта | Визуализация процесса работы, ограничение объемов незавершенной работы, управление процессом, включение циклов обратной связи |
| Роли | Владелец продукта, Scrum-мастер, команда разработчиков | Нет обязательных ролей |

**Сравнение досок:**

- У спринтов в Scrum есть дата начала и дата окончания, в то время как в Kanban работа ведется без перерыва. В Scrum наша цель — закончить спринт, в Kanban — задачу.
- В команде Scrum четко разграничены роли (владелец продукта, команда разработчиков и Scrum-мастер), а в Kanban формальные роли отсутствуют. Обе методики требуют от команд навыков самоорганизации.
- Доска Kanban используется на протяжении всего жизненного цикла проекта, а доска Scrum обнуляется и обновляется после каждого спринта.
- Доска Scrum содержит определенное количество заданий, которые нужно выполнить к заданному сроку.
- Доски Kanban дают больше свободы в том, что касается заданий и времени их выполнения. В зависимости от потребностей можно менять приоритеты, людей, ответственных за выполнение заданий, и содержание заданий.

## Scruban и Kanplan

Но что, если вашей команде больше всего подойдет комбинация методик Scrum и Kanban? Или ей необходимо перейти со Scrum на Kanban? В таком случае идеальным выбором будет Scrumban. Эта смешанная методика имеет

множество вариантов применения, однако чаще всего команды, работающие со Scrumban, берут из Scrum спринты с бэклогом, а из Kanban — ограничения незавершенной работы и время цикла. (Примечание. Время цикла — это время, затрачиваемое командой на обработку задачи.)

А если команда не работает итерациями, но нуждается в бэклоге, она может использовать Kanplan (или включить функцию бэклога Kanban) в Jira Software.

**Водопадная модель (Waterfall Model)**

Последовательный процесс разработки, при котором разработка перетекает плавно вниз (как водопад), следуя этапам инициации, анализа, дизайна, разработки, тестирования и поддержки. При следовании водопадной модели вы должны последовательно переходить от одного этапа к другому, без каких-либо пересечений или итеративных шагов.

По данным исследования PMI, 12% компаний используют методологию Waterfall на постоянной основе, а 40% респондентов утверждают, что часто к ней обращаются. А по данным LiquidPlanner, каскадную модель используют 25% организаций.

Количество этапов в Waterfall варьируется от компании к компании, но общий подход выглядит следующим образом:

- Создание концепции. Первая фаза жизненного цикла разработки ПО (SDLC) включает в себя анализ затрат и результатов и оценку масштабов проекта.
- Подготовка. Набор команды, определение целей и задач.
- Анализ. Изучение объема работ и требований.
- Дизайн. Создание прототипа и согласование его с заказчиком.
- Разработка/написание кода. Создание ПО на основе утвержденного прототипа.
- Тестирование. Готовый продукт тестируют.
- Реализация. Продукт выходит на рынок.
- Техническое обслуживание. Устранение выявленных недочетов и поддержка.

Преимущества каскадной модели

- Требует меньшей координации благодаря последовательным процессам с четко определенными этапами.
- Понятные этапы проекта позволяют четко определить зависимости между работами.
- Стоимость проекта можно оценить после определения требований.
- Больше внимания уделяется документированию проекта и требований.
- Этап проектирования, предшествующий написанию любого ПО, выполняется более методично и лучше структурирован.
- Среди преимуществ методологии можно выделить чёткую структуру и предсказуемый рабочий процесс. Это позволяет легко оценить затраты и прикинуть сроки выполнения до начала проекта. Резиденты Quora утверждают, что такие тонкости важны для финансовых и страховых компаний или компаний, которые разрабатывают «железо».
- Кроме того, модель подразумевает документирование каждого этапа. Это помогает создавать базу для последующих проектов. Также большое количество отчетности позволяет в любой момент показать заказчику или руководству, на какой стадии находится продукт.

Недостатки каскадной модели

- Сложнее разбить работу на части и организовать совместную работу, поскольку из-за более строгой последовательности этапов команды узко специализированы.
- Риск потери времени из-за задержек и препятствий при переходе на следующий этап.
- Дополнительные требования к набору участников специализированных команд для разных этапов проекта, в отличие от методики agile, которая способствует формированию более многофункциональных команд.
- Дополнительные накладные расходы на коммуникацию во время передачи работы на следующий этап.
- Владение продуктом и вовлеченность могут быть не такими сильными, как при использовании методики agile, поскольку основное внимание уделяется текущему этапу.
- Клиенты часто не знают, чего они действительно хотят, пока не взглянут на прототип.

# PMBok @todo

| Области знаний | Группы процессов управления проектом | | | | |
|---|---|---|---|---|---|
| | Группа процессов инициации | Группа процессов планирования | Группа процессов исполнения | Группа процессов мониторинга и контроля | Группа процессов закрытия |
| **4. Управление интеграцией проекта** | 4.1 Разработка устава проекта | 4.2 Разработка плана управления проектом | 4.3 Руководство и управление работами проекта | 4.4 Мониторинг и контроль работ проекта<br>4.5 Интегрированный контроль изменений | 4.6 Закрытие проекта или фазы |
| **5. Управление содержанием проекта** | | 5.1 Планирование управления содержанием<br>5.2 Сбор требований<br>5.3 Определение содержания<br>5.4 Создание ИСР | | 5.5 Подтверждение содержания<br>5.6 Контроль содержания | |
| **6. Управление сроками проекта** | | 6.1 Планирование управления расписанием<br>6.2 Определение операций<br>6.3 Определение последовательности операций<br>6.4 Оценка ресурсов операций<br>6.5 Оценка длительности операций<br>6.6 Разработка расписания | | 6.7 Контроль расписания | |
| **7. Управление стоимостью проекта** | | 7.1 Планирование управления стоимостью<br>7.2 Оценка стоимости<br>7.3 Определение бюджета | | 7.4 Контроль стоимости | |
| **8. Управление качеством проекта** | | 8.1 Планирование управления качеством | 8.2 Обеспечение качества | 8.3 Контроль качества | |
| **9. Управление человеческими ресурсами проекта** | | 9.1 Планирование управления человеческими ресурсами | 9.2 Набор команды проекта<br>9.3 Развитие команды проекта<br>9.4 Управление командой проекта | | |
| **10. Управление коммуникациями проекта** | | 10.1 Планирование управления коммуникациями | 10.2 Управление коммуникациями | 10.3 Контроль коммуникаций | |
| **11. Управление рисками проекта** | | 11.1 Планирование управления рисками<br>11.2 Идентификация рисков<br>11.3 Качественный анализ рисков<br>11.4 Количественный анализ рисков<br>11.5 Планирование реагирования на риски | | 11.6 Контроль рисков | |
| **12. Управление закупками проекта** | | 12.1 Планирование управления закупками | 12.2 Проведение закупок | 12.3 Контроль закупок | 12.4 Закрытие закупок |
| **13. Управление заинтересованными сторонами проекта** | 13.1 Определение заинтересованных сторон | 13.2 Планирование управления заинтересованными сторонами | 13.3 Управление вовлечением заинтересованных сторон | 13.4 Контроль вовлечения заинтересованных сторон | |

### Lean @todo

Термин означает «бережливая разработка». Его корни уходят глубоко в историю компании Toyota и её подходов к решению задач. В компании вносят только те изменения, которые приносят пользу, требуют минимум затрат и отнимают не более 30% запланированного времени.

По данным исследования PMI, 8% компаний постоянно используют принципы Lean, а 26% часто к ним обращаются.

Принципы Lean:

- Устранение лишнего: того, что не приносит пользы.
- Упор на обучение: цикличная разработка, обратная связь с клиентом.
- Решения принимаются на основе фактов, а не прогнозов.
- Целостность во всем: от информирования заказчика до рефакторинга.
- Полномасштабное видение: важно оценивать проект как целое, а не по частям.

Когда команда следует принципам бережливой разработки, она не просто выполняет задачи, а стремится сделать продукт с наименьшим количеством ошибок. В своём исследовании компания BBC обнаружила, Lean повышает скорость разработки ПО на 37% и снижает количество багов на 24%.

Команда GlobalLuxSoft отмечает, что бережливую разработку стоит применять, только если к проекту подключены опытные разработчики, так как обучение на ходу оказывается невозможным и ставит создание продукта под угрозу.

### Six Sigma @todo

Шесть сигм - это философия управления, разработанная компанией Motorola и которая делает акцент на установке чересчур высоких целей, сборе данных и анализе результатов для снижения дефектов в продуктах и услугах. Это высокотехнологичная методика точной настройки бизнес-процессов, применяемая с целью минимизировать вероятность возникновения дефектов в операционной деятельности.

### PRINCE2 @todo

# Управление продуктами

Управление продуктом — это организационная функция, регулирующая каждый этап жизненного цикла продукта, от разработки до позиционирования и ценообразования, исходя из продукта и интересов клиента. Чтобы создать наилучший продукт, менеджеры по продукту представляют в организации интересы клиента и следят за тем, чтобы в полной мере учитывалась рыночная ситуация.

Дорожная карта продукта — это стратегия долгосрочного развития продукта или решения. С помощью дорожных карт владельцы продуктов описывают будущие функциональные возможности продукта и устанавливают сроки, к которым эти новые возможности будут выпущены. Дорожная карта продукта имеет принципиально важное значение для понимания того, как краткосрочные усилия соответствуют долгосрочным бизнес-целям.

Как построить дорожную карту продукта?

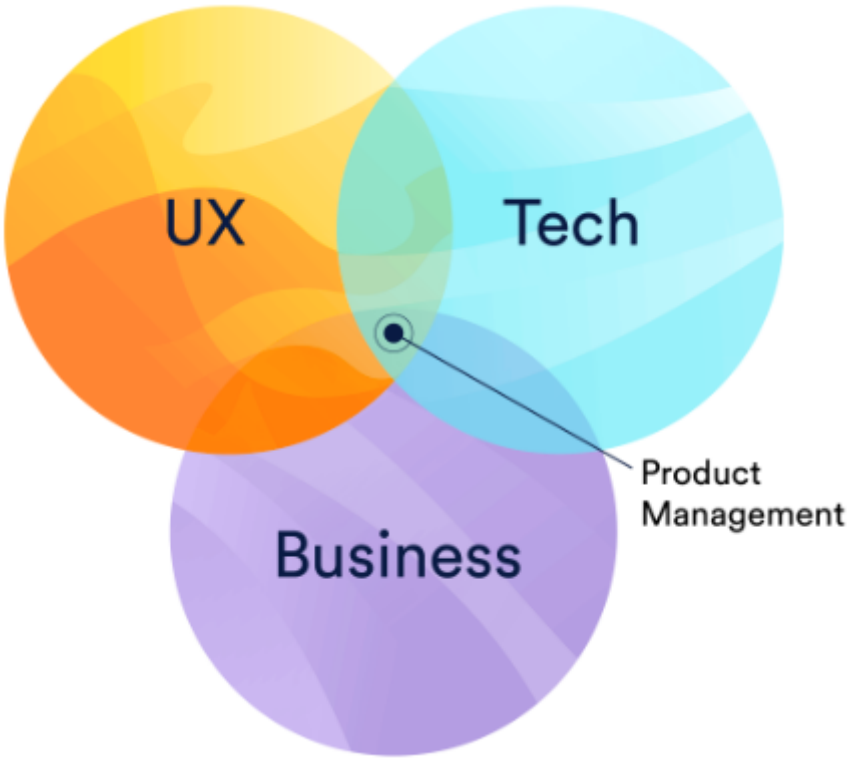
Чтобы построить дорожную карту, владельцы продуктов должны учитывать траектории рынка, предложения ценности покупателю, стратегические цели и ограничения трудовых ресурсов. Поняв эти факторы, владелец продукта может приступить к работе со своей командой и начать расставлять приоритеты в работах и эпиках на дорожной карте.

Содержимое дорожной карты будет зависеть от ее аудитории: дорожная карта для команды разработчиков может охватывать только один продукт, тогда как дорожная карта для руководителей может описывать несколько продуктов. В зависимости от размера и структуры организации одна дорожная карта может быть предназначена для нескольких групп, работающих над одним и тем же продуктом. Внешняя дорожная карта часто описывает несколько продуктов, связанных с одним ключевым моментом или потребностью клиента.

Основная идея заключается в том, чтобы создать дорожную карту, которую сможет легко понять аудитория. Слишком тщательная или недостаточная проработка деталей в дорожной карте может дать чересчур поверхностное представление о продукте или, что еще хуже, отпугнуть читателя. Дорожная карта с нужным количеством деталей и некоторой визуальной привлекательностью способна обеспечить необходимую вовлеченность основных заинтересованных сторон.

## Кто такой менеджер по продукту?

Менеджер по продукту — это человек, определяющий потребности клиентов и более крупные бизнес-цели, которые можно будет достичь с помощью продукта или возможности. Кроме того, он описывает критерии успеха продукта и объединяет команду для воплощения этой концепции в реальность.




Если говорить в целом, хороший менеджер по продукту занимается несколькими задачами.

- Определение и представление потребностей пользователей.
- Мониторинг рынка и анализ конкурентов.
- Создание концепции продукта.
- Согласование концепции продукта с заинтересованными сторонами.
- Установка приоритетов для функциональных возможностей продукта.
- Формирование единого образа мысли в больших командах, чтобы каждый участник мог принимать самостоятельные решения.

**Менеджер по продукту и владелец продукта**

| Менеджер по продукту | Владелец продукта |
|---|---|
| Работает с внешними заинтересованными сторонами | Работает с внутренними заинтересованными сторонами |
| Помогает определить концепцию продукта | Помогает командам выполнять работу на основе общей концепции |
| Намечает критерии успеха | Намечает план для достижения успеха |
| Отвечает за концепцию, маркетинг, ROI | Отвечает за бэклог команды и реализацию работы |
| Работает на концептуальном уровне | Участие в повседневной деятельности |

## Советы и рекомендации для эффективной работы менеджера по продуктам

- Безжалостно расставляйте приоритеты
- Знайте текущую ситуацию
- Предоставьте команде возможность принимать собственные решения
- Научитесь оказывать влияние без полномочий

## Советы для нового менеджера по продукту: первая неделя

Первая неделя любого менеджера по продукту должна включать непосредственное взаимодействие и общение с пользователями.

В течение первой недели обязательно ознакомьтесь с технологией. Поговорите с командой об ограничениях платформы и проблемах, с которыми они сталкиваются. Познакомьтесь с базовой архитектурой.

Если вы пришли в новую компанию, пообщайтесь с несколькими пользователями. Даже с теми, кто перестал пользоваться продуктом из-за плохого мнения о нем. Более того, таким пользователям нужно уделить особое внимание.

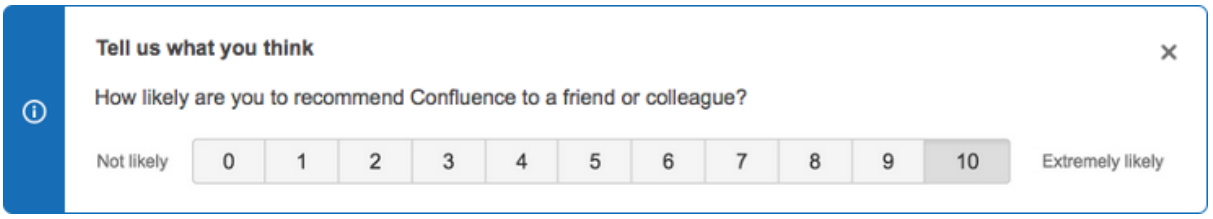
Вы не только узнаете кое-что новое о продукте и о базе пользователей, но и станете той точкой соприкосновения, которая нужна пользователю для сохранения лояльности к вашему продукту. Считайте, что это самое экономичное вложение в удовлетворение запросов клиентов, которое вы можете сделать.

### NPS

Расшифровывается как Net Promoter Score (индекс потребительской лояльности). NPS — это показатель лояльности, который определяет отношение клиентов к вашему продукту и ваши действия по этому поводу.

Это особенно ценный механизм обратной связи с клиентами для менеджеров работающих с продуктами, распространение которых опирается на мощность «сарафанного радио». Отличительной особенностью NPS по сравнению с другими инструментами количественного взаимодействия, например оценкой удовлетворенности клиентов, является то, что он позволяет легко и быстро собрать отзывы, обеспечивающие корректный и измеримый результат. Он содержит всего один вопрос, а результат можно сравнивать с другими продуктами.

Опрос NPS состоит из одного простого вопроса: «Какова вероятность того, что вы порекомендуете <наш продукт> друзьям или коллегам?»

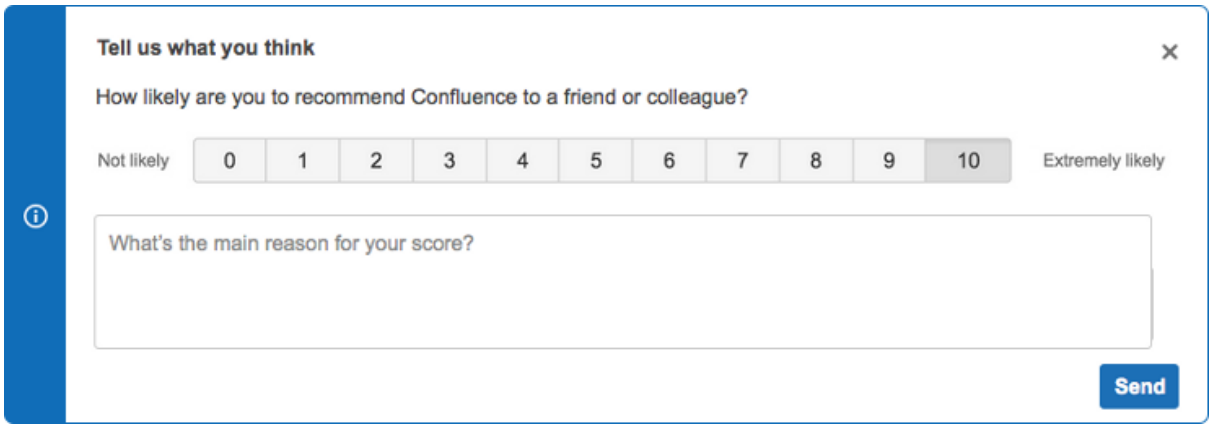


После сбора через опрос NPS достаточного количества ответов их делят на три группы: Promoters (сторонники) (9–10), Passives (нейтральные пользователи) (7–8) и Detractors (критики) (0–6).

Затем вычисляется индекс NPS. Для этого из процентного количества сторонников вычитается процент критиков. Например, если на 100 ответов у вас получилось 40 сторонников, 25 нейтральных пользователей и 35 критиков, тогда ваш индекс NPS будет (40% – 35%) = +5%. Значения индекса NPS находятся в диапазоне от –100% до +100%.

о NPS — это всего лишь число, оно не скажет вам, почему клиенты довольны или недовольны вашим продуктом.

Именно поэтому после выбора оценки мы просим дать отзыв.




https://www.atlassian.com/ru/agile/product-management/how-to-prioritize-features-using-net-promoter-scores

## Аналитика продукта

Аналитика продукта — это исследование взаимодействия пользователей с продуктом или сервисом. С помощью этого процесса команды по продукту могут отслеживать и анализировать вовлеченность и действия пользователей, а также представлять эти данные в наглядной форме. Таким образом команды могут улучшать и оптимизировать продукт или сервис.

Руководителю продукта важно знать ответы на вопросы вроде
«Сколько времени пользователи проводят в продукте каждый день?»,
«Какие действия они совершают чаще всего?»
«Какие функции наименее востребованы?».
Это поможет понять пользователей и определить, как сделать работу с продуктом удобнее.

**Тестирование будущего до его наступления**

Рабочий процесс обычно выглядит следующим образом:

- Определяем четкую гипотезу относительно изменения в продукте, например «мы ожидаем, что при увеличении размера поля для комментариев количество комментариев увеличится на 5 %».
- Выполняем самую дешевую реализацию этого изменения и загружаем любые необходимые аналитические события для проверки этой гипотезы.
- Развертываем изменение в подгруппе клиентов для проведения A/B-тестирования.
- Затаив дыхание, ждем результатов.
- Выполняем детальный разбор результатов, прибегая к помощи аналитика при работе над более сложными изменениями, и определяем успешность изменения.

# Инструменты PM

## Инструменты для личной продуктивности

Todolist - личный таск трекер с множеством возможностей.

Google calendar - планирование встреч, напоминания о них

Loom — приложение записывает со звуком видео с экрана и/или веб-камеры, которое потом можно отправить по почте или поделиться ссылкой.

Canva — отличный сервис для создания презентаций, информационных буклетов, различной инфографики и многого другого. Радует большое количество шаблонов и современный дизайн.

Лайтшот - https://prnt.sc/ инструмент для генерации ссылок на скриншоты.

Read Aloud — расширение для браузера, которое превращает текстовый файл в аудио. Может пригодиться, если надо быстро ознакомиться с документом не читая его — например, на ходу или когда вы за рулем.

## Инструменты для командной продуктивности

ПО для трека прогресса и отображения задач, таск-системы:
➔ Asana
➔ Jira
➔ Trello
➔ Redmine
➔ Mantish

### Bitbucket

Это инструмент для хостинга кода и совместной работы на основе Git, предназначенный для команд. Лучшие в своем классе интеграции Jira и Trello для Bitbucket создают для всей команды разработчиков единое пространство, в котором ее участники вместе работают над проектом. Ваша команда может совместно работать над кодом, начиная с появления идеи до выпуска в облако, контролировать качество кода с помощью автоматического тестирования и уверенно развертывать его.
https://www.atlassian.com/ru/software/bitbucket/guides/getting-started/overview

## Документация

ПО для хранения документации:
➔ Notion
➔ GoogleDocs
➔ Slite
➔ Confluence

Основные документы:
- Устав проекта

- Документ обоснования проекта
- SRS (Software Requirement Specification)//SOW - ТЗ, спецификация, требования
- Журнал изменений (Change Requests)
- Risk Chart
- Action plan
- Follow-up // Summary - документ с договоренностями после разговора/созвона/встречи

## Impact map

https://scrumtrek.ru/blog/product-management/3326/impact-mapping-guide/

**Для чего:**

- Для формирования бэклога
- Для определения рисков
- Для выявления конфликтов интересов заинтересованных лиц
- Для генерации гипотез по развитию продукта

**Как строить Impact Map?**

Для построения карты влияния (IM) вам потребуется:

Инструмент визуализации: стикеры или электронный инструмент вроде Miro или Mural

- Заинтересованные лица: пользователи, представители бизнеса, технические эксперты и т.д.
- Команда разработки
- Исследователь
- 1-2 часа
- Вся карта влияния строится по отношению к цели, которую необходимо достичь. Ваша задача — последовательно ответить на четыре ключевых вопроса: Зачем? Кто? Как? Что?

Делается итеративно в 3-4 захода.

**Пример использования Impact Mapping**

Приложение YoTask — мобильное приложение для ведения «списка» взаимодействия со структурой и контроля показателей работы сети.

Сначала строим полноценную ветку ответов на вопросы. Не обсуждайте детали реализации “что”, задача упражнения — построить карту, а не детальный план.


Далее продолжаем строить отростки первой ветки:


После переходим к другим веткам (Кто?) и делаем по аналогии:


В данном примере Impact map строился с каждым (Кто?) отдельно, для экономии времени заинтересованных лиц.

По итогу мы получили: цель, пользовательские истории которые ведут к цели и задачи, которые необходимо реализовать для создания пользовательского опыта.

Кстати, вопреки распространенному заблуждению, чтобы сформировать бэклог из карты влияния, необходимо взять не конечную колонку “Задачи”, а колонки “Кто?” и “Как?” — которые образуют собой пользовательские истории.


## Бэклог

Бэклог продукта — это перечень рабочих задач, расположенных в порядке важности, для команды разработчиков. Его составляют на основе дорожной карты и требований в ней.

Инициативы дорожной карты делятся на несколько эпиков, а каждый эпик содержит несколько требований и пользовательских историй.

Владелец продукта составляет из этих пользовательских историй единый список для команды разработчиков.

Хотя расстановкой приоритетов занимается владелец продукта, в процесс вовлечены и другие стороны. Успешность бэклога зависит от вклада и обратной связи, предоставленной клиентами, дизайнерами и командой разработчиков.

Владельцы продукта должны пересматривать бэклог перед каждым собранием по планированию итерации, чтобы уточнить расстановку приоритетов и внести изменения на основе выводов, сделанных в результате последней итерации. Регулярный пересмотр бэклога в кругах специалистов по Agile часто называют «грумингом» или «ведением бэклога» (некоторые используют термин «уточнение бэклога»).

Когда бэклог становится достаточно большим, владельцам продукта приходится выделять в нем группы краткосрочных и долгосрочных задач. Краткосрочные задачи нужно досконально проработать, прежде чем присвоить им этот статус. Для этого нужно составить полноценные пользовательские истории, обсудить все детали совместной работы с дизайнерами и разработчиками и оценить сложность разработки. Долгосрочные задачи могут быть продуманы не до конца, однако если команда разработчиков даст им приблизительную оценку, это поможет

расставить приоритеты. Ключевое слово здесь — «приблизительная». Оценки поменяются, когда команда получит полное понимание долгосрочных задач и приступит к их выполнению.

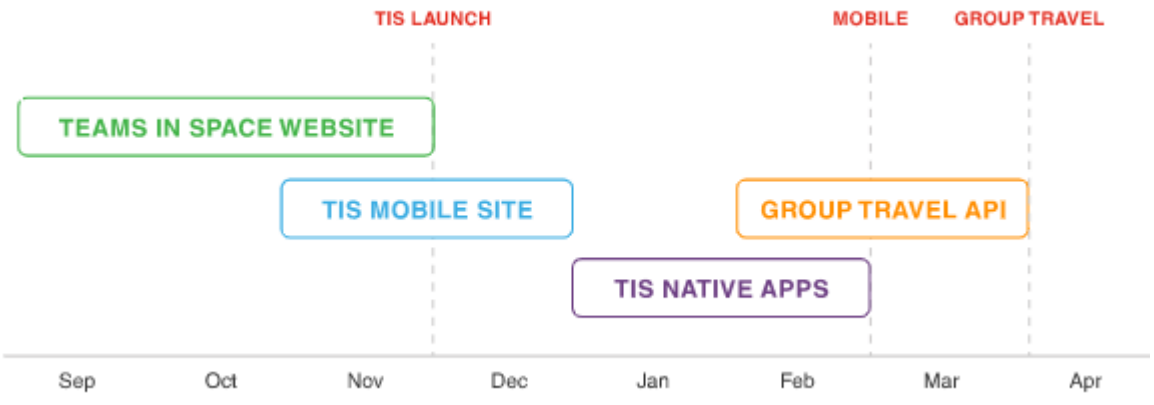
Кроме того, на основе бэклога продукта планируются итерации. В бэклог должны быть включены все рабочие задачи:

- пользовательские истории,
- баги,
- изменения в дизайне,
- технический долг,
- запросы клиентов,
- действия намеченные по итогам ретроспективы, и т. д.

## Роадмап (Дорожная карта)

Дорожная карта продукта — это стратегия долгосрочного развития продукта или решения. С помощью дорожных карт владельцы продуктов описывают будущие функциональные возможности продукта и устанавливают сроки, к которым эти новые возможности будут выпущены. В процессе agile-разработки дорожная карта обеспечивает необходимый контекст для ежедневной работы команды и потому должна меняться вслед за изменениями в конкурентной среде.

Чтобы составить дорожную карту, владельцы продуктов принимают во внимание возможное развитие ситуации на рынке, предложения по формированию ценности и технические ограничения. Достаточно глубоко изучив эти факторы, владельцы выносят на дорожную карту инициативы и сроки. Ниже приведен пример очень простой дорожной карты для команды по продукту. Инициативы представлены цветными надписями в прямоугольниках, а сроки указаны с помощью красных отметок контрольных точек.




Сделайте дорожную карту доступной в режиме онлайн и вносите в нее изменения по мере необходимости. В итоге команда получит единый достоверный источник информации.
Большинство инструментов для совместной работы, разработанных в таких целях, автоматически рассылают всем участникам проекта уведомления об изменении дорожной карты.

Вот как это принято делать: разделите инициативы в бэклоге продукта на эпики, затем разбейте их на требования и пользовательские истории. Видя эти связи, владельцы продуктов и команда разработчиков смогут без труда принимать краткосрочные решения, которые не станут помехой для работы в будущем.

Agile-разработка, в свою очередь, сопряжена с тремя видами рисков.

- Если дорожная карта будет часто меняться, команда может потерять веру в то, что руководство способно принимать стратегические решения.
- Если дорожную карту обновлять недостаточно часто, продукт может выйти на рынок слишком поздно и не удовлетворить накопившийся спрос.
- Долгосрочные задачи могут казаться слишком громоздкими для небольших итераций. Чтобы компенсировать это, команда начинает дробить работу на мелкие части и в результате слишком сильно концентрируется на краткосрочных итогах.

Нужно найти для дорожной карты идеальный баланс между текущими тактическими планами и стратегическими долгосрочными целями. Для этого лучше всего пересматривать дорожные карты раз в три месяца, при необходимости корректировать и рассылать всем остальным. Такой способ подходит организациям любого размера.

https://www.atlassian.com/ru/software/jira/guides/roadmaps/basic-roadmaps

При составлении и презентации дорожной карты помните следующее:
1. Не нужно громких слов — старайтесь передать сущность
2. Будьте осторожны с обязательствами
3. Создавайте реалистичные планы
4. Мыслите масштабно, но начинайте с малого
5. Создайте бизнес-сценарий
6. Найдите баланс между рутиной и мотивирующими задачами
7. Не забывайте думать, что последует за продуктом с минимальным функционалом и выпуском первой версии

**Рекомендации по созданию отличных дорожных карт**
Создание и поддержка дорожных карт продуктов — это непрерывный процесс, которым можно заниматься вместе с командой. Существует несколько простых способов, позволяющих добиться успешного результата.

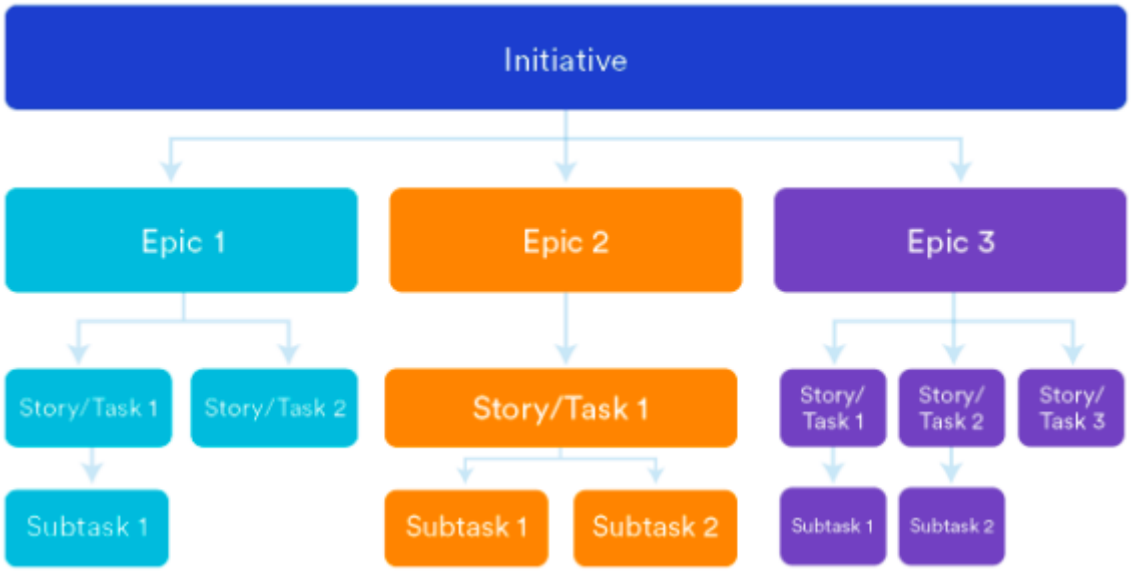
- Подробностей в дорожной карте должно быть ровно столько, сколько требуется аудитории.
- В дорожной карте должен сохраняться баланс: она в равной степени должна ориентироваться на краткосрочные тактические задачи и на их связи с долгосрочными целями.
- Регулярно просматривайте дорожные карты и вносите коррективы при изменении планов.

- Убедитесь, что у всех есть доступ к дорожной карте (и что к ней регулярно обращаются).
- Оставайтесь на связи с заинтересованными сторонами на всех уровнях, чтобы обеспечить согласованность действий.

## Инициативы, Эпики, User stories

Что такое истории, эпики и инициативы?

- Инициативы — это ряд эпиков, объединенных общей целью.
- Эпики — это крупные этапы работы, которые можно разбить на несколько небольших заданий (историй).
- Истории, или пользовательские истории, — это краткое изложение требований или запросов, составленное с точки зрения конечного пользователя.




Пример эпиков в инициативе:
Предположим, ваша ракетная компания хочет в этом году сократить стоимость запуска в космос на 5 %. Такая цель идеально походит на роль инициативы, так как за один эпик с этой масштабной задачей не справиться. Инициативу можно разделить на такие эпики, как «сократить потребление топлива на этапе запуска на 1 %», «увеличить частоту запусков в квартал с 3 до 4» и «уменьшить значение температуры на всех терморегуляторах в экономичном режиме с 22 до 20 градусов Цельсия».

## Epics

Почти всегда эпик выполняется за несколько спринтов. По мере того, как команда получает больше информации об эпике из отзывов разработчиков и клиентов, меняется набор пользовательских историй в нем. Это главная особенность эпиков agile: объем работ можно менять на основании сообщений от клиентов и прогресса команды.

С практической точки зрения эпик составляет верхний уровень в иерархии работы. Однако понимание связей между эпиком и другими agile-структурами позволяет получить представление о повседневной работе разработчиков и ее месте в более общей картине.

- Дорожная карта продукта — это стратегия долгосрочного развития продукта или решения.
- Тема — это цель организации, которой подчинены эпики и инициативы.
- Дорожная карта продукта представляется в виде набора инициатив, наложенных на временную шкалу.

- Инициативы разбиваются на эпики, чтобы было проще прослеживать связь между повседневной работой команды (представленной в виде небольших историй) и общими целями компании.

Выполнение ряда эпиков приводит к выполнению конкретной инициативы, в результате чего продукт продолжает развиваться и улучшаться в соответствии с организационными темами, требованиями рынка и клиентов.


При создании нового эпика можно воспользоваться другими средствами планирования и организации, которые уже есть у вашей команды. Цели или OKR команды на квартал прекрасно подойдут в качестве основы для эпиков. Создавая эпик, помните о следующем.

- Отчетность. Создавайте эпики для тех проектов, которые привлекут внимание менеджеров и руководителей.
- История. Эпики и составляющие их истории должны складываться в своего рода рассказ о вашем пути к той стадии, на которой возможность или продукт находятся в данный момент.
- Культура. При выборе размера и степени детализации эпика отталкивайтесь от культуры вашей организации.
- Время. Большинство команд разработчиков прибегает к оценке сложности работы, но мы рекомендуем проверить ее продолжительность. На эпик должна уходить пара недель. Это не слишком много и не слишком мало.

**User Story**

Пользовательская история — это наименьшая единица работы в методике agile. Это конечная цель, а не возможность, сформулированная с точки зрения пользователя ПО.

Пользовательская история — это описание функциональной возможности ПО простыми, общими словами, составленное с точки зрения конечного пользователя или клиента.

Пользовательская история пишется с целью разъяснить, как именно выполнение рабочей задачи приведет к созданию конкретной ценности для клиента. «Клиентами» не обязательно должны быть сторонние конечные пользователи в привычном смысле слова. Эту роль могут на себя примерять внутренние клиенты или коллеги из организации, которые рассчитывают на вашу команду.

Пользовательские истории состоят из нескольких предложений, описывающих требуемый результат простым языком и в общих чертах. Они не содержат мелочей. Требования появятся позже, когда команда обсудит их и придет к согласию.

Пользовательские истории также составляют значительные элементы методик Agile, такие как эпики и инициативы. Эпики — это большие рабочие задачи, которые делятся на несколько историй. Группа эпиков образует инициативу. Благодаря этим крупным структурам каждодневные усилия команды разработчиков (в работе над историями) ведут к достижению целей организации, выраженных в эпиках и инициативах.

- Истории удерживают акцент на пользователе. Список дел поможет команде сосредоточиться на актуальных задачах, в то время как с набором историй участники смогут направить усилия на решение проблем реальных пользователей.
- Истории создают условия для совместной работы. Когда определена конечная цель, команда может совместными усилиями найти лучшее решение для клиента и лучший способ достижения этой цели.
- Истории подталкивают к поиску нестандартных решений. Истории заставляют команду подходить критически и творчески к выбору наилучшего пути достижения конечной цели.
- Истории задают динамику. Выполнив очередную историю, команда разработчиков справляется с небольшой задачей и радуется промежуточному успеху, который помогает двигаться дальше.

### Как пользоваться

Как правило, историю пишет владелец продукта, менеджер по продукту или руководитель группы проектов, после чего она отправляется на проверку.

В ходе собрания по планированию спринта или итерации команда решает, какие истории она выполнит в ходе этого спринта. На этом этапе команды обсуждают требования каждой пользовательской истории и связанные функциональные возможности. Это шанс проявить свои навыки и творческий потенциал и внести вклад в воплощение истории в жизнь вашей командой. По завершении согласования требования добавляются в историю.

Еще на собраниях оценивают истории на основании их сложности или времени, которое нужно потратить на выполнение. Размер истории должен позволять выполнить ее за один спринт, поэтому в ходе оценки каждой

истории команда следит, чтобы слишком трудоемкие или затратные по времени истории разбивались на меньшие части.

## Как написать пользовательскую историю

Не существует универсального способа по вычленению историй в составе эпика, но есть множество отличных вариантов.

- **Роль или тип пользователя.** Создайте уникальную историю для каждого типа пользователя. «Ускоренный вход в систему для новых посетителей», «ускоренный вход в систему для существующих клиентов» и т. д.
- **Упорядоченные этапы.** Разделите процесс на этапы и создайте историю для каждого.
- **Культура.** Ориентируясь на то, как заведено в команде, решите, будет ли история быстро выполнимой задачей или проектом на неделю.
- **Время.** Если в команде не принято иначе, отводите на выполнение истории не больше одного спринта.

При написании пользовательских историй держите в уме следующее.

- **Критерии готовности работы.** Как правило, история считается «выполненной», когда пользователь может сделать то, что было запрошено. Тем не менее, четко сформулируйте цель.
- **Краткое описание задач и подзадач.** Определите, какие конкретно этапы нужно пройти и кто несет ответственность за каждый из них.
- **Обратная связь.** Поддерживайте связь с пользователями, чтобы увидеть проблему или потребность их глазами. Зачем гадать, если можно услышать историю из уст самих клиентов?
- **Время.** Истории должны выполняться за один спринт, поэтому истории, которые могут занимать недели или месяцы, следует разбивать на несколько историй поменьше. Как вариант, считайте их самостоятельными эпиками.

Сформулировав пользовательские истории, позаботьтесь о том, чтобы они были доступны всей команде.

## Шаблон и примеры пользовательских историй

Пользовательские истории часто представлены в виде простого предложения следующего вида:

«Как [тип клиента], [хочу то-то], [чтобы делать что-то]».

Давайте разберем эту формулировку.

**«Как [тип клиента]»**: для кого мы выполняем эту работу? Нам не так важна должность, сколько личность, что стоит за типом клиента.
**«Хочу то-то»**: в этой части заключается намерение пользователя — не возможностей, которыми он пользуется. Чего пользователь хочет добиться? В этом утверждении не должно быть ни слова о способах реализации. Если вы

описываете какую-либо деталь пользовательского интерфейса, игнорируя цель пользователя, вы упускаете суть.
**«Чтобы делать что-то»**: какое место отведено этому сиюминутному желанию клиента в более широком масштабе? Какую пользу в целом хочет извлечь клиент? Какую крупную проблему нужно решить?
Пользовательские истории могут выглядеть, например, следующим образом.

- Как Макс, я хочу пригласить друзей, чтобы мы вместе могли пользоваться этим замечательным сервисом.
- Как Саша, я хочу организовать свою работу, чтобы лучше контролировать ситуацию.
- Как менеджер, я хочу видеть, как продвигается работа у моих коллег, чтобы можно было составлять более точные отчеты о наших успехах и неудачах.

### User story mapping
https://scrumtrek.ru/blog/product-management/3498/user-story-mapping-guide/

## CJM (Client Journey Map)
https://scrumtrek.ru/blog/product-management/3214/customer-journey-map-guide/

## Схемы
ПО для составления схем
➔ MindManager
➔ XMind
➔ Miro
➔ FlowMapp
➔ Canva
➔ Figma

## Метрики
### Story Points
https://community.atlassian.com/t5/Agile-articles/Six-experts-sound-off-on-story-points-the-evolution-of-agile/ba-p/1553590
Многие команды agile предпочитают оценку сложности в очках. Очки сложности отражают общие трудозатраты, необходимые, чтобы полностью реализовать элемент бэклога продукта или выполнить любую другую рабочую задачу.

Команды начисляют очки в зависимости от сложности и объема работы, а также сопутствующих рисков или неопределенности. Эти числовые значения нужны для того, чтобы более эффективно разбить работу на небольшие части и избавиться от неопределенности. Благодаря такому подходу со временем команды понимают, сколько они могут сделать за отведенное время, вырабатывают общее представление и придерживаются его.

- При выборе дат не учитывается работа, не относящаяся к проекту напрямую, а ведь она неизбежно появляется. Отправка электронных

сообщений, проведение встреч и собеседований — все это может отнимать время участника команды.
- На выбор дат значительное влияние оказывают эмоции человека. Относительная оценка работы позволяет максимально исключить эмоциональную привязанность.
- Каждая команда подходит к оценке сложности работы немного по-своему, а значит, скорость каждой команды (измеренная в очках) тоже будет немного иная, чем у других. Это, в свою очередь, означает, что никто не сможет оказывать давление на команду, апеллируя к какому-то эталону скорости.
- Договорившись о соответствии между трудозатратами и сложностью в очках, вы сможете быстро и без дальнейших споров распределить очки.
- Количество очков, которое получат участники команды за решение проблем, зависит от сложности задачи, а не от затраченного времени. Следовательно, участники команды будут заинтересованы в повышении эффективности, а не в расходе времени.

Чтобы владельцу продукта было проще оценить усилия для выполнения каждой рабочей задачи, нужно правильно провести оценку и определить приоритет каждой задачи с учетом полученных данных.

Отдельное задание не должно занимать более 16 часов работы (если вы оцениваете сложность в очках, можете договориться о верхнем пределе, скажем, в 20 очков). Оценить более сложные рабочие задачи с должной долей уверенности практически невозможно. А уверенность очень важна, особенно когда речь идет о задачах с наиболее высоким приоритетом в бэклоге. Если сложность какой-либо задачи выходит за верхний предел команды (16 часов или 20 очков), это сигнал, что ее нужно разбить на более мелкие составляющие и провести оценку повторно.

Покер планирования

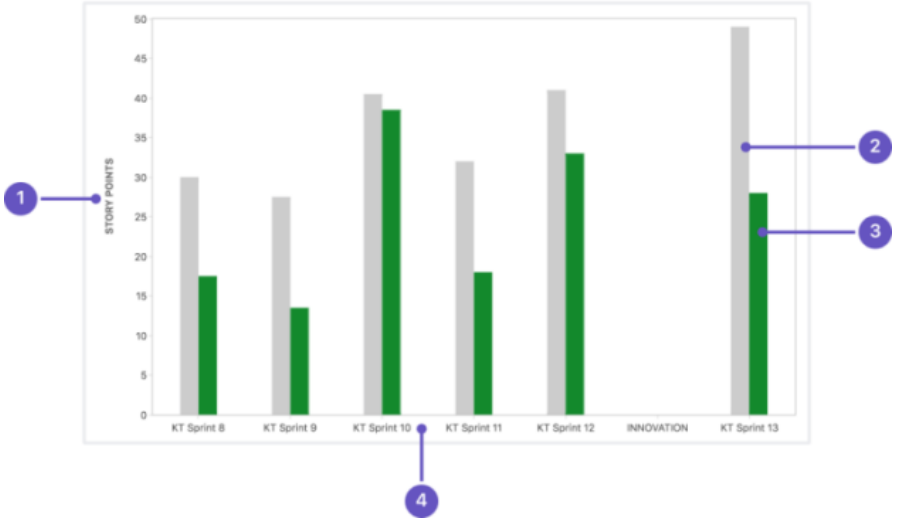
В ходе процедуры команда кратко разбирает каждую задачу из бэклога, после чего каждый участник мысленно выставляет ей оценку. Затем каждый поднимает карточку с номером, который соответствует оценке. Если все приходят к одному мнению, отлично! Если нет, уделите время (не слишком много, пару минут), чтобы понять, чем обоснованы разные оценки. Не стоит забывать, что оценка не должна быть скрупулезной. Если команда зашла в дебри, сделайте короткий перерыв и верните обсуждение на более общий уровень.
Установите приложение Planning Poker чтобы попробовать.

**Скорость команды**

То, сколько времени команда может выделить на спринт, по сути, определяется скоростью, то есть тем, какой объем работы команда способна выполнить за определенное время, а также производительностью, то есть степенью доступности команды.

Диаграмма скорости показывает величину ценности, созданной в ходе спринта. Это помогает в прогнозировании объема работ, который команда способна выполнить в будущих спринтах. Скорость команды можно выяснить только после совместного проведения нескольких спринтов всей командой.

Пример диаграммы скорости, на котором представлена (1) оценка сложности работы; (2) общий принятый объем, то есть совокупная оценка всех задач в спринте; (3) сложность завершенных задач; (4) названия завершенных спринтов.




**Производительность команды**
Спланировать производительность можно, например, собрав данные о доступности каждого участника команды по нескольким спринтам и суммировав их, чтобы получить процент от общей производительности.

**Тип работ**
При планировании спринта целенаправленно определяйте, за какую работу возьмется ваша команда, опираясь на разбивку по типам работ. В этом случае, даже если в бэклоге окажется большой объем технического долга и работы над качеством, вы сможете подойти к делу стратегически, запланировав спринт по техническому долгу или подняв планку контроля качества.

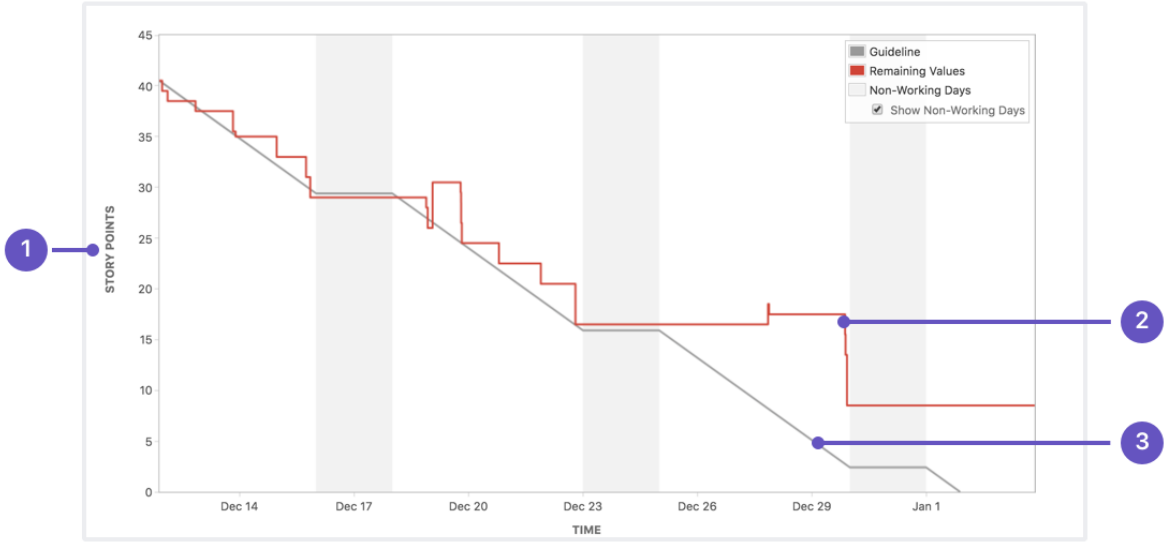
**Диаграмма Burndown для спринта (Диаграмма сгорания)**
Такая диаграмма дает представление о ходе выполнения работы в спринте. На временной шкале показан объем оставшейся работы, измеренный в очках сложности или часах. Диаграмма помогает прогнозировать способность команды закончить работу в назначенный срок и позволяет предотвратить увеличение объема работ. Если на диаграмме Burndown наблюдается резкий спад, это может говорить о неточной оценке работ.

Шаг 1. Определите принцип оценки работы в команде
Принцип оценки — это единица измерения, с помощью которой команда будет оценивать сложность работы. В Jira Software сложность работы можно оценивать в очках за пользовательскую историю, часах или придумать собственный принцип.

Шаг 2. Оцените сложность своих задач
В Agile оценка сложности подразумевает измерение размера бэклога команды или отдельной рабочей задачи.

Шаг 3. Отслеживайте прогресс команды с помощью диаграмм Burndown
В этом отчете показан объем работы, которую предстоит выполнить за спринт. С его помощью можно следить, сколько всего работы осталось в спринте, и предсказывать вероятность достижения цели спринта.

Ниже приведена диаграмма Burndown для спринта в Jira, на которой представлена (1) оценка работы; (2) остаточная величина, то есть суммарный объем оставшейся работы в спринте; (3) направляющая, которая служит ориентиром для команды.




1. Принцип оценки.
   На вертикальной оси указывается выбранный принцип оценки сложности.
2. Объем оставшейся работы.
   Красной линией показан объем оставшейся работы по оценкам команды.
3. Контрольная линия.
   Серая линия отображает приблизительное положение, которое занимала бы команда, если бы работа выполнялась линейно, без задержек или опережения графика. Если красная линия находится ниже серой, поздравляем — все идет к тому, что команда выполнит работу целиком до конца спринта. Но это не значит, что успех гарантирован. При отслеживании прогресса команды нужно уметь правильно пользоваться имеющейся информацией.

## Показатели по багам

Количество обнаруженных дефектов:

- во время разработки;
- после поставки продукта клиентам;
- людьми, не входящими в состав команды;
- количество дефектов, перенесенных в будущий релиз;

- количество входящих запросов в службу поддержки от клиентов;
- покрытие кода автоматизированными тестами (в процентах).

**Earned Value Management, EVM (Метод освоенного объема)**
Он является важным способом измерения производительности и прогресса проекта. Рассчитывается на основе объема проекта, графика и стоимости.
https://planneroffice.com.ua/blog/17/earned_value_method

**Ключевые показатели**

- Рентабельность инвестиций (ROI) для бизнеса — компании измеряют ее различными способами в зависимости от целей, в том числе учитывая рост доходов, ежемесячное активное использование (MAU) и многое другое.
  https://www.calltouch.ru/blog/glossary/roi-return-on-investment/

- Удовлетворенность клиентов — показатели опросов, такие как индекс потребительской лояльности (NPS) и индекс удовлетворенности клиентов (CSAT), могут быть полезны для отслеживания успешности проекта. Постоянно высокий показатель удовлетворенности клиентов по каждому релизу показывает, что scrum-команда действительно поставляет ценный продукт.

- Удовлетворенность команды — просто спросив участников о степени их заинтересованности в проекте и взаимодействии с командой, можно выявить такие проблемы, как текучесть кадров, отток специалистов и недовольство разработчиков.

**Метрики проекта по методике Google:**
Happiness (Счастье)
— пользовательское удовлетворение;
— ощущение, что продуктом легко пользоваться;
— net promoter score.
Engagement (Вовлечение)
— количество визитов пользователя в неделю;
— количество фото, загружаемых юзером в день;
— количество лайков и шэров.
Adoption (Принятие)
— обновления до новой версии;
— созданные пользователем подписки;
— покупки сделанные новыми пользователями в приложении.
Retention (Возвращаемость)
— количество пользователей, остающихся активными с течением времени
— churn
— повторные покупки
Task Success (Успех ключевых задач)
— успешные поиски;
— время загрузки фотографии;

— полностью заполненный пользователем профиль

**Продуктовые метрики**

MAU или активная месячная аудитория.
DAU или активная дневная аудитория.
MNU — количество новых пользователей за месяц.
Utilisation или конверсия в пользователя.
Frequency или частота использования.
MCU — количество пользователей, которые навсегда вас покинули (отток).
CPA - стоимость действия
CPC - стоимость клиента

## Диаграмма Ганта

Это инструмент управления проектами, иллюстрирующий то, как выполняется запланированная работа с течением времени. Диаграмма Ганта может включать даты начала и завершения заданий, контрольные точки, зависимости между заданиями, исполнителей и не только.

Диаграммы Ганта можно использовать для отслеживания организационной работы в проекте. Зависимости между заданиями гарантируют, что приступить к выполнению нового задания можно только после завершения другого.

Диаграммы Ганта могут быть построены в Jira, MS Project, Gant Chart, Google Tabs или любом другом инструменте.

Пример:

| | НАЗВАНИЕ ЗАДАЧИ | Ответсвенный | ПРОЦЕНТ ВЫПОЛНЕНИЯ | 1 КВАРТАЛ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Январь | | | Февраль | | | | Март | | | | |
| | | | | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 |
| | **Виджет** | | | | | | | | | | | | | | |
| | Сохранение данных при шаге назад | | 100% | | | | | | | | | | | | |
| | Редактирование в корзине | | 100% | | | | | | | | | | | | |
| | Скролл карточек | | 100% | | | | | | | | | | | | |
| | Невозможность провести оплату в отсутвии остатков | Андрей | 100% | | | | | | | | | | | | |
| | Уникальные символы в имени клиента | Рома | 100% | | | | | | | | | | | | |
| | История переотправок | Саша | 100% | | | | | | | | | | | | |
| | Запуск Я.Путешествия | все | 100% | | | | | | | | | | | | |
| | Настройка вопрос ответ, оферта | Саша | 100% | | | | | | | | | | | | |
| | Изменение способа отправки при переотправке | Саша | 100% | | | | | | | | | | | | |
| | Ввод количества сертификатов | Андрей | 100% | | | | | | | | | | | | |
| | Сделать календарь с понедельника | Андрей | 100% | | | | | | | | | | | | |
| | Виджет для Пятерочки (Благотворительность) фронт | Андрей | 100% | | | | | | | | | | | | |
| | Виджет для Пятерочки (Благотворительность) бэк | Рома | 100% | | | | | | | | | | | | |
| | Расчет суммы в корзине | Андрей | 100% | | | | | | | | | | | | |
| | Корректировка формы ввода имени, валидация данных для Тибериума | Рома, Андрей | 100% | | | | | | | | | | | | |
| | Генерация QR-кодов | Саша, Андрей | 100% | | | | | | | | | | | | |
| 1 | Поменять ссылку в письмах | Рома | 100% | | | | | | | | | | | | |

## Critical Path Method, CPM

Его суть в том, что работу над проектом нельзя продолжать, пока не будет устранена проблема-блокер.

## PERT Chart (Диаграмма PERT)

Program Evaluation Review Technique
Техника оценки и анализа проектов. Она представляет собой инструмент управления проектами, используемый для простой организации задач и сроков.

## RACI Chart (Диаграмма RACI)

RACI означает Responsible, Accountable, Consulted, Informed (ответственный, подотчетный, консультируемый, информированный). Эта диаграмма позволяет распределить роли и обязанности между заинтересованными сторонами и членами команды в отношении деятельности по проекту.

## Work Breakdown Structure, WBS

Иерархическая структура работ.
Это легко усваиваемая и организованная разбивка проекта на иерархические разделы и подзадачи.

# Реестр рисков

Пример:

| Риск | Описание | Причина | Воздействие | Вероятность | Степень влияния 0-1 | Категория | Ответственный | Оценка потерь/прибыли | Способ митигации | Стоимость митигации |
|---|---|---|---|---|---|---|---|---|---|---|
| Технологии | | | | | | | | | | |
| Поломка компьютера у члена команды | Угроза | Износ/Повреждение | 20% | 30% | 0,06 | Контролируемый | PM | 20-70т.р. Уточнение расчетов | Основной сценарий: Покупка нового/починка Запасной сценарий: Находим компьютер среди знакомых | 20/70т.р. |
| Потеря данных по проектам в поддержке | Угроза | Облачное хранилище схлопнулось | 80% | 10% | 0,08 | Неконтролируемый | PM | 50т.р./м | Основной сценарий: Иметь резервные копии на других серверах и личных носителях | 10т.р. |
| Люди | | | | | | | | | | |
| Потеря члена команды, например программиста | Угроза | Уход на другую работу/Конфликт | 40% | 70% | 0,28 | Контролируемый | PM | 40т.р. | Основной сценарий: Иметь в запасе средства чтобы поднять з/п Запасной сценарий:Иметь программиста готового заменить имеющегося | 50т.р. |
| Риск непринятия результата | Угроза | Несоответствие требованиям заказчика, по причине непонимания | 40% | 30% | 0,12 | Контролируемый | Ген.Дир. | 50% от стоимости проекта | Основной сценарий: Введение стандартов взаимодействия с клиентом, обеспечение своевременного согласования | 50т.р. |
| Процессы | | | | | | | | | | |
| Ошибка разработчика смещает сроки | Угроза | Человеческий фактор/ Недостаточная квалификация/Недостаточная мотивация | 10% | 30% | 0,03 | Контролируемый | Ген.Дир. | Пенни согласно договору с клиентом | Введение KPI, итераций. Сверхурочная работа при необходимости. | x*t специалиста |
| Недостигнутые KPI по рекламной кампании | Угроза | Ошибка маркетолога | 60% | 30% | 0,04 | Контролируемый | Маркетолог | 100т.р. (50% от средней ежемесячной прибыли) | Создать условия для повышения конверсии в 10000 чел., ведение блога, подключение CPA сетей. | 80 т.р.*6 месяцев=480т.р. |
| Внешние факторы | | | | | | | | | | |
| У заказчика нет возможности заплатить | Угроза | Нехватка средств | 40% | 10% | 0,04 | Неконтролируемый | PM | 50% от стоимости проекта | В договоре обозначить ответственность заказчика на такой случай. Иметь запас средств на уплату ФОТ и аренды. | 500т.р. |
| Лояльность клиента приносит крупный проект | Возможность | Рекомендация | 80% | 30% | 0,24 | Неконтролируемый | Ген.Дир. | до 3млн.р. | Наличие проверенных аутсорсеров, которых при крупном заказе можно перевести на контракт. | 0 |

## Платформы для поиска вакансий и нетворкинга

➔ Indeed
➔ LinkedIn
➔ WeWork
➔ Glassdoor
➔ Career Builder
➔ IdeaList
➔ Dice
➔ Monster.com
➔ ZipRecruiter
➔ GulfTalent
➔ Angel.co
➔ Bayt.com
➔ Jobble
➔ Trud.com
➔ RemoteJobs

# Сбор требований

Менеджеры по продукту обычно сначала составляют документ с требованиями к продукту.

Шаблон требований к продукту - https://www.atlassian.com/ru/software/confluence/templates/product-requirements

В документе с требованиями к продукту описывается продукт, который предстоит создать. В нем указывается назначение продукта, его возможности, функции и принцип работы.


Затем документ передают заинтересованным сторонам (чтобы те внесли свой вклад) — командам технического профиля и бизнес-командам, которые будут помогать в создании, запуске или продвижении продукта на рынке.

Следуйте следующим советам:

- Пусть при разговоре с клиентами присутствуют дизайнер и разработчик. Так они узнают информацию от клиента из первых уст, а не из заметок владельца продукта. Это также даст им возможность лучше прощупать почву, пока тема еще свежа в памяти клиента.
- Пусть типы клиентов разрабатывает и использует вся команда. У каждого участника команды есть своя уникальная точка зрения и свои соображения, и каждому участнику нужно понимать, как типы клиентов определяют разработку продукта.
- Пусть расстановкой приоритетов для неполадок и ведением бэклога также занимается вся команда. Воспользуйтесь этой отличной возможностью, чтобы ввести всех в курс происходящего и дать понять, почему владелец продукта расставил приоритеты в работе именно так, а не иначе.

При составлении документа с требованиями рекомендуется использовать один и тот же шаблон для всей команды.

# Бюджетирование

Разбор методики Unit-экономика
https://skillbox.ru/media/management/razbor-metodiki-yunitekonomika/

Управление стоимостью проекта
https://ppt-online.org/387238

# Культура в команде

## Культура выполнения работы

В эффективных командах выполнение каждой рабочей задачи подчиняется четко сформулированным методикам и принципам разработки. Чтобы оценить свой процесс выполнения работы и убедиться, что он идеально подходит команде, ответьте на следующие вопросы.

- Хорошо ли прорабатывают пользовательские истории владелец продукта, дизайнер и команда разработчиков, прежде чем начать работу над ними?
- Все ли понимают ценности и принципы разработки, принятые в команде, и следуют им?
- Установлены ли для проверки кода, автоматических тестов и непрерывной интеграции понятные критерии и требования, на основе которых формируется устойчивая культура Agile-разработки?
- Много ли обнаруживается багов после того, как команда завершает работу над историей? Другими словами, действительно ли завершена работа, отмеченная как «завершенная»?

Мы распределяем наше время так:

- 70% времени: новые функции
- 15% времени: исправление ошибок
- 15% времени: технический долг

Совет от профессионалов: элементы с самым высоким процентом всегда имеют тенденцию превосходить элементы с более низким процентом. Таким образом, мы начинаем с более низких процентов в начале нашего спринта, прежде чем перейти к поставке функций позже в спринте. Также не помешает одержать несколько быстрых побед в начале спринта, чтобы все чувствовали себя хорошо и продуктивно.

## Как оптимизировать удаленную работу в команде

Успех удаленной команды зависит от взаимного доверия, обмена информацией и сотрудничества.

Полезным для распределенной Scrum-команды окажется продуманный план коммуникаций, содержащий:

- согласованные условия удаленной работы;
- описание способа связи с другими участниками команды по нерабочим вопросам;
- описание согласованного подхода к организации собраний;
- порядок сообщения участниками команды сведений о доступности;
- перечень инструментов для совместной работы, которые следует использовать.

## Дисциплина в команде

1. Получил задачу – подтвердил в комментарии, что понял задачу и принимаешь в работу.

2. Не понял задачу – уточни у постановщика, согласуйте вид результата исполнения. Зафиксируй в комментариях.

3. Согласен со сроками в задаче – подтверди сроки. Сделай все возможное, чтобы выполнить задачу в срок.

4. Сроки не определены постановщиком – предложи свои.

5. Не согласен с определенными сроками – обсуди сроки с постановщиком, согласуйте решение.

6. Фиксируй договоренности в комментариях к задаче.

7. Фиксируй в задаче количество затраченных часов с комментарием чем конкретно занимался.

8. Выполнил задачу – перетащи задачу в соответствующую колонку в канбане и сообщи постановщику, что задача выполнена.

9. Не успеваешь выполнить задачу в согласованный срок – оповести постановщика заранее, а не в ожидаемый день готовности. Обсуди сроки с постановщиком, согласуйте решение проблемы. Зафиксируйте новые условия в комментариях.

10. Задача поставлена не по адресу – попроси постановщика ее переназначить.

11. Выполнение приоритетной задачи что-либо блокирует - не сиди без дела, бери в работу следующую задачу из спринта или обратись с вопросом чем заняться к своему руководителю.

12. Требуется участие/консультация коллеги - обратись, можно в общем чате проекта, чтобы PM был в курсе.

Статья для разработчиков о agile философии - https://www.atlassian.com/ru/agile/software-development/developer

# Разработка

## Общее

**Основные модели разработки ПО:**

● Code and fix — модель кодирования и устранения ошибок;
● Waterfall Model — каскадная модель, или «водопад»;
● V-model — V-образная модель, разработка через тестирование;
● Incremental Model — инкрементная модель;
● Iterative Model — итеративная (или итерационная) модель;
● Spiral Model — спиральная модель;
● Chaos model — модель хаоса;
● Prototype Model — прототипная модель

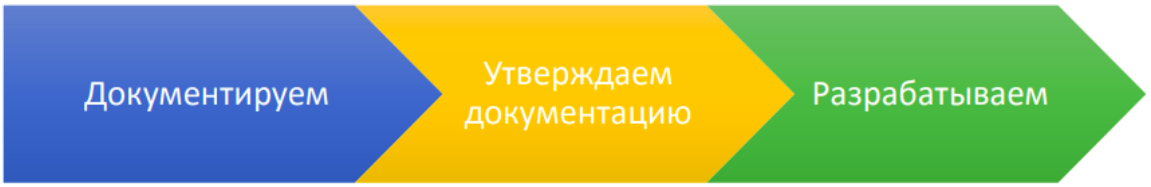
**Agile:**

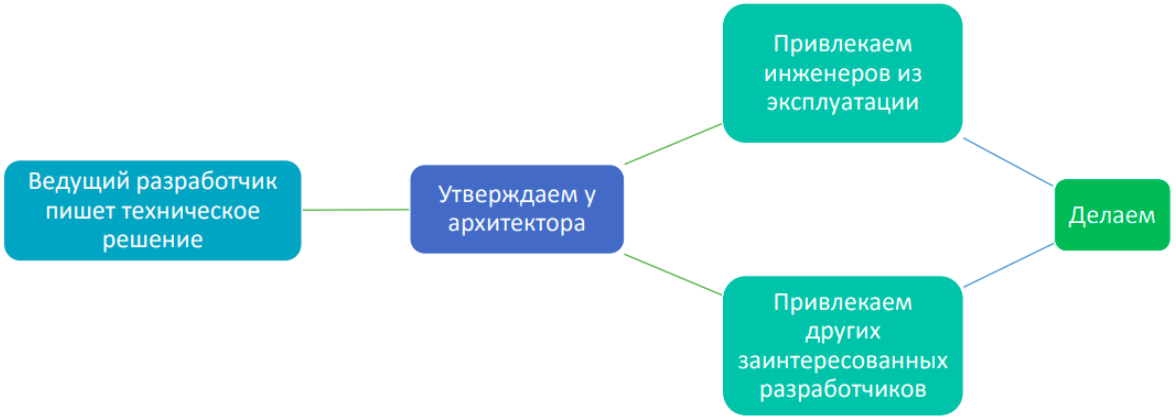
● экстремальное программирование (Extreme Programming, XP);
● бережливая разработка программного обеспечения (Lean);
● фреймворк для управления проектами Scrum;
● разработку, управляемую функциональностью (Feature-driven development, FDD);
● разработку через тестирование (Test-driven development, TDD);
● методологию «чистойкомнаты» (Cleanroom Software Engineering);
● итеративно-инкрементальный метод разработки (OpenUP);
● методологию разработки Microsoft Solutions Framework (MSF);
● метод разработки динамических систем (Dynamic Systems Development Method,
DSDM);
● метод управления разработкой Kanban

**Окружения среды разработки:**

● Local - компьютер разработчика
● Development - сервер разработки выступающий как песочница, где разработчик может выполнить unit-тестирование
● Integration - основа для построения CI, или для тестирования сайд-эффектов разработчиком
● Testing/Test/Internal Acceptance - окружение в котором выполняется тестирование интерфейса. Команда контроля качества проверяет что новый код не будет иметь влияния на существующую функциональность системы после деплоя нового кода в тестовое окружение.
● Staging/Stage/Model/Pre-Production/Demo - зеркало прод окружения
● Production/Live - серверы конечных пользователей/клиентов

Подходы к внедрению изменений:




## Сервер

Основное направление — поддержка интернет-ресурсов.
Количество задач, возлагаемых на сервер велико. Вот несколько сценариев, в которых необходимы подобные устройства:

- хостинг сайтов;
- разработка веб-приложений;
- платформа для приема и отправки электронных писем;
- хранение файлов;
- создание общего рабочего пространства для сотрудников одной фирмы;
- создание шлюзов (proxy или VPN), заменяющих информацию о подключившемся компьютере на другую;
- добыча криптовалюты

Подбор серверных систем происходит по сформулированным запросам заказчиков. Благодаря широкому ассортименту не просто с ходу подобрать подходящее устройство, чтобы оно удовлетворяло пожеланиям клиента и оперативно справлялось с поставленными задачами, поэтому ориентируются на:

- мощность;
- габариты;
- надежность;
- управляемость;
- масштабируемость;
- бюджет;

● готовность к работе

Типы серверов

- Сервер приложений
- Сервер базы данных
- Фаерволы, брендмауэры

## База данных

Система управления базами данных (СУБД) является субъектом управления и программой, а база данных объектом управления и собственно данными, которыми управляет СУБД.

● БД - это данные хранимые в структурированном виде. Обычно в виде набора файлов.
● СУБД - это программа или библиотека, которая умеет с этими файлами работать.


## Hosting

- **Виртуальный хостинг** (Shared)

Сервер, на котором расположено множество сайтов. Они используют одинаковое программное обеспечение и имеют равные возможности. На одном сервере может находиться около тысячи сайтов. Как правило, на таком хостинге располагают небольшие веб-ресурсы, не требующие больших мощностей и дискового пространства. Благодаря своей низкой стоимости и отсутствию необходимости в администрировании сервера, виртуальный хостинг — самый распространенный выбор среди пользователей.

- **Виртуальный выделенные сервер (VPS-сервер)**

VPS отличается от виртуального хостинга тем, что владелец такого сервера может устанавливать и настраивать любое программное обеспечение. По сути, управление VPS-сервером не отличается от управления физическим

сервером. Сайту на VPS выделены определенные ресурсы в соответствии с тарифом.

- **Выделенные физический сервер (Dedicated)**

Пользователю дается отдельный сервер в дата-центре под самостоятельное управление с возможностью устанавливать любую операционную систему, менять программное обеспечение под свои нужды. Такой хостинг используется для масштабных ресурсоемких проектов с высокими требованиями и хорошо подойдет для крупных интернет-магазинов с высокой посещаемостью или, например, онлайн-игр.

- **Облачный хостинг (Cloud hosting)**

Облачный хостинг — объединённая система серверов, на которых располагаются клиентские сайты. Таким образом, выделяемые для клиента мощности не ограничиваются одним сервером, а распределяются сразу между несколькими серверами. Это обеспечивает бесперебойную работу ресурса вне зависимости от выхода из строя какого-то одного сервера. Кроме повышенной производительности, облачный хостинг привлекает клиентов и другими преимуществами. Во-первых, пользователь платит только за те ресурсы, которые использовал — цена услуги зависит от объемов потребления мощностей. Во-вторых, на облачном хостинге при изменении нагрузки выделенные ресурсы увеличиваются или уменьшаются автоматически.

- **Colocation**

Размещение оборудования клиента в дата-центре провайдера Colocation буквально переводится как «соразмещение», что точно описывает этот вид хостинг-услуг. Ваш сервер располагается на технологической площадке провайдера (дата-центре), который обеспечивает его высокоскоростным интернет-каналом и заботится о прочих необходимых условиях содержания.

## GIT

Система контроля версий.
Это сервис или программа, в которой хранятся данные о продукте, чтобы в любой момент можно было вернуться к любому этапу его создания. Самый известный сервис для этого — Git.
Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создает ссылку на предыдущую версию идентичного файла, который уже сохранен. Git представляет свои данные как, скажем, поток снимков.

## Код-ревью

Это процесс проверки кода, который позволяет: выявить → ошибки, пропуски, уязвимости и стилистические недочеты (с точки зрения проекта или принятых в команде правил).

Когда разработчик заканчивает выполнение задачи, другой разработчик анализирует получившийся код, принимая в расчет следующие вопросы.

- Нет ли в коде очевидных логических ошибок?
- Полностью ли пригоден код для всех сценариев использования, описанных в требованиях к коду?
- Достаточно ли покрывают добавленный код новые автоматические тесты? Нужно ли переписать существующие автоматические тесты для учета изменений в коде?
- Отвечает ли код требованиям существующего руководства по оформлению?

Проверки кода должны быть частью существующего рабочего процесса команды.

Выполняйте проверку до слияния

Обязательная проверка кода до его слияния с вышестоящей веткой дает уверенность в том, что в рабочую среду не попадет непроверенный код. Это значит, что спорные архитектурные решения, принятые в 2 часа ночи, и ошибки, которые допустит стажер в использовании шаблона проектирования, будут выявлены прежде, чем смогут повлечь за собой долгосрочные (и досадные) последствия для приложения.

Когда разработчики знают, что их код будет проверять коллега по команде, они прикладывают дополнительные усилия, чтобы код успешно прошел все тесты и был написан максимально хорошо, чтобы проверяющий не столкнулся с трудностями. Благодаря такой осознанности сам процесс написания кода становится более отлаженным и, как результат, протекает быстрее.

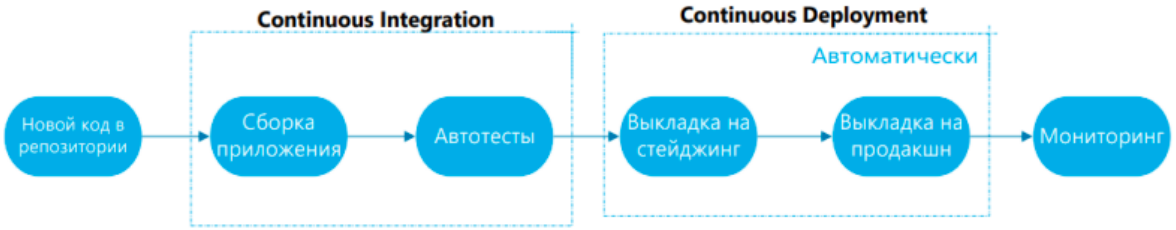
## DEVOPS

https://www.atlassian.com/ru/devops/what-is-devops

DevOps - это подход, методология и даже культура и философия процесса разработки, при котором программисты, тестировщики и системные администраторы могут работать над продуктом быстрее и эффективнее. Подход помогает снизить ошибки при передаче проекта от разработчиков к тестировщикам и сисадминам и наладить между ними взаимодействие. В основе лежит идея, что разработка, тестирование и эксплуатация цифровых продуктов — это единый, бесшовный и циклический процесс.

Это набор практик, нацеленных на активное взаимодействие и интеграцию специалистов по разработке и эксплуатации.

Нацелена на то, чтобы помогать организациям быстрее создавать и обновлять программные продукты.




Контейнеризация помогает:

- увеличить скорость выкладки кода на сервер, так как контейнер при запуске

уже готов к работе

- быстро вернуться к предыдущей версии, если в новой обнаружились проблемы:

закачиваем старые контейнеры — и снова все работает

- В итоге программисты сосредоточены на своей главной задаче: пишут код. А девопс помогает пройти весь остальной путь: собирает проверенные коды в контейнер, отправляет его на сервер и добавляет к уже работающей программе


**Мерж реквест / Пул реквест** - мерж-реквест или пулл-реквест создается в системе управления git-репозиториями. Это запрос на мерж одной ветки в другую, подобно задаче, назначаемый на какого-либо исполнителя. GitHub и Bitbucket используют термин «пулл-реквест»,

потому что первое необходимое действие — сделать пулл предлагаемой ветки.

**Деплой** - это развертывание/помещение исполняемого кода на сервер, где он будет работать.

**Deploy** - это целый процесс действий, которые делают программный продукт готовым к использованию:

➔ выпуск;
➔ установка;
➔ активация;
➔ адаптация;
➔ обновление;
➔ исправление ошибок и другие.

Виртуальная машина — это полноценная операционная система внутри другой ОС, с собственным ядром и другими изолированными ресурсами. Контейнер — не готовый «компьютер», а лишь изолированный механизм для запуска одного приложения.

В отличие от аппаратной виртуализации, контейнеризация обеспечивает разделение ресурсов не на аппаратном уровне, а на базе ядра операционной системы. Контейнеры более легковесны, менее требовательны и полностью зависимы от «материнской» ОС, чем VM.

## CI/CD

Continuous Integration, Continuous Delivery — непрерывная интеграция и доставка — это технология автоматизации тестирования и доставки новых модулей разрабатываемого проекта заинтересованным сторонам (разработчики, аналитики, инженеры качества, конечные пользователи).

Как выполняется настройка CI/CD, процесс: https://losst.ru/nastrojka-gitlab-ci-cd

Pipeline - организованный процесс миграции кода из репозитория до конечного сервера через ci/cd.

**Этапы CI/CD:**

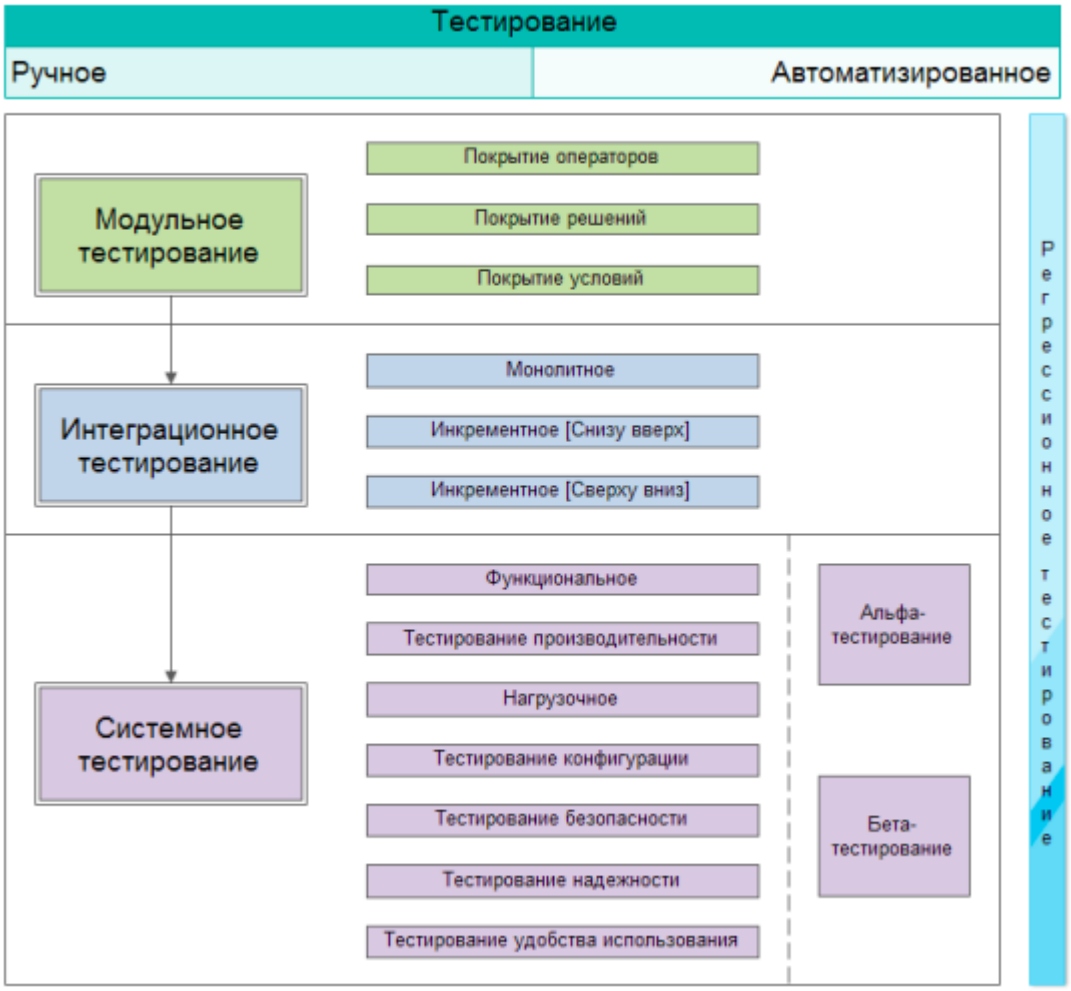
- **Написание кода.**
  Каждый из разработчиков пишет код своего модуля, проводит ручное тестирование, а затем соединяет результат работы с текущей версией проекта в основной ветке. Для контроля версий используется система Git, либо аналогичные решения. Когда участники команды опубликуют код своих модулей в основной ветке, начнется следующий этап.
- **Сборка.**
  Система контроля версий запускает автоматическую сборку и тестирование проекта. Триггеры для начала сборки настраиваются командой индивидуально — фиксация изменений в основной ветке проекта, сборка по расписанию, по запросу и т.д. Для автоматизации сборки используется Jenkins, либо аналогичный продукт.
- **Ручное тестирование.**

Когда CI система успешно проверила работоспособность тестовой версии, то код отправляется тестировщикам для ручного обследования. При этом тестовая сборка получает номер кандидата для дальнейшего релиза продукта.

- **Релиз.**
  По итогам ручного тестирования сборка получает исправления, а итоговый номер версии кандидата повышается.
- **Развертывание.**
  На этом этапе рабочая версия продукта для клиентов автоматически публикуется на production серверах разработчика. После этого клиент может взаимодействовать с программой и ознакомиться с ее функционалом.
- **Поддержка и мониторинг.**
  Конечные пользователи начинают работать с продуктом. Команда разработки поддерживает его и анализирует пользовательский опыт.

# Тестирование

Выделяют 4 основных уровня тестирования:
- Компонентное/модульное тестирование (Component/Unit Testing)
- Интеграционное тестирование (Integration Testing)
- Системное тестирование (System Testing)
- Приемочное тестирование (Acceptance Testing)



**Дымовые тесты (Smoke Tests)**: выполняются каждый раз, когда мы получаем новый билд (версию), проекта (системы) на тестирование, при этом считая её относительно нестабильной. Нам нужно убедиться что критически важные функции AUT (Application Under Test) работают согласно ожиданиям. Идея данного вида тестирования заключается в том, чтобы выявить серьезные проблемы как можно раньше, и отклонить этот билд (вернуть на доработку) на раннем этапе тестирования, чтобы не углубляться в долгие и сложные тесты, не затрачивая тем самым время на заведомо бракованное ПО.

**Санитарное тестирование:** используется каждый раз, когда мы получаем относительно стабильный билд ПО, чтобы определить работоспособность в деталях. Иными словами, здесь проходит валидация того, что важные части функциональности системы работают согласно требованиям на низком уровне.

**Ре-тест**: проводится в случае, если фича/функциональность уже имела дефекты, и эти дефекты были недавно исправлены

**Регрессионные тесты**: собственно то, что занимает львиную долю времени и для чего существует автоматизация тестирования. Проводится регрессионное тестирование AUT тогда, когда нужно убедиться что новые (добавленные) функции приложения / исправленные дефекты не оказали влияния на текущую, уже существующую функциональность, работавшую (и протестированную) ранее.

**Юнит-тесты** - это тесты белого ящика, которые проверяют отдельные части кода, такие как функции, методы и классы. Они должны быть написаны на том же языке, что и тестируемый продукт и храниться в том же репозитории. Они часто прогоняются как часть сборки, чтобы сразу же увидеть успешно ли завершается тест или нет.

**Интеграционные тесты** - это тесты черного ящика, которые проверяют, что интеграционные точки между компонентами продукта работают корректно. Тестируемый продукт должен быть в активной фазе и развернут на тестовом окружении. Тесты сервиса часто являются именно тестами интеграционного уровня.

**End-to-end тесты** - это тесты черного ящика, которые проверяют пути выполнения системы. Их можно рассматривать как многошаговые интеграционные тесты. Тесты для веб-интерфейса часто являются именно end-to-end тестами.

A/B-тестирование
https://habr.com/ru/post/233911/

## Оптимизация процессов

## Сдача проекта

# Траектории развития в смежные области для PM

Вариативно, в зависимости от компании, сферы и рынка, у IT PM могут быть следующие карьерные линии:
1. Project manager - program manager - Portfolio Manager - Head of PMO
2. Project manager - Delivery Manager - Delivery Director
3. Project manager - Product manager - Product Owner (не путать с ролью в скрам) - Chief product owner.
4. Операционные руководители вроде COO//CIO//CTO// CEO.
5. BDM (Business Development Manager)
6. Chief Product Owner – это уже ответственность за экосистему продуктов или целый ряд линейки продуктов

# Рекомендуемая литература

1. Deadline Том Демарко
2. Мифический человеком месяц, Брукс
3. Теория Ограничений Голдратта
4. Критическая цепь
5. Фасилитация Вилкинсона.
6. Путь скрам мастера, Шохова.
7. Скрам без ошибок, Голдштейн.
8. Постигая agile”
9. Лидер и племя
10. Dao Toyota
11. Гении и аутсайдеры
12. Проджект-менеджмент: как быть профессионалом, А. Минкевич,С. Дерцап
13. Управление высокотехнологичными программами и проектами
14. От хорошего к великому
15. Вдохновленные
16. Жесткий менеджмент, Ден Кенади
17. Возможно всё! Джон Вон Эйкен
18. Джедайские техники конструктивного общения, Александр Орлов
19. Регулярный менеджмент, Александр Фридман
20. Персональное управленческое искусство, Владимир Тарасов
21. Управление проектом на одной странице
22. Ненасильственное общение
23. Психбольница в руках пациентов,  Алан Купер
24. Интерфейс, Алан Купер
25. Сборники статей от Альпина паблишер: управление командой, методы принятия решений, эмоциональный интеллект.
26. Гид HBR разрешение конфликтов

# Для ищущих работу

## Часто задаваемые вопросы на интервью

**11 вопросов, которые нужно подготовить в 100% случаях**

1. Расскажи о себе (2-3 минуты) (задает тон всему интервью и дальнейшему разговору на собеседовании)
2. 3 главных достижения
3. 3 провала (и чему вы научились/как вышли из ситуации провала/что выявили)
4. 3 сильные стороны// 3 слабые стороны (подкрепленные примером и историей, как вы работаете над слабыми сторонами)
5. Почему именно эта индустрия
6. Почему именно эта компания (предварительно нужно познакомиться с сайтом компании и ее миссией, видением, понять, какие продукты выпускает и чем занимается)
7. Пример лидерства?

Расскажите о ситуации, когда вы проявили себя как лидера

8. Пример, как вы кого-либо переубедили (Например, когда команде и вам пришлось убеждать клиента в перспективности предлагаемого вами решения). На примере важно показать ваше умение быть убедительным в нужных ситуациях.
9. Пример конфликтной ситуации
10. Пример работы под давлением
11. Пример “entrepreneurial drive” (как вы что-то с нуля создали, сделали, внедрили и какие результаты были получены по итогу вашей инициативы)

**В чем разница между проектом и продуктом?**

Проект это временное предприятие, которое подразумевает наличие конечного результата, срока его достижения и ресурсы для его достижения.

Продукт — это по сути товар или услуга, который призван удовлетворять потребности потребителя.

**Какие этапы есть в жизненном цикле ПО**

- Формирование требований к ПО на стадии анализа,
- проектирование,
- реализация,
- тестирование,
- внедрение,
- эксплуатация и сопровождение,
- снятие с эксплуатации.

**Что такое стек?**

Это набор инструментов, которые команда применяет при работе в проектах. Он включает в себя языки программирования, фреймворки,

системы управления базами данных, компиляторы и другие инструменты.

**Что происходит, когда пользователь вводит запрос в браузер?**

Пользователь вводит в браузере адрес сайта>браузер ищет IP адрес сервера (такая информация хранится в распределенной системе серверов — DNS)

Не обнаружив подходящих записей в кэше, браузер формирует запрос к DNS-серверам, расположенным в интернете.

Например, если нужно найти IP-адрес сайта mail.vc.ru, браузер спрашивает у ближайшего DNS-сервера «Какой IP-адрес у сайта mail.imperium.im?».

Сервер может ответить: «Я не знаю про mail.imperium.im, но знаю сервер, который отвечает за imperium.im». Запрос переадресовывается дальше, на сервер «выше», пока в итоге один из серверов не найдет ответ об IP-адресе для сайта.

Как только браузер узнал IP-адрес нужного сервера, он пытается установить с ним соединение. В большинстве случаев для этого используется специальный протокол — TCP.

TCP — это набор правил, который описывает способы соединения между устройствами, форматы отправки запросов, действия в случае потери данных и так далее.

После установки соединения браузер отправляет специальный запрос, в котором просит сервер отправить данные для отображения страницы. В этом запросе содержится информация о самом браузере, временные файлы, требования к соединению и так далее.

Задача браузера — как можно подробнее объяснить серверу, какая именно информация ему нужна.

Сервер получил запрос от браузера с подробным описанием того, что ему требуется.

Теперь ему нужно обработать этот запрос. Этой задачей занимается специальное

серверное программное обеспечение — например, nginx или Apache. Чаще всего такие программы принято называть веб-серверами.

Веб-сервер в свою очередь перенаправляет запрос на дальнейшую обработку к программе-обработчику — например, PHP, Ruby или ASP.NET.

Программа внимательно изучает содержимое запроса — например, понимает, в каком формате нужно отправить ответ и какие именно файлы нужны. И собирает ответ.

Когда ответ сформирован, он отправляется веб-сервером обратно браузеру.

В ответе как правило содержится контент для отображения веб-страницы, информация о типе сжатия данных, способах кэширования, файлы cookie, которые нужно записать и так

далее.

Браузер распаковывает полученный ответ и постепенно начинает отображать полученный контент на экране пользователя — этот процесс называется рендерингом.

Сначала браузер загружает только основную структуру HTML-страницы.

Затем последовательно проверяет все теги и отправляет дополнительные

GET-запросы для получения с сервера различных элементов — картинки, файлы, скрипты, таблицы стилей и так далее. Поэтому по мере загрузки страницы браузер и сервер продолжают обмениваться между собой информацией.
Параллельно с этим на компьютер как правило сохраняются статичные файлы пользователя — чтобы при следующем посещении не загружать их заново и быстрее отобразить пользователю содержимое страницы.

Как только рендеринг завершен — пользователю отобразится полностью загруженная страница сайта.

**Расскажите о анализе сложной проблемы и решении которое вы нашли?**

**Расскажите о сложном проекте и как вы с ним справились?**

Когда мы работали над созданием виджета по продаже подарочных сертификатов, мои коллеги продали его Яндекс.Еде, хотя продукт был еще не готов. Таким образом наш дедлайн сильно сместился в сторону сжатия. В связи с этим наша команда(разработки и я) на протяжении 3 недель работала на износ, мы работали по выходным и по будням до поздна. Мне приходилось постоянно поддерживать в команде вовлеченность и доносить значимость того, что мы делаем. Не смотря на это, мы задержались на 3 дня, что не так критично учитывая обстоятельства. Забавно, что в итоге, после, мы практически полностью переписали сервис.

**Как вы мотивируете команду?**

Вовлекайте команду
Давайте сотруднику почувствовать, что он вносит вклад в общее дело ...
Вдохновляйте коллектив своей верой и оптимизмом ...
Ставьте четкие цели, постепенно поднимая планку ...
Давайте регулярный фидбек, отмечайте позитивные качества каждого

**Какую самую большую ошибку вы допустили при управлении проектом?**

**Есть ли у вас опыт проведения ретроспектив, разборов полетов и т.д.? Расскажите о нем.**

**Как ускорить процесс разработки если сроки горят?**

https://vc.ru/services/329766-kak-uskorit-process-razrabotki-produkta-esli-sroki-goryat-neskolko-prakticheskih-sovetov

**Методологии**

**Минусы Scrum**

- Сложности при заключении договоров. Scrum в принципе не подразумевает наличие фиксированного бюджета и фиксированного

технического задания, что затрудняет юридическое оформление такого рода договоренностей;
- Большое количество исключений. Специалисты в этой области считают данную методологию неприменимой для работы с государственными заказами, а также совершенно нерабочей при низкой квалификации команды, заниженных сроках работ или бюджете, некомпетентном менеджере проекта. В то время как другие методологии позволяют завершить проект при подобных условиях, хотя и на низком уровне;
- Узкая специализация методов. Так, например, если использовать Scrum при разработке сайтов, этапы дизайна и контента уже будут выходить за рамки методологии и требовать совершенно иного подхода.

**Отличие Scrum и Agile**

Основное отличие Scrum и Agile в том, что: Agile — это общая философия, стиль управления проектами, а Scrum — один из нескольких методов, используемых для реализации этого стиля, agile-фреймворк.

Ключевые сходства этих двух определений, отчетливо делающие Scrum Agile-процессом:

Краткосрочные циклы разработки.
Сосредоточенность на взаимодействии между людьми, сотрудничестве и общении.
Способность адаптироваться к изменениям и обратной связи.
Вот что отличает Scrum от других методологий Agile:

Работа разбита на спринты продолжительностью от одной до четырех недель.
Бэклог продукта содержит записи о том, какую работу необходимо выполнить.
Наличие Scrum-мастера и владельца продукта (product manager).
Члены команды проводят короткое «ежедневное собрание по обновлению Scrum».

**Что такое RAID в рамках управления проектами?**

RAID расшифровывается как риски (R, risks), допущения (A, assumptions), проблемы (I, issues) и зависимости (D, dependencies). Под аббревиатурой RAID понимают жизненно важные компоненты, которые должен знать любой project-менеджер. Во-первых, всегда есть риски, поэтому управляющий проектом должен уметь выбирать меньшее из зол. Во-вторых, пока допущения по действиям и оценкам ясны не будут, вероятность возникновения неприятных ситуаций весьма велика. В-третьих, проблемы, как и зависимости, ограничивают выбор действий.

**Что такое метод критического пути?**

Под критическим путём понимают все задачи, которые определяют окончательную дату запуска проекта. И если из-за одной задачи

график нарушается на день, весь проект задерживается на день. А ведь иногда возникают задачи, которые выходят за рамки критического пути. Для решения таковых в расписании должны быть предусмотрены специальные временные резервы.

**Что такое закон Парето?**

Это метод статистического анализа, помогающий прожект-менеджеру определять приоритеты между несколькими действиями. Закон основан на том, что «20 % усилий дают 80 % результата». В количественном анализе метод применяют, чтобы определить 20 % причин, которые создадут 80 % проблем.

**Какие вы знаете творческие методы решения проблем?**

При возникновении сложной ситуации важно вовлечь в её решение всю команду. Тут неплохой результат может дать классический мозговой штурм, а дополнит его система «шести шляп мышления».

**Как вы оцениваете применение 3-точечных методов оценки?**

Эта методика основана на 3-х различных оценочных значениях, что позволяет улучшать результат. Ожидаемое значение определяется посредством вычисления среднего из трёх:

— оптимистичного (O, optimistic value);
— пессимистичного (P, pessimistic value);
— наиболее вероятного (M, most likely value).

**На что должен обращать внимание project-менеджер для повышения эффективности управления задачами?**

Выделяют следующие сферы знаний:

— управление предметной областью;
— оптимизация затрат;
— планирование распределения ресурсов;
— контроль материально-технического обеспечения;
— управление рисками;
— улучшение процесса работы;
— сравнительный анализ производительности;
— автоматизация;
— планирование показателей качества;
— анализ первопричин отклонения от плана.

**Что такое тройственная ограниченность?**

Реализация любого проекта сопровождается ограничениями. Основные три фактора — это время, объём и стоимость. Мы можем образовать из них треугольник, где каждый фактор — это вершина, а качество — это центр:

1) проекты надо выполнять в рамках запланированного бюджета;
2) проекты надо выполнять вовремя;
3) проекты должны отвечать согласованному объёму;
4) проекты должны отвечать требованиям клиента.

**Что такое система поддержки решений?**

Система этого типа часто используется в корпорациях и основана на информационной системе управления. Есть два типа СПР — неструктурированная и структурированная. СПР соединяет множество блок-схем в организационном процессе, её можно сформулировать и построить и статистически, и стохастически.

**Что такое диаграмма рыбьей кости?**

Она же диаграмма Исикавы и причинно-следственных связей. Отображает первопричины определённого события. В контексте управления инструмент применяется для мониторинга рисков и контроля качества.

**Что такое интегрированная модель зрелости процессов ПО (CMMI)?**

CMMI — совершенствование процессов и видов деятельности в компаниях разных размеров. Содержит набор рекомендаций в виде практик. Их применение, по мнению разработчиков модели, даёт возможность достичь целей, необходимых для полной реализации поставленных задач в определенных областях деятельности.

**Что такое метод «Шести сигм»?**

«Шесть сигм» — это показатель качества, который стремится к совершенству. Метод основан на данных для устранения дефектов. Зрелость производственного процесса описывается в качестве σ-рейтинга отклонений либо процентом бездефектной продукции на выходе. При грамотном применении помогает завершить проект вовремя и снижать риск провала.

**Что такое реестр рисков?**

Это документ, включающий в себя все выявленные риски проекта, а также список обязательных и потенциальных действий.

**Как определять объем работ?**

Для этого используют методики разбивки продукта, анализа требований, системной инженерии, системного анализа, анализа стоимости и анализа альтернатив. Последний реализуется путём мозгового штурма либо парного сопоставления.

**Что такое система распределения работ?**

Система распределения работ определяет перечень действий, необходимых для проекта, включая вспомогательные действия. Система основана на уровнях, причём вся работа требует чёткого понимания. Используется дневник распределения работ, включающий в себя дополнительные детали. Когда всё известно о работе и зависимостях, тогда оценки времени и усилий будут точны.

## Какие вопросы стоит задать рекрутеру

Обязательные:
0. Причина открытия вакансии?
1. Какие возможности роста предусмотрены в компании на обсуждаемой должности?/Каков типичный карьерный путь человека на этой должности?
2. Какие предусматриваются график и формат работы (удаленка/офис/гибрид)?
3. С какими членами команды происходит ежедневная коммуникация?
4. Какие бенефиты предоставляет компания (дмс, английский, спорт, корпоративы, оплата питания)?
5. Какие варианты оформления сотрудников возможны?

Про позицию:
1. Как будет выглядеть мой типичный рабочий день?
2. По каким критериям в вашей компании оценивается эффективность сотрудника на данной должности?
3. С какими сложностями я столкнусь на этой должности?
4. Каким вы видите идеального кандидата на эту позицию?

Про компанию:
1. Почему Вам (рекрутеру) нравится работать в этой компании?
2. Как вы можете охарактеризовать рабочую среду компании?
3. Как выстроено внутреннее обучение сотрудников в компании?

Про 💶💶💶:
1. Как выстраивается зарплата?
2. Бывают ли овертаймы и как они оплачиваются?
3. Как происходит процесс повышения зарплаты?

На еще что обратить внимание:
• Дистанция между вами и рекрутером. Так вы поймете, какой стиль общения принят в компании.
• Дают ли вам всю нужную информацию или избегают ответов и кормят абстракциями?
• Ментальный метч. Конечно, собеседование — это стресс. Но если вы чувствуете себя комфортно — это хороший знак.
• Психологическое состояние сотрудников. Бывают разные ситуации, но если усталость потенциальных коллег бросается в глаза, обязательно спросите про овертаймы.

**by Aleksandr Konovalov**
**Telegram LinkedIn Mail**