Управление проектами

Agile, Scrum, Kanban — управление проектами и командами

UNIKA ACADEMY

Учебник курса · 2026

Блок 1: Основы управления проектами

Урок 1.1: Что такое управление проектами – принципы и роли

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Управление проектами – это не просто модное словосочетание из корпоративного мира. Это дисциплина, которая определяет, будет ли ваша идея реализована вовремя, в рамках бюджета и с нужным качеством – или же утонет в хаосе, переработках и конфликтах. По данным PMI (Project Management Institute), организации, которые инвестируют в проверенные практики управления проектами, теряют в 28 раз меньше денег из-за провальных инициатив.

В этом уроке мы заложим фундамент: разберём, что такое проект (и чем он отличается от операционной деятельности), какие международные стандарты существуют (PMBOK, PRINCE2, ISO 21500), какие роли формируют проектную команду и какие принципы определяют успех. Это базовые знания, без которых невозможно двигаться дальше.

Независимо от того, управляете ли вы IT-разработкой, маркетинговой кампанией, строительством или запуском стартапа – принципы остаются универсальными. После этого урока вы будете понимать терминологию, видеть структуру любого проекта и знать, какую роль вы занимаете (или хотите занять) в проектной команде.

Что такое проект и чем он отличается от процесса

Определение проекта

Согласно PMBOK 7th Edition (PMI, 2021), **проект** – это временное предприятие, направленное на создание уникального продукта, услуги или результата. Ключевые характеристики:

- **Временность** – у проекта есть чёткое начало и конец
- **Уникальность** – результат проекта не является повторением чего-то существующего
- **Прогрессивная детализация** – детали уточняются по мере продвижения
- **Ограниченность ресурсов** – бюджет, время, люди конечны

Проект vs Операционная деятельность

Характеристика	Проект	Операционная деятельность
Длительность	Временная	Постоянная
Результат	Уникальный	Повторяющийся
Цель	Достичь результата и завершиться	Поддерживать бизнес-процессы
Команда	Формируется под проект	Постоянный штат
Пример	Запуск нового продукта	Ежедневная обработка заказов

Примеры проектов из разных сфер

- **IT:** Разработка мобильного приложения для интернет-банкинга
- **Маркетинг:** Запуск рекламной кампании к Чёрной пятнице
- **Строительство:** Возведение жилого комплекса
- **Образование:** Создание онлайн-курса (именно то, что вы сейчас изучаете)
- **Стартап:** Запуск MVP и привлечение первых 100 клиентов

Международные стандарты управления проектами

PMBOK (Project Management Body of Knowledge)

Разработан PMI (США). Самый распространённый стандарт в мире. PMBOK 7th Edition (2021) перешёл от процессного подхода к принципам:

12 принципов PMBOK 7: 1. Будьте усердным, уважительным и заботливым распорядителем 2. Создавайте среду для совместной работы 3. Эффективно взаимодействуйте со стейкхолдерами 4. Фокусируйтесь на ценности 5. Распознавайте, оценивайте и реагируйте на системные взаимодействия 6. Демонстрируйте лидерское поведение 7. Адаптируйте подход к контексту 8. Встраивайте качество в процессы и результаты 9. Управляйте сложностью 10. Оптимизируйте реакцию на риски 11. Принимайте адаптивность и устойчивость 12. Содействуйте изменениям для достижения видения будущего состояния

PRINCE2 (Projects IN Controlled Environments)

Британский стандарт, особенно популярен в Европе и государственном секторе. Основан на 7 принципах, 7 темах и 7 процессах. Ключевое отличие – чёткое разделение ответственности между уровнями управления (стратегический, тактический, операционный).

ISO 21500

Международный стандарт, обеспечивающий общую рамку для управления проектами. Менее детальный, чем PMBOK или PRINCE2, но признан на международном уровне.

Ключевые роли в проектной команде

Проектный менеджер (PM)

Центральная фигура. Отвечает за планирование, выполнение, мониторинг и закрытие проекта. Ключевые компетенции PM по PMI Talent Triangle:

- **Техническое управление проектами** – инструменты, методологии, процессы
- **Лидерство** – мотивация команды, управление конфликтами, принятие решений
- **Стратегическое и бизнес-управление** – понимание бизнес-контекста, ROI, стратегии

Спонсор проекта

Руководитель верхнего уровня, который обеспечивает финансирование, принимает ключевые решения и устраняет эскалированные проблемы. Без сильного спонсора проекты часто буксуют.

Заказчик (Product Owner в Agile)

Тот, кто определяет требования к результату. В Agile – Product Owner, который приоритизирует бэклог и принимает результаты спринтов.

Команда проекта

Специалисты, выполняющие работу: разработчики, дизайнеры, аналитики, тестировщики, маркетологи. В зависимости от методологии команда может быть функциональной (каждый в своём отделе) или кросс-функциональной (все вместе).

Стейкхолдеры

Все, кого затрагивает проект: пользователи, руководство, регуляторы, смежные отделы. Подробнее разберём в Уроке 1.9.

RACI-матрица

Инструмент для распределения ответственности: - **R (Responsible)** – исполнитель, делает работу - **A (Accountable)** – ответственный, принимает решение - **C (Consulted)** – консультант, с кем советуются - **I (Informed)** – информируемый, кого ставят в известность

Задача	PM	Спонсор	Разработчик	Дизайнер
Утверждение ТЗ	A	I	C	C
Разработка MVP	A	I	R	R
Приёмка результата	R	A	I	I

Треугольник проектных ограничений

Классическая модель “Iron Triangle” определяет три основных ограничения любого проекта:

- **Scope (Содержание)** – что именно нужно сделать
- **Time (Время)** – к какому сроку
- **Cost (Стоимость)** – за какой бюджет

В центре – **Quality (Качество)**. Изменение одного ограничения неизбежно влияет на другие. Если заказчик добавляет функционал (scope), то увеличивается либо срок, либо бюджет, либо страдает качество.

Современные фреймворки добавляют ещё три ограничения: **Risk (Риски)**, **Resources (Ресурсы)** и **Customer Satisfaction (Удовлетворённость заказчика)**, формируя “звезду” из шести ограничений.

Практический пример

Клиент просит добавить чат-бота на сайт (scope +). Варианты: 1. Сдвинуть дедлайн на 2 недели (time +) 2. Нанять дополнительного разработчика (cost +) 3. Убрать другую фичу из скоупа (scope –) 4. Сделать бота минимальным, без AI (quality –)

Задача PM – показать заказчику эти варианты и принять обоснованное решение.

Инструменты для управления проектами: обзор

Современный PM использует специализированный софт. Краткий обзор:

- **Jira** – стандарт в IT, мощная система для Agile-команд, спринты, бэклоги, Kanban-доски
- **Asana** – универсальный инструмент для любых команд, отличные визуализации и автоматизации
- **Notion** – гибкий инструмент для документации, баз знаний, трекинга задач и вики проекта
- **Monday.com** – визуально привлекательный инструмент с кастомными воркфлоу
- **MS Project** – классика для Waterfall-проектов, диаграммы Ганта, критический путь
- **Trello** – простые Kanban-доски, подходит для небольших команд и личных проектов
- **ClickUp** – all-in-one решение, пытается заменить все остальные инструменты

В этом курсе мы будем работать с Notion (как базой знаний) и Jira/Trello (как трекерами задач). Выбор инструмента зависит от размера команды, методологии и бюджета.

Что дальше

В следующем уроке (1.2) мы разберём жизненный цикл проекта – от первоначальной идеи до формального закрытия. Вы узнаете, через какие фазы проходит каждый проект и какие ключевые артефакты создаются на каждом этапе.

Урок 1.1 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.2: Жизненный цикл проекта – от идеи до закрытия

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Каждый проект, от разработки мобильного приложения до строительства моста, проходит через определённые фазы. Понимание жизненного цикла проекта – это как наличие карты перед путешествием: вы знаете, где находитесь, что уже позади и что впереди. Без этого понимания менеджеры часто застревают в одной фазе, пропускают критические шаги или начинают выполнение работ без надлежащего планирования.

PMBOK описывает пять групп процессов (инициация, планирование, исполнение, мониторинг и контроль, закрытие), а PRINCE2 выделяет семь процессов с акцентом на управление по стадиям. В реальном мире эти модели адаптируются под конкретный контекст: стартап не будет следовать тому же циклу, что и государственный проект.

В этом уроке мы пройдём весь путь проекта от начала до конца, разберём ключевые артефакты каждой фазы и узнаем, какие решения являются критическими на каждом этапе. Вы получите практическую карту, которую сможете применить к любому проекту.

Фаза 1: Инициация проекта

Что происходит

На этом этапе рождается идея и принимается решение: стоит ли этот проект ресурсов? Главные вопросы: зачем этот проект нужен, кому он нужен и выполним ли он в принципе?

Ключевые артефакты

Project Charter (Устав проекта) – главный документ инициации. Содержит: - Название и описание проекта - Бизнес-обоснование (зачем?) - Цели и критерии успеха - Высокоуровневые требования - Высокоуровневые риски - Предварительный бюджет и сроки - Назначение проектного менеджера - Подпись спонсора

Business Case (Бизнес-кейс) – экономическое обоснование. Включает анализ ROI, период окупаемости, NPV (чистая приведённая стоимость). В PRINCE2 бизнес-кейс – это живой документ, который обновляется на протяжении всего проекта.

Feasibility Study (Технико-экономическое обоснование) – для крупных проектов: техническая реализуемость, доступность ресурсов, правовые ограничения.

Пример: Устав проекта для запуска e-commerce

Поле	Содержание
Название	Запуск интернет-магазина бытовой техники
Цель	Увеличить продажи на 30% через онлайн-канал
Бюджет	$80,000
Срок	4 месяца
PM	Иванова А.С.
Спонсор	Директор по развитию
Критерий успеха	500 заказов в первый месяц после запуска

Решение Go/No-Go

По итогам инициации принимается решение: запускаем проект (Go) или отклоняем (No-Go). Около 30% идей отсеиваются на этом этапе – и это нормально. Лучше отказаться от слабого проекта на старте, чем потратить ресурсы впустую.

Фаза 2: Планирование

Что происходит

Самая объёмная фаза по количеству артефактов. Здесь создаётся дорожная карта: что делать, кто делает, когда, за какие деньги и с какими рисками.

Ключевые артефакты

- **WBS (Work Breakdown Structure)** – декомпозиция работ (подробнее в Уроке 1.5)
- **Расписание проекта** – диаграмма Ганта, критический путь (Урок 1.6)
- **Бюджет проекта** – детальная смета (Урок 1.8)
- **План управления рисками** – реестр рисков, стратегии реагирования (Урок 1.7)
- **Коммуникационный план** – кто, кому, что и когда сообщает
- **План управления качеством** – стандарты и процедуры проверки
- **План управления ресурсами** – люди, оборудование, материалы

Правило “1:10:100”

Ошибка, обнаруженная на этапе планирования, стоит 1 единицу ресурсов на исправление. Та же ошибка на этапе исполнения стоит 10 единиц. На этапе закрытия или после запуска – 100 единиц. Поэтому инвестиции в качественное планирование всегда окупаются.

Типичная ошибка

“Давайте не будем тратить время на планирование и сразу начнём делать” – самая дорогая фраза в проектном управлении. Статистика PMI: проекты с формальным планированием завершаются успешно в 2.5 раза чаще.

Фаза 3: Исполнение

Что происходит

Команда выполняет запланированные работы. PM координирует, решает проблемы, управляет коммуникациями и мотивирует команду. На эту фазу приходится 60-80% бюджета проекта.

Ключевые активности PM

- Координация команды и распределение задач
- Проведение регулярных статус-митингов
- Управление коммуникациями со стейкхолдерами
- Управление закупками и подрядчиками
- Развитие команды и решение конфликтов
- Обеспечение качества результатов

Daily Standup (ежедневная встреча)

В Agile-командах – 15-минутная встреча, где каждый отвечает на три вопроса: 1. Что сделал вчера? 2. Что планирую сегодня? 3. Есть ли блокеры?

В Waterfall-проектах – еженедельные статус-митинги с отчётностью по KPI.

Фаза 4: Мониторинг и контроль

Что происходит

Эта фаза идёт параллельно с исполнением. PM отслеживает отклонения от плана и принимает корректирующие действия.

Ключевые метрики

EVM (Earned Value Management) – метод оценки прогресса проекта: - **PV (Planned Value)** – плановая стоимость запланированных работ - **EV (Earned Value)** – плановая стоимость выполненных работ - **AC (Actual Cost)** – фактические затраты - **SPI (Schedule Performance Index)** = EV/PV (< 1 = отставание от графика) - **CPI (Cost Performance Index)** = EV/AC (< 1 = перерасход бюджета)

Управление изменениями

Запросы на изменение (Change Requests) – неизбежная часть любого проекта. Формальный процесс: 1. Инициатор подаёт запрос на изменение 2. PM анализирует влияние на scope, time, cost, quality 3. Change Control Board (CCB) принимает решение 4. Утверждённые изменения вносятся в план

Без формального процесса управления изменениями проект страдает от “scope creep” – неконтролируемого разрастания содержания.

Фаза 5: Закрытие проекта

Что происходит

Формальное завершение проекта. Многие PM пренебрегают этой фазой, но она критически важна для накопления опыта и формального освобождения ресурсов.

Ключевые активности

- **Приёмка результатов** – формальное подтверждение заказчиком, что результат соответствует требованиям
- **Lessons Learned (Извлечённые уроки)** – ретроспектива: что сработало, что нет, что улучшить
- **Архивация документации** – все артефакты сохраняются для будущих проектов

- **Освобождение ресурсов** – команда переходит на другие проекты
- **Финальный отчёт** – сравнение план/факт по всем параметрам
- **Celebration** – празднование завершения (недооценённый, но важный элемент)

Шаблон Lessons Learned

Категория	Что планировали	Что получилось	Причина отклонения	Рекомендация
Сроки	4 месяца	5 месяцев	Задержка с дизайном	Закладывать буфер 20% на креативные задачи
Бюджет	$80K	$75K	Нашли более дешёвого подрядчика	Проводить тендер среди 3+ подрядчиков
Качество	0 критических багов	2 бага на запуске	Недостаточное тестирование	Добавить фазу UAT минимум 2 недели

Предиктивный vs Адаптивный жизненный цикл

В зависимости от степени неопределённости проекта выбирается тип жизненного цикла:

Предиктивный (Waterfall): - Фазы идут последовательно - Требования фиксируются в начале - Подходит для проектов с низкой неопределённостью (строительство, производство)

Адаптивный (Agile): - Работа ведётся короткими итерациями (спринтами) - Требования уточняются по ходу - Подходит для проектов с высокой неопределённостью (IT, инновации)

Гибридный: - Комбинация предиктивного и адаптивного подходов - Наиболее распространён в реальной практике - Пример: планирование по Waterfall, разработка по Agile

Подробнее сравнение Waterfall и Agile разберём в следующем уроке (1.3).

Что дальше

В уроке 1.3 мы глубоко погрузимся в сравнение двух основных подходов – Waterfall и Agile. Вы узнаете, когда какой подход применять, и разберёте реальные кейсы выбора методологии.

Урок 1.2 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.3: Waterfall vs Agile – когда что использовать

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Спор между Waterfall и Agile – это, пожалуй, самая горячая дискуссия в мире управления проектами. Одни утверждают, что Agile – единственный правильный подход, другие считают, что Waterfall незаменим для серьёзных проектов. Правда, как обычно, посередине: каждый подход имеет свои сильные стороны и свою область применения.

Waterfall (каскадная модель) появился в 1970-х годах и доминировал десятилетия. Agile Manifesto был опубликован в 2001 году, и с тех пор гибкие методологии завоевали огромную популярность, особенно в IT. По данным State of Agile Report 2024, 71% организаций используют Agile в той или иной форме. Но при этом Waterfall по-прежнему доминирует в строительстве, промышленности и государственных проектах.

В этом уроке мы не просто сравним два подхода – мы дадим вам конкретный фреймворк для принятия решения, какой подход использовать в вашем проекте. Плюс разберём популярные Agile-фреймворки: Scrum, Kanban и SAFe.

Waterfall: каскадная модель

Принцип работы

Фазы проекта выполняются последовательно, каждая начинается после завершения предыдущей. Как водопад: вода течёт только вниз, вернуться назад нельзя (или очень дорого).

Последовательность фаз: 1. Сбор требований (Requirements) 2. Проектирование (Design) 3. Разработка (Implementation) 4. Тестирование (Testing) 5. Развёртывание (Deployment) 6. Поддержка (Maintenance)

Преимущества Waterfall

- **Предсказуемость** – чёткие сроки, бюджет и результат определены заранее
- **Документация** – полная документация на каждом этапе
- **Простота управления** – линейный процесс легко отслеживать
- **Подходит для фиксированных контрактов** – scope, time, cost зафиксированы
- **Регуляторные требования** – в медицине, авиации, госсекторе часто обязателен

Недостатки Waterfall

- **Негибкость** – изменения требований после начала разработки очень дороги
- **Поздняя обратная связь** – заказчик видит результат только в конце
- **Высокий риск** – если требования поняты неверно, вся работа может быть бесполезна
- **Долгий Time-to-Market** – продукт выходит на рынок только после полного завершения

Когда использовать Waterfall

- Требования стабильны и хорошо определены
- Проект имеет фиксированный scope, бюджет и сроки
- Регуляторная среда требует полной документации
- Технология хорошо изучена, нет неопределённости
- Пример: строительство здания, внедрение ERP-системы по утверждённому ТЗ

Agile: гибкий подход

Agile Manifesto (2001)

4 ценности Agile: 1. **Люди и взаимодействие** важнее процессов и инструментов 2. **Работающий продукт** важнее исчерпывающей документации 3. **Сотрудничество с заказчиком** важнее согласования условий контракта 4. **Готовность к изменениям** важнее следования первоначальному плану

12 принципов Agile подчёркивают: ранняя и непрерывная поставка ценности, приветствие изменений, частая поставка работающего продукта, самоорганизующиеся команды, регулярная рефлексия.

Scrum – самый популярный Agile-фреймворк

Роли: - Product Owner – определяет ЧТО делать (бэклог, приоритеты) - Scrum Master – обеспечивает КАК работает процесс (устраняет препятствия) - Development Team – 5-9 человек, кросс-функциональная команда

Артефакты: - Product Backlog – упорядоченный список всех требований - Sprint Backlog – задачи на текущий спринт - Increment – работающий результат спринта (потенциально готовый к релизу)

События (церемонии): - Sprint Planning – планирование спринта (2-4 часа) - Daily Scrum – ежедневная встреча (15 мин) - Sprint Review – демонстрация результатов (1-2 часа) - Sprint Retrospective – ретроспектива процесса (1-1.5 часа)

Спринт длится 1-4 недели (чаще всего 2 недели).

Kanban

Альтернатива Scrum, основанная на визуализации потока работы:

- **Kanban-доска** – колонки: To Do, In Progress, Review, Done
- **WIP-лимиты** – ограничение количества задач в работе одновременно
- **Непрерывный поток** – нет спринтов, задачи берутся по мере готовности
- **Метрики:** Lead Time (время от запроса до выполнения), Cycle Time (время в работе), Throughput (количество задач за период)

Kanban подходит для команд поддержки, DevOps, и ситуаций, где спринты неудобны.

SAFe (Scaled Agile Framework)

Для крупных организаций, где Agile нужно масштабировать на десятки или сотни команд. Уровни: Team, Program (ART), Large Solution, Portfolio. Используют Boeing, Cisco, Intel и другие корпорации.

Сравнительная таблица

Критерий	Waterfall	Agile (Scrum)
Подход к требованиям	Фиксированы в начале	Эволюционируют
Поставка ценности	В конце проекта	Каждый спринт (2-4 недели)
Роль заказчика	Утверждает ТЗ, принимает результат	Активно участвует постоянно
Документация	Полная, формальная	Минимально достаточная
Управление изменениями	Формальный Change Request	Естественная часть процесса
Риск	Высокий (поздняя обратная связь)	Низкий (ранняя обратная связь)
Планирование	Детальное в начале	Прогрессивная детализация
Метрики	EVM, Gantt, Milestones	Velocity, Burndown, Lead Time
Размер команды	Любой	5-9 человек (Scrum Team)
Контракт	Fixed Price	Time & Material

Фреймворк выбора подхода: Cynefin + Stacey Matrix

Cynefin Framework (Дейв Сноуден)

Помогает определить тип проблемы:

- **Simple (Простая)** – причинно-следственные связи очевидны. Используй best practices. Waterfall.
- **Complicated (Сложная)** – нужна экспертиза для анализа. Good practices. Waterfall или гибрид.
- **Complex (Комплексная)** – причинно-следственные связи видны только постфактум. Emergent practices. Agile.
- **Chaotic (Хаотичная)** – нет причинно-следственных связей. Novel practices. Быстрые эксперименты.

Stacey Matrix

Два измерения: определённость требований и определённость технологии.

- **Оба определены** – Waterfall

- **Требования неопределённы, технология известна** – Agile
- **Технология неопределённа, требования ясны** – прототипирование + Waterfall
- **Оба неопределённы** – Lean Startup, дизайн-мышление, Agile

Практический чек-лист выбора

Ответьте на 5 вопросов: 1. Могу ли я зафиксировать все требования в начале? (Да = Waterfall) 2. Нужна ли частая обратная связь от заказчика? (Да = Agile) 3. Есть ли регуляторные требования к документации? (Да = Waterfall) 4. Насколько велика неопределённость? (Высокая = Agile) 5. Какой контракт? (Fixed Price = Waterfall, T&M = Agile)

Гибридный подход: реальность 2026 года

В реальной практике чистый Waterfall или чистый Agile встречается редко. Большинство компаний используют гибридный подход:

- **Water-Scrum-Fall:** Планирование и закрытие по Waterfall, разработка по Scrum
- **Agile с гейтами:** Agile-спринты с контрольными точками (gates) для стейкхолдеров
- **Disciplined Agile (DA):** Фреймворк PMI, который помогает выбрать правильные практики для контекста

Пример: банк разрабатывает новое мобильное приложение. Высокоуровневое планирование и архитектура – Waterfall (2 месяца). Разработка фич – Scrum-спринты по 2 недели (4 месяца). Регуляторное тестирование и деплой – Waterfall (1 месяц).

Что дальше

В уроке 1.4 мы разберём, как правильно ставить цели проекта, используя фреймворки SMART, OKR и KPI. Без чётких целей ни Waterfall, ни Agile не спасут проект от провала.

Урок 1.3 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.4: Постановка целей – SMART, OKR, KPI

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Проект без чётких целей – как корабль без компаса. Команда может работать невероятно усердно, но если нет ясного понимания, куда вы идёте и как измерить прогресс, усилия будут потрачены впустую. Исследование Гарвардского университета показало: люди, записывающие свои цели, достигают их в 10 раз чаще тех, кто этого не делает.

В управлении проектами существуют три ключевых фреймворка для постановки и отслеживания целей: SMART – для формулирования конкретных целей, OKR – для стратегического выравнивания команды, и KPI – для измерения эффективности. Эти инструменты не конкурируют друг с другом – они дополняют друг друга и часто используются совместно.

В этом уроке вы научитесь формулировать цели, которые действительно работают, связывать их с измеримыми показателями и каскадировать от стратегии компании до конкретных задач проекта. Мы разберём каждый фреймворк на практических примерах из разных отраслей.

SMART: фундамент целеполагания

Что такое SMART

SMART – это акроним, описывающий критерии качественно сформулированной цели. Разработан Джорджем Дораном в 1981 году и с тех пор стал стандартом де-факто.

- **S (Specific)** – Конкретная. Что именно нужно достичь? Кто участвует? Где?
- **M (Measurable)** – Измеримая. Как понять, что цель достигнута? Какие числа?
- **A (Achievable)** – Достижимая. Реально ли это с имеющимися ресурсами?
- **R (Relevant)** – Значимая. Почему это важно? Как связано со стратегией?
- **T (Time-bound)** – Ограниченная по времени. К какому сроку?

Примеры: плохо vs хорошо

Плохая цель	Проблема	SMART-цель
Увеличить продажи	Нет цифр, нет срока	Увеличить онлайн-продажи на 25% за Q3 2026 через запуск нового e-commerce сайта
Улучшить приложение	Неконкретно	Снизить время загрузки главной страницы с 4с до 1.5с к 1 мая 2026
Привлечь больше клиентов	Нет метрики	Привлечь 500 новых платящих клиентов через контекстную рекламу за $50 CAC к концу Q2 2026
Повысить качество	Абстрактно	Снизить количество критических багов в продакшне с 15 до 3 в месяц к августу 2026

Углублённый пример: проект редизайна сайта

Общая цель проекта (SMART): "Провести полный редизайн корпоративного сайта с миграцией на Next.js, увеличив конверсию посетитель-лид с 2.1% до 4.5% и снизив bounce rate с 65% до 40%, к 15 сентября 2026 года, в рамках бюджета $45,000."

Эта цель проверяема по каждому критерию SMART: - S: редизайн корпоративного сайта, миграция на Next.js - M: конверсия 2.1% -> 4.5%, bounce rate 65% -> 40% - A: бюджет $45K, стек Next.js – реалистично для команды - R: увеличение конверсии напрямую влияет на выручку - T: к 15 сентября 2026

OKR: стратегическое выравнивание

Что такое OKR

OKR (Objectives and Key Results) – фреймворк, разработанный Энди Гроувом в Intel и популяризированный Джоном Дорром в Google. Используется Google, LinkedIn, Twitter, Spotify, Uber и тысячами других компаний.

- **Objective (Цель)** – качественное, вдохновляющее описание того, чего хотим достичь. Амбициозное, но понятное.
- **Key Results (Ключевые результаты)** – 2-5 количественных метрик, которые показывают, достигнута ли цель. Каждый KR имеет числовое значение.

Принципы OKR

1. **Амбициозность** – OKR должны быть stretch goals. Достижение 70% считается хорошим результатом
2. **Прозрачность** – все OKR компании видны всем сотрудникам
3. **Каскадирование** – OKR компании -> OKR отдела -> OKR команды -> OKR сотрудника
4. **Каденция** – обычно квартальные OKR, пересматриваются каждые 3 месяца
5. **Не привязаны к бонусам** – OKR не должны влиять на KPI зарплаты, иначе люди будут ставить заниженные цели

Пример OKR для проекта запуска мобильного приложения

Objective: Запустить мобильное приложение, которое станет основным каналом взаимодействия с клиентами

Key Results: - KR1: Достичь 10,000 скачиваний в первый месяц после запуска - KR2: Добиться рейтинга 4.5+ в App Store и Google Play - KR3: Перевести 30% текущих клиентов на использование приложения - KR4: Обеспечить retention rate 40% на 30-й день

Каскадирование OKR: от компании к проекту

Уровень компании: Objective: Стать лидером рынка цифровых финансовых услуг в регионе - KR1: Увеличить долю рынка с 12% до 20% - KR2: Достичь NPS 60+

Уровень проекта (мобильное приложение): Objective: Создать лучший мобильный банкинг в стране - KR1: 10,000 скачиваний за первый месяц - KR2: Рейтинг 4.5+ в сторах - KR3: 30% клиентов в приложении

Уровень команды (разработка): Objective: Обеспечить безупречное качество приложения - KR1: 0 критических багов при запуске - KR2: Время отклика API < 200ms - KR3: Покрытие тестами > 80%

KPI: метрики эффективности

Что такое KPI

KPI (Key Performance Indicators) – ключевые показатели эффективности, которые измеряют прогресс к цели. В отличие от OKR, KPI – это постоянно отслеживаемые метрики, часто привязанные к операционной деятельности.

Типы KPI в проектном управлении

Lagging indicators (запаздывающие) – результат уже произошёл: - Выручка проекта - ROI - Количество завершённых проектов - Удовлетворённость клиента (CSAT)

Leading indicators (опережающие) – предсказывают будущий результат: - Velocity команды (сколько story points за спринт) - Code coverage (покрытие тестами) - Pipeline (количество проектов в воронке) - Количество блокеров в спринте

KPI-дашборд проекта: шаблон

KPI	Целевое значение	Текущее значение	Статус
Прогресс по задачам	60% (к середине проекта)	55%	Жёлтый
Расход бюджета	< $22,500 (50% от бюджета)	$21,800	Зелёный
Критические баги	0	2	Красный
Velocity команды	40 SP/спринт	38 SP/спринт	Жёлтый
Удовлетворённость стейкхолдеров	> 8/10	8.5/10	Зелёный

RAG-статус (Red-Amber-Green)

- **Зелёный** – в рамках плана, отклонение < 5%
- **Жёлтый (Amber)** – незначительное отклонение 5-15%, нужно внимание
- **Красный** – критическое отклонение > 15%, требуется эскалация

Как связать SMART, OKR и KPI вместе

Модель интеграции

1. **OKR** задают стратегическое направление (квартал/год)
2. **SMART** формулируют конкретные цели проекта
3. **KPI** отслеживают прогресс в реальном времени

Пример интеграции: проект улучшения клиентского сервиса

OKR (квартал): Objective: Создать лучший клиентский опыт на рынке - KR1: Увеличить NPS с 35 до 55 - KR2: Снизить среднее время ответа с 4 часов до 30 минут - KR3: Достичь FCR (First Contact Resolution) 80%

SMART-цель проекта: "Внедрить систему тикетов Zendesk и AI-чатбота, обеспечив время первого ответа < 2 минут и FCR > 75% к 1 июля 2026, бюджет $30,000."

KPI (еженедельный мониторинг): - Среднее время первого ответа - FCR (First Contact Resolution Rate) - CSAT (Customer Satisfaction Score) - Количество тикетов в очереди - Процент тикетов, решённых AI-ботом

Инструменты для целеполагания и отслеживания

- **Notion** – OKR-трекер, дашборды, база знаний
- **Gtmhub (Quantive)** – специализированный OKR-инструмент
- **Jira** – KPI через дашборды и отчёты
- **Google Sheets / Excel** – KPI-дашборды для небольших команд
- **Looker / Tableau / Power BI** – продвинутая аналитика для крупных проектов
- **Asana Goals** – встроенный OKR-функционал

Что дальше

В уроке 1.5 мы разберём WBS (Work Breakdown Structure) – технику декомпозиции целей проекта на конкретные задачи. Вы научитесь разбивать любой проект на управляемые пакеты работ.

Урок 1.4 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.5: Work Breakdown Structure – декомпозиция задач

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Вы поставили SMART-цели, определили OKR и выбрали методологию. Теперь встаёт главный вопрос: как превратить большую цель в конкретные задачи, которые можно назначить людям и отслеживать? Ответ – Work Breakdown Structure (WBS), один из самых мощных и при этом самых недооценённых инструментов проектного управления.

WBS – это иерархическая декомпозиция всего объёма работ проекта на управляемые пакеты. Проще говоря, вы берёте слона и режете его на стейки. Без WBS проектный план – это набор размытых задач типа "сделать дизайн" или "разработать бэкенд". С WBS каждая задача конкретна, измерима и назначаема.

В PMBOK WBS является обязательным артефактом фазы планирования. Это основа для оценки сроков (Урок 1.6), бюджета (Урок 1.8) и распределения ответственности. В этом уроке вы научитесь строить WBS для любого проекта, используя правило 100% и принцип 8/80.

Что такое WBS

Определение

Work Breakdown Structure (Иерархическая структура работ) – это ориентированная на результат иерархическая декомпозиция работ, которые должна выполнить команда проекта для достижения целей проекта. Каждый уровень WBS представляет всё более детальное определение работ.

Структура WBS

Уровень 0: Проект (весь проект целиком) **Уровень 1:** Фазы или основные компоненты (deliverables) **Уровень 2:** Подкомпоненты **Уровень 3:** Рабочие пакеты (Work Packages) – самый нижний уровень WBS **Уровень 4+:** Активности (Activities) – конкретные задачи внутри рабочих пакетов (это уже за пределами WBS, в расписании)

Правило 100%

Суммарный объём работ на каждом уровне должен составлять 100% работ родительского элемента. Ничего лишнего (gold plating), ничего пропущенного. Если вы разбили “Разработка сайта” на “Фронтенд” и “Бэкенд”, но забыли “Тестирование” и “Деплой” – правило 100% нарушено.

Правило 8/80

Каждый рабочий пакет (самый нижний уровень WBS) должен занимать от 8 до 80 часов работы. Если меньше 8 – слишком мелкая детализация, усложняет управление. Если больше 80 – нужно декомпозировать дальше.

Как строить WBS: пошаговый алгоритм

Шаг 1: Определите основные deliverables (результаты)

Начните с вопроса: “Какие крупные результаты должен дать этот проект?” Это будут элементы Уровня 1.

Пример для проекта “Запуск мобильного приложения”: 1. Аналитика и проектирование 2. UX/UI дизайн 3. Разработка 4. Тестирование 5. Запуск и маркетинг 6. Управление проектом

Шаг 2: Декомпозируйте каждый deliverable

Разбейте каждый элемент Уровня 1 на подкомпоненты.

1. Аналитика и проектирование: - 1.1 Исследование рынка и конкурентов - 1.2 Сбор и документирование требований - 1.3 Техническое задание (ТЗ) - 1.4 Архитектура приложения

2. UX/UI дизайн: - 2.1 User Research и персоны - 2.2 User Flow и Wireframes - 2.3 UI-дизайн (макеты экранов) - 2.4 Дизайн-система и UI Kit - 2.5 Прототип (интерактивный)

3. Разработка: - 3.1 Настройка инфраструктуры (CI/CD, серверы) - 3.2 Backend API - 3.3 Frontend (мобильное приложение) - 3.4 Интеграция с внешними сервисами - 3.5 Админ-панель

4. Тестирование: - 4.1 Юнит-тесты - 4.2 Интеграционное тестирование - 4.3 UAT (User Acceptance Testing) - 4.4 Нагрузочное тестирование - 4.5 Исправление багов

5. Запуск и маркетинг: - 5.1 Подготовка стор-листинга (App Store, Google Play) - 5.2 Бета-тестирование - 5.3 Маркетинговая кампания - 5.4 PR и медиа - 5.5 Запуск (Go-Live)

6. Управление проектом: - 6.1 Координация команды - 6.2 Отчётность стейкхолдерам - 6.3 Управление рисками - 6.4 Управление изменениями

Шаг 3: Детализируйте до рабочих пакетов

Каждый подкомпонент разбивается на конкретные рабочие пакеты (Work Packages). Например:

3.2 Backend API: - 3.2.1 Проектирование базы данных - 3.2.2 Разработка API аутентификации - 3.2.3 Разработка API каталога товаров - 3.2.4 Разработка API корзины и заказов - 3.2.5 Разработка API уведомлений - 3.2.6 Документация API (Swagger)

Каждый из этих рабочих пакетов занимает 8-80 часов и может быть назначен конкретному исполнителю.

Подходы к декомпозиции

По deliverables (результатам) – рекомендуемый

Структура строится вокруг результатов проекта: что будет создано? Это самый распространённый и рекомендуемый PMBOK подход.

По фазам

Структура повторяет жизненный цикл: Инициация -> Планирование -> Разработка -> Тестирование -> Запуск. Подходит для Waterfall-проектов.

По командам/отделам

Структура отражает организацию: Дизайн-команда, Backend-команда, Frontend-команда, QA-команда. Удобно для матричных организаций.

По географии/локациям

Для глобальных проектов: Москва, Ташкент, Дубай. Каждая локация – отдельная ветка WBS.

WBS Dictionary (Словарь WBS)

Каждый рабочий пакет описывается в WBS Dictionary:

Поле	Пример
ID пакета	3.2.2
Название	Разработка API аутентификации
Описание	Реализация регистрации, входа, JWT-токенов, восстановления пароля, OAuth2
Ответственный	Иванов А. (Backend Lead)
Оценка трудозатрат	40 часов
Зависимости	3.2.1 (Проектирование БД)
Критерии приёмки	Все эндпоинты работают, тесты пройдены, документация в Swagger
Риски	Сложность интеграции OAuth2 с внешними провайдерами

Инструменты для создания WBS

- **Microsoft Project** – классический инструмент, мощная WBS + Gantt
- **Miro / FigJam** – визуальные доски для brainstorming WBS в команде
- **MindMeister / XMind** – mind maps, отлично подходят для визуализации иерархии
- **WBS Schedule Pro** – специализированный инструмент для WBS
- **Jira** – компоненты (Components) и эпики (Epics) формируют WBS-подобную структуру
- **Notion** – иерархические базы данных, страницы и подстраницы
- **Excel / Google Sheets** – для простых проектов, нумерованный список

WBS в Jira

В Jira WBS реализуется через иерархию: - **Initiative** (Уровень 0) = Проект - **Epic** (Уровень 1) = Фаза / Deliverable - **Story** (Уровень 2) = Рабочий пакет - **Sub-task** (Уровень 3) = Конкретная задача

Типичные ошибки при создании WBS

1. **Забыть управление проектом** – координация, отчётность, управление рисками – это тоже работа, и она должна быть в WBS

2. **Слишком глубокая декомпозиция** – 7+ уровней обычно избыточны. 3-4 уровня достаточно для большинства проектов
3. **Включение действий вместо результатов** – WBS описывает ЧТО (дизайн главной страницы), а не КАК (открыть Figma, создать фрейм...)
4. **Нарушение правила 100%** – что-то забыли или добавили лишнее
5. **Создание WBS в одиночку** – WBS должна создаваться с участием команды, иначе будут пропуски

Что дальше

В уроке 1.6 мы используем WBS как основу для планирования сроков. Вы научитесь строить диаграмму Ганта, определять зависимости между задачами и находить критический путь проекта.

Урок 1.5 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.6: Планирование сроков – диаграмма Ганта и критический путь

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Вы построили WBS и знаете, какие работы нужно выполнить. Теперь критический вопрос: сколько времени займёт проект и в каком порядке выполнять задачи? Ошибка в планировании сроков – одна из самых дорогих в проектном управлении. По данным Standish Group, 52% проектов завершаются позже запланированного срока. Причина чаще всего не в лени команды, а в некачественном планировании.

В этом уроке мы разберём два фундаментальных инструмента: диаграмму Ганта (визуальное расписание проекта) и метод критического пути (CPM – Critical Path Method). Вместе они позволяют не только составить реалистичное расписание, но и понять, какие задачи определяют длительность всего проекта, а где есть запас времени.

Вы также узнаете о техниках оценки трудозатрат (экспертная оценка, Planning Poker, трёхточечная оценка), типах зависимостей между задачами и инструментах для создания расписания проекта.

Типы зависимостей между задачами

Прежде чем строить расписание, нужно определить зависимости – логические связи между задачами.

Четыре типа зависимостей (Precedence Diagramming Method)

FS (Finish-to-Start) – самый распространённый (90% случаев). Задача B начинается после завершения задачи A. - Пример: Тестирование начинается после завершения разработки

FF (Finish-to-Finish) – задача B завершается одновременно с задачей A. - Пример: Документация завершается вместе с разработкой

SS (Start-to-Start) – задача B начинается одновременно с задачей A. - Пример: Разработка фронтенда и бэкенда начинаются одновременно

SF (Start-to-Finish) – задача B завершается, когда начинается задача A (редкий тип). - Пример: Старая система выводится из эксплуатации, когда новая запускается

Типы зависимостей по природе

- **Обязательные (hard logic)** – физически невозможно иначе. Нельзя тестировать то, что не написано.

- **Дискреционные (soft logic)** – best practice, но можно изменить. Обычно дизайн делают до разработки, но можно параллельно.
- **Внешние** – зависимость от внешних факторов. Ожидание одобрения регулятора, поставка оборудования.

Диаграмма Ганта

Что это

Диаграмма Ганта – это горизонтальная столбчатая диаграмма, где каждая задача представлена полосой, расположенной на временной оси. Придумана Генри Гантом в 1910-х годах и с тех пор является самым популярным инструментом визуализации расписания проекта.

Элементы диаграммы Ганта

- **Ось Y** – список задач (из WBS)
- **Ось X** – временная шкала (дни, недели, месяцы)
- **Полосы** – длительность каждой задачи
- **Стрелки** – зависимости между задачами
- **Ромбы** – вехи (milestones) – ключевые контрольные точки без длительности
- **Процент завершения** – прогресс по каждой задаче
- **Критический путь** – задачи, выделенные красным

Пример: диаграмма Ганта для редизайна сайта

Задача	Начало	Конец	Длительность	Зависимость
1. Исследование и аналитика	1 марта	14 марта	2 недели	–
2. UX-проектирование	15 марта	28 марта	2 недели	1 (FS)
3. UI-дизайн	29 марта	18 апреля	3 недели	2 (FS)
4. Разработка фронтенда	19 апреля	16 мая	4 недели	3 (FS)
5. Разработка бэкенда	19 апреля	9 мая	3 недели	2 (FS)
6. Интеграция	17 мая	23 мая	1 неделя	4, 5 (FS)
7. Тестирование	24 мая	6 июня	2 недели	6 (FS)
ВЕХА: Запуск	7 июня	7 июня	0	7 (FS)

Обратите внимание: задачи 4 и 5 идут параллельно (обе зависят от задачи 2, а не друг от друга). Это сокращает общий срок проекта.

Метод критического пути (CPM)

Что такое критический путь

Критический путь – это самая длинная последовательность зависимых задач от начала до конца проекта. Он определяет минимально возможную длительность проекта. Задержка любой задачи на критическом пути автоматически сдвигает дату завершения проекта.

Как найти критический путь

Шаг 1: Forward Pass (Прямой проход) Двигаемся от начала к концу, вычисляя самые ранние даты начала (ES) и окончания (EF) каждой задачи. - ES = максимальный EF предшественников - EF = ES + Duration

Шаг 2: Backward Pass (Обратный проход) Двигаемся от конца к началу, вычисляя самые поздние даты начала (LS) и окончания (LF). - LF = минимальный LS последователей - LS = LF - Duration

Шаг 3: Вычисление Float (Резерв времени) - Total Float = LS - ES (или LF - EF) - Задачи с Float = 0 лежат на критическом пути

Пример расчёта

Используем пример выше:

Задача	Duration	ES	EF	LS	LF	Float	На крит. пути?
1. Аналитика	2 нед	0	2	0	2	0	Да
2. UX	2 нед	2	4	2	4	0	Да
3. UI	3 нед	4	7	4	7	0	Да
4. Frontend	4 нед	7	11	7	11	0	Да
5. Backend	3 нед	4	7	8	11	4	Нет
6. Интеграция	1 нед	11	12	11	12	0	Да
7. Тестирование	2 нед	12	14	12	14	0	Да

Критический путь: 1 -> 2 -> 3 -> 4 -> 6 -> 7 = 14 недель

Задача 5 (Backend) имеет Float = 4 недели – может задержаться на 4 недели без влияния на проект.

Техники оценки трудозатрат

Экспертная оценка

Самый простой метод: спросить эксперта. Опасность – когнитивные искажения (оптимизм, якорение). Рекомендация: привлекать 2-3 экспертов и усреднять.

Трёхточечная оценка (PERT)

Для каждой задачи определяются три оценки: - **O (Optimistic)** – лучший случай - **M (Most Likely)** – наиболее вероятный - **P (Pessimistic)** – худший случай

Формула PERT: E = (O + 4M + P) / 6

Стандартное отклонение: SD = (P - O) / 6

Пример: Разработка API - O = 2 недели, M = 3 недели, P = 7 недель - E = (2 + 12 + 7) / 6 = 3.5 недели - SD = (7 - 2) / 6 = 0.83 недели

Planning Poker (Agile)

Команда оценивает задачи с помощью карт Фибоначчи (1, 2, 3, 5, 8, 13, 21...). Каждый показывает свою карту одновременно, обсуждаются расхождения, повторяют до консенсуса. Это устраняет якорение и даёт более точные оценки.

Аналоговая оценка

Используя данные прошлых проектов: "В прошлый раз аналогичная задача заняла 3 недели, значит и сейчас заложим 3 недели". Работает, если есть качественные исторические данные.

Буферы и резервы

Закон Паркинсона

"Работа заполняет всё отведённое для неё время." Если дать задачу на 5 дней, она займёт 5 дней, даже если можно сделать за 3.

Типы буферов

- **Project Buffer** – буфер в конце проекта (обычно 10-20% от общей длительности)
- **Feeding Buffer** – буфер перед входом некритической задачи в критический путь
- **Resource Buffer** – предупреждение ресурсов о предстоящей работе на критическом пути

Метод критической цепи (CCPM, Голдратт)

Вместо буферов на каждую задачу – один общий буфер на весь проект. Оценки задач урезаются на 50%, а сэкономленное время формирует проектный буфер. Управление идёт по потреблению буфера.

Инструменты для планирования сроков

- **Microsoft Project** – золотой стандарт: Gantt, CPM, Resource Leveling, EVM
- **GanttPro** – облачный инструмент, простой и доступный
- **TeamGantt** – онлайн-диаграммы Ганта для команд
- **Jira + Advanced Roadmaps** – для Agile-проектов, таймлайн эпиков
- **Asana Timeline** – визуальный таймлайн проекта
- **Monday.com** – Gantt-представление задач
- **Notion** – Timeline view в базах данных

Что дальше

В уроке 1.7 мы разберём управление рисками – как идентифицировать потенциальные проблемы до того, как они произойдут, и подготовить планы реагирования.

Урок 1.6 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.7: Управление рисками – идентификация и митигация

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Если что-то может пойти не так – оно пойдёт не так. Закон Мёрфи в проектном управлении работает с пугающей точностью. Но опытный PM не боится рисков – он ими управляет. По данным PMI Pulse of the Profession, организации с зрелыми практиками управления рисками завершают на 14% больше проектов в срок и на 12% – в рамках бюджета.

Риск – это неопределённое событие, которое, если произойдёт, окажет положительное (opportunity) или отрицательное (threat) влияние на проект. Да, риски бывают и положительными – возможность завершить проект раньше, получить дополнительное финансирование, найти более эффективную технологию. Но чаще всего, когда говорят об управлении рисками, имеют в виду именно угрозы.

В этом уроке вы научитесь системно идентифицировать риски, оценивать их по вероятности и влиянию, приоритизировать через матрицу рисков и разрабатывать стратегии реагирования. Это один из самых практичных навыков PM – он спасает проекты, карьеры и бюджеты.

Процесс управления рисками по PMBOK

PMBOK описывает 7 процессов управления рисками:

1. **Plan Risk Management** – определить подход к управлению рисками
2. **Identify Risks** – выявить потенциальные риски
3. **Perform Qualitative Risk Analysis** – оценить вероятность и влияние
4. **Perform Quantitative Risk Analysis** – численный анализ (Monte Carlo и др.)
5. **Plan Risk Responses** – разработать стратегии реагирования
6. **Implement Risk Responses** – реализовать запланированные действия
7. **Monitor Risks** – отслеживать риски и выявлять новые

В этом уроке мы сфокусируемся на процессах 2, 3 и 5 – они дают максимальную практическую пользу.

Идентификация рисков

Техники выявления рисков

Мозговой штурм (Brainstorming) Соберите команду и задайте вопрос: "Что может пойти не так?" Правила: никакой критики, максимум идей, фиксируйте всё. После – группируйте и приоритизируйте.

SWOT-анализ Strengths (Сильные стороны), Weaknesses (Слабые стороны), Opportunities (Возможности), Threats (Угрозы). Помогает увидеть риски в контексте проекта и организации.

Анализ допущений (Assumptions Analysis) Запишите все допущения проекта ("предполагаем, что команда будет в полном составе", "предполагаем, что API партнёра стабилен") и проверьте: что если допущение окажется ложным?

Checklist из прошлых проектов Используйте Lessons Learned из завершённых проектов. Если в прошлом проекте были проблемы с интеграцией – этот риск стоит учесть.

Диаграмма Исикавы (Fishbone / Причинно-следственная) Категории причин рисков: People (Люди), Process (Процессы), Technology (Технология), External (Внешние факторы). Для каждой категории – конкретные риски.

Pre-mortem анализ (Гэри Клейн) Представьте, что проект провалился. Каждый участник пишет причины провала. Мощная техника, потому что снимает психологический барьер "у нас всё будет хорошо".

Категории рисков

Категория	Примеры рисков
Технические	Новая технология не работает как ожидалось, баги в критических компонентах
Ресурсные	Увольнение ключевого разработчика, нехватка бюджета
Внешние	Изменение законодательства, действия конкурентов, пандемия
Организационные	Смена приоритетов руководства, реорганизация
Управленческие	Scope creep, нечёткие требования, слабая коммуникация
Коммерческие	Изменение рыночных условий, курса валют

Оценка рисков: матрица вероятности и влияния

Качественная оценка

Каждый риск оценивается по двум параметрам:

Вероятность (Probability): - Очень низкая (1) – < 10% - Низкая (2) – 10-30% - Средняя (3) – 30-50% - Высокая (4) – 50-70% - Очень высокая (5) – > 70%

Влияние (Impact): - Незначительное (1) – < 5% бюджета/сроков - Низкое (2) – 5-10% - Среднее (3) – 10-20% - Высокое (4) – 20-40% - Критическое (5) – > 40% или провал проекта

Матрица рисков (Risk Matrix)

Risk Score = Probability x Impact

	Влияние 1	Влияние 2	Влияние 3	Влияние 4	Влияние 5
Вероятность 5	5	10	15	20	25
Вероятность 4	4	8	12	16	20
Вероятность 3	3	6	9	12	15
Вероятность 2	2	4	6	8	10
Вероятность 1	1	2	3	4	5

Зоны: - 1-4: Низкий риск (зелёная зона) – принять или мониторить - 5-12: Средний риск (жёлтая зона) – разработать план реагирования - 15-25: Высокий риск (красная зона) – немедленные действия, эскалация

Стратегии реагирования на риски

Для угроз (негативных рисков)

Избежание (Avoid) Изменить план проекта, чтобы исключить риск. Например: не использовать непроверенную технологию, заменить на проверенную.

Снижение (Mitigate) Уменьшить вероятность или влияние. Например: провести прототипирование рискованного компонента на ранней стадии.

Передача (Transfer) Переложить последствия на третью сторону. Например: страхование, outsourcing рискованной части, fixed-price контракт с подрядчиком.

Принятие (Accept) Осознанно принять риск, не предпринимая проактивных действий. Бывает активное (создать contingency reserve) и пассивное (просто мониторить).

Для возможностей (положительных рисков)

Использование (Exploit) – гарантировать реализацию возможности **Усиление (Enhance)** – увеличить вероятность **Разделение (Share)** – привлечь партнёра для совместного использования **Принятие (Accept)** – если возможность реализуется – отлично, нет – ничего страшного

Реестр рисков: шаблон

ID	Риск	Категория	Вероятность	Влияние	Score	Стратегия	Действия	Владелец	Статус
R01	Увольнение ведущего разработчика	Ресурсы	3	5	15	Снижение	Документирование кода, парное программирование, подготовка замены	PM	Активен
R02	API партнёра нестабилен	Технический	4	3	12	Снижение	Создать mock-сервер, написать интеграционные тесты, иметь fallback	Tech Lead	Активен
R03	Scope creep от заказчика	Управленческий	4	4	16	Избежание	Зафиксировать scope в контракте, формальный Change Request процесс	PM	Активен

ID	Риск	Категория	Вероятность	Влияние	Score	Стратегия	Действия	Владелец	Статус
R04	Задержка оборудования	Внешний	2	3	6	Принятие	Мониторить статус поставки, иметь план B с облачным решением	Ops	Монитори
R05	Курс валюты вырастет на 20%+	Коммерческий	2	4	8	Передача	Зафиксировать курс в контракте, валютная оговорка	Финансы	Монитори

Contingency и Management Reserves

Contingency Reserve (Резерв на непредвиденные обстоятельства)

Закладывается на известные риски (identified risks). Рассчитывается на основе EMV (Expected Monetary Value):

EMV = Probability x Impact (в деньгах)

Пример: Риск задержки – вероятность 30%, стоимость последствий $10,000. EMV = 0.3 x $10,000 = $3,000

Сумма EMV всех рисков = Contingency Reserve.

Management Reserve

Закладывается на неизвестные риски (unknown unknowns) – те, которые невозможно предвидеть. Обычно 5-15% от бюджета проекта. Управляется спонсором, не PM.

Мониторинг рисков

Risk Review Meeting

Регулярная встреча (еженедельно или каждый спринт): - Обзор активных рисков – изменились ли вероятность/влияние? - Новые риски – появились ли? - Закрытые риски – какие риски больше неактуальны? - Триггеры – появились ли ранние признаки наступления риска?

Risk Burndown Chart

График, показывающий суммарный Risk Score проекта во времени. В идеале – снижается по мере продвижения проекта.

Что дальше

В уроке 1.8 мы разберём бюджетирование проекта – как рассчитать стоимость, создать бюджет и контролировать расходы с помощью EVM-метрик.

Урок 1.7 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.8: Бюджетирование проекта

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Деньги – кровеносная система любого проекта. Без грамотного бюджетирования даже блестяще спланированный проект может провалиться: закончились средства на полпути, расходы вышли из-под контроля, спонсор недоволен, команда демотивирована. По статистике PMI, 43% проектов превышают первоначальный бюджет, а средний перерасход составляет 27%.

Бюджетирование проекта – это не просто "посчитать, сколько стоит". Это комплексный процесс, включающий оценку стоимости (cost estimating), определение бюджета (cost budgeting) и контроль расходов (cost control). В PMBOK этому посвящена целая область знаний – Project Cost Management.

В этом уроке вы научитесь оценивать стоимость проекта различными методами, формировать бюджет с учётом резервов, строить S-curve для визуализации расходов и контролировать бюджет с помощью Earned Value Management (EVM). Это навыки, которые отличают junior PM от senior.

Типы затрат проекта

По отношению к проекту

Прямые затраты (Direct Costs): - Зарплата команды проекта - Оборудование и материалы - Лицензии ПО - Подрядчики и фрилансеры - Командировки

Косвенные затраты (Indirect Costs / Overhead): - Аренда офиса (доля проекта) - Административные расходы - HR-затраты - Амортизация оборудования - Корпоративные лицензии

По поведению

Постоянные (Fixed): не зависят от объёма работ – аренда, лицензии **Переменные (Variable):** растут с объёмом – зарплата за овертайм, облачные ресурсы

Sunk Costs (Невозвратные затраты)

Уже потраченные деньги, которые нельзя вернуть. Важное правило: sunk costs не должны влиять на решение о продолжении проекта. Если проект убыточен – его нужно закрыть, даже если уже потрачено $100K. Это психологически сложно, но экономически верно.

Методы оценки стоимости

Аналоговая оценка (Analogous Estimating)

Основана на данных прошлых проектов. "Прошлый сайт стоил $50K, этот похож – значит тоже около $50K."

- Точность: +/- 35-50%
- Когда использовать: ранние стадии, мало данных
- Плюс: быстро, дёшево
- Минус: низкая точность

Параметрическая оценка (Parametric Estimating)

Использует статистические зависимости. "Стоимость 1 экрана приложения = $2,000. У нас 25 экранов. Итого: $50,000."

- Точность: +/- 15-25%
- Когда использовать: есть надёжные метрики из прошлых проектов
- Плюс: объективная, масштабируемая
- Минус: нужны качественные исторические данные

Оценка "снизу вверх" (Bottom-Up Estimating)

Оценка каждого рабочего пакета из WBS, затем суммирование. Самый точный метод.

- Точность: +/- 5-15%
- Когда использовать: детальное планирование, есть полная WBS
- Плюс: высокая точность
- Минус: трудоёмко, требует время

Трёхточечная оценка (Three-Point Estimating)

Аналогично PERT для сроков:

E = (O + 4M + P) / 6

Где O = оптимистичная, M = наиболее вероятная, P = пессимистичная оценка.

Пример: Разработка модуля оплаты - O = $8,000 - M = $12,000 - P = $25,000 - E = (8,000 + 48,000 + 25,000) / 6 = $13,500

Формирование бюджета проекта

Структура бюджета

Cost Baseline (Базовый план стоимости) – утверждённый бюджет, по которому измеряется прогресс:

Сумма оценок рабочих пакетов (WBS) + Contingency Reserve (резерв на известные риски, обычно 10-15%) = Cost Baseline + Management Reserve (резерв на неизвестные риски, обычно 5-10%) = Project Budget (Бюджет проекта)

Пример бюджета: проект разработки e-commerce

Категория	Статья	Сумма
Команда	PM (4 мес x $4,000)	$16,000
Команда	Backend-разработчик (4 мес x $5,000)	$20,000
Команда	Frontend-разработчик (3 мес x $4,500)	$13,500
Команда	UX/UI дизайнер (2 мес x $4,000)	$8,000
Команда	QA-инженер (2 мес x $3,500)	$7,000
Лицензии	Figma, Jira, GitHub, AWS	$3,500
Подрядчики	Копирайтер, фотограф	$4,000
Маркетинг	Запуск (контекст, SMM)	$5,000
Subtotal		**$77,000**
Contingency (12%)		$9,240
Cost Baseline		**$86,240**
Management Reserve (8%)		$6,899
Project Budget		**$93,139**

S-Curve: визуализация расходов

Что такое S-Curve

S-Curve (S-образная кривая) показывает кумулятивные расходы проекта во времени. Называется так, потому что имеет характерную S-образную форму:

- **Начало проекта** – расходы невелики (планирование, аналитика)
- **Середина** – расходы нарастают стремительно (активная разработка)
- **Конец** – расходы замедляются (тестирование, закрытие)

Использование S-Curve

На одном графике отображаются три линии: - **PV (Planned Value)** – запланированные расходы (baseline) - **EV (Earned Value)** – стоимость выполненных работ - **AC (Actual Cost)** – фактические расходы

Это основа Earned Value Management.

Earned Value Management (EVM)

Ключевые метрики EVM

Базовые показатели: - **BAC (Budget at Completion)** – общий бюджет проекта - **PV (Planned Value)** – сколько работ должно быть выполнено к дате X по плану - **EV (Earned Value)** – плановая стоимость фактически выполненных работ - **AC (Actual Cost)** – сколько реально потрачено

Индексы эффективности: - **CPI (Cost Performance Index)** = EV / AC - CPI > 1: экономия бюджета - CPI = 1: в рамках бюджета - CPI < 1: перерасход

- **SPI (Schedule Performance Index)** = EV / PV
  - SPI > 1: опережение графика
  - SPI = 1: по графику
  - SPI < 1: отставание

Отклонения: - **CV (Cost Variance)** = EV - AC (отклонение по стоимости) - **SV (Schedule Variance)** = EV - PV (отклонение по срокам)

Прогнозы: - **EAC (Estimate at Completion)** = BAC / CPI (прогноз итоговой стоимости) - **ETC (Estimate to Complete)** = EAC - AC (сколько ещё нужно потратить) - **TCPI (To-Complete Performance Index)** = (BAC - EV) / (BAC - AC) (требуемая эффективность)

Пример расчёта EVM

Проект с BAC = $100,000. На 50% по срокам:

- PV = $50,000 (по плану должны были выполнить работ на $50K)
- EV = $40,000 (фактически выполнено работ на $40K по плану)
- AC = $48,000 (реально потрачено $48K)

Расчёт: - CPI = 40,000 / 48,000 = 0.83 – перерасход 17%! - SPI = 40,000 / 50,000 = 0.80 – отставание 20%! - CV = 40,000 - 48,000 = -$8,000 – перерасход $8K - SV = 40,000 - 50,000 = -$10,000 – отстаём на $10K работ - EAC = 100,000 / 0.83 = $120,482 – прогноз: проект будет стоить $120K вместо $100K

Вывод: проект в красной зоне – нужны корректирующие действия.

Инструменты бюджетирования

- **Microsoft Project** – встроенное управление ресурсами и бюджетом, EVM
- **Google Sheets / Excel** – для большинства проектов достаточно
- **Smartsheet** – шаблоны бюджетов и автоматические расчёты
- **Monday.com** – Budget Tracker, визуализация затрат
- **Harvest / Toggl** – трекинг времени команды, автоматический расчёт стоимости
- **QuickBooks / Xero** – для связки с бухгалтерией
- **Jira + Tempo** – плагин для учёта времени и бюджета в Jira

---

Типичные ошибки бюджетирования

1. **Не закладывать резервы** – contingency reserve 10-15% обязателен
2. **Забывать о косвенных затратах** – аренда, лицензии, администрирование
3. **Оптимистичные оценки** – команда всегда недооценивает трудозатраты
4. **Не отслеживать расходы** – узнать о перерасходе в конце проекта = поздно
5. **Sunk cost fallacy** – продолжать вкладывать в провальный проект “потому что уже потрачено”
6. **Не пересчитывать EAC** – бюджет нужно обновлять еженедельно/помесячно

Что дальше

В уроке 1.9 мы разберём управление стейкхолдерами – как определить, кто влияет на ваш проект, и управлять их ожиданиями. Ведь именно стейкхолдеры принимают решения по бюджету.

Урок 1.8 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.9: Стейкхолдеры – управление ожиданиями

Блок 1: Основы управления проектами Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Вы можете идеально спланировать проект, уложиться в бюджет и сроки, но если ключевые стейкхолдеры недовольны – проект будет считаться провальным. По данным PMI, неэффективное управление стейкхолдерами является причиной провала 30% проектов. Это не технический навык – это навык работы с людьми, и он, пожалуй, самый сложный в арсенале проектного менеджера.

Стейкхолдер (stakeholder, заинтересованная сторона) – это любое лицо, группа или организация, которые могут влиять на проект или подвергаются влиянию проекта. Это не только заказчик и команда – это руководство, пользователи, регуляторы, смежные отделы, подрядчики, и даже конкуренты.

В этом уроке вы научитесь идентифицировать стейкхолдеров, анализировать их интересы и влияние, строить стратегии вовлечения и управлять коммуникациями так, чтобы все ключевые лица были информированы, вовлечены и удовлетворены. Мы разберём матрицу Power/Interest, план коммуникаций и реальные стратегии работы с “трудными” стейкхолдерами.

Идентификация стейкхолдеров

Кто является стейкхолдером?

Задайте себе вопросы: - Кто инициировал проект и финансирует его? - Кто будет использовать результат проекта? - Кто будет поддерживать и развивать результат после проекта? - Кто предоставляет ресурсы (людей, оборудование, бюджет)? - Чья работа изменится из-за проекта? - Кто может заблокировать или замедлить проект? - Кто регулирует деятельность (законы, стандарты)? - Кого затронут последствия проекта?

Категории стейкхолдеров

Внутренние: - Спонсор проекта - Руководство компании (CEO, CTO, CFO) - Функциональные менеджеры - Команда проекта - Смежные отделы (маркетинг, продажи, HR, юристы) - PMO (Project Management Office)

Внешние: - Заказчик / Клиент - Конечные пользователи - Подрядчики и поставщики - Регуляторы (государственные органы) - Партнёры - СМИ и общественность - Конкуренты

Stakeholder Register (Реестр стейкхолдеров)

ID	Стейкхолдер	Роль	Организация	Интересы	Ожидания	Влияние	Отношение
S01	Иванов А.Н.	Спонсор	Совет директоров	ROI, сроки	Запуск к 1 сентября	Высокое	Поддерживающий
S02	Петрова М.Б.	CTO	IT-департамент	Архитектура, безопасность	Масштабируемое решение	Высокое	Нейтральный
S03	Сидоров К.Р.	Начальник отдела продаж	Продажи	Удобство CRM	Без прерывания работы	Среднее	Скептический
S04	Конечные пользователи	Клиенты	–	Удобство, скорость	Интуитивный интерфейс	Низкое	Неизвестно
S05	Регулятор	ЦБ	Госорган	Соответствие ФЗ-152	Полное соответствие	Высокое	Нейтральный

Анализ стейкхолдеров: матрица Power/Interest

Матрица Менделоу (Power/Interest Grid)

Два измерения: - **Power (Власть/Влияние)** – насколько стейкхолдер может влиять на проект - **Interest (Интерес)** – насколько стейкхолдер заинтересован в проекте

	Низкий интерес	Высокий интерес
Высокая власть	Keep Satisfied (Удовлетворять)	Manage Closely (Управлять активно)
Низкая власть	Monitor (Мониторить)	Keep Informed (Информировать)

Стратегии по квадрантам

Manage Closely (Высокая власть + Высокий интерес): - Регулярные личные встречи - Вовлечение в принятие ключевых решений - Проактивная коммуникация - Примеры: спонсор, заказчик, CEO

Keep Satisfied (Высокая власть + Низкий интерес): - Периодические отчёты о статусе - Не перегружать деталями - Быстро реагировать на запросы - Примеры: CFO, совет директоров, регулятор

Keep Informed (Низкая власть + Высокий интерес): - Регулярные рассылки, дашборды - Вовлечение в обсуждения - Учёт обратной связи - Примеры: конечные пользователи, команда поддержки

Monitor (Низкая власть + Низкий интерес): - Минимальная коммуникация - Мониторинг изменения статуса - Примеры: широкая общественность, косвенно затронутые отделы

Дополнительные модели анализа

Salience Model (Mitchell, Agle & Wood): Три атрибута: Power (Власть), Legitimacy (Легитимность), Urgency (Срочность). Стейкхолдеры классифицируются по комбинации атрибутов (от latent до definitive).

Influence/Impact Matrix: Вместо власти и интереса используются влияние на проект и воздействие проекта на стейкхолдера.

Стратегии вовлечения стейкхолдеров

Уровни вовлечённости (PMBOK)

1. **Unaware (Неосведомлённый)** – не знает о проекте
2. **Resistant (Сопротивляющийся)** – знает, но против

3. **Neutral (Нейтральный)** – знает, но не поддерживает и не сопротивляется
4. **Supportive (Поддерживающий)** – поддерживает проект
5. **Leading (Лидирующий)** – активно продвигает проект

Stakeholder Engagement Assessment Matrix

Стейкхолдер	Текущий уровень	Желаемый уровень	Разрыв	Действия
Спонсор	Supportive	Leading	1	Вовлечь в key decisions, дать публичную роль
CTO	Neutral	Supportive	1	Показать технические преимущества, вовлечь в выбор стека
Отдел продаж	Resistant	Neutral/Supportive	2	Провести demo, показать выгоды, обучить
Пользователи	Unaware	Supportive	3	Фокус-группы, бета-тестирование, коммуникация о выгодах

Работа с сопротивлением

Почему стейкхолдеры сопротивляются: - Страх потерять контроль или влияние - Неприятие изменений (change fatigue) - Негативный опыт прошлых проектов - Конфликт интересов - Недостаток информации

Стратегии преодоления: 1. **Слушайте** – поймите реальные причины сопротивления 2. **Вовлекайте** – дайте возможность влиять на решения 3. **Показывайте выгоды** – что стейкхолдер получит лично (WIIFM: What's In It For Me) 4. **Предоставляйте доказательства** – демо, пилот, данные 5. **Найдите союзника** – попросите лояльного стейкхолдера того же уровня поддержать 6. **Эскалируйте** – если сопротивление блокирует проект, эскалируйте спонсору

Коммуникационный план

Формула эффективной коммуникации

Кто –> Кому –> Что –> Когда –> Как –> Обратная связь

Шаблон коммуникационного плана

Стейкхолдер	Тип коммуникации	Частота	Канал	Ответственный	Содержание
Спонсор	Статус-отчёт	Еженедельно	Личная встреча 30 мин	PM	Прогресс, риски, решения
Steering Committee	Ревью	Ежемесячно	Презентация 1 час	PM + Team Lead	KPI, бюджет, прогноз
Команда	Daily Standup	Ежедневно	Slack/Teams	Scrum Master	Прогресс, блокеры
Заказчик	Sprint Review	Каждые 2 нед	Zoom + демо	PM	Результаты спринта
Пользователи	Newsletter	Ежемесячно	Email	Маркетинг	Прогресс, анонсы
Руководство	Дашборд	Realtime	Jira/Notion	Автоматически	Метрики проекта

Правила коммуникации с стейкхолдерами

1. **No Surprises Rule** – стейкхолдер не должен узнавать о проблемах от третьих лиц
2. **Escalation Path** – чёткий путь эскалации: Team Lead -> PM -> Спонсор -> Steering Committee
3. **Right Level of Detail** – CEO не нужны детали багов, а разработчику – финансовые отчёты

4. **Active Listening** – 80% слушаем, 20% говорим
5. **Document Everything** – решения, договорённости, изменения – письменно

Конфликты между стейкхолдерами

Типичные конфликты

- Заказчик хочет больше фич, а спонсор – уложиться в бюджет
- Отдел продаж хочет запуск быстрее, а QA – больше тестирования
- CTO хочет новый стек, а команда предпочитает проверенный

Методы разрешения (от предпочтительного к крайнему)

1. **Collaborate (Сотрудничество)** – win-win решение, удовлетворяющее обе стороны
2. **Compromise (Компромисс)** – обе стороны уступают частично
3. **Smooth (Сглаживание)** – акцент на общих интересах, избегание конфронтации
4. **Force (Принуждение)** – решение принимает тот, кто имеет власть
5. **Withdraw (Уклонение)** – отложить решение (иногда оправдано, но часто опасно)

Что дальше

В финальном уроке блока (1.10) вы соберёте все знания вместе и создадите полноценный план проекта от А до Я: устав, WBS, расписание, бюджет, риски и стейкхолдеры.

Урок 1.9 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Урок 1.10: Проект — план проекта от А до Я

Блок 1: Основы управления проектами Длительность видео: 10-12 минут **Тип:** Проектный урок

Введение

Это финальный урок блока, и он полностью практический. Мы объединим все знания из 9 уроков и создадим полный план проекта — от определения целей до управления рисками и стейкхолдерами.

К концу урока у вас будет: 1. Документ Project Charter (устав проекта) 2. Work Breakdown Structure (WBS) 3. Диаграмма Ганта с критическим путём 4. Реестр рисков 5. Карта стейкхолдеров

Всё это — на одном реальном примере, который вы сможете адаптировать под любой свой проект.

Кейс для практики

Проект: Запуск интернет-магазина

Параметр	Значение
Цель	Запустить работающий интернет-магазин за 8 недель
Бюджет	$5,000

Параметр	Значение
Команда	PM (вы), дизайнер, разработчик, маркетолог
Заказчик	CEO компании
Результат	Работающий магазин с 50+ товарами и настроенной рекламой

Шаг 1: Project Charter (Устав проекта)

Project Charter — это документ, который формально авторизует проект и даёт PM полномочия.

Шаблон

```
# PROJECT CHARTER

## Название проекта
Запуск интернет-магазина [Бренд]

## Цель проекта (SMART)
Запустить работающий интернет-магазин на Shopify с каталогом
из 50+ товаров и настроенной рекламой к [дата]
в рамках бюджета $5,000.

## Обоснование
Компания теряет $15K/мес потенциальной выручки без
онлайн-канала продаж. Конкуренты уже онлайн.

## Scope (объём)
Включено:
- Shopify магазин с кастомной темой
- 50+ товаров с фото и описаниями
- Интеграция с платёжными системами
- SEO-оптимизация
- Запуск рекламы (Facebook + Google)

Не включено:
- Мобильное приложение
- Кастомная разработка
- Складская логистика

## Бюджет
- Shopify: $39/мес
- Тема: $380
- Фото: $500
- Реклама: $2,000
- Дизайн: $1,000
- Резерв (10%): $500
- ИТОГО: $4,919

## Ключевые milestone
1. Неделя 2: Тема настроена, дизайн утверждён
2. Неделя 4: Все товары загружены
3. Неделя 6: Оплата, доставка, юридические страницы
4. Неделя 7: Тестирование
5. Неделя 8: Запуск + реклама

## Команда
| Роль | Имя | Ответственность |
|---|---|---|
| Project Manager | [Вы] | Координация, сроки, бюджет |
```

```
| Дизайнер | [Имя] | Визуал, фото, тема |
| Разработчик | [Имя] | Настройка, интеграции |
| Маркетолог | [Имя] | SEO, реклама, контент |

## Спонсор проекта
CEO — утверждает бюджет и ключевые решения

## Риски верхнего уровня
- Задержка контента (фото, описания)
- Проблемы с платёжным шлюзом
- Превышение рекламного бюджета
```

Шаг 2: Work Breakdown Structure

Разбейте проект на фазы → задачи → подзадачи:

```
1. ПОДГОТОВКА (Неделя 1–2)
   1.1 Анализ конкурентов
   1.2 Определение ЦА
   1.3 Выбор и покупка домена
   1.4 Регистрация Shopify
   1.5 Выбор и установка темы
   1.6 Брендинг (логотип, цветовая схема)

2. КОНТЕНТ (Неделя 2–4)
   2.1 Фотосессия товаров
   2.2 Написание описаний
   2.3 Создание коллекций
   2.4 Загрузка товаров
   2.5 Написание страниц About, FAQ

3. НАСТРОЙКА (Неделя 4–6)
   3.1 Подключение оплаты
   3.2 Настройка доставки
   3.3 Юридические страницы
   3.4 Email-уведомления
   3.5 SEO-оптимизация

4. ТЕСТИРОВАНИЕ (Неделя 6–7)
   4.1 Тестовые заказы
   4.2 Проверка на мобильных
   4.3 Проверка скорости
   4.4 UAT (приёмочное тестирование)

5. ЗАПУСК (Неделя 7–8)
   5.1 Снятие пароля
   5.2 Подключение Analytics
   5.3 Запуск рекламы
   5.4 Мониторинг первых заказов
   5.5 Сбор обратной связи
```

Шаг 3: Диаграмма Ганта

Создание в бесплатных инструментах

Инструмент	Цена	Диаграмма Ганта
Google Sheets	Бесплатно	Ручная (условное форматирование)

Инструмент	Цена	Диаграмма Ганта
Notion	Бесплатно	Timeline view
ClickUp	Free plan	Встроенная
TeamGantt	Free до 1 проекта	Специализированная
Monday.com	Free до 2 пользователей	Встроенная

Критический путь

Из нашего WBS критический путь:

Анализ конкурентов → Выбор темы → Фотосессия → Загрузка товаров → Настройка оплаты → Тестирование → Запуск

Любая задержка на критическом пути сдвигает весь проект. Эти задачи требуют максимального внимания.

Зависимости

- Фотосессия (2.1) не может начаться до определения ЦА (1.2)
- Загрузка товаров (2.4) зависит от фотосессии (2.1) и описаний (2.2)
- Тестирование (4.x) не начнётся до завершения настройки (3.x)
- Реклама (5.3) запускается только после снятия пароля (5.1)

Шаг 4: Реестр рисков

#	Риск	Вероятность	Влияние	Стратегия	Действие
R1	Задержка фото	Высокая	Высокое	Митигация	Начать фото в неделю 1, иметь резерв 3 дня
R2	Проблемы с оплатой	Средняя	Высокое	План B	Подготовить PayPal как backup
R3	Превышение бюджета	Средняя	Среднее	Митигация	10% резерв + еженедельный контроль
R4	Недоступность дизайнера	Низкая	Высокое	Митигация	Согласовать расписание заранее
R5	Низкая конверсия после запуска	Высокая	Среднее	Принятие	Запланировать A/B тесты в первый месяц

Шаг 5: Карта стейкхолдеров

Матрица Власть/Интерес

Стейкхолдер	Власть	Интерес	Стратегия
CEO (спонсор)	Высокая	Высокий	Manage closely — еженедельные отчёты
Команда разработки	Низкая	Высокий	Keep informed — daily standup
Клиенты (будущие)	Низкая	Низкий	Monitor — собирать фидбек после запуска
Бухгалтерия	Средняя	Низкий	Keep satisfied — отчёт по бюджету

Коммуникационный план

Аудитория	Формат	Частота	Канал
CEO	Статус-отчёт	Еженедельно (Пт)	Email + встреча
Команда	Standup	Ежедневно	Slack/встреча
CEO	Milestone review	По факту	Встреча
Все	Ретроспектива	После запуска	Встреча

Итоги блока

За 10 уроков вы освоили фундамент управления проектами:

1. **Принципы и роли** — что такое PM и зачем он нужен
2. **Жизненный цикл** — от идеи до закрытия
3. **Waterfall vs Agile** — когда что использовать
4. **Цели** — SMART, OKR, KPI
5. **WBS** — декомпозиция задач
6. **Сроки** — диаграмма Ганта и критический путь
7. **Риски** — идентификация и митигация
8. **Бюджет** — планирование и контроль
9. **Стейкхолдеры** — управление ожиданиями
10. **Практика** — полный план проекта

В следующем блоке — Agile, Scrum и Kanban: современные методологии для быстрых итераций.

Урок 1.10 из 10 | Блок 1: Основы управления проектами | Курс: Управление проектами

Блок 2: Agile, Scrum и Kanban

Урок 2.1: Agile Manifesto – ценности и принципы

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

В 2001 году семнадцать разработчиков собрались на горнолыжном курорте Snowbird в штате Юта и за три дня сформулировали документ, который изменил индустрию разработки ПО – а затем и управление проектами в целом. Этот документ – Agile Manifesto (Манифест гибкой разработки). Он состоит всего из 4 ценностей и 12 принципов, но за ними стоит фундаментальный сдвиг мышления: от планирования к адаптации, от документации к работающему продукту, от контрактов к сотрудничеству.

Почему это важно сейчас? По данным отчёта Standish Group (CHAOS Report 2020), Agile-проекты имеют втрое более высокий показатель успешности по сравнению с Waterfall-проектами. Компании от Spotify до Сбербанка используют Agile-подходы. Но при этом больше половины организаций, заявляющих об "Agile-трансформации", на практике просто переименовали совещания и добавили стикеры на доску.

В этом уроке мы разберём Agile Manifesto на атомарном уровне: каждую ценность, каждый принцип, их практическое применение, типичные ошибки и антипаттерны. После этого урока вы будете понимать не только ЧТО такое Agile, но и ПОЧЕМУ он работает – и когда он НЕ работает.

Историческая справка: до Agile

До 2001 года доминировал Waterfall (каскадная модель). Проект шёл линейно: Требования -> Проектирование -> Разработка -> Тестирование -> Внедрение. Каждая фаза завершалась полностью, прежде чем начиналась следующая.

Проблемы Waterfall: - Заказчик видел результат только в конце (через 6-18 месяцев) - Изменение требований стоило катастрофически дорого - Команда работала в изоляции от пользователей - 60-70% функций в готовом продукте не использовались

Попытки решить эти проблемы привели к появлению легковесных методологий: Extreme Programming (XP), Crystal, DSDM, Feature-Driven Development. Agile Manifesto объединил их общие принципы.

4 ценности Agile Manifesto

Манифест формулирует 4 пары ценностей в формате: "Мы ценим **X** больше, чем Y". Важно: Y не отвергается полностью, но X имеет приоритет.

Приоритет	Меньший приоритет	Практический смысл
Люди и взаимодействие	Процессы и инструменты	Лучшая доска задач не спасёт токсичную команду
Работающий продукт	Исчерпывающая документация	Рабочий прототип ценнее 200 страниц ТЗ
Сотрудничество с заказчиком	Согласование условий контракта	Регулярная обратная связь вместо формальной приёмки
Готовность к изменениям	Следование первоначальному плану	План – это гипотеза, а не закон

Ценность 1: Люди и взаимодействие > Процессы и инструменты

Даже идеальный процесс бесполезен, если люди не общаются. PM, который прячется за Jira-тикетами вместо живого разговора, нарушает эту ценность. Практика: ежедневные стендапы, парное программирование, ретроспективы – всё это про взаимодействие.

Ценность 2: Работающий продукт > Исчерпывающая документация

Документация нужна, но она не должна заменять работающий софт. Если команда 3 месяца писала спецификацию и ещё не написала ни строчки кода – что-то пошло не так. Практика: каждые 1-4 недели – инкремент работающего продукта.

Ценность 3: Сотрудничество с заказчиком > Согласование условий контракта

Вместо "подписал ТЗ – не трогай" – постоянный диалог. Заказчик участвует в планировании спринтов, видит демо, даёт обратную связь. Практика: Product Owner как голос заказчика в команде.

Ценность 4: Готовность к изменениям > Следование плану

Требования БУДУТ меняться. Рынок изменится, конкурент выпустит фичу, пользователи попросят другое. Agile-команда воспринимает изменения как конкурентное преимущество, а не как угрозу. Практика: гибкий бэклог, перепланирование каждый спринт.

12 принципов Agile

За 4 ценностями стоят 12 принципов, которые конкретизируют подход:

1. **Высший приоритет – удовлетворение заказчика через раннюю и непрерывную поставку ценного ПО.** Не ждите 6 месяцев. Показывайте результат каждые 2 недели.
2. **Приветствуйте изменение требований, даже на поздних стадиях.** Agile-процессы используют изменения для конкурентного преимущества заказчика.

3. **Поставляйте работающее ПО часто** – от пары недель до пары месяцев, с предпочтением более коротких сроков.
4. **Бизнес и разработка должны работать вместе ежедневно** на протяжении всего проекта.
5. **Стройте проекты вокруг мотивированных людей.** Дайте им среду и поддержку, доверяйте им выполнение работы.
6. **Самый эффективный способ обмена информацией – личный разговор (face-to-face).**
7. **Работающее ПО – главный показатель прогресса.**
8. **Agile-процессы способствуют устойчивой разработке.** Спонсоры, разработчики и пользователи должны поддерживать постоянный темп бесконечно.
9. **Постоянное внимание к техническому совершенству и качественному дизайну** повышает гибкость.
10. **Простота – искусство максимизации объёма невыполненной работы** – является необходимой.
11. **Лучшие архитектуры, требования и дизайн** появляются из самоорганизующихся команд.
12. **Команда регулярно размышляет о том, как стать эффективнее**, и соответственно корректирует своё поведение.

Agile – это не методология, а мышление

Распространённое заблуждение

Agile часто путают с конкретной методологией (Scrum, Kanban). Но Agile – это **набор ценностей и принципов**, а Scrum и Kanban – **фреймворки**, которые реализуют эти ценности на практике.

Уровень	Что это	Примеры
Философия	Agile Manifesto	4 ценности + 12 принципов
Фреймворк	Конкретная система правил	Scrum, Kanban, XP
Практики	Конкретные техники	Daily Standup, Sprint Review, User Stories
Инструменты	ПО для реализации	Jira, Trello, Miro

Когда Agile работает лучше всего

Модель Cynefin (Дэйв Сноуден) помогает определить, когда Agile подходит:

- **Простые проблемы** (очевидная причинно-следственная связь) – процедуры, чек-листы. Waterfall подходит.
- **Сложные проблемы** (cause-effect через анализ) – экспертное планирование. Waterfall или гибридный подход.
- **Комплексные проблемы** (непредсказуемость, эмерджентность) – **Agile идеален**. Эксперименты, итерации, быстрая обратная связь.
- **Хаотические проблемы** (кризис) – сначала действуй, потом анализируй.

Большинство IT-проектов находятся в зоне комплексных проблем, поэтому Agile стал стандартом в отрасли.

Антипаттерны: “Agile-театр”

Организации часто внедряют Agile поверхностно, не меняя культуру. Признаки “Agile-театра”:

- Стендапы проводятся, но длятся 40 минут и превращаются в отчёт менеджеру
- Спринты есть, но скоуп меняется посреди спринта руководством
- Ретроспективы проводятся, но ничего не меняется по их итогам
- Есть Scrum Master, но он же и менеджер, который раздаёт задачи
- Backlog существует, но приоритеты меняет не Product Owner, а директор по звонку

Настоящий Agile требует организационных изменений: автономии команд, доверия, плоской иерархии, культуры экспериментов.

Что дальше

В уроке 2.2 мы разберём Scrum – самый популярный Agile-фреймворк. Вы узнаете о трёх ролях (Product Owner, Scrum Master, Developers), пяти событиях и трёх артефактах, которые составляют основу Scrum.

Урок 2.1 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.2: Scrum – роли, артефакты, церемонии

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Scrum – самый популярный Agile-фреймворк в мире. По данным State of Agile Report 2023, 87% Agile-команд используют Scrum или его вариации. Фреймворк создан Кеном Швабером и Джеффом Сазерлендом в начале 1990-х и описан в Scrum Guide – документе всего на 13 страниц, который определяет все правила игры.

Scrum строится на эмпирическом контроле процессов: вместо того чтобы предсказывать будущее, команда работает короткими итерациями (спринтами), регулярно инспектирует результат и адаптирует план. Три столпа эмпиризма в Scrum: **прозрачность** (все видят реальное состояние), **инспекция** (регулярная проверка прогресса) и **адаптация** (корректировка курса).

В этом уроке мы полностью разберём структуру Scrum: три роли (Scrum Team), три артефакта и пять событий (церемоний). Это каркас, на котором строится вся работа Scrum-команды.

Три роли в Scrum

Scrum Guide 2020 определяет единую Scrum Team, состоящую из трёх ролей. Нет иерархии, нет субкоманд.

Product Owner (Владелец продукта)

Один человек, отвечающий за максимизацию ценности продукта и управление Product Backlog.

Обязанности: - Формулирование и приоритизация элементов Product Backlog - Определение цели продукта (Product Goal) - Обеспечение прозрачности бэклога для всех стейкхолдеров - Принятие или отклонение результатов работы команды

Ключевое правило: Product Owner – это ОДИН человек, не комитет. Он может делегировать работу, но ответственность лежит на нём. Если руководство обходит Product Owner и напрямую ставит задачи команде – Scrum ломается.

Scrum Master

Служащий лидер (servant-leader), который отвечает за внедрение Scrum и устранение препятствий.

Обязанности: - Коучинг команды в самоорганизации и кросс-функциональности - Помощь Product Owner в управлении бэклогом - Устранение impediments (блокеров) – от технических проблем до организационных барьеров - Фасилитация Scrum-событий - Работа с организацией: продвижение Agile-культуры

Важно: Scrum Master – это НЕ менеджер проекта. Он не раздаёт задачи, не контролирует людей, не является “начальником стендапа”. Он создаёт условия, в которых команда работает максимально эффективно.

Developers (Разработчики)

В Scrum Guide 2020 термин Developers обозначает всех членов команды, создающих инкремент: программистов, тестировщиков, дизайнеров, аналитиков – любых специалистов.

Характеристики: - Кросс-функциональность: команда обладает всеми навыками для создания инкремента - Самоорганизация: команда сама решает КАК выполнить работу - Рекомендуемый размер: 3-9 человек (без PO и SM) - Ответственность за создание Sprint Backlog, соблюдение Definition of Done и адаптацию плана

Роль	Фокус	Вопрос	Антипаттерн
Product Owner	ЧТО делать	“Какая фича принесёт больше ценности?”	Комитет из 5 человек решает приоритеты
Scrum Master	КАК работать	“Что мешает команде? Как улучшить процесс?”	SM = менеджер, раздающий задачи
Developers	КАК сделать	“Как реализовать это технически?”	Команда не принимает решений, ждёт указаний

Три артефакта Scrum

Артефакты обеспечивают прозрачность. Каждый артефакт имеет связанное обязательство (commitment).

Product Backlog (Бэклог продукта)

Упорядоченный список всего, что может понадобиться в продукте. Единственный источник требований. Живой документ, который постоянно уточняется (Backlog Refinement).

Обязательство: Product Goal – долгосрочная цель продукта, к которой стремится Scrum Team.

Структура типичного элемента бэклога: - Заголовок (User Story или описание задачи) - Описание / критерии приёмки - Приоритет (порядок в бэклоге) - Оценка (Story Points) - Статус

Sprint Backlog (Бэклог спринта)

Набор элементов Product Backlog, выбранных для спринта, + план их выполнения + Sprint Goal. Принадлежит Developers. Только команда может добавлять или удалять элементы Sprint Backlog.

Обязательство: Sprint Goal – единая цель спринта, которая создаёт фокус и согласованность.

Пример Sprint Goal: “Пользователь может зарегистрироваться, войти и восстановить пароль” – а не список из 12 задач.

Increment (Инкремент)

Готовый, работающий кусок продукта, который добавляет ценность ко всем предыдущим инкрементам. Инкремент должен быть потенциально готовым к релизу.

Обязательство: Definition of Done (DoD) – формальное описание качества, которому должен соответствовать инкремент.

Пример Definition of Done: - Код написан и прошёл code review - Юнит-тесты написаны и проходят (покрытие >80%) - Интеграционные тесты пройдены - Документация обновлена - Деплой на staging выполнен - Product Owner принял функционал

Пять событий (церемоний) Scrum

Все события – это формальные возможности для инспекции и адаптации.

1. Sprint (Спринт)

Контейнер для всех остальных событий. Фиксированный период (1-4 недели, чаще всего 2 недели), в течение которого создаётся инкремент.

Правила спринта: - Длительность спринта не меняется - Sprint Goal не меняется - Качество не снижается - Scope может уточняться (через переговоры PO и Developers) - Только Product Owner может отменить спринт (крайне редко)

2. Sprint Planning (Планирование спринта)

Открывает спринт. Команда отвечает на 3 вопроса: - ЗАЧЕМ этот спринт ценен? (Sprint Goal) - ЧТО можно сделать в этом спринте? (элементы из Product Backlog) - КАК будет выполнена работа? (декомпозиция на задачи)

Timebox: максимум 8 часов для месячного спринта (4 часа для двухнедельного).

3. Daily Scrum (Ежедневный стендап)

15-минутная встреча Developers для синхронизации и планирования работы на день. Подробнее в Уроке 2.4.

4. Sprint Review (Обзор спринта)

Демонстрация инкремента стейкхолдерам, получение обратной связи, адаптация Product Backlog. Подробнее в Уроке 2.4.

5. Sprint Retrospective (Ретроспектива спринта)

Команда анализирует процесс работы и определяет улучшения. Подробнее в Уроке 2.5.

Событие	Участники	Timebox (2-нед. спринт)	Результат
Sprint Planning	Вся Scrum Team	4 часа	Sprint Goal + Sprint Backlog
Daily Scrum	Developers	15 минут	План на день
Sprint Review	Scrum Team + стейкхолдеры	2 часа	Обратная связь, обновлённый Backlog
Retrospective	Scrum Team	1.5 часа	Actionable improvements

Scrum Flow: как всё работает вместе

Последовательность одного спринта:

1. **Sprint Planning** – выбираем цель и работу
2. **Daily Scrum** – каждый день синхронизируемся
3. **Разработка** – команда создаёт инкремент
4. **Sprint Review** – показываем результат, получаем фидбэк
5. **Sprint Retrospective** – улучшаем процесс
6. **Повторяем** – новый спринт начинается сразу после завершения предыдущего

Между спринтами нет перерывов. Product Backlog Refinement (груминг) происходит непрерывно в течение спринта, занимая до 10% времени команды.

Типичные ошибки при внедрении Scrum

1. **Scrum Master = менеджер проекта.** SM фасилитирует, а не управляет. Если он назначает задачи – это не Scrum.
2. **Product Owner – по совместительству.** PO на полставки не может полноценно управлять бэклогом.
3. **Нет Definition of Done.** Без DoD “готово” означает разное для разных людей.
4. **Спринты как мини-Waterfall.** Неделя 1 – аналитика, неделя 2 – разработка, неделя 3 – тест. Это не Scrum.
5. **Пропуск ретроспектив.** Без рефлексии нет улучшений. Это ключевой принцип эмпиризма.

Что дальше

В уроке 2.3 мы глубоко погрузимся в Sprint Planning – как правильно планировать спринт, формулировать Sprint Goal и декомпозировать работу на задачи.

Урок 2.2 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.3: Sprint Planning – планирование спринта

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Sprint Planning – это событие, которое открывает каждый спринт и задаёт направление работы команды на ближайшие 1-4 недели. Именно здесь формулируется Sprint Goal, выбираются элементы Product Backlog и создаётся план их реализации. Плохое планирование спринта приводит к размытому фокусу, перегрузке команды и невыполненным обещаниям. Хорошее – к чёткому пониманию цели, реалистичному объёму работы и мотивированной команде.

По Scrum Guide 2020, Sprint Planning отвечает на три вопроса: ЗАЧЕМ этот спринт ценен? ЧТО можно сделать? КАК будет выполнена работа? В этом уроке мы разберём каждый вопрос, научимся формулировать качественный Sprint Goal и проведём практическое планирование на примере реального проекта.

Структура Sprint Planning

Timebox и участники

Параметр	Значение
Timebox	8 часов (4-нед. спринт) / 4 часа (2-нед. спринт)
Участники	Вся Scrum Team (PO + SM + Developers)
Фасилитатор	Scrum Master
Вход	Приоритизированный Product Backlog, Velocity команды, DoD
Выход	Sprint Goal + Sprint Backlog

Три темы Sprint Planning

Scrum Guide 2020 структурирует планирование вокруг трёх тем (topics):

Тема 1: ЗАЧЕМ (Why) Product Owner объясняет, почему этот спринт ценен для продукта и бизнеса. Команда совместно формулирует Sprint Goal. Это самая важная часть планирования – без цели спринт превращается в список задач.

Тема 2: ЧТО (What) Developers обсуждают с Product Owner, какие элементы Product Backlog можно включить в спринт. Учитываются: приоритет (порядок в бэклоге), оценки (Story Points), velocity команды, доступность людей.

Тема 3: КАК (How) Developers декомпозируют выбранные элементы на конкретные задачи (tasks). Каждая задача – 4-8 часов работы. Это даёт команде план и уверенность в достижимости Sprint Goal.

Sprint Goal: сердце спринта

Что такое Sprint Goal

Sprint Goal – это единая цель спринта, которая создаёт фокус и согласованность. Sprint Goal неизменен в течение спринта. Он даёт гибкость: конкретные задачи могут меняться, но цель остаётся.

Плохие Sprint Goals

- “Закрыть 15 тикетов из бэклога” – это не цель, а список
- “Работать над модулем оплаты” – слишком размыто
- “Сделать всё, что не успели в прошлом спринте” – нет фокуса
- “Sprint 14” – это нумерация, не цель

Хорошие Sprint Goals

- “Пользователь может оформить заказ и оплатить картой” – конкретный пользовательский сценарий
- “Административная панель готова для первого бета-тестирования” – измеримый результат
- “Время загрузки главной страницы сокращено до < 2 секунд” – конкретная метрика
- “Интеграция с платёжной системой Stripe завершена и протестирована” – чёткий deliverable

Формула Sprint Goal

Попробуйте шаблон: **“К концу спринта [кто] сможет [что делать], что позволит [бизнес-ценность]”**

Пример: “К концу спринта администратор сможет управлять каталогом товаров (CRUD + фильтры), что позволит начать загрузку контента для бета-запуска.”

Определение объёма спринта

Capacity Planning (Планирование ёмкости)

Перед выбором задач нужно определить, сколько работы команда реально может сделать.

Шаг 1: Определите доступность

Участник	Рабочие дни	Отпуск/больничный	Другие проекты	Доступные дни
Алексей (Backend)	10	0	0	10
Мария (Frontend)	10	2	0	8
Дмитрий (QA)	10	0	3	7
Итого	30	2	3	25

Шаг 2: Вычтите накладные расходы Обычно 20-30% времени уходит на митинги, code review, помощь коллегам, обучение. Если доступно 25 человеко-дней, реальная ёмкость – 17-20 человеко-дней.

Шаг 3: Используйте Velocity Velocity – это среднее количество Story Points, которые команда закрывает за спринт. Если последние 3 спринта velocity = 34, 38, 32, то средняя velocity = 34.7. Берите в спринт ~35 Story Points.

Типичная ошибка: переоценка ёмкости

По данным исследований, команды систематически завышают свою ёмкость на 20-40%. Правило: лучше взять меньше и добавить задачи в середине спринта, чем не выполнить Sprint Goal.

Практический алгоритм Sprint Planning

Подготовка (до Sprint Planning)

1. **Product Owner** приоритизирует бэклог, верхние 15-20 элементов детализированы (refined)
2. **Developers** знакомятся с верхушкой бэклога заранее (на Backlog Refinement)
3. **Scrum Master** готовит данные: velocity, capacity, результаты прошлого спринта

Во время Sprint Planning

Первые 15-20 минут: Sprint Goal - PO представляет бизнес-контекст: что важно для продукта именно сейчас? - Команда обсуждает и формулирует Sprint Goal - Goal фиксируется и больше не меняется

Следующие 60-90 минут: выбор элементов - PO представляет верхние элементы бэклога по порядку приоритета - Developers задают вопросы, уточняют критерии приёмки - Команда оценивает, укладываются ли элементы в ёмкость - Формируется список элементов Sprint Backlog

Последние 60-90 минут: декомпозиция - Developers разбивают каждый элемент на задачи (tasks) - Каждая задача = 4-8 часов - Задачи фиксируются на Sprint Board (Jira, Trello, физическая доска) - Команда подтверждает реалистичность плана

Пример Sprint Planning

Проект: мобильное приложение для доставки еды. Спринт 5, 2 недели.

Sprint Goal: “Пользователь может найти ресторан, выбрать блюда и оформить заказ (без оплаты).”

Выбранные элементы:

ID	User Story	Story Points
US-042	Как пользователь, я хочу искать рестораны по названию и кухне	5
US-043	Как пользователь, я хочу видеть меню ресторана с ценами	8
US-044	Как пользователь, я хочу добавлять блюда в корзину	5
US-045	Как пользователь, я хочу оформить заказ с выбором адреса доставки	8
US-046	Как пользователь, я хочу видеть историю заказов	3
BUG-012	Фикс: крэш при повороте экрана на странице профиля	2
TECH-008	Рефакторинг: переход на новый API картографии	5
Итого		**36 SP**

Velocity команды: 34-38 SP. Объём реалистичен.

Антипаттерны Sprint Planning

1. **Планирование без Product Owner.** Команда гадает о приоритетах вместо того, чтобы услышать их от PO.
2. **Нет Sprint Goal.** Спринт превращается в “мешок задач” без фокуса.
3. **Перегрузка спринта.** “Возьмём побольше, чтобы был запас” – верный путь к burnout и невыполнению.
4. **Планирование без оценок.** Команда берёт задачи “на глазок”, без учёта velocity.
5. **Слишком длинное планирование.** Если Planning затягивается на весь день – значит, бэклог плохо проработан (Refinement провален).
6. **Микроменеджмент.** PO или SM указывают команде, КАК делать работу. Developers решают сами.

Что дальше

В уроке 2.4 мы разберём два ключевых ежедневных и еженедельных события: Daily Scrum (ежедневный стендап) и Sprint Review (обзор спринта). Вы узнаете, как проводить их эффективно и избегать превращения в скучную отчётность.

Урок 2.3 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.4: Daily Standup и Sprint Review

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Два события Scrum определяют ежедневный пульс команды и еженедельную связь с заказчиком: Daily Scrum (ежедневный стендап) и Sprint Review (обзор спринта). Оба критически важны, и оба чаще всего проводятся неправильно. Daily Scrum превращается в скучный отчёт менеджеру, а Sprint Review – в формальную презентацию, после которой ничего не меняется.

В этом уроке мы разберём оба события по Scrum Guide 2020: их цель, формат, типичные ошибки и практические рекомендации для эффективного проведения. Вы научитесь проводить 15-минутный стендап, который реально помогает команде, и Sprint Review, который даёт ценную обратную связь.

Daily Scrum (Ежедневный стендап)

Цель и формат

Daily Scrum – это 15-минутное событие для Developers, которое проводится каждый рабочий день в одно и то же время, в одном и том же месте. Цель – инспектировать прогресс к Sprint Goal и адаптировать Sprint Backlog.

Параметр	Значение
Timebox	15 минут (строго)
Участники	Developers (обязательно), SM (фасилитатор), PO (опционально, наблюдатель)
Частота	Каждый рабочий день
Формат	Стоя (поэтому “standup”), у доски или экрана с Sprint Board

Классический формат: три вопроса

Традиционно каждый участник отвечает на три вопроса:

1. **Что я сделал вчера** для достижения Sprint Goal?
2. **Что я планирую сделать сегодня** для достижения Sprint Goal?
3. **Есть ли блокеры** (impediments), которые мешают прогрессу?

Современный формат: фокус на Sprint Goal

Scrum Guide 2020 не предписывает формат трёх вопросов. Многие опытные команды переключаются на другие форматы:

Walk the Board – команда смотрит на Sprint Board справа налево (от “Done” к “To Do”) и обсуждает каждый элемент: - Что нужно, чтобы этот элемент продвинулся вперёд? - Кто над ним работает? Нужна ли помощь? - Есть ли блокеры?

Этот формат фокусирует на задачах, а не на людях, и помогает выявить застрявшие элементы.

Focus on Sprint Goal – команда оценивает прогресс к Sprint Goal: - Насколько мы близки к Sprint Goal? - Какие риски могут помешать? - Нужно ли скорректировать план?

Что НЕ является Daily Scrum

- **Не отчёт менеджеру.** Daily Scrum – это синхронизация МЕЖДУ разработчиками, а не доклад руководству.
- **Не решение проблем.** Если обсуждение затягивается – выносите в отдельную встречу (“parking lot”).
- **Не статус-митинг.** Фокус на Sprint Goal, а не на перечислении выполненных тикетов.
- **Не планёрка.** Решения о том, кто что делает, принимаются командой, а не менеджером.

Антипаттерны Daily Scrum

Антипаттерн	Проблема	Решение
“Отчёт начальнику”	Люди говорят для менеджера, а не для команды	SM напоминает: “Говорите друг с другом, не со мной”
“Монолог на 5 минут”	Один человек занимает всё время	Строгий timebox: 1-2 минуты на человека
“Технические дебаты”	Обсуждение архитектуры на 30 минут	Parking lot: “Обсудим после стендапа”
“Все молчат”	Никто не говорит о проблемах	SM задаёт наводящие вопросы, безопасная среда
“Пропуск стендапа”	“Сегодня нечего сказать”	Daily проводится КАЖДЫЙ день, даже если кажется бесполезным
“Async-only стендап”	Только текст в Slack, нет живого общения	Для распределённых команд: видеозвонок 1-2 раза в неделю

Советы для эффективного Daily Scrum

1. **Стойте.** Физический дискомфорт от стояния естественно ограничивает время.
2. **Начинайте вовремя.** Не ждите опоздавших. Уважайте время тех, кто пришёл.
3. **Используйте доску.** Визуальная Sprint Board (физическая или Jira) – фокус обсуждения.
4. **Parking lot.** Записывайте вопросы для обсуждения после стендапа.
5. **Ротация фасилитатора.** Не только SM – пусть каждый член команды по очереди ведёт Daily.

Sprint Review (Обзор спринта)

Цель и формат

Sprint Review – событие в конце спринта, где Scrum Team демонстрирует результаты работы стейкхолдерам и получает обратную связь. Это НЕ формальная презентация и НЕ отчёт – это рабочая встреча для инспекции инкремента и адаптации Product Backlog.

Параметр	Значение
Timebox	4 часа (4-нед. спринт) / 2 часа (2-нед. спринт)
Участники	Вся Scrum Team + ключевые стейкхолдеры
Фасилитатор	Product Owner (обычно)
Вход	Инкремент продукта, Sprint Goal
Выход	Обратная связь, обновлённый Product Backlog

Структура Sprint Review

Часть 1: Контекст (5-10 минут) - Product Owner напоминает Sprint Goal - Краткий обзор: что планировали, что сделали, что не сделали и почему

Часть 2: Демонстрация инкремента (30-60 минут) - Команда показывает РАБОТАЮЩИЙ продукт (не слайды, не мокапы) - Демо проводится на staging-среде или в продакшене - Стейкхолдеры могут сами “потрогать” продукт - Только функционал, соответствующий Definition of Done

Часть 3: Обратная связь и обсуждение (20-40 минут) - Стейкхолдеры задают вопросы, предлагают изменения - Обсуждение рыночных условий, конкурентов, новых возможностей - PO фиксирует все предложения для Product Backlog

Часть 4: Адаптация (10-15 минут) - PO обновляет Product Backlog на основе полученной обратной связи - Обсуждение приоритетов следующего спринта - Прогноз: что примерно будет готово к следующему релизу

Кого приглашать на Sprint Review

- **Обязательно:** вся Scrum Team
- **Рекомендуется:** заказчик / представитель бизнеса, конечные пользователи (если возможно), смежные команды
- **Опционально:** руководство, маркетинг, поддержка

Чем больше реальных пользователей видят демо – тем ценнее обратная связь.

Антипаттерны Sprint Review

Антипаттерн	Проблема	Решение
“PowerPoint Review”	Слайды вместо работающего продукта	Только живое демо, только работающий софт
“Нет стейкхолдеров”	Команда показывает друг другу	Активно приглашать стейкхолдеров, менять время под них
“Только позитив”	Скрытие неудач и проблем	Прозрачность: что не сделали и почему – это нормально
“Формальная приёмка”	Review = подписание акта	Review – это обратная связь, а не gate
“Без последствий”	Фидбэк не попадает в бэклог	PO обязан зафиксировать каждое предложение

Советы для эффективного Sprint Review

1. **Готовьте демо заранее.** Определите, кто показывает какой функционал. Прогоните демо за 30 минут до Review.
2. **Создайте тестовые данные.** Демо с пустой базой или “Lorem ipsum” не впечатляет. Подготовьте реалистичные данные.
3. **Задавайте вопросы стейкхолдерам.** Не просто показывайте – спрашивайте: “Это то, что вы ожидали? Что бы вы изменили?”
4. **Фиксируйте всё.** Каждый комментарий стейкхолдера -> элемент Product Backlog.
5. **Празднуйте успехи.** Sprint Review – это возможность признать работу команды.

Daily Scrum и Sprint Review: связь

Оба события – про инспекцию и адаптацию, но на разных уровнях:

Аспект	Daily Scrum	Sprint Review
Частота	Ежедневно	Раз в спринт
Фокус	Прогресс к Sprint Goal	Инкремент продукта
Участники	Developers	Scrum Team + стейкхолдеры
Адаптация	Sprint Backlog (план на день)	Product Backlog (бэклог продукта)
Цикл обратной связи	24 часа	1-4 недели

Что дальше

В уроке 2.5 мы разберём Sprint Retrospective – событие, направленное на непрерывное улучшение процесса работы команды. Вы узнаете форматы ретроспектив, научитесь фасилитировать их и превращать обсуждения в конкретные действия.

Урок 2.4 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.5: Retrospective – непрерывное улучшение

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Sprint Retrospective – последнее событие спринта и, по мнению многих Agile-коучей, самое важное. Именно ретроспектива реализует 12-й принцип Agile Manifesto: “Команда регулярно размышляет о том, как стать эффективнее, и корректирует своё поведение.” Без ретроспектив команда обречена повторять одни и те же ошибки.

По данным исследования Scrum Alliance, команды, которые проводят ретроспективы регулярно и внедряют хотя бы одно улучшение каждый спринт, показывают на 24% более высокую производительность через 6 месяцев. Но есть и обратная сторона: ретроспективы, которые проводятся формально (“всё хорошо, улучшений нет”), не только бесполезны – они деморализуют команду.

В этом уроке вы научитесь проводить ретроспективы, которые приводят к реальным изменениям: выберете формат, создадите безопасную среду и превратите обсуждения в конкретные действия.

Основы Sprint Retrospective

Цель и формат

Sprint Retrospective – это возможность для Scrum Team инспектировать себя и создать план улучшений для следующего спринта.

Параметр	Значение
Timebox	3 часа (4-нед. спринт) / 1.5 часа (2-нед. спринт)
Участники	Вся Scrum Team (PO + SM + Developers)
Фасилитатор	Scrum Master
Вход	Опыт прошедшего спринта
Выход	Actionable improvements (конкретные действия)

Три вопроса ретроспективы

Классический формат строится вокруг трёх вопросов:

1. **Что прошло хорошо?** – практики, которые стоит сохранить
2. **Что можно улучшить?** – проблемы, неэффективности, боли
3. **Какие конкретные действия мы предпримем?** – 1-3 улучшения для следующего спринта

Безопасная среда: основа успеха

Ретроспектива работает только если люди чувствуют себя в безопасности. Если разработчик боится сказать “наш процесс deployment сломан” из-за реакции менеджера – ретроспектива бесполезна.

Правила безопасной среды: - Vegas Rule: “Что сказано на ретро – остаётся на ретро” - Нет обвинений: обсуждаем процессы, а не людей - Все мнения равны: голос джуниора так же важен, как голос тимлида - Prime Directive (Норм Керт): “Независимо от того, что мы обнаружим, мы понимаем и искренне верим, что каждый делал лучшее, на что был способен, учитывая известную информацию, навыки и ресурсы”

Форматы ретроспектив

Монотонность убивает ретроспективы. Меняйте формат каждые 2-4 спринта.

Формат 1: Start / Stop / Continue

Самый простой формат для начинающих команд.

Колонка	Вопрос	Пример
Start	Что мы должны начать делать?	Начать проводить code review до обеда
Stop	Что мы должны прекратить?	Прекратить добавлять задачи в середине спринта
Continue	Что мы должны продолжать?	Продолжать парное программирование на сложных задачах

Формат 2: Mad / Sad / Glad

Эмоциональный формат, помогающий выявить скрытые проблемы.

- **Mad (Злость):** Что вас раздражало в этом спринте?
- **Sad (Грусть):** Что вас расстроило или разочаровало?
- **Glad (Радость):** Что порадовало, что было классно?

Формат 3: 4L – Liked, Learned, Lacked, Longed For

- **Liked:** Что понравилось?
- **Learned:** Чему научились?
- **Lacked:** Чего не хватило?
- **Longed For:** О чём мечтали (но не было)?

Формат 4: Sailboat (Парусник)

Визуальная метафора – команда = парусник, плывущий к цели.

- **Ветер (Wind):** Что двигало нас вперёд?
- **Якорь (Anchor):** Что тормозило?
- **Скалы (Rocks):** Какие риски впереди?
- **Остров (Island):** Куда мы плывём? (цель)

Рисуется на доске или в Miro, каждый приклеивает стикеры.

Формат 5: Timeline

Команда рисует хронологию спринта и отмечает ключевые события: что было хорошо (зелёные стикеры), что плохо (красные), что нейтрально (жёлтые). Позволяет увидеть паттерны: например, последние два дня спринта всегда стрессовые.

Пошаговый алгоритм проведения ретроспективы

Подготовка (15 минут до)

1. Выберите формат (не повторяйте предыдущий)
2. Подготовьте доску: физическую или Miro/FigJam
3. Подготовьте данные: velocity спринта, burndown chart, количество багов
4. Проверьте статус action items прошлой ретроспективы

Шаг 1: Check-in (5 минут)

Короткое упражнение для вовлечения. Примеры: - “Одним словом опишите этот спринт” (каждый говорит одно слово) - “Оцените спринт от 1 до 5 пальцев” (покажите одновременно) - “Какой суперсилой вы пользовались в этом спринте?”

Шаг 2: Сбор данных (15-20 минут)

Каждый участник пишет стикеры (физические или в Miro) по выбранному формату. Один стикер = одна мысль. Писать молча, 5-7 минут. Затем каждый озвучивает свои стикеры и приклеивает на доску.

Шаг 3: Группировка и голосование (10 минут)

Группируйте похожие стикеры в кластеры. Дайте каждому кластеру название. Затем dot-voting: каждый участник получает 3 точки (голоса) и ставит их на самые важные кластеры. Топ-2-3 кластера – приоритеты для обсуждения.

Шаг 4: Обсуждение (20-30 минут)

Обсудите приоритетные кластеры: - В чём корневая причина проблемы? (используйте “5 Why”) - Что конкретно мы можем сделать? - Кто возьмёт на себя ответственность? - Как мы измерим улучшение?

Шаг 5: Action Items (10 минут)

Сформулируйте 1-3 конкретных действия (не больше!). Каждый action item должен быть SMART:

Action Item	Ответственный	Дедлайн	Метрика успеха
Внедрить автоматический линтер в CI/CD	Алексей	Середина следующего спринта	0 стилевых замечаний на code review
Перенести Daily Scrum на 10:00 вместо 9:00	Scrum Master	Со следующего спринта	Все на месте, опоздания = 0

Шаг 6: Check-out (5 минут)

“Что вы вынесли из этой ретроспективы? Одно предложение.”

Работа с action items

Главная причина провала ретроспектив – action items не выполняются. Решения:

1. **Добавляйте action items в Sprint Backlog** следующего спринта. Это работа, которая требует времени.
2. **Ограничивайте количество.** 1-3 action items, не больше. Лучше сделать одно улучшение, чем запланировать десять и не сделать ни одного.
3. **Проверяйте статус.** Начинайте каждую ретроспективу с проверки: что из прошлых action items было сделано?
4. **Визуализируйте.** Создайте доску “Improvements” в Jira/Trello/Notion – отдельный бэклог улучшений.

Антипаттерны ретроспектив

1. **“Всё хорошо, улучшений нет.”** Если команда не находит что улучшить – значит, нет безопасной среды или нет честности.
2. **“Одни и те же проблемы каждый спринт.”** Action items не выполняются. Решение: добавлять в Sprint Backlog.
3. **“Ретро = жалобы без решений.”** SM должен переводить жалобы в конкретные действия.
4. **“Пропускаем ретро, нет времени.”** Ретроспектива – это инвестиция в эффективность. Пропуск = техдолг по процессам.
5. **“Менеджер на ретро.”** Если руководитель присутствует и оценивает – люди замолчат. Ретро – для команды.

Что дальше

В уроке 2.6 мы перейдём к Kanban – другому популярному Agile-фреймворку. Вы узнаете, как визуализировать поток работы, управлять WIP-лимитами и когда Kanban лучше Scrum.

Урок 2.5 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.6: Kanban – визуализация потока работы

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Kanban – второй по популярности Agile-фреймворк после Scrum. Но в отличие от Scrum, Kanban не предписывает роли, спринты или церемонии. Его сила – в визуализации потока работы и ограничении незавершённой работы (WIP). Kanban зародился на заводах Toyota в 1940-х годах как система управления производством и был адаптирован для IT Дэвидом Андерсоном в 2007 году.

Почему Kanban стоит изучать, даже если вы используете Scrum? Во-первых, многие команды используют Scrumban (гибрид). Во-вторых, Kanban-доска – универсальный инструмент визуализации, который работает в любом фреймворке. В-третьих, для некоторых типов работы (поддержка, DevOps, маркетинг) Kanban подходит лучше, чем Scrum.

В этом уроке мы разберём 6 практик и 4 принципа Kanban, научимся проектировать Kanban-доску, устанавливать WIP-лимиты и измерять эффективность потока.

4 принципа Kanban

Kanban Method (Дэвид Андерсон) строится на четырёх фундаментальных принципах:

1. **Начните с того, что делаете сейчас.** Kanban не требует революции. Не меняйте роли, не ломайте процессы. Начните с визуализации текущего потока.
2. **Договоритесь об эволюционном улучшении.** Маленькие, инкрементальные изменения вместо радикальных трансформаций. Это снижает сопротивление.
3. **Уважайте текущие роли, ответственности и должности.** Kanban не назначает новых ролей (в отличие от Scrum). Существующая структура сохраняется.
4. **Поощряйте лидерство на всех уровнях.** Улучшения инициируются не только менеджерами, но и каждым членом команды.

6 практик Kanban

Практика 1: Визуализация потока работы

Основа Kanban – доска, на которой каждая задача представлена карточкой, а колонки отображают этапы процесса.

Минимальная Kanban-доска:

To Do	In Progress	Done
Задача 1	Задача 4	Задача 7
Задача 2	Задача 5	Задача 8
Задача 3		

Продвинутая Kanban-доска для разработки:

Backlog	Analysis	Development	Code Review	Testing	Staging	Done
US-15	US-12	US-10	US-08	US-06	US-04	US-01

Backlog	Analysis	Development	Code Review	Testing	Staging	Done
US-16		US-11	US-09			US-02
US-17						US-03

Практика 2: Ограничение WIP (Work In Progress)

WIP-лимит – максимальное количество задач, которые могут одновременно находиться в колонке. Это ключевая практика Kanban, отличающая его от простой доски задач.

Зачем нужны WIP-лимиты: - Мультитаскинг снижает продуктивность на 20-40% (исследование APA) - WIP-лимиты вынуждают завершать начатое, прежде чем начинать новое - Они выявляют узкие места (bottlenecks) в процессе - Они сокращают Lead Time (время от запроса до доставки)

Как установить WIP-лимиты:

Правило большого пальца: WIP-лимит колонки = количество людей, работающих на этом этапе + 1 (буфер).

Колонка	Люди	WIP-лимит
Analysis	1 аналитик	2
Development	3 разработчика	4
Code Review	2 ревьюера	3
Testing	1 QA	2

Если колонка заполнена до лимита – новые задачи в неё не добавляются. Команда сначала помогает завершить текущие задачи.

Практика 3: Управление потоком

Команда постоянно мониторит поток работы и устраняет препятствия: - Если задачи накапливаются перед колонкой (bottleneck) – нужно увеличить пропускную способность этого этапа - Если колонка часто пуста (starving) – предыдущий этап работает слишком медленно - Цель – равномерный, предсказуемый поток

Практика 4: Сделайте политики явными

Определите правила перемещения задач между колонками: - Критерии входа: что нужно, чтобы задача попала в колонку? - Критерии выхода: что нужно, чтобы задача покинула колонку? - Definition of Done для каждого этапа

Пример: - “Development -> Code Review”: код написан, юнит-тесты проходят, PR создан - “Code Review -> Testing”: минимум 2 approve, все замечания исправлены - “Testing -> Done”: все тест-кейсы пройдены, нет критических багов

Практика 5: Внедряйте петли обратной связи

Kanban не предписывает конкретные митинги, но рекомендует регулярные “каденции” (cadences):

- **Daily Kanban Meeting** (ежедневно, 15 мин) – обзор доски, фокус на заблокированных задачах
- **Replenishment Meeting** (еженедельно/по необходимости) – пополнение бэклога, приоритизация
- **Service Delivery Review** (раз в 2 недели) – анализ метрик, SLA
- **Operations Review** (ежемесячно) – кросс-командная координация
- **Retrospective** (по необходимости) – улучшение процесса

Практика 6: Улучшайте совместно, эволюционируйте экспериментально

Используйте данные (метрики) для принятия решений об изменениях. Каждое изменение – это эксперимент: внедрите, измерьте, оцените результат.

Метрики Kanban

Lead Time

Время от момента, когда задача попала в систему (Backlog), до её завершения (Done). Включает время ожидания.

Cycle Time

Время от начала активной работы над задачей (In Progress) до завершения (Done). Не включает время ожидания в бэклоге.

Throughput

Количество задач, завершённых за период (неделя, месяц). Позволяет прогнозировать, когда будет готова фича.

Cumulative Flow Diagram (CFD)

График, показывающий количество задач в каждом состоянии на каждый день. Позволяет визуально увидеть: - Растёт ли WIP? - Есть ли bottleneck (расширение полосы)? - Стабилен ли поток?

Метрика	Что измеряет	Формула
Lead Time	Время доставки	Дата Done - Дата создания
Cycle Time	Время работы	Дата Done - Дата начала работы
Throughput	Пропускная способность	Количество Done / период
WIP	Незавершённая работа	Карточки в In Progress

Kanban vs Scrum

Аспект	Scrum	Kanban
Итерации	Фиксированные спринты (1-4 недели)	Непрерывный поток
Роли	PO, SM, Developers	Нет предписанных ролей
Планирование	Sprint Planning	Пополнение по необходимости
Изменения	Scope спринта фиксирован	Задачи можно добавлять в любой момент
Ограничение работы	Sprint (timebox)	WIP-лимиты
Метрики	Velocity, Burndown	Lead Time, Cycle Time, CFD
Лучше подходит для	Разработка продуктов, кросс-функциональные команды	Поддержка, DevOps, непрерывный поток задач

Scrumban: лучшее из двух миров

Многие команды комбинируют Scrum и Kanban: - Спринты и церемонии из Scrum - WIP-лимиты и визуализация из Kanban - Pull-система: задачи “вытягиваются” командой, а не “выталкиваются” менеджером

Проектирование Kanban-доски: пошаговое руководство

1. **Определите этапы вашего процесса.** Пройдите путь задачи от “идеи” до “готово”. Какие этапы она проходит?
2. **Создайте колонки.** Каждый этап = колонка. Добавьте Backlog в начале и Done в конце.
3. **Установите WIP-лимиты.** Начните с формулы “люди + 1” для каждой колонки.
4. **Определите политики.** Для каждой границы между колонками: что нужно для перемещения?
5. **Визуализируйте типы задач.** Используйте цвета: баги – красные, фичи – синие, техдолг – жёлтые.
6. **Добавьте swimlanes.** Горизонтальные дорожки для разных потоков: “Срочно”, “Обычный”, “Фоновый”.

Что дальше

В уроке 2.7 мы разберём User Stories – основной способ описания требований в Agile. Вы узнаете, как писать качественные User Stories, определять критерии приёмки и использовать формат INVEST.

Урок 2.6 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.7: User Stories – как писать требования

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

User Story (пользовательская история) – основной способ описания требований в Agile. В отличие от традиционных спецификаций (ТЗ, SRS), User Stories описывают функционал с точки зрения пользователя: кому нужна эта фича, что она делает и зачем. Формат кажется простым, но именно в его простоте – сила.

User Stories были придуманы Кентом Беком в рамках Extreme Programming (XP) и стали стандартом для всех Agile-фреймворков. По данным VersionOne (State of Agile Report), 58% Agile-команд используют User Stories как основной формат бэклога.

В этом уроке вы научитесь писать качественные User Stories по формату “Как [кто], я хочу [что], чтобы [зачем]”, определять критерии приёмки (Acceptance Criteria), применять модель INVEST и структурировать бэклог через эпики и темы.

Формат User Story

Классическая формула

“Как [роль], я хочу [действие], чтобы [ценность]”

- **Роль (кто):** конкретный тип пользователя, не абстрактный “пользователь”
- **Действие (что):** конкретная функция или возможность
- **Ценность (зачем):** бизнес-ценность или мотивация пользователя

Примеры User Stories

Плохая User Story	Хорошая User Story	
Сделать авторизацию	Как зарегистрированный пользователь, я хочу войти по email и паролю, чтобы получить доступ к своему личному кабинету	
Добавить фильтры	Как покупатель, я хочу фильтровать товары по цене, размеру и цвету, чтобы быстрее найти нужный продукт	
Сделать отчёты	Как менеджер отдела продаж, я хочу видеть еженедельный отчёт по конверсии, чтобы оценить эффективность команды	
Оптимизация БД	Как администратор системы, я хочу, чтобы поиск по каталогу работал менее 1 секунды, чтобы пользователи не уходили из-за медленной загрузки	

Три “C” User Story (Рон Джеффрис)

1. **Card (Карточка):** краткое описание на карточке (физической или в Jira). Не полная спецификация, а приглашение к разговору.
2. **Conversation (Разговор):** детали обсуждаются лично между PO и Developers. Карточка – это напоминание о разговоре, а не замена.
3. **Confirmation (Подтверждение):** критерии приёмки (Acceptance Criteria), которые подтверждают, что история выполнена.

Acceptance Criteria (Критерии приёмки)

Acceptance Criteria – это конкретные условия, при выполнении которых User Story считается завершённой. Без них “готово” означает разное для PO, разработчика и тестировщика.

Формат Given-When-Then (Gherkin)

Given [предусловие], **When** [действие], **Then** [ожидаемый результат]

Пример для User Story “Как покупатель, я хочу добавить товар в корзину”:

- **Given** пользователь находится на странице товара, **When** он нажимает кнопку “В корзину”, **Then** товар добавляется в корзину и отображается счётчик товаров в шапке
- **Given** товар уже в корзине, **When** пользователь нажимает “В корзину” повторно, **Then** количество этого товара увеличивается на 1
- **Given** товар отсутствует на складе, **When** пользователь открывает страницу товара, **Then** кнопка “В корзину” неактивна и отображается надпись “Нет в наличии”

Формат чек-листа

Более простой формат для команд, не использующих BDD:

User Story: Как покупатель, я хочу оформить заказ с доставкой

Acceptance Criteria: - Пользователь может выбрать адрес доставки из сохранённых или ввести новый - Стоимость доставки рассчитывается автоматически при вводе адреса - Минимальная сумма заказа для бесплатной доставки = 3000 руб. - Пользователь получает подтверждение заказа на email - Заказ отображается в личном кабинете со статусом “Оформлен”

Модель INVEST

Майк Кон предложил акроним INVEST для оценки качества User Stories:

Критерий	Расшифровка	Пример нарушения
I – Independent	Независимая от других историй	“Сначала сделай US-10, потом US-11” (зависимость)
N – Negotiable	Обсуждаемая (не жёсткий контракт)	“Сделать точно как в макете, без обсуждений”
V – Valuable	Ценная для пользователя/бизнеса	“Рефакторинг модуля X” (техническая, нет пользовательской ценности)
E – Estimable	Оцениваемая (можно оценить трудозатраты)	“Улучшить UX” (слишком размыто)
S – Small	Достаточно маленькая для спринта	“Реализовать полный модуль CRM” (эпик, не история)
T – Testable	Тестируемая (есть чёткие criteria)	“Сделать красивый дизайн” (субъективно)

Как проверить по INVEST

Задайте шесть вопросов: 1. Можно ли реализовать эту историю отдельно от других? (I) 2. Могут ли PO и Developers обсудить детали? (N) 3. Получит ли пользователь реальную ценность? (V) 4. Может ли команда оценить объём работы? (E) 5. Можно ли закрыть

историю за один спринт? (S) 6. Можно ли написать тест, который подтвердит готовность? (T)

Иерархия: Тема -> Эпик -> User Story -> Задача

User Stories – это средний уровень детализации. Для структурирования бэклога используется иерархия:

Тема (Theme) – стратегическое направление Пример: “Улучшение процесса покупки”

Эпик (Epic) – большая функциональная область, которая разбивается на User Stories Пример: “Корзина и оформление заказа”

User Story – конкретная пользовательская возможность, реализуемая за 1 спринт Пример: “Как покупатель, я хочу добавить товар в корзину”

Задача (Task/Sub-task) – техническая задача для реализации User Story Пример: “Создать API endpoint POST /cart/items”

Пример декомпозиции эпика

Эпик: Регистрация и аутентификация

ID	User Story	SP
US-001	Как новый пользователь, я хочу зарегистрироваться по email, чтобы создать аккаунт	5
US-002	Как зарегистрированный пользователь, я хочу войти по email и паролю	3
US-003	Как пользователь, я хочу восстановить забытый пароль по email	5
US-004	Как пользователь, я хочу войти через Google/Apple, чтобы не вводить пароль	8
US-005	Как пользователь, я хочу включить двухфакторную аутентификацию	8
US-006	Как пользователь, я хочу выйти из аккаунта на всех устройствах	3

Специальные типы историй

Technical Stories (Технические истории)

Не все задачи имеют прямого пользователя. Технические задачи (рефакторинг, миграция, инфраструктура) описываются: - “Как разработчик, я хочу мигрировать с MySQL на PostgreSQL, чтобы улучшить производительность JSON-запросов” - Или как enabler: “Для поддержки US-042 необходимо настроить CI/CD pipeline”

Spike (Исследование)

Timeboxed исследовательская задача для снижения неопределённости: - “Spike: исследовать 3 платёжных провайдера (Stripe, PayPal, CloudPayments) – 8 часов” - Результат: не код, а решение (какой провайдер выбрать и почему)

Bug (Дефект)

Баги тоже можно описывать в формате: - “Как покупатель, я ожидаю, что при нажатии ‘Оплатить’ транзакция завершится, но вместо этого появляется ошибка 500” - Или в формате: Steps to Reproduce -> Expected -> Actual

---

Типичные ошибки при написании User Stories

1. **“Как пользователь” для всех историй.** Используйте конкретные роли: покупатель, администратор, курьер, менеджер.
2. **Пропуск части “чтобы”.** Без ценности история – просто задача. “Чтобы” заставляет думать о мотивации пользователя.
3. **Слишком крупные истории.** Если история не помещается в спринт – это эпик. Декомпозируйте.
4. **Технический язык.** User Story должна быть понятна PO и стейкхолдерам, а не только разработчикам.
5. **Отсутствие Acceptance Criteria.** Без AC невозможно протестировать и принять историю.

Что дальше

В уроке 2.8 мы разберём Story Points и Velocity – как оценивать сложность User Stories и прогнозировать, сколько работы команда может выполнить за спринт.

Урок 2.7 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.8: Story Points и Velocity – оценка и прогнозирование

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Одна из самых сложных задач в управлении проектами – точная оценка трудозатрат. Традиционный подход (“эта задача займёт 40 часов”) систематически ошибается: люди плохо оценивают абсолютное время, но хорошо сравнивают задачи между собой. Именно на этом принципе строится система Story Points.

Story Points – это относительная единица оценки сложности User Story. Не часы, не дни, а абстрактная мера, которая учитывает сложность, объём работы и неопределённость. Velocity – это среднее количество Story Points, которое команда закрывает за спринт. Вместе Story Points и Velocity дают мощный инструмент прогнозирования: когда будет готова фича? Сколько спринтов нужно до релиза?

В этом уроке вы научитесь оценивать задачи в Story Points, проводить Planning Poker, рассчитывать Velocity и прогнозировать сроки с учётом неопределённости.

Story Points: что это и зачем

Почему не часы?

Проблемы оценки в часах: - **Субъективность:** задача, которую сеньор сделает за 4 часа, джуниор будет делать 16 часов - **Закон Паркинсона:** работа заполняет всё отведённое время. Оценка в часах создаёт потолок - **Давление менеджмента:** “Ты сказал 8 часов, прошло 12 – почему?” Часы превращаются в обязательства - **Ошибка планирования (Planning Fallacy):** люди систематически недооценивают время на 25-50%

Что такое Story Points

Story Points – это относительная оценка, которая учитывает три фактора:

1. **Сложность (Complexity):** насколько технически сложна задача?
2. **Объём работы (Effort):** сколько работы нужно сделать?
3. **Неопределённость (Uncertainty/Risk):** насколько мы уверены в понимании задачи?

Story Points работают по принципу относительного сравнения. Команда выбирает “эталонную” задачу (обычно небольшую, хорошо понятную), присваивает ей, скажем, 3 Story Points, и все остальные задачи оценивает относительно этого эталона.

Шкала Фибоначчи

Для оценки используется модифицированная последовательность Фибоначчи: **1, 2, 3, 5, 8, 13, 21, 40, 100**

Почему Фибоначчи, а не линейная шкала (1, 2, 3, 4, 5...)? - Чем больше задача, тем выше неопределённость - Разница между 1 и 2 SP более значима, чем между 20 и 21 - Нет смысла спорить: “это 14 или 15?” – если больше 13, то 21

Story Points	Значение	Пример
1	Тривиальная задача	Изменить текст кнопки
2	Простая задача	Добавить валидацию email
3	Средняя задача	Создать форму обратной связи
5	Выше среднего	Реализовать поиск с фильтрами
8	Сложная задача	Интеграция с платёжной системой
13	Очень сложная задача	Реализовать систему уведомлений (email + push + in-app)
21+	Слишком большая	Требует декомпозиции на несколько историй

Правило: если оценка > 13 SP, историю нужно разбить на более мелкие.

Planning Poker

Planning Poker – самый популярный метод групповой оценки в Agile. Изобретён Джеймсом Греннингом, популяризирован Майком Коном.

Процесс Planning Poker

Шаг 1: Каждый участник получает набор карт с числами Фибоначчи (1, 2, 3, 5, 8, 13, 21). Можно использовать приложения: Scrum Poker, PlanningPokerOnline.com.

Шаг 2: Product Owner представляет User Story и отвечает на вопросы команды.

Шаг 3: Каждый участник выбирает карту с оценкой и кладёт рубашкой вверх.

Шаг 4: Все одновременно переворачивают карты.

Шаг 5: Если оценки совпадают (или близки: 5 и 8) – фиксируем. Если разброс большой (3 и 13) – участники с минимальной и максимальной оценкой объясняют свою логику. Затем второй раунд.

Шаг 6: Обычно 2-3 раундов достаточно для консенсуса.

Правила Planning Poker

- Оценивают только Developers (не PO, не SM, не менеджеры)
- Одновременное раскрытие карт предотвращает “якорение” (когда мнение авторитета влияет на остальных)
- Обсуждение расхождений – самая ценная часть. Часто выясняется, что люди по-разному понимают задачу
- Timebox: 3-5 минут на историю. Если дольше – историю нужно уточнить

Альтернативные методы оценки

T-Shirt Sizing (Размеры футболок) Быстрая предварительная оценка: S, M, L, XL. Используется при грубой оценке большого бэклога.

Размер	Story Points (примерно)
S	1-2

Размер	Story Points (примерно)
M	3-5
L	8-13
XL	21+ (декомпозировать)

Affinity Estimation (Группировка по подобию) Все истории раскладываются на столе. Команда молча группирует их по сложности (простые -> средние -> сложные -> очень сложные). Затем присваивает каждой группе число Фибоначчи. Подходит для оценки 20-50+ историй за 1-2 часа.

#NoEstimates Радикальный подход: вместо оценки Story Points команда считает количество завершённых историй (throughput). Если все истории примерно одного размера (2-5 SP), то throughput – достаточный предиктор.

Velocity: измерение скорости команды

Что такое Velocity

Velocity – это сумма Story Points всех User Stories, которые команда завершила (Done) в спринте. Только завершённые истории – частично сделанные не считаются.

Пример:

Спринт	Запланировано (SP)	Завершено (SP)	Velocity
Sprint 1	30	25	25
Sprint 2	28	30	30
Sprint 3	32	28	28
Sprint 4	30	32	32
Sprint 5	32	27	27
Среднее			**28.4**

Как использовать Velocity

Для планирования спринта: берите в спринт объём работы, близкий к средней Velocity. Если средняя Velocity = 28, не планируйте 40 SP.

Для прогнозирования релиза: если в бэклоге 120 SP до релиза, а Velocity = 28 SP/спринт, то потребуется примерно 120/28 = 4.3 спринта (~9 недель при 2-нед. спринтах).

Для отслеживания тренда: если Velocity падает спринт за спринтом – команда выгорает, растёт техдолг или есть организационные проблемы.

Правила работы с Velocity

1. **Velocity – метрика для команды, не для менеджмента.** Нельзя сравнивать Velocity разных команд. Story Points субъективны: “5” одной команды не равно “5” другой.
2. **Velocity стабилизируется через 3-5 спринтов.** Первые спринты – калибровка. Не делайте выводов по одному спринту.
3. **Не используйте Velocity как KPI.** Если Velocity станет целью, команда начнёт “надувать” оценки: то, что раньше было 3 SP, станет 8 SP.
4. **Частично завершённые истории = 0 SP.** Это мотивирует закрывать начатое, а не начинать новое.

Burndown Chart

Burndown Chart – визуализация прогресса спринта. Ось X – дни спринта, ось Y – оставшиеся Story Points.

Идеальная линия – равномерное снижение от полного объёма до нуля.

Реальная линия – фактический прогресс. Паттерны: - **Линия выше идеальной** – команда отстаёт. Нужно обсудить на Daily Scrum. - **Линия ниже идеальной** – команда опережает. Можно взять дополнительные истории. - **Плоская линия** – ничего не закрывается. Блокеры? Слишком крупные истории? - **Резкое падение в конце** – всё делается в последний момент. Mini-Waterfall.

Burnup Chart (альтернатива)

Burnup Chart показывает две линии: сколько сделано (растёт вверх) и общий объём (может расти при добавлении задач). Преимущество: видно изменения скоупа.

Release Planning с использованием Velocity

Прогнозирование с диапазоном

Вместо точной даты релиза дайте диапазон на основе оптимистичной и пессимистичной Velocity:

Параметр	Значение
Оставшийся объём	200 SP
Оптимистичная Velocity (лучший спринт)	35 SP
Средняя Velocity	28 SP
Пессимистичная Velocity (худший спринт)	22 SP
Оптимистичный прогноз	200/35 = 5.7 спринтов
Реалистичный прогноз	200/28 = 7.1 спринтов
Пессимистичный прогноз	200/22 = 9.1 спринтов

Итого: релиз через 6-9 спринтов (12-18 недель). Такой диапазон честнее, чем “точно к 15 марта”.

Что дальше

В уроке 2.9 мы разберём масштабирование Agile для крупных организаций: фреймворки SAFe, LeSS и Nexus. Как работать по Agile, когда у вас не 7 человек, а 70 или 700?

Урок 2.8 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.9: Масштабирование Agile – SAFe, LeSS, Nexus

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Scrum работает отлично для одной команды из 5-9 человек. Но что делать, когда над продуктом работают 5, 10 или 50 команд? Как координировать работу, управлять зависимостями и сохранять Agile-принципы в масштабе организации? Для решения этих задач были созданы фреймворки масштабирования Agile: SAFe, LeSS и Nexus.

По данным 16th Annual State of Agile Report, 53% организаций используют или планируют использовать фреймворки масштабирования. SAFe лидирует (37%), LeSS (5%) и Nexus (4%) занимают вторую и третью позиции. В этом уроке мы разберём ключевые идеи каждого фреймворка, их сильные и слабые стороны, и определим, когда какой подход применять.

Почему масштабирование Agile – это сложно

Проблемы масштаба

Когда над продуктом работает одна Scrum-команда, координация минимальна. При добавлении каждой новой команды сложность растёт экспоненциально:

Количество команд	Связи между командами	Уровень координации
1	0	Минимальный
2	1	Низкий
3	3	Средний
5	10	Высокий
10	45	Очень высокий
20	190	Критический

Формула: количество связей = n*(n-1)/2, где n – число команд.

Ключевые вызовы масштабирования

1. **Зависимости между командами.** Команда А не может двигаться, пока Команда Б не завершит API.
2. **Единый Product Backlog vs множество бэклогов.** Кто приоритизирует? Один PO на 50 разработчиков?
3. **Интеграция инкрементов.** Как объединить работу 10 команд в единый продукт?
4. **Архитектурная целостность.** Без координации каждая команда создаёт свою архитектуру.
5. **Культурное сопротивление.** Менеджеры среднего звена теряют контроль при Agile-трансформации.

SAFe (Scaled Agile Framework)

Обзор

SAFe – самый популярный и самый детальный фреймворк масштабирования. Создан Дином Леффингуэллом. Текущая версия – SAFe 6.0. SAFe предоставляет полную структуру: от команды до портфеля.

Уровни SAFe

SAFe организован в несколько уровней (конфигураций):

Essential SAFe (минимальная конфигурация): - **Team Level:** Agile-команды работают по Scrum или Kanban - **ART (Agile Release Train):** 5-12 команд (50-125 человек), синхронизированных в единый поток поставки - **PI (Program Increment):** 8-12 недель (обычно 5 спринтов по 2 недели + Innovation & Planning sprint)

Large Solution SAFe: - Для продуктов, требующих нескольких ART (более 125 человек) - Solution Train координирует несколько ART

Portfolio SAFe: - Стратегический уровень: управление портфелем продуктов - Lean Portfolio Management, Value Streams, Strategic Themes

Ключевые элементы SAFe

Agile Release Train (ART): Виртуальная организация из 5-12 команд, работающих над одним Value Stream. У ART есть: - **Release Train Engineer (RTE)** – “Chief Scrum Master” для всего ART - **Product Management** – приоритизация на уровне ART - **System Architect** – техническая целостность

PI Planning (Program Increment Planning): Двухдневное мероприятие, где все команды ART собираются вместе и планируют следующий PI (8-12 недель). Это “сердцебиение” SAFe.

Агенда PI Planning: - День 1: бизнес-контекст, видение продукта, драфт планов команд - День 2: финализация планов, определение зависимостей, голосование за уверенность (Confidence Vote)

Confidence Vote: Каждый участник показывает от 1 до 5 пальцев. Если средняя оценка < 3, планы пересматриваются.

Плюсы и минусы SAFe

Плюсы	Минусы
Полная, детальная система	Бюрократичность, много ролей и артефактов
Хорошо работает в крупных организациях	Сложность внедрения (12-18 месяцев)
PI Planning создаёт выравнивание	Может стать “Waterfall в Agile-обёртке”
Богатая база знаний и сертификации	Дорогие сертификации и консалтинг
Поддержка портфельного уровня	Может задавить культуру самоорганизации

LeSS (Large-Scale Scrum)

Обзор

LeSS (Крэг Ларман и Бас Водде) – минималистичный подход: “Scrum, но больше”. Вместо добавления новых ролей и процессов LeSS убирает лишнее и расширяет стандартный Scrum на несколько команд.

Конфигурации LeSS

LeSS (2-8 команд): - Один Product Owner, один Product Backlog - Общий Sprint Review и общая ретроспектива - Каждая команда проводит свои Sprint Planning и Daily Scrum - Общий Definition of Done

LeSS Huge (8+ команд): - Добавляются Area Product Owners для каждой области продукта - Requirement Areas разделяют бэклог на тематические области - Сохраняется единый Product Backlog и общий Product Owner

Ключевые принципы LeSS

1. **Больше с меньшим (More with LeSS).** Не добавляйте процессы – упрощайте.
2. **Lean Thinking.** Устраняйте потери (waste): ненужные митинги, передачи, ожидание.
3. **Системное мышление.** Оптимизируйте целое, а не части.
4. **Эмпирический контроль.** Прозрачность, инспекция, адаптация – как в базовом Scrum.
5. **Фокус на клиенте.** Каждая команда понимает конечного пользователя.

Плюсы и минусы LeSS

Плюсы	Минусы
Простота, минимум дополнительных ролей	Требует сильного Product Owner
Сохраняет дух Scrum	Один PO на 2-8 команд – узкое горлышко
Кросс-функциональные feature-команды	Требует серьёзной реструктуризации организации
Фокус на техническом совершенстве	Мало поддержки на портфельном уровне

Nexus (Scrum.org)

Обзор

Nexus (Кен Швабер и Scrum.org) – фреймворк для масштабирования Scrum на 3-9 команд, работающих над одним продуктом. Nexus максимально близок к оригинальному Scrum и добавляет минимум новых элементов.

Структура Nexus

Nexus Integration Team (NIT): Команда, ответственная за координацию интеграции между Scrum-командами. Включает: - Product Owner (общий для всех команд) - Scrum Master (может быть общим или отдельным для NIT) - Один или несколько членов Scrum-команд

NIT не делает работу за команды – она обеспечивает, что работа команд интегрируется.

События Nexus:

Событие Nexus	Назначение	Timebox
Nexus Sprint Planning	Координация планов между командами	4 часа
Nexus Daily Scrum	Синхронизация между командами (представители)	15 минут
Nexus Sprint Review	Общий обзор интегрированного инкремента	2 часа
Nexus Sprint Retrospective	Общая ретроспектива + ретро каждой команды	3 часа

Артефакт: Nexus Sprint Backlog Объединённый Sprint Backlog, показывающий зависимости между командами.

Плюсы и минусы Nexus

Плюсы	Минусы
Максимальная близость к Scrum	Ограничен 3-9 командами
Минимум новых ролей (только NIT)	Нет портфельного уровня
Фокус на интеграции	Менее детальный, чем SAFe
Бесплатный Nexus Guide	Меньше community и ресурсов

Сравнение фреймворков

Критерий	SAFe	LeSS	Nexus
Масштаб	50-10000+	10-3000+	15-80
Количество команд	Без ограничений	2-8 (LeSS) / 8+ (LeSS Huge)	3-9
Сложность внедрения	Высокая	Средняя	Низкая
Дополнительные роли	RTE, Product Management, System Architect	Area PO (только LeSS Huge)	Nexus Integration Team
Портфельный уровень	Да	Нет	Нет
Философия	Детальный фреймворк	Минималистичный Scrum	Расширение Scrum
Лучше подходит для	Крупные организации, регулируемые отрасли	Организации, готовые к радикальным изменениям	3-9 команд, один продукт

Как выбрать фреймворк

Выбирайте SAFe, если: - Организация крупная (500+ разработчиков) - Нужна поддержка на уровне портфеля - Менеджмент хочет детальную систему с метриками - Регулируемая отрасль (банки, госсектор, авиация)

Выбирайте LeSS, если: - Организация готова к серьёзной реструктуризации - Хотите сохранить дух Scrum в масштабе - Один продукт, несколько команд - Сильный Product Owner с видением продукта

Выбирайте Nexus, если: - 3-9 команд над одним продуктом - Хотите минимальных изменений к существующему Scrum - Главная проблема – интеграция работы между командами - Нет необходимости в портфельном управлении

Что дальше

В уроке 2.10 мы проведём практический проект: запустим Scrum-команду с нуля. Вы создадите Product Backlog, проведёте Sprint Planning, смоделируете спринт и Sprint Review.

Урок 2.9 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Урок 2.10: Проект: запуск Scrum-команды

Блок 2: Agile, Scrum и Kanban Длительность видео: 8-10 минут **Тип:** Практический проект

Введение

Это финальный урок Блока 2. Все предыдущие уроки дали вам теоретическую базу: Agile Manifesto, Scrum, Kanban, User Stories, Story Points, Velocity, масштабирование. Теперь пора применить всё на практике. В этом уроке мы пройдём полный цикл запуска Scrum-команды: от формирования команды и создания Product Backlog до проведения Sprint Planning, симуляции спринта и Sprint Review.

Вы можете выполнить этот проект индивидуально (симулируя роли) или в группе (идеально 3-5 человек). Результатом будет готовый набор артефактов: Product Backlog, Sprint Backlog, Sprint Board, Definition of Done и протокол Sprint Review.

Шаг 1: Выбор продукта и формирование команды

Выбор продукта

Для проекта выберите один из сценариев или придумайте свой:

Сценарий	Описание	Целевая аудитория
A: Мобильное приложение для доставки еды	MVP: каталог ресторанов, корзина, оформление заказа	Горожане 20-45 лет
B: Платформа онлайн-обучения	MVP: каталог курсов, просмотр уроков, тесты	Студенты и специалисты
C: Система управления задачами	MVP: проекты, задачи, доска (Kanban), уведомления	Небольшие команды (3-15 чел.)
D: Маркетплейс услуг	MVP: каталог специалистов, бронирование, отзывы	Заказчики и фрилансеры

Распределение ролей

Роль	Обязанности в проекте	Кто выполняет
Product Owner	Создание Product Backlog, приоритизация, Sprint Goal	Участник 1 (или вы)
Scrum Master	Фасилитация церемоний, контроль процесса	Участник 2 (или вы)

Роль	Обязанности в проекте	Кто выполняет
Developer 1	Оценка задач, декомпозиция, Sprint Board	Участник 3 (или вы)
Developer 2	Оценка задач, декомпозиция, Sprint Board	Участник 4 (или вы)
Developer 3	Оценка задач, декомпозиция, Sprint Board	Участник 5 (или вы)

При индивидуальном выполнении вы “надеваете” каждую роль по очереди.

Шаг 2: Создание Product Backlog

Задание

Создайте Product Backlog из 15-20 User Stories для выбранного продукта.

Шаблон User Story

Для каждой User Story заполните:

Поле	Описание
ID	Уникальный номер (US-001, US-002...)
User Story	Как [роль], я хочу [действие], чтобы [ценность]
Acceptance Criteria	3-5 критериев приёмки
Priority	Must / Should / Could / Won’t (MoSCoW)
Story Points	Оценка по шкале Фибоначчи (1, 2, 3, 5, 8, 13)

Пример Product Backlog (Сценарий B: онлайн-обучение)

ID	User Story	Priority	SP
US-001	Как студент, я хочу зарегистрироваться по email, чтобы получить доступ к платформе	Must	5
US-002	Как студент, я хочу войти в аккаунт по email и паролю	Must	3
US-003	Как студент, я хочу просматривать каталог курсов с описаниями и рейтингами	Must	5
US-004	Как студент, я хочу смотреть видеоуроки внутри платформы	Must	8
US-005	Как студент, я хочу отмечать уроки как пройденные и видеть прогресс	Must	5
US-006	Как студент, я хочу проходить тесты после каждого урока	Should	8
US-007	Как преподаватель, я хочу создавать курсы и добавлять уроки	Must	13
US-008	Как преподаватель, я хочу видеть статистику по студентам	Should	5

ID	User Story	Priority	SP
US-009	Как студент, я хочу оставлять отзывы о курсах	Should	3
US-010	Как студент, я хочу получать сертификат по завершении курса	Could	5
US-011	Как студент, я хочу искать курсы по категориям и ключевым словам	Must	5
US-012	Как администратор, я хочу модерировать контент и пользователей	Should	8
US-013	Как студент, я хочу сохранять курсы в избранное	Could	2
US-014	Как студент, я хочу получать уведомления о новых уроках	Could	5
US-015	Как студент, я хочу общаться с преподавателем в чате	Could	13
Итого			**93 SP**

Оценка через Planning Poker

Проведите оценку каждой User Story: 1. PO описывает историю и критерии приёмки 2. Developers задают вопросы 3. Каждый показывает карту (или записывает число) 4. При расхождении – обсуждение и повторная оценка 5. Фиксация консенсусной оценки

Шаг 3: Definition of Done

Задание

Составьте Definition of Done для вашей команды.

Шаблон DoD

Definition of Done (v1.0): - [ ] Код написан и соответствует стандартам кодирования команды - [ ] Code Review пройден (минимум 1 approve) - [ ] Юнит-тесты написаны и проходят - [ ] Acceptance Criteria выполнены - [ ] Функционал задеплоен на staging-среду - [ ] Нет критических и major-багов - [ ] Документация обновлена (API, README) - [ ] Product Owner провёл приёмку

Обсудите DoD в команде и адаптируйте под свой контекст.

Шаг 4: Sprint Planning

Задание

Проведите Sprint Planning для Спринта 1 (2 недели).

Тема 1: Sprint Goal

Сформулируйте Sprint Goal по формуле: “К концу спринта [кто] сможет [что], что позволит [ценность].”

Пример: “К концу спринта студент сможет зарегистрироваться, войти и просмотреть каталог курсов, что позволит начать бета-тестирование платформы.”

Тема 2: Выбор элементов

Определите Velocity для первого спринта (предположение). Для новой команды используйте правило: Velocity = 30-40% от теоретической ёмкости.

Пример: 3 разработчика * 10 дней * 6 часов продуктивной работы = 180 часов. 30-40% = 54-72 часа ~ 25-35 SP (грубая оценка для первого спринта).

Выберите User Stories для Sprint 1 (в порядке приоритета):

ID	User Story	SP	Включена
US-001	Регистрация по email	5	Да
US-002	Вход по email и паролю	3	Да
US-003	Каталог курсов	5	Да
US-004	Просмотр видеоуроков	8	Да
US-011	Поиск по курсам	5	Да
US-005	Прогресс по урокам	5	Да
Итого Sprint 1		**31 SP**	

Тема 3: Декомпозиция на задачи

Разбейте каждую User Story на конкретные задачи (4-8 часов каждая):

US-001: Регистрация по email (5 SP) - Создать модель User в базе данных (4 ч) - Разработать API регистрации (POST /auth/register) (4 ч) - Создать UI формы регистрации (4 ч) - Добавить валидацию email и пароля (frontend + backend) (4 ч) - Написать тесты (unit + integration) (4 ч)

Шаг 5: Sprint Board

Задание

Создайте Sprint Board в Jira, Trello, Notion или на физической доске.

Структура доски:

To Do	In Progress	Code Review	Testing	Done
US-001: Модель User				
US-001: API регистрации				
US-001: UI формы				
US-001: Валидация				
US-001: Тесты				
US-002: API входа				
...				

WIP-лимиты: - In Progress: 4 - Code Review: 3 - Testing: 2

Шаг 6: Симуляция спринта

Daily Scrum (симуляция)

Проведите 2-3 “Daily Scrum” (можно за один подход, симулируя дни 1, 5 и 9 спринта).

Для каждого "дня" переместите задачи на доске и обсудите: - Что было сделано? - Есть ли блокеры? - Идём ли мы к Sprint Goal?

Burndown Chart (симуляция)

Заполните Burndown Chart на основе симуляции:

День	Оставшиеся SP (план)	Оставшиеся SP (факт)
1	28	31
2	25	29
3	22	26
4	19	23
5	16	18
6	13	15
7	9	12
8	6	8
9	3	4
10	0	0

Шаг 7: Sprint Review

Задание

Проведите Sprint Review (15-20 минут в симуляции).

Агенда Sprint Review:

1. **Sprint Goal** – напомнить цель спринта (2 мин)
2. **Что сделано / не сделано** – список завершённых User Stories (3 мин)
3. **Демо** – показать результат (5-10 мин)
4. **Обратная связь** – что бы изменили стейкхолдеры? (5 мин)
5. **Обновление Product Backlog** – новые элементы, изменение приоритетов (3 мин)

Протокол Sprint Review

Заполните протокол:

Параметр	Значение
Sprint	Sprint 1
Sprint Goal	Студент может зарегистрироваться, войти и просмотреть каталог курсов
Запланировано	31 SP (6 User Stories)
Завершено	31 SP (6 User Stories)
Velocity	31
Новые элементы бэклога	US-016: Фильтрация курсов по длительности (3 SP)
Изменения приоритетов	US-006 (тесты) повышена до Must

Шаг 8: Sprint Retrospective

Задание

Проведите ретроспективу в формате Start / Stop / Continue.

Start	Stop	Continue
Code Review в тот же день	Откладывать тестирование на конец спринта	Ежедневные стендапы
Автоматизировать деплой на staging	Работать над 3+ задачами параллельно	Парное программирование на сложных задачах

Action Items:

Action Item	Ответственный	Дедлайн
Настроить автоматический деплой (CI/CD)	Developer 1	Sprint 2, день 3
Ввести правило: Code Review в течение 4 часов	Scrum Master	Sprint 2, день 1

Чек-лист сдачи проекта

Убедитесь, что вы создали все артефакты:

- [ ] Product Backlog (15-20 User Stories с Acceptance Criteria, приоритетами и Story Points)
- [ ] Definition of Done (8+ критериев)
- [ ] Sprint Goal для Sprint 1
- [ ] Sprint Backlog (5-8 User Stories, декомпозированные на задачи)
- [ ] Sprint Board (To Do / In Progress / Code Review / Testing / Done)
- [ ] Burndown Chart (заполненный по дням спринта)
- [ ] Протокол Sprint Review
- [ ] Результаты Retrospective (Start/Stop/Continue + Action Items)

Заключение блока

Поздравляю! Вы прошли Блок 2 полностью. Теперь вы знаете Agile Manifesto, умеете работать по Scrum и Kanban, писать User Stories, оценивать задачи в Story Points и прогнозировать сроки с помощью Velocity. В Блоке 3 мы перейдём к инструментам управления проектами: Jira, Asana, Notion и другие системы, которые автоматизируют всё, что мы изучили.

Урок 2.10 из 10 | Блок 2: Agile, Scrum и Kanban | Курс: Управление проектами

Блок 3: Инструменты и лидерство

Урок 3.1: Jira – настройка и работа с проектами

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Jira от Atlassian – это стандарт индустрии для управления проектами в IT. По данным Atlassian, более 75 000 компаний используют Jira: от стартапов до корпораций уровня NASA, Spotify, Airbnb. Если вы работаете в IT-компании, вероятность встретить Jira – около 65% (по данным State of Agile Report).

Но парадокс в том, что большинство команд используют лишь 20-30% возможностей Jira, при этом тратя часы на ручные операции, которые можно автоматизировать за 5 минут. В этом уроке мы пройдём от создания проекта до продвинутых workflow, автоматизации и dashboards. Вы научитесь настраивать Jira так, чтобы она работала НА вас, а не вы на неё.

Типы проектов в Jira

Jira Cloud предлагает два класса проектов:

Характеристика	Team-managed (Next-gen)	Company-managed (Classic)
Кто настраивает	Любой участник команды	Jira-администратор
Сложность	Простая	Гибкая, но сложная
Workflow	Упрощённый	Полностью кастомный
Подходит для	Небольших команд (3-10 чел.)	Крупных организаций, сложных процессов
Интеграции	Базовые	Полные (500+ приложений)
Рекомендация	Старт, обучение, пилотные проекты	Продакшн, масштабирование

Шаблоны проектов

- **Scrum Board** – для спринтов: бэклог, спринты, доска, Burndown Chart
- **Kanban Board** – для потока: непрерывная работа, WIP-лимиты, Cumulative Flow Diagram
- **Bug Tracking** – для QA: баги, приоритеты, severity, версии
- **Project Management** – для не-IT: задачи, дедлайны, диаграмма Ганта (с Jira Plans)

Создание и настройка проекта: пошагово

Шаг 1: Создание проекта

1. Нажмите “Projects” -> “Create Project”
2. Выберите шаблон (Scrum для этого урока)
3. Укажите: название, ключ (KEY – префикс задач, например PM-001), тип (Team-managed)
4. Пригласите участников команды

Шаг 2: Настройка типов задач (Issue Types)

Тип задачи	Назначение	Пример
Epic	Крупная функциональность (1-3 месяца)	“Система оплаты”
Story	Пользовательская история (1-2 недели)	“Как покупатель, я хочу оплатить заказ картой”
Task	Техническая задача	“Настроить интеграцию с платёжным шлюзом”
Bug	Дефект	“Ошибка 500 при оплате на мобильном”
Sub-task	Подзадача к Story/Task	“Написать API-эндпоинт /payments”

Шаг 3: Настройка полей задачи

Обязательные поля для каждой задачи: - **Summary** – краткое название - **Description** – детальное описание (используйте шаблон: Контекст / Задача / Acceptance Criteria) - **Priority** – Highest, High, Medium, Low, Lowest - **Assignee** – ответственный - **Story Points** – оценка сложности (для Stories) - **Sprint** – к какому спринту привязана - **Epic Link** – к какому Epic принадлежит - **Labels** – теги для фильтрации (frontend, backend, design, devops)

Workflow: настройка жизненного цикла задачи

Workflow определяет, через какие статусы проходит задача и кто может менять статус.

Стандартный Scrum Workflow

```
To Do -> In Progress -> In Review -> Testing -> Done
```

Расширенный Workflow (рекомендация)

```
Backlog -> Selected for Sprint -> In Progress -> Code Review -> QA Testing -> UAT -> Done
```

Статус	Кто работает	Что происходит
Backlog	PO	Задача в бэклоге, ждёт приоритизации
Selected for Sprint	SM	Задача выбрана для текущего спринта
In Progress	Developer	Разработчик активно работает
Code Review	Developer (другой)	Pull Request на ревью
QA Testing	QA Engineer	Тестирование на staging
UAT	PO / Stakeholder	Приёмочное тестирование
Done	–	Задача закрыта, соответствует DoD

Правила перехода (Transitions)

- Из “In Progress” в “Code Review” можно перейти только если заполнено поле “Pull Request URL”
- Из “QA Testing” нельзя вернуть в “In Progress” без комментария (описание бага)
- В “Done” может перевести только QA или PO

Доска (Board): настройка визуализации

Колонки доски

Настройте колонки в соответствии с Workflow. Каждая колонка = один статус.

Swimlanes (дорожки)

Используйте swimlanes для группировки: - **По Epic** – видно прогресс каждой большой функциональности - **По Assignee** – видна загрузка каждого разработчика - **По Priority** – критичные задачи сверху

WIP-лимиты

Обязательно установите WIP-лимиты для Kanban-досок: - In Progress: 2 * количество разработчиков - Code Review: количество разработчиков - QA Testing: 3-5

Фильтры и JQL (Jira Query Language)

JQL – это язык запросов Jira. Освоение JQL – это суперсила PM.

Базовые запросы

Запрос	Описание
`project = PM AND status = "In Progress"`	Все задачи в работе
`assignee = currentUser() AND sprint in openSprints()`	Мои задачи в текущем спринте
`priority = Highest AND status != Done`	Критичные незакрытые задачи
`created >= -7d AND project = PM`	Задачи, созданные за последнюю неделю
`updated <= -14d AND status != Done`	Задачи без обновлений 2+ недели (“зависшие”)

Продвинутые запросы

Запрос	Описание
`sprint in openSprints() AND remainingEstimate > 0 AND status = "In Progress"`	Задачи с оставшимся временем
`issuetype = Bug AND priority in (Highest, High) AND fixVersion = "2.0"`	Критичные баги для релиза 2.0
`labels in (frontend) AND sprint in openSprints() ORDER BY priority DESC`	Frontend-задачи текущего спринта по приоритету

Сохраняйте фильтры для повторного использования и создания dashboards.

Dashboards: информационные панели

Dashboard – это набор виджетов (gadgets), показывающих состояние проекта.

Рекомендуемые гаджеты для PM Dashboard

Гаджет	Что показывает	Для кого
Sprint Burndown	Прогресс текущего спринта	SM, команда
Velocity Chart	Velocity за последние 5-8 спринтов	PO, SM
Pie Chart (по статусам)	Распределение задач по статусам	PM
Two-Dimensional Filter (Assignee x Status)	Загрузка каждого участника	SM
Created vs Resolved	Создано vs закрыто задач за период	PM, руководство
Filter Results	Список критичных незакрытых задач	PM, PO

Автоматизация в Jira

Jira Automation – встроенный инструмент для автоматизации рутинных действий.

Топ-5 автоматизаций для PM

1. **Автоприсвоение**: Когда задача переходит в “Code Review” -> автоматически назначить на лида команды
2. **Уведомление о блокерах**: Когда задача получает флаг “Blocked” -> отправить сообщение в Slack-канал команды

3. **Закрытие Sub-tasks**: Когда все sub-tasks задачи переведены в Done -> перевести родительскую задачу в Done
4. **Напоминание о зависших задачах**: Если задача в “In Progress” более 3 дней -> отправить напоминание assignee
5. **Автосоздание задачи на деплой**: Когда Sprint переведён в “Complete” -> создать задачу “Deploy Sprint N to Production”

Формат правила автоматизации

```
Триггер (WHEN) -> Условие (IF) -> Действие (THEN)
```

Пример: WHEN issue transitions to “Done” -> IF issue type = “Story” -> THEN send Slack message “Story {summary} завершена! :tada:”

Интеграции Jira

Инструмент	Интеграция	Ценность
Confluence	Связь задач с документацией	Спецификации, протоколы встреч
Bitbucket / GitHub	Связь коммитов и PR с задачами	Трейсабельность кода
Slack	Уведомления о статусах	Быстрая коммуникация
Figma	Связь макетов с задачами	Дизайн-спецификации
TestRail	Связь тест-кейсов с Stories	QA-покрытие

Что дальше

В уроке 3.2 мы разберём Notion – инструмент “всё-в-одном”, который совмещает документацию, базы данных, Kanban-доски и вики в едином пространстве. Вы узнаете, как использовать Notion для управления проектами и когда он лучше Jira.

Урок 3.1 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.2: Notion – всё-в-одном для управления проектами

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Notion – это инструмент, который объединяет документацию, базы данных, Kanban-доски, вики и трекер задач в одном пространстве. По данным Notion, их платформой пользуются более 30 миллионов человек и команды таких компаний как Figma, Pixar, Samsung. Notion стал феноменом, потому что решает главную проблему: разрозненность информации. Вместо 5 инструментов (Google Docs + Trello + Confluence + Sheets + Slack pinned messages) – один Notion.

Но Notion – это конструктор. Без правильной архитектуры он превращается в хаос из вложенных страниц, которые никто не может найти. В этом уроке мы построим систему управления проектами в Notion с нуля: от структуры workspace до связанных баз данных, шаблонов и формул.

Архитектура Notion Workspace

Иерархия контента

```
Workspace (команда/компания)
  └── Teamspace (отдел/проект)
       └── Page (страница)
            └── Database (база данных)
                 └── Database Item (запись = страница)
                      └── Блоки (текст, таблицы, embed...)
```

Рекомендуемая структура для PM

Раздел	Содержание	Тип
Home	Дашборд с ключевыми метриками и ссылками	Страница
Projects	Все проекты (база данных)	Database
Tasks	Все задачи (база данных, связана с Projects)	Database
Sprints	Спринты (база данных, связана с Tasks)	Database
Team Wiki	Онбординг, процессы, стандарты, DoD	Страница с вложенными
Meeting Notes	Протоколы встреч (база данных)	Database
Retrospectives	Ретроспективы по спринтам	Database

Базы данных: основа системы

База данных в Notion – это не просто таблица. Это коллекция страниц с настраиваемыми свойствами (полями), которую можно отображать в разных представлениях (views).

Свойства базы данных Tasks

Свойство	Тип	Назначение
Task Name	Title	Название задачи
Status	Select	To Do / In Progress / Review / Done
Priority	Select	Highest / High / Medium / Low
Assignee	Person	Ответственный
Due Date	Date	Дедлайн
Project	Relation	Связь с базой Projects
Sprint	Relation	Связь с базой Sprints
Story Points	Number	Оценка сложности
Tags	Multi-select	frontend, backend, design, bug, feature
Created	Created time	Автоматическая дата создания
Progress	Formula	Автоматический расчёт (на основе sub-tasks)

Представления (Views)

Одну базу данных можно отображать в разных форматах:

Представление	Формат	Применение
Board View	Kanban-доска	Ежедневная работа: перетаскивание задач
Table View	Таблица	Массовое редактирование, экспорт

Представление	Формат	Применение
Calendar View	Календарь	Планирование по дедлайнам
Timeline View	Ганта-подобная диаграмма	Зависимости и сроки
List View	Список	Быстрый обзор
Gallery View	Карточки	Контент-план, дизайн-задачи

Фильтры и сортировка

Каждое представление можно фильтровать и сортировать. Примеры полезных представлений:

- **My Tasks** – фильтр: Assignee = Me, Status != Done. Сортировка: Priority DESC, Due Date ASC
- **Sprint Board** – фильтр: Sprint = Current Sprint. Группировка: Status. Вид: Board
- **Overdue Tasks** – фильтр: Due Date < Today, Status != Done. Сортировка: Due Date ASC
- **Backlog** – фильтр: Sprint = empty, Status = To Do. Сортировка: Priority DESC

Связанные базы данных (Relations & Rollups)

Relation – связь между базами

Связь Tasks -> Projects означает: каждая задача принадлежит проекту. Это как foreign key в SQL.

Rollup – агрегация по связи

Rollup позволяет вычислять агрегаты по связанным записям:

База	Rollup	Формула	Результат
Projects	Total Tasks	Count all (Tasks)	45
Projects	Completed Tasks	Count where Status = Done	32
Projects	Completion %	Percent where Status = Done	71%
Projects	Total Story Points	Sum of Story Points (Tasks)	120
Sprints	Sprint Velocity	Sum of SP where Status = Done	34

Формулы в Notion

Notion поддерживает формулы для вычисляемых полей:

- **Days Until Deadline**: `dateBetween(prop("Due Date"), now(), "days")`
- **Is Overdue**: `if(prop("Due Date") < now() and prop("Status") != "Done", true, false)`
- **Priority Score**: `if(prop("Priority") == "Highest", 4, if(prop("Priority") == "High", 3, if(prop("Priority") == "Medium", 2, 1)))`

Шаблоны: ускорение создания контента

Database Templates

Создайте шаблоны для повторяющихся типов записей:

Шаблон: User Story

```
## User Story
Как [роль], я хочу [действие], чтобы [ценность].
```

```
## Acceptance Criteria
- [ ] Критерий 1
- [ ] Критерий 2
- [ ] Критерий 3

## Technical Notes
[Технические детали для разработчика]

## Design
[Ссылка на макет в Figma]
```

Шаблон: Meeting Notes

```
## Участники
@Имя1, @Имя2, @Имя3

## Повестка
1. Тема 1
2. Тема 2

## Решения
- Решение 1 — ответственный: @Имя — дедлайн: /дата

## Action Items
- [ ] Задача 1 — @Имя — дедлайн
- [ ] Задача 2 — @Имя — дедлайн

## Следующая встреча
Дата: /дата | Время: | Формат:
```

Шаблон: Sprint Retrospective

```
## Sprint N | Дата

### Что прошло хорошо
- ...

### Что можно улучшить
- ...

### Action Items
| Действие | Ответственный | Дедлайн | Статус |
|---|---|---|---|
| | | | |
```

PM Dashboard в Notion

Создайте главную страницу – дашборд проектного менеджера:

Структура дашборда

1. **Заголовок**: Команда / Проект / Текущий спринт
2. **Метрики** (callout-блоки): Задач в спринте | Выполнено | Velocity | Дней до конца спринта
3. **Sprint Board** (linked database, Board view, фильтр: текущий спринт)
4. **My Tasks Today** (linked database, List view, фильтр: Assignee = Me, Due Date = Today)
5. **Overdue Tasks** (linked database, Table view, фильтр: Due Date < Today, Status != Done)
6. **Quick Links**: Confluence, Figma, Slack-канал, Git-репозиторий

Linked Database vs Original Database

Linked Database – это "представление" (view) существующей базы данных на другой странице. Данные не дублируются. Изменения в linked database отражаются в оригинале и наоборот. Используйте linked databases на дашбордах.

Notion AI и автоматизация

Notion AI

- Генерация summary из длинных заметок встреч
- Автозаполнение описаний задач
- Перевод контента для мультиязычных команд
- Извлечение action items из протоколов

Автоматизация (Notion Automations)

Триггер	Действие	Пример
Status changed to "Done"	Update property	Установить "Completed Date" = Today
New page added	Send notification	Уведомить PM о новой задаче
Due Date = Today	Send notification	Напомнить assignee о дедлайне
Status changed to "In Progress"	Update property	Установить "Start Date" = Today

---

Notion vs Jira: когда что использовать

Критерий	Notion	Jira
Документация	Отлично	Через Confluence (отдельный продукт)
Kanban/Scrum	Хорошо	Отлично
JQL-подобные запросы	Фильтры (проще)	JQL (мощнее)
Workflow	Базовый (Status)	Полный (transitions, validators, conditions)
Автоматизация	Базовая	Мощная (Jira Automation)
Отчётность	Формулы, Rollups	Dashboards, гаджеты, Velocity Chart
Стоимость	Free (до 10 чел.) / $10/user	Free (до 10 чел.) / $8.15/user
Лучше для	Небольших команд, стартапов, контент-проектов	IT-команд, Scrum/Kanban, enterprise

Рекомендация: Notion для документации + вики + лёгкого трекинга. Jira для полноценного Scrum/Kanban с автоматизацией workflow. Многие команды используют оба инструмента.

Что дальше

В уроке 3.3 мы сравним Trello, Asana и ClickUp – три популярные альтернативы. Вы узнаете, чем они отличаются, и научитесь выбирать инструмент под конкретный тип проекта и размер команды.

Урок 3.2 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.3: Trello, Asana, ClickUp – выбор инструмента

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

На рынке десятки инструментов для управления проектами, и каждый год появляются новые. Но топ-3 среди “универсальных” (не считая Jira и Notion, которые мы разобрали) – это Trello, Asana и ClickUp. У каждого – своя философия, свои сильные стороны и свои ограничения.

Ошибка, которую допускают 90% команд: выбирают инструмент по рекомендации знакомых или по красивому лендингу, а через 3 месяца мигрируют на другой, теряя данные и время. Правильный подход – выбирать инструмент на основе критериев: тип проекта, размер команды, бюджет, сложность процессов.

В этом уроке мы разберём Trello, Asana и ClickUp по 10 критериям, дадим матрицу выбора и покажем реальные сценарии использования.

Trello: простота и визуальность

Философия

Trello (Atlassian, 2011) построен вокруг одной метафоры: Kanban-доска с карточками. Минимальный порог входа – можно начать работать через 2 минуты без обучения.

Ключевые возможности

Возможность	Описание	
Доски (Boards)	Kanban-доска с колонками	
Карточки (Cards)	Задачи с описанием, чеклистами, вложениями, дедлайнами	
Power-Ups	Плагины: Calendar, Gantt, Custom Fields, Voting	
Butler	Встроенная автоматизация (правила, кнопки, расписание)	
Шаблоны	100+ готовых шаблонов досок	

Сильные стороны

- Минимальный порог входа – обучение за 5 минут
- Визуальность – drag-and-drop, интуитивный интерфейс
- Бесплатный план – до 10 досок, неограниченное количество карточек
- Butler автоматизация – мощная, без кода
- Идеален для: маркетинг, контент, HR-процессы, личная продуктивность

Ограничения

- Нет встроенных Story Points, Velocity, Burndown
- Нет Timeline/Gantt без Power-Up
- Ограниченная отчётность
- Плоская структура – нет иерархии задач (epic -> story -> sub-task)
- Не масштабируется для 20+ человек с комплексными процессами

Asana: структура и масштаб

Философия

Asana (2012, основана со-основателем Facebook) фокусируется на “work management” – управлении работой на уровне организации. Если Trello – это доска, то Asana – это операционная система работы.

Ключевые возможности

Возможность	Описание	
Projects	Списки, доски, Timeline, календарь	
Tasks & Subtasks	Иерархия задач, зависимости	
Portfolios	Обзор нескольких проектов в одном месте	
Workload	Управление загрузкой команды	
Goals	OKR-трекинг на уровне компании	
Rules	Автоматизация (триггеры + действия)	
Forms	Приём запросов от внешних и внутренних клиентов	

Сильные стороны

- Множество представлений одного проекта: List, Board, Timeline, Calendar
- Portfolios – портфельное управление (обзор всех проектов с прогрессом)
- Workload – визуализация загрузки каждого участника
- Goals & Milestones – стратегическое планирование
- Forms – приём запросов (баги, запросы от отделов)
- Масштабируется до 100+ человек

Ограничения

- Отсутствие встроенного Scrum (нет спринтов, Velocity, Burndown)
- Timeline и Portfolios только в платных планах ($10.99+/user/month)
- Интерфейс может быть перегружен для маленьких команд
- Нет встроенной вики/документации (нужен отдельный инструмент)

ClickUp: всё и сразу

Философия

ClickUp (2017) позиционирует себя как “one app to replace them all” – замена всех инструментов. Максимальная функциональность: задачи, документы, цели, whiteboards, чат, time tracking – всё в одном.

Ключевые возможности

Возможность	Описание	
Spaces / Folders / Lists	Трёхуровневая иерархия организации	
Views (15+)	Board, List, Table, Gantt, Timeline, Calendar, Mind Map, Workload...	
Docs	Встроенные документы (как Notion)	
Goals	OKR-трекинг	
Whiteboards	Виртуальные доски	
Time Tracking	Встроенный трекер времени	

Возможность	Описание	
Sprints	Встроенный Scrum с Velocity и Burndown	
Automations	100+ шаблонов автоматизаций	
AI	ClickUp Brain – AI-ассистент	

Сильные стороны

- Максимальная функциональность в одном инструменте
- 15+ представлений – больше, чем у любого конкурента
- Встроенные спринты с Velocity и Burndown (чего нет у Trello и Asana)
- Встроенные документы (не нужен Confluence/Notion)
- Щедрый бесплатный план
- Активное развитие – новые функции каждый месяц

Ограничения

- Сложность – кривая обучения круче, чем у Trello и Asana
- Производительность – интерфейс может тормозить на больших проектах
- “Feature bloat” – перегруженность функциями, которые не все нужны
- Частые обновления интерфейса – нужно регулярно переучиваться
- Менее стабилен, чем Jira или Asana (более молодой продукт)

Сравнительная матрица

Критерий	Trello	Asana	ClickUp	Jira	Notion
Простота	5/5	3/5	2/5	2/5	3/5
Kanban	5/5	4/5	5/5	5/5	4/5
Scrum (спринты, Velocity)	1/5	2/5	4/5	5/5	2/5
Документация/вики	1/5	1/5	4/5	1/5 (Confluence)	5/5
Портфельное управление	1/5	5/5	4/5	4/5 (Plans)	3/5
Автоматизация	3/5	4/5	4/5	5/5	2/5
Отчётность	2/5	4/5	4/5	5/5	3/5
Бесплатный план	4/5	3/5	5/5	4/5	4/5
Масштабируемость	2/5	5/5	3/5	5/5	3/5
Кривая обучения	1 день	3-5 дней	5-7 дней	7-14 дней	3-5 дней

Матрица выбора: какой инструмент для какого случая

Сценарий	Рекомендация	Почему
Фрилансер или команда 1-3 человека	Trello	Простота, бесплатный план, мгновенный старт
Маркетинговая команда (5-15 чел.)	Asana	Portfolios, Timeline, Forms, масштабируемость
IT-стартап (5-15 чел., Scrum)	ClickUp или Jira	Спринты, Velocity, Burndown

Сценарий	Рекомендация	Почему
IT-компания (20+ чел., enterprise)	Jira + Confluence	Стандарт индустрии, глубокие workflow, JQL
Контент-проект / Документация	Notion	Всё-в-одном: docs + tasks + wiki
Мультипроектное управление	Asana	Portfolios + Goals + Workload
Максимальный функционал, один инструмент	ClickUp	15+ views, docs, goals, time tracking
Не-IT команда (HR, Sales, Operations)	Trello или Asana	Низкий порог входа, нет лишних IT-фич

Стоимость (2024-2025)

Инструмент	Бесплатный	Базовый платный	Business/Premium
Trello	10 досок, без ограничений по картам	$5/user/month	$10/user/month
Asana	15 чел., ограниченные views	$10.99/user/month	$24.99/user/month
ClickUp	Без ограничений по задачам	$7/user/month	$12/user/month
Jira	10 users, базовый Scrum/Kanban	$8.15/user/month	$16/user/month
Notion	10 чел., базовые блоки	$10/user/month	$18/user/month

Как мигрировать между инструментами

Если вы уже используете один инструмент и хотите перейти на другой:

1. **Экспорт**: большинство инструментов поддерживают экспорт в CSV
2. **Импорт**: Asana, ClickUp, Notion имеют встроенные импортёры из конкурентов
3. **Параллельный период**: запустите новый инструмент на 2-4 недели параллельно
4. **Обучение команды**: заложите 1-2 недели на онбординг
5. **Архив старого инструмента**: не удаляйте сразу, оставьте в read-only на 3 месяца

Что дальше

В уроке 3.4 мы перейдём от инструментов к коммуникации: как выстроить общение в команде через Slack, регулярные встречи и документацию, чтобы информация не терялась, а решения принимались быстро.

Урок 3.3 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.4: Коммуникация в команде – Slack, встречи, документация

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

По данным исследования PMI Pulse of the Profession, 29% проектов проваливаются из-за плохой коммуникации. Это причина номер один, опережающая нечёткие требования (13%) и нехватку ресурсов (10%). При этом средний сотрудник тратит 28% рабочего

времени на email и 19% на поиск информации (McKinsey).

Проблема не в отсутствии инструментов – Slack, Teams, Zoom, Google Meet, email, Confluence. Проблема в отсутствии системы. Когда информация размазана по 5 каналам, решения принимаются в личных сообщениях, а протоколы встреч никто не пишет – проект обречён на повторяющиеся вопросы, потерянные решения и "а мы же об этом договаривались".

В этом уроке мы построим коммуникационный стек проекта: от структуры Slack-каналов до формата встреч и системы документирования решений.

Коммуникационный план проекта

Первый документ, который PM создаёт на старте проекта (после Устава) – это коммуникационный план.

Шаблон коммуникационного плана

Тип коммуникации	Канал	Частота	Участники	Ответственный	Артефакт
Daily Standup	Slack huddle / Zoom	Ежедневно, 10:00	Dev-команда, SM	Scrum Master	–
Sprint Planning	Zoom + Miro	Каждые 2 недели	PO, SM, Developers	PO	Sprint Backlog
Sprint Review	Zoom (демо)	Каждые 2 недели	Вся команда + стейкхолдеры	PO	Протокол Review
Sprint Retro	Zoom + FigJam	Каждые 2 недели	Dev-команда, SM	SM	Action Items
Статус-отчёт	Email / Confluence	Еженедельно (пт)	PM -> руководство	PM	Status Report
1-on-1	Zoom / офлайн	2 раза в месяц	PM <-> каждый участник	PM	Заметки 1-on-1
Экстренная связь	Телефон / Telegram	По необходимости	PM <-> PO / SM	PM	–
Документация проекта	Confluence / Notion	Постоянно	Вся команда	PM	Wiki проекта

Slack: структура каналов

Slack (или аналоги: Microsoft Teams, Mattermost) – центр оперативной коммуникации.

Рекомендуемая структура каналов

Канал	Назначение	Кто участвует	Правила
#project-general	Общие объявления проекта	Вся команда	Только важное, без флуда
#project-dev	Технические обсуждения	Developers, SM	Код, архитектура, технические решения
#project-design	Дизайн-обсуждения	Designers, PO	Макеты, UI/UX-решения
#project-standup	Асинхронные стендапы	Dev-команда	Формат: Сделал / Планирую / Блокеры
#project-alerts	Автоматические уведомления	Все	Jira-триггеры, CI/CD, мониторинг
#project-random	Неформальное общение	Все	Мемы, поздравления, командный дух

Правила использования Slack

1. **Threads** – обсуждение в тредах, а не в основном канале

2. **@channel / @here** – только для критичных сообщений
3. **Реакции** – используйте эмодзи для подтверждения (прочитал, согласен, сделаю)
4. **Pins** – закрепляйте важные решения и ссылки
5. **DMs** – минимизируйте личные сообщения для проектных решений (решения должны быть видны команде)

Асинхронные стендапы

Для распределённых команд вместо 15-минутного созвона используйте бот (Standuply, Geekbot) в Slack:

Каждый день в 10:00 бот задаёт 3 вопроса: 1. Что ты сделал вчера? 2. Что планируешь сегодня? 3. Есть ли блокеры?

Ответы публикуются в #project-standup. SM просматривает и реагирует на блокеры.

Встречи: формат и регламент

Принципы эффективных встреч

Принцип	Описание	Антипаттерн
Повестка до встречи	Отправить за 24 часа до встречи	“Давайте обсудим” без конкретики
Таймбокс	Жёсткий лимит времени	Встреча на 30 минут, длится 1.5 часа
Протокол	Записать решения и action items	“Мы обсудили, но никто не записал”
Минимум участников	Только те, кто нужен	15 человек на встрече, говорят двое
Правило “Два ботинка”	Если встречу можно заменить Slack-сообщением – замените	Созвон ради созвона

Форматы встреч

Встреча	Длительность	Формат	Ключевой результат
Daily Standup	15 мин	Синхронный или асинхронный	Блокеры выявлены
Sprint Planning	2-4 часа	Zoom + Jira + Miro	Sprint Backlog
Sprint Review	1 час	Демо + обсуждение	Обратная связь стейкхолдеров
Sprint Retro	1-1.5 часа	FigJam / Miro	Action Items
Backlog Grooming	1-2 часа	PO + Developers	Уточнённые User Stories
Architecture Review	1-2 часа	Whiteboard / Miro	Технические решения
1-on-1	30 мин	Неформальный	Обратная связь, мотивация

Шаблон протокола встречи

```
Дата: ___
Участники: ___
Цель встречи: ___

Обсуждённые темы:
1. [Тема] — [Решение]
2. [Тема] — [Решение]

Action Items:
| Действие | Ответственный | Дедлайн |
|---|---|---|
| | | |
```

```
Следующая встреча: [дата, время]
```

Документация: система “Single Source of Truth”

Главный принцип документации – Single Source of Truth (SSOT): один источник истины. Каждый тип информации хранится в ОДНОМ месте.

Карта документации проекта

Тип документа	Где хранится	Кто обновляет	Частота
Устав проекта (Charter)	Confluence / Notion	PM	Один раз на старте
Product Backlog	Jira / ClickUp	PO	Постоянно
Sprint Backlog	Jira / ClickUp	SM + Developers	Каждый спринт
Техническая документация	Confluence / Notion / README	Developers	По мере разработки
API-документация	Swagger / Postman	Backend-разработчик	По мере разработки
Дизайн-макеты	Figma	Дизайнер	По мере дизайна
Протоколы встреч	Confluence / Notion	PM / SM	После каждой встречи
Статус-отчёты	Confluence / Email	PM	Еженедельно
Definition of Done	Confluence / Notion	SM + команда	Ревизия каждый квартал
Onboarding Guide	Confluence / Notion	PM	Обновление при изменениях

Принципы документации

1. **Пиши для будущего себя**: через 6 месяцев вы забудете контекст
2. **Структура > объём**: короткий структурированный документ лучше длинного хаотичного
3. **Шаблоны**: используйте шаблоны для повторяющихся документов
4. **Версионирование**: помечайте дату последнего обновления
5. **Ссылки, а не копии**: не копируйте информацию – ставьте ссылку на источник

Статус-отчёт: формат и пример

Еженедельный статус-отчёт – основной инструмент коммуникации PM с руководством.

Шаблон статус-отчёта

Раздел	Содержание
Проект	[Название]
Период	[Дата – Дата]
Общий статус	Зелёный / Жёлтый / Красный
Прогресс	Выполнено X из Y задач (Z%)
Достижения за неделю	3-5 пунктов
Риски и проблемы	Риск – Влияние – План митигации
Планы на следующую неделю	3-5 пунктов
Бюджет	Потрачено X из Y (Z%)

Раздел	Содержание
Нужна помощь	Конкретные запросы к руководству

RAG-статус (Red / Amber / Green)

Цвет	Значение	Действие
Зелёный	Проект идёт по плану	Продолжаем
Жёлтый	Есть риски, нужно внимание	PM работает над решением
Красный	Проект в опасности	Эскалация, нужна помощь руководства

Что дальше

В уроке 3.5 мы разберём управление удалёнными командами: как выстроить процессы, когда команда работает в разных часовых поясах, и как сохранить продуктивность и командный дух на расстоянии.

Урок 3.4 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.5: Удалённые команды – управление на расстоянии

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

После 2020 года удалённая работа стала нормой. По данным Buffer (State of Remote Work 2024), 98% опрошенных хотят работать удалённо хотя бы часть времени. GitLab (2000+ сотрудников, 65 стран, полностью удалённая компания) показывает, что remote-first модель работает на масштабе. Но статистика обратной стороны тоже показательна: 25% удалённых сотрудников страдают от одиночества, 22% – от невозможности “отключиться” от работы, а 50% менеджеров считают, что продуктивность удалённой команды ниже (Owl Labs).

Проблема не в удалённости как таковой. Проблема – в попытке управлять удалённой командой так же, как офисной. Remote-команда требует другого подхода: асинхронная коммуникация, документирование по умолчанию, осознанное построение культуры, чёткие ожидания по результатам.

В этом уроке мы разберём фреймворк управления удалённой командой: от выбора модели работы до конкретных практик, инструментов и решения типичных проблем.

Модели удалённой работы

Модель	Описание	Примеры компаний	Плюсы	Минусы
Remote-first	Удалёнка по умолчанию, офис – опция	GitLab, Automattic, Zapier	Равные условия для всех	Сложнее построить культуру
Hybrid	Часть дней в офисе, часть – удалённо	Google, Microsoft, Сбер	Баланс гибкости и общения	“Граждане двух сортов” (офис vs remote)
Remote-friendly	Офис основной, удалёнка допускается	Многие компании	Легче управлять	Удалённые чувствуют себя изолированными

Модель	Описание	Примеры компаний	Плюсы	Минусы
Fully distributed	Нет офиса, команда в разных странах/городах	Basecamp, Doist (Todoist)	Доступ к глобальным талантам	Часовые пояса, юридические сложности

Ключевое правило

Если хотя бы один человек в команде работает удалённо – вся команда должна работать “как удалённая”. Иначе возникает асимметрия: офисные сотрудники принимают решения у кофемашины, а удалённый узнаёт о них через день.

Асинхронная коммуникация: главный принцип

Синхронная vs Асинхронная

Характеристика	Синхронная	Асинхронная
Определение	Все участвуют одновременно	Каждый отвечает, когда удобно
Примеры	Zoom-звонок, Slack huddle, стендап	Slack-сообщение, Notion-комментарий, Loom-видео
Плюсы	Быстрая обратная связь, нюансы	Учитывает часовые пояса, даёт время подумать
Минусы	Требует совпадения по времени, прерывает	Медленнее, может теряться контекст

Когда синхронно, когда асинхронно

Ситуация	Рекомендация
Принятие критичного решения	Синхронно (встреча)
Разрешение конфликта	Синхронно (видео, не текст)
Обновление статуса	Асинхронно (Slack, standup-бот)
Code Review	Асинхронно (GitHub / GitLab)
Обсуждение требований	Асинхронно (Notion-документ) + синхронно (meeting для вопросов)
Ретроспектива	Синхронно (нужна живая дискуссия)
1-on-1	Синхронно (видео)
Объявление	Асинхронно (Slack #general)

Правило GitLab

“Write things down” – записывай всё. Если решение принято устно (на встрече, в звонке) – оно не считается принятым, пока не задокументировано. Это ключевой принцип remote-first.

Часовые пояса: практические решения

Overlap Hours (перекрывающиеся часы)

Определите “overlap window” – 3-4 часа в день, когда все члены команды онлайн. Все синхронные встречи – только в это окно.

Пример для команды Москва + Ташкент + Тбилиси:

Город	Часовой пояс	Рабочий день	Overlap
Москва	UTC+3	10:00-19:00	
Ташкент	UTC+5	10:00-19:00	
Тбилиси	UTC+4	10:00-19:00	
Overlap		**12:00-17:00 MSK**	5 часов

Для более разнесённых команд (Москва + Лондон + Нью-Йорк) overlap может быть всего 1-2 часа. В этом случае максимизируйте асинхронную коммуникацию.

Практические правила

1. **Все встречи – в overlap hours**
2. **Не ожидайте мгновенного ответа** – установите SLA: ответ в Slack – в течение 4 часов (в рабочее время)
3. **Записывайте встречи** – тот, кто не смог присутствовать, посмотрит запись
4. **Ротация неудобного времени** – если overlap попадает на раннее утро для одного пояса, чередуйте время встреч
5. **Используйте World Clock** – плагин в Slack или worldtimebuddy.com

---

Инструменты для удалённой работы

Категория	Инструменты	Назначение
Мессенджер	Slack, Microsoft Teams	Оперативная коммуникация
Видеозвонки	Zoom, Google Meet, Around	Синхронные встречи
Трекер задач	Jira, ClickUp, Asana	Управление работой
Документация	Notion, Confluence	Knowledge base
Видеосообщения	Loom, Tella	Асинхронные демо и объяснения
Виртуальная доска	Miro, FigJam	Brainstorm, ретроспективы
Дизайн	Figma	Совместная работа над макетами
Код	GitHub, GitLab	Код, PR, Code Review
Time tracking	Toggl, Clockify	Учёт рабочего времени
Виртуальный офис	Gather, SpatialChat	Ощущение присутствия

Loom – секретное оружие PM

Loom позволяет записать короткое видео с экраном и камерой за 2 минуты. Идеально для: - Объяснение сложных задач (вместо длинного текста) - Демо-функционала для стейкхолдеров - Feedback на дизайн-макеты - Онбординг нового сотрудника

Культура удалённой команды

Проблемы и решения

Проблема	Решение
Изоляция и одиночество	Виртуальные coffee chats (2 человека, random pairing, 15 мин)
Микроменеджмент	Управление по результатам, а не по часам (Results-Only Work Environment)
Размытие границ работа/жизнь	Чёткое рабочее время, право на disconnect

Проблема	Решение
Отсутствие доверия	Прозрачность: публичные каналы, открытые доски, еженедельные демо
Сложный онбординг	Buddy system: каждому новичку – наставник на 2-4 недели
Нет командного духа	Виртуальные тимбилдинги, ежеквартальные офлайн-встречи (offsites)

Remote Team Agreement

Составьте “командное соглашение” – документ, который фиксирует правила работы:

Пункт	Содержание
Рабочие часы	Core hours: 12:00-17:00 MSK, остальное – гибко
Время ответа	Slack: 4 часа (рабочее время), Email: 24 часа
Встречи	Камера включена (рекомендуется), запись обязательна
Документация	Все решения – в Notion/Confluence, не в DMs
Disconnect	После 19:00 и в выходные – не пишем (кроме P0-инцидентов)
1-on-1	Каждые 2 недели, 30 минут
Обратная связь	Открытая, конструктивная, по модели SBI (Situation-Behavior-Impact)

Метрики эффективности удалённой команды

Метрика	Как измерять	Целевое значение
Velocity (Scrum)	Jira / ClickUp	Стабильная или растущая
Cycle Time	Jira (от In Progress до Done)	Снижение или стабильность
Response Time	Slack analytics	< 4 часов в рабочее время
Meeting Load	Google Calendar audit	< 25% рабочего времени
Employee Satisfaction	Анонимные опросы (ежемесячно)	> 7/10
Documentation Coverage	Confluence / Notion audit	Все решения задокументированы

Важно: не измеряйте “часы онлайн”

Трекинг времени “мышь двигается” – это микроменеджмент, разрушающий доверие. Измеряйте результат: Velocity, Cycle Time, качество (количество багов), удовлетворённость клиента.

Что дальше

В уроке 3.6 мы разберём конфликты в проектах: почему они неизбежны, как их предотвращать и разрешать, и когда конфликт – это на самом деле хорошо для команды.

Урок 3.5 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.6: Конфликты в проектах – предотвращение и решение

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Конфликты в проектах неизбежны. По данным CPP Global (создатели Myers-Briggs), сотрудники тратят в среднем 2.8 часа в неделю на конфликты – это 359 миллиардов долларов потерянной продуктивности ежегодно только в США. При этом 85% сотрудников сталкиваются с конфликтами на работе, а 29% – практически постоянно.

Но конфликт – это не всегда плохо. Исследования Кэтлин Эйзенхардт (Stanford) показали, что команды с умеренным уровнем "когнитивных конфликтов" (споры об идеях, подходах, решениях) принимают более качественные решения, чем команды, где все всегда согласны. Проблема не в конфликтах, а в неумении их разрешать.

В этом уроке мы разберём типы конфликтов, модель их эскалации, 5 стратегий разрешения, и конкретные скрипты для PM, который оказался в роли медиатора.

Типы конфликтов в проектах

Тип	Описание	Пример	Опасность
Когнитивный (Task Conflict)	Споры об идеях, подходах, решениях	"REST vs GraphQL для нашего API"	Низкая (полезен!)
Процессный (Process Conflict)	Разногласия о том, КАК работать	"Спринты 1 неделя vs 2 недели"	Средняя
Межличностный (Relationship Conflict)	Личная неприязнь, обиды	"Он всегда критикует мой код"	Высокая
Ролевой (Role Conflict)	Неясность обязанностей	"Кто принимает решение – PO или PM?"	Средняя
Ресурсный (Resource Conflict)	Конкуренция за ресурсы	"Нам нужен этот разработчик, а не вашему проекту"	Высокая

Когнитивный конфликт – это ХОРОШО

Команда, где никто не спорит, – это команда, где никто не думает (или боится высказаться). PM должен поощрять когнитивные конфликты и мгновенно гасить межличностные.

Модель эскалации конфликта (Глассл)

Фридрих Глассл описал 9 стадий эскалации конфликта, которые можно сгруппировать в 3 фазы:

Фаза 1: Win-Win (ещё можно решить внутри команды)

Стадия	Описание	Признаки
1. Напряжение	Различия во мнениях, лёгкий дискомфорт	Обсуждения становятся горячее
2. Дебаты	Поляризация, попытка убедить	"Мы" vs "они", чёрно-белое мышление
3. Действия	От слов к действиям (игнорирование, обход)	Один человек принимает решение без обсуждения

Фаза 2: Win-Lose (нужен медиатор)

Стадия	Описание	Признаки
4. Коалиции	Формируются лагеря, ищут союзников	Разговоры “за спиной”, группировки
5. Потеря лица	Публичные обвинения	Переход на личности, токсичные комментарии
6. Угрозы	Ультиматумы	“Если не сделаете по-моему, я уйду с проекта”

Фаза 3: Lose-Lose (нужно вмешательство руководства)

Стадия	Описание	Признаки
7. Деструкция	Цель – навредить другой стороне	Саботаж, блокирование работы
8. Фрагментация	Разрушение команды	Люди увольняются, проект останавливается
9. Тотальная война	Обе стороны готовы пострадать, лишь бы навредить	Полный крах проекта

Задача PM: не допустить перехода из Фазы 1 в Фазу 2. Если конфликт на стадии 1-3, PM может разрешить его самостоятельно. На стадии 4-6 нужен внешний медиатор. На стадии 7+ – вмешательство руководства.

---

5 стратегий разрешения конфликтов (Thomas-Kilmann)

Модель Thomas-Kilmann описывает 5 стратегий поведения в конфликте, основанных на двух осях: assertiveness (настойчивость) и cooperativeness (кооперативность).

Стратегия	Настойчивость	Кооперация	Когда использовать	Когда НЕ использовать
Соперничество (Competing)	Высокая	Низкая	Критические решения, безопасность, P0-инцидент	Долгосрочные отношения
Уступка (Accommodating)	Низкая	Высокая	Вопрос неважен для вас, сохранение отношений	Принципиальные вопросы
Избегание (Avoiding)	Низкая	Низкая	Эмоции накалены, нужна пауза, вопрос решится сам	Критичные вопросы, дедлайн горит
Компромисс (Compromising)	Средняя	Средняя	Равная сила сторон, временное решение	Когда возможна коллаборация
Сотрудничество (Collaborating)	Высокая	Высокая	Важны и результат, и отношения, есть время	Тривиальные вопросы, нет времени

Рекомендация для PM

В 80% случаев стремитесь к **сотрудничеству** (win-win). Используйте **компромисс** как запасной вариант. **Соперничество** – только в кризисных ситуациях (P0-инцидент, вопрос безопасности). **Избегание** – когда эмоции зашкаливают и нужна пауза (но вернитесь к проблеме!).

Роль PM как медиатора

Пошаговый алгоритм медиации

Шаг 1: Выслушайте обе стороны отдельно (1-on-1) - Дайте каждому высказаться без прерываний - Задавайте открытые вопросы: “Расскажи, как ты видишь ситуацию?” - Не принимайте чью-то сторону на этом этапе

Шаг 2: Определите корневую причину - Часто видимый конфликт – симптом. Реальная причина глубже: - “Он плохо делает код” -> На самом деле: нет Code Review стандартов - “Она не слушает мои идеи” -> На самом деле: PO не привлекает разработчиков к обсуждению - “Он срывает дедлайны” -> На самом деле: нереалистичные оценки без буфера

Шаг 3: Проведите совместную встречу - Установите правила: говорить о фактах, не о людях; слушать до конца - Каждая сторона описывает: “Ситуация – Что я чувствую – Что я хочу” - PM фасилитирует, не судит

Шаг 4: Найдите решение - Задайте вопрос: “Что нужно, чтобы мы оба были довольны результатом?” - Зафиксируйте договорённости письменно - Определите action items и follow-up

Шаг 5: Follow-up - Через 1-2 недели проведите check-in: “Как ситуация? Договорённости соблюдаются?” - Если конфликт вернулся – эскалируйте

Предотвращение конфликтов: превентивные меры

Мера	Описание	Когда внедрять
Чёткие роли и обязанности (RACI)	Кто Responsible, Accountable, Consulted, Informed	На старте проекта
Team Agreement	Правила работы, принятые командой	На старте проекта
Регулярные 1-on-1	Раннее выявление напряжения	Каждые 2 недели
Ретроспективы	Безопасное пространство для обратной связи	Каждый спринт
Definition of Done	Единое понимание “готовности”	На старте проекта
Код-ревью стандарты	Правила ревью: что комментировать, а что нет	На старте разработки
Обратная связь по SBI	Situation-Behavior-Impact (не “ты плохой”, а “в ситуации X ты сделал Y, и это повлияло Z”)	Постоянно

Модель SBI для обратной связи

Компонент	Плохой пример	Хороший пример (SBI)
Situation	“Ты вечно...”	“На вчерашнем стендапе...”
Behavior	“...всё критикуешь”	“...ты прервал Алексея трижды, не дав ему закончить мысль”
Impact	“...и всех бесишь”	“...и он не смог рассказать о блокере, который обнаружился позже и стоил нам 4 часа”

Типичные конфликты в проектах и их решение

Конфликт	Корневая причина	Решение
Разработчик vs QA: “Это не баг”	Нет Definition of Done	Совместно создать DoD с критериями приёмки
PO vs Dev: “Вы делаете не то”	Нечёткие User Stories	Grooming: детализация AC до спринта
PM vs Руководство: “Добавьте фичу в спринт”	Scope creep	Показать Impact: что выпадет, если добавить
Dev vs Dev: “Мой подход лучше”	Нет архитектурных стандартов	ADR (Architecture Decision Records)

Конфликт	Корневая причина	Решение
Команда vs Дедлайн: “Мы не успеваем”	Нереалистичная оценка	Пересмотр скоупа (MoSCoW), буфер 20-30%

Что дальше

В уроке 3.7 мы разберём мотивацию команды: KPI, OKR, нематериальные стимулы, теории мотивации и конкретные инструменты PM для поддержания продуктивности и вовлечённости.

Урок 3.6 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.7: Мотивация команды — от KPI до нематериальных стимулов

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Мотивированная команда работает в 2-3 раза эффективнее немотивированной. Но мотивация — это не «повысить зарплату». Исследования Дэниела Пинка показали, что после определённого уровня дохода деньги перестают быть главным мотиватором. На первый план выходят автономия, мастерство и смысл.

PM не управляет зарплатами, но он управляет средой, в которой люди работают. В этом уроке разберём: как мотивировать команду без бюджета, как правильно ставить KPI, и какие ошибки убивают мотивацию.

Теории мотивации для PM

Модель Дэниела Пинка (Drive)

Фактор	Что значит	Как PM может влиять
Autonomy	Свобода в выборе как делать работу	Давайте задачи, не диктуйте способ выполнения
Mastery	Возможность расти и улучшаться	Сложные задачи, обучение, обратная связь
Purpose	Понимание зачем это нужно	Связывайте задачи с большой целью

Двухфакторная модель Герцберга

Гигиенические факторы (устраняют неудовлетворённость)	Мотиваторы (создают удовлетворённость)
Зарплата	Признание достижений
Условия работы	Интересная работа
Отношения с коллегами	Ответственность
Политика компании	Карьерный рост
Безопасность	Профессиональное развитие

Ключевой инсайт: Устранение гигиенических факторов не мотивирует, но их отсутствие демотивирует.

KPI для команды

Правила хороших KPI

1. **Измеримые** — конкретное число, а не «улучшить качество»
2. **Под контролем** — человек может повлиять на результат
3. **Сбалансированные** — не только скорость, но и качество
4. **Понятные** — каждый знает свои KPI и как они считаются
5. **Обновляемые** — пересматриваются каждый квартал

KPI по ролям

Роль	KPI	Как считать
Разработчик	Velocity (Story Points/sprint)	Среднее за 3 спринта
Разработчик	Bugs per release	Баги, найденные после деплоя
Дизайнер	Task completion rate	% задач, закрытых в срок
QA	Test coverage	% покрытия тестами
Маркетолог	Leads generated	Количество квалифицированных лидов
PM	On-time delivery	% проектов, сданных в срок

Ошибки с KPI

- **Слишком много** — 3-5 KPI максимум на человека
- **Только количественные** — «100 коммитов в неделю» убивает качество
- **Не обсуждаются** — KPI должны быть согласованы, не навязаны
- **Не пересматриваются** — то, что важно в Q1, может быть неважно в Q3

Нематериальная мотивация

10 способов мотивации без бюджета

#	Способ	Как применить
1	Публичное признание	Отметить на standup, в Slack, на ретроспективе
2	Интересные задачи	Дать возможность поработать с новой технологией
3	Автономия	«Реши эту задачу как считаешь лучше»
4	Менторство	Старший помогает младшему (выигрывают оба)
5	Карьерные разговоры	1-on-1 про цели и развитие (ежеквартально)
6	Голос в решениях	Спрашивать мнение команды перед решениями
7	Гибкость	Удалёнка, гибкий график, async-работа
8	Челленджи	Хакатоны, innovation time, side projects
9	Прозрачность	Делитесь метриками, планами, контекстом
10	Благодарность	Простое «спасибо» работает лучше, чем кажется

1-on-1 встречи

- **Частота:** каждые 1-2 недели, 30 минут

- **Формат:** не статус-апдейт, а разговор о человеке
- **Вопросы:**
  - Что тебе нравится в работе? Что нет?
  - Чему хочешь научиться?
  - Чем я могу помочь?
  - Есть ли что-то, что мешает?

Демотиваторы: что убивает мотивацию

Демотиватор	Почему вредит	Как исправить
Микроменеджмент	Убивает автономию	Делегируйте результат, не процесс
Отсутствие фидбека	Люди не знают, хорошо ли работают	Регулярная обратная связь
Бессмысленные задачи	Не видят связь с целью	Объясняйте контекст и impact
Переработки	Выгорание	Защищайте команду от scope creep
Несправедливость	Потеря доверия	Прозрачные правила для всех
Токсичные люди	Отравляют атмосферу	Решайте быстро, не игнорируйте

Что дальше

Вы знаете, как мотивировать команду. В следующем уроке разберём лидерство PM — как перейти от менеджера задач к лидеру, за которым хотят идти.

Урок 3.7 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.8: Лидерство PM — от менеджера к лидеру

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Практический урок

Введение

Менеджер управляет задачами. Лидер вдохновляет людей. Лучшие PM — это и менеджеры, и лидеры. Они не просто следят за сроками и бюджетом, они создают среду, в которой команда хочет делать лучшую работу.

По данным Google (Project Oxygen), лучшие менеджеры — это не лучшие технари, а те, кто умеет слушать, давать обратную связь и создавать психологическую безопасность. В этом уроке разберём: как PM может стать лидером, а не просто «трекером задач».

Менеджер vs Лидер

Менеджер	Лидер
Управляет задачами	Управляет людьми
Контролирует	Доверяет
Следует процессам	Создаёт процессы

Менеджер	Лидер
Реагирует на проблемы	Предвидит проблемы
Говорит «сделайте это»	Говорит «давайте сделаем это»
Использует власть позиции	Использует влияние
Фокус на КАК	Фокус на ЗАЧЕМ

PM как Servant Leader

Scrum Guide определяет Scrum Master как Servant Leader — лидера, который служит команде. Этот принцип применим к любому PM:

- **Убирает препятствия** вместо создания новых
- **Защищает команду** от внешнего давления
- **Создаёт условия** для лучшей работы
- **Развивает людей** а не только проект

---

5 навыков PM-лидера

1. Эмоциональный интеллект (EQ)

Компонент	Что значит	Как развивать
Самосознание	Понимать свои эмоции	Рефлексия, дневник
Саморегуляция	Управлять реакциями	Пауза перед ответом
Эмпатия	Понимать чувства других	Активное слушание
Социальные навыки	Строить отношения	Networking, менторство
Мотивация	Внутренний drive	Цели, purpose

2. Обратная связь

SBI-модель (Situation-Behavior-Impact):

- **Situation:** «На вчерашнем demo...»
- **Behavior:** «...ты представил фичу без подготовки...»
- **Impact:** «...и клиент не понял ценность, мы потеряли возможность»

Правила обратной связи: - Давайте вовремя (не через месяц) - Конкретно (не «работай лучше») - Приватно для негативной, публично для позитивной - Про поведение, не про личность

3. Принятие решений

Ситуация	Метод	Когда
Горит дедлайн	Авторитарное	PM решает сам
Технический выбор	Консультативное	PM спрашивает экспертов
Процессные изменения	Консенсус	Команда решает вместе
Стратегические вопросы	Делегирование стейкхолдеру	PM эскалирует

4. Storytelling

- Числа убеждают, истории вдохновляют
- Формат: «Было → Стало → Результат»

- Используйте storytelling для: kickoff, demo, ретроспектив, отчётов

5. Психологическая безопасность

Google (Project Aristotle) обнаружил, что **психологическая безопасность** — фактор №1 эффективности команды.

Что это: среда, где люди не боятся: - Задавать «глупые» вопросы - Признавать ошибки - Предлагать новые идеи - Не соглашаться с руководством

Как создать: - Признавайте свои ошибки первыми - Благодарите за вопросы и предложения - Не наказывайте за честность - Реагируйте на проблемы, а не на людей

Стили лидерства

Стиль	Когда использовать	Когда избегать
Директивный	Кризис, новая команда	Опытная команда
Коучинг	Развитие людей	Когда нет времени
Поддерживающий	Опытная команда	Когда нужен контроль
Делегирующий	Самостоятельная команда	Новички
Визионерский	Изменения, новое направление	Рутинные задачи
Демократический	Когда нужен buy-in	Срочные решения

Лучшие лидеры переключаются между стилями в зависимости от ситуации и зрелости команды.

Карьерный путь PM

Уровень	Роль	Фокус	Опыт
Junior	Project Coordinator	Задачи и сроки	0-2 года
Middle	Project Manager	Проекты и команды	2-5 лет
Senior	Senior PM	Программы и стратегия	5-8 лет
Lead	Head of PM / PMO	Процессы и методология	8-12 лет
Executive	VP of Operations / COO	Бизнес и стратегия	12+ лет

Что дальше

Вы освоили навыки лидерства для PM. В следующем уроке разберём сертификации — PMP, PSM, PRINCE2 — какие стоит получить и как подготовиться.

Урок 3.8 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.9: Сертификации — PMP, PSM, PRINCE2

Блок 3: Инструменты и лидерство Длительность видео: 8-10 минут **Тип:** Обзорный урок

Введение

Сертификация PM — это доказательство ваших знаний и навыков для работодателей и клиентов. По данным PMI, сертифицированные PMP зарабатывают в среднем на 25% больше, чем несертифицированные коллеги.

Но сертификаций много: PMP, PSM, PRINCE2, SAFe, CAPM и другие. Какую выбрать? Зависит от вашей специализации, опыта и карьерных целей.

В этом уроке разберём основные сертификации, требования, стоимость и стратегию подготовки.

Основные сертификации

Сравнение

Сертификация	Организация	Фокус	Опыт	Стоимость	Сложность
PMP	PMI	Классический PM	3-5 лет	$555	Высокая
CAPM	PMI	Начинающий PM	0-3 года	$300	Средняя
PSM I	Scrum.org	Scrum Master	0+ лет	$200	Средняя
PSM II	Scrum.org	Advanced Scrum	2+ года	$250	Высокая
CSM	Scrum Alliance	Scrum Master	0+ лет	$1,000-1,500	Низкая
PRINCE2 Foundation	Axelos	Waterfall PM	0+ лет	$350-500	Средняя
PRINCE2 Practitioner	Axelos	Advanced PM	1+ год	$400-600	Высокая
SAFe Agilist	Scaled Agile	Enterprise Agile	3+ лет	$1,000	Средняя
PMI-ACP	PMI	Agile PM	2+ лет	$435	Средняя

PMP (Project Management Professional)

Самая признаваемая в мире

- **Организация:** PMI (Project Management Institute)
- **Признаётся:** глобально, во всех индустриях
- **Holderов:** 1.4+ миллиона по всему миру
- **Зарплата:** +25% в среднем (vs non-PMP)

Требования

Образование	Опыт PM	Обучение
4-year degree	36 месяцев (3 года)	35 часов
Без degree	60 месяцев (5 лет)	35 часов

Экзамен

- **180 вопросов** за 230 минут
- **Формат:** multiple choice + drag & drop
- **Проходной балл:** не публикуется (примерно 65-70%)
- **Домены:** People (42%), Process (50%), Business Environment (8%)
- **Фокус 2024+:** 50% Predictive + 50% Agile

Подготовка

Ресурс	Тип	Стоимость
PMBOK 7th Edition	Книга	$100 (или бесплатно для членов PMI)
Rita Mulcahy PMP Exam Prep	Книга	$65
Joseph Phillips (Udemy)	Курс	$15-30
PMI Study Hall	Практические вопросы	$79
Prepcast Simulator	Mock exams	$69

Стратегия: 2-3 месяца подготовки, 1-2 часа в день.

PSM (Professional Scrum Master)

Почему PSM, а не CSM

PSM (Scrum.org)	CSM (Scrum Alliance)
$200 (только экзамен)	$1,000-1,500 (тренинг обязателен)
Нет обязательного тренинга	2-дневный тренинг обязателен
Сложный экзамен (85% проходной)	Лёгкий экзамен (после тренинга)
Не требует продления	Продление каждые 2 года ($100)
Более уважаемая в tech	Более известная в корпорациях

Экзамен PSM I

- **80 вопросов** за 60 минут
- **Проходной балл:** 85%
- **Формат:** multiple choice, true/false, multiple answer
- **Основа:** Scrum Guide 2020 (13 страниц)

Подготовка

1. Прочитайте Scrum Guide 2020 (3-5 раз)
2. Пройдите Open Assessments на scrum.org (бесплатно)
3. Mikhail Lapshin's PSM Quiz (бесплатно)
4. Подготовка: 2-4 недели

PRINCE2

Когда выбрать PRINCE2

- Работаете в Великобритании, Австралии, Европе
- Государственные и корпоративные проекты
- Waterfall/Hybrid методологии
- Процессно-ориентированный подход

Два уровня

	Foundation	Practitioner
Фокус	Знание терминологии и процессов	Применение на практике
Вопросов	60	68
Проходной	55% (33/60)	55% (38/68)
Открытая книга	Нет	Да
Стоимость	$350-500	$400-600

Как выбрать сертификацию

Матрица выбора

Если вы...	Рекомендация
Начинающий PM (0-2 года)	CAPM или PSM I
PM с опытом (3+ лет)	PMP
Scrum-команда	PSM I → PSM II
Корпорация / госсектор	PRINCE2
Enterprise Agile	SAFe Agilist
Agile PM	PMI-ACP
Хотите быстро и дёшево	PSM I ($200, 2-4 недели)
Хотите максимальный ROI	PMP ($555, 2-3 месяца)

Порядок получения

1. **PSM I** — быстро, дёшево, полезно → первая сертификация
2. **PMP** — через 3-5 лет опыта → максимальный карьерный boost
3. **SAFe / PSM II** — специализация по мере роста

Что дальше

Вы знаете, какие сертификации существуют и какую выбрать. В финальном уроке мы объединим все знания и создадим план управления реальным проектом.

Урок 3.9 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами

Урок 3.10: Финальный проект — управление реальным проектом

Блок 3: Инструменты и лидерство Длительность видео: 10-12 минут **Тип:** Проектный урок

Введение

Это финальный урок курса. За 30 уроков вы изучили всё — от основ PM до Agile, от инструментов до лидерства. Теперь объединим знания в практический кейс: полный цикл управления реальным проектом.

Вы получите: чек-листы, шаблоны и пошаговый план, который можно применить к любому проекту — от запуска продукта до организации мероприятия.

Кейс: Запуск нового продукта

Контекст

Параметр	Значение
Проект	Запуск мобильного приложения для фитнес-клуба
Команда	7 человек (PM, 2 разработчика, дизайнер, QA, маркетолог, PO)
Срок	12 недель (3 спринта по 4 недели или 6 спринтов по 2 недели)
Бюджет	$50,000
Методология	Scrum (2-недельные спринты)

Фаза 1: Инициация (Неделя 0)

Project Charter

- **Цель (SMART):** Запустить MVP мобильного приложения с функциями бронирования, расписания и абонемента к [дата]
- **Scope:** Авторизация, расписание, бронирование, оплата, push-уведомления
- **Out of scope:** Социальные функции, видео-тренировки, геймификация
- **Стейкхолдеры:** CEO (спонсор), тренеры, клиенты клуба
- **Бюджет:** $50,000 (разработка $30K, дизайн $8K, маркетинг $7K, резерв $5K)
- **Риски:** Интеграция с кассой, задержка дизайна, низкий adoption

Kick-off встреча

Агенда: 1. Представление проекта и команды (10 мин) 2. Цель и scope (15 мин) 3. Timeline и milestone (10 мин) 4. Роли и ответственности (10 мин) 5. Инструменты и процессы (10 мин) 6. Вопросы (15 мин)

Фаза 2: Планирование (Неделя 1)

Product Backlog

#	User Story	Priority	Story Points
1	Как клиент, я хочу зарегистрироваться по email/Google	Must	5
2	Как клиент, я хочу видеть расписание занятий	Must	3
3	Как клиент, я хочу забронировать занятие	Must	8
4	Как клиент, я хочу оплатить абонемент	Must	13
5	Как клиент, я хочу получать push-уведомления	Should	5
6	Как тренер, я хочу видеть список записавшихся	Should	3
7	Как админ, я хочу управлять расписанием	Should	8
8	Как клиент, я хочу видеть историю посещений	Could	3

Sprint Planning (Sprint 1)

- **Sprint Goal:** Авторизация + просмотр расписания
- **Velocity (ожидаемая):** 20 SP
- **Stories:** #1 (5 SP) + #2 (3 SP) + Design System (8 SP) + Backend Setup (4 SP) = 20 SP

Definition of Done

- Код написан и прошёл code review
- Unit тесты написаны и проходят
- QA протестировал и подтвердил
- Дизайн соответствует макетам
- Документация обновлена
- PO принял на Sprint Review

Фаза 3: Выполнение (Недели 2-10)

Daily Standup (15 мин, каждый день)

- Что сделал вчера?
- Что буду делать сегодня?
- Есть ли блокеры?

Sprint Review (каждые 2 недели)

- Демо готовой функциональности стейкхолдерам
- Обратная связь от PO и CEO
- Обновление backlog на основе фидбека

Retrospective (каждые 2 недели)

Формат «Start, Stop, Continue»: - **Start:** что начать делать? - **Stop:** что перестать делать? - **Continue:** что продолжить делать?

Burndown Chart

Отслеживайте прогресс каждый день: - Ось X: дни спринта - Ось Y: оставшиеся Story Points - Ideal line vs Actual line - Если Actual выше Ideal — спринт под угрозой

Фаза 4: Мониторинг

Еженедельный статус-репорт

```
## Статус-репорт: [Дата]

### Общий статус: 🟢 На графике

### Прогресс
- Sprint 3/6 в процессе
- 45/100 SP выполнено (45%)
- Бюджет: $22K из $50K использовано (44%)

### Достижения недели
- Авторизация завершена и протестирована
- Дизайн расписания утверждён
- API бронирования готов на 70%

### Риски
```

```
- 🟡 Интеграция с платёжной системой — задержка ответа от провайдера
- Митигация: параллельно тестируем альтернативный провайдер

### План на следующую неделю
- Завершить бронирование
- Начать интеграцию с оплатой
- QA: тестирование авторизации + расписание
```

Управление рисками

Риск	Статус	Действие
Задержка платёжной интеграции	🟡 Active	Тестируем backup провайдера
Дизайнер болеет	🟢 Resolved	Фрилансер подключён
Scope creep (CEO хочет чат)	🟡 Active	Перенесли в v2.0

Фаза 5: Закрытие (Недели 11-12)

Launch Checklist

- [ ] Все Must stories выполнены
- [ ] QA regression testing пройдено
- [ ] Performance testing (< 3 сек загрузка)
- [ ] Security audit
- [ ] App Store / Google Play submission
- [ ] Marketing launch plan активирован
- [ ] Support team обучена
- [ ] Monitoring и alerting настроены
- [ ] Rollback plan подготовлен

Project Retrospective

Вопрос	Ответы команды
Что пошло хорошо?	Scrum работал, коммуникация, качество
Что можно улучшить?	Estimation, stakeholder management
Что мы узнали?	Интеграции всегда дольше, чем кажется
Что сделаем по-другому?	Начнём интеграции раньше, буфер 20%

Lessons Learned

Документируйте для будущих проектов: - Какие оценки были точными, какие нет - Какие риски реализовались - Какие процессы работали - Рекомендации для следующего проекта

Итоги курса

За 30 уроков (3 блока) вы освоили:

Блок 1: Основы управления проектами - Принципы, жизненный цикл, Waterfall vs Agile - SMART, OKR, WBS, Ганта, критический путь - Риски, бюджет, стейкхолдеры

Блок 2: Agile, Scrum и Kanban - Agile Manifesto, Scrum роли и церемонии - Sprint Planning, Daily, Review, Retro - Kanban, User Stories, Story Points - SAFe, LeSS, Nexus

Блок 3: Инструменты и лидерство - Jira, Notion, Trello, Asana, ClickUp - Коммуникация, удалённые команды, конфликты - Мотивация, лидерство, сертификации

Теперь у вас есть знания, инструменты и практический опыт. Управляйте проектами профессионально!

Урок 3.10 из 10 | Блок 3: Инструменты и лидерство | Курс: Управление проектами