**Каскадная методология**

1. Основы каскадной методологии

1.1 Объяснение каскадной модели (Waterfall) простыми словами

Waterfall (водопад) — это линейный подход к управлению проектами, где работа делится на последовательные этапы. Каждый этап должен быть полностью завершён до перехода к следующему.

Аналогия: Представьте строительство дома:

* Сначала создают проект (архитектура).
* Затем заливают фундамент.
* Возводят стены.
* Проводят коммуникации.
* Делают отделку.

Нельзя начать отделку, пока не построены стены. Так же и в Waterfall — нельзя тестировать продукт, пока не готово ТЗ и код.

1.2. Основные этапы Waterfall

1. Сбор и анализ требований (что нужно сделать?)
2. Проектирование (как это будет работать?)
3. Разработка (написание кода/создание продукта)
4. Тестирование (проверка на ошибки)
5. Внедрение (запуск продукта)
6. Поддержка (исправление багов, обновления)

1.3. Преимущества и недостатки Waterfall vs Agile

| Критерии | Waterfall | Agile |
| --- | --- | --- |
| Гибкость | Низкая (трудно вносить изменения) | Высокая (итерации, адаптация) |
| Документация | Полная (ТЗ, спецификации) | Минимальная (гибкие требования |
| Риски | Высокие (ошибки обнаруживаются поздно) | Низкие (тестируется на каждой итерации) |
| Сроки и бюджет | Чёткие (фиксированный план) | Гибкие (приоритеты могут меняться) |
| Применение | Проекты с ясными требованиями (строительство, госпроекты) | IT, стартапы, динамичные проекты |

1.4. Кто и когда предложил каскадную модель?

1970 — Уинстон Ройс (Winston Royce) в статье "Managing the Development of Large Software Systems" описал Waterfall как нежелательный, но распространённый подход.

1985 — Стандарт DOD-STD-2167 (военные проекты США) закрепил Waterfall как обязательный для госзаказов.

1990-2000-е — Критика Waterfall в IT из-за негибкости, рост популярности Agile.

2. Практическое применение Waterfall

2.1. Как составить план проекта по Waterfall? Пример структуры

1. Инициация

* Определение целей, стейкхолдеров.
* Оценка бюджета и сроков.

1. Анализ требований

* Документ ТЗ (Техническое задание).

1. Проектирование

* Архитектура, схемы, спецификации.

1. Разработка

* Написание кода/создание продукта.

1. Тестирование

* Тест-кейсы, отчёт о багах.

1. Внедрение

* Развёртывание, обучение пользователей.

1. Поддержка

* Исправление ошибок, обновления.

2.2. Какие документы обязательны на каждом этапе?

* ТЗ (Техническое задание) — этап анализа.
* SRS (Software Requirements Specification) — детальные требования.
* Дизайн-документ — архитектура, схемы БД.
* Тест-план и отчёты — этап тестирования.
* User Manual — руководство пользователя.

2.3. Инструменты для управления Waterfall

* MS Project — для диаграмм Ганта.
* Excel — для трекинга задач и бюджета.
* Jira (в классическом режиме) — если нужен трекинг, но без Agile-доски.
* Confluence — для хранения документации.

3. Waterfall в разных сферах

3.1. Где чаще всего применяется Waterfall?

* Строительство (нельзя менять проект после заливки фундамента).
* Аэрокосмическая отрасль (NASA, Boeing — строгие стандарты).
* Госпроекты (бюджетные, с жёсткими требованиями).
* Фармацевтика (длительные тесты, сертификации)

3.2. Можно ли использовать Waterfall в IT?

Да, но редко:

Подходит:

* Проекты с фиксированными требованиями (например, банковские системы).
* Legacy-системы (где важна стабильность).

Не подходит:

* Стартапы (требования меняются).
* Веб-разработка (быстрые изменения).

3.3. Гибридные модели (Water-Scrum-Fall)

* Water-Scrum-Fall — требования и дизайн по Waterfall, разработка по Scrum.
* Канбан + Waterfall — гибкое управление задачами внутри жёсткого плана.

4. Проблемы и решения в Waterfall

4.1. Что делать, если на этапе тестирования выявлены ошибки в требованиях?

Варианты:

* Вернуться на этап анализа → задержка сроков.
* Заплатить за доработку → увеличение бюджета.

Как избежать:

* Тщательный анализ требований на старте.
* Прототипирование перед разработкой.

4.2. Как управлять изменениями в Waterfall?

* Change Request (CR) — формальный запрос на изменение.
* Impact Analysis — оценка влияния на сроки/бюджет.

4.3. Критика Waterfall: аргументы за и против

Плюсы:

* Чёткий план, прогнозируемый бюджет.
* Подходит для регулируемых отраслей (медицина, авиация).

Минусы:

* Нет гибкости.
* Ошибки обнаруживаются слишком поздно.
* Долгая разработка (клиент ждёт годами).

5. Сравнение с другими методологиями

5.1. Waterfall vs Agile: когда что выбирать?

| Waterfall | Agile |
| --- | --- |
| Чёткие требования (строительство) | Изменчивые требования (стартапы) |
| Долгосрочные проекты (5+ лет) | Короткие итерации (2-4 недели) |
| Жёсткий контроль бюджета | Гибкое финансирование |

5.2. Чем Waterfall отличается от V-модели?

* V-модель — это Waterfall + раннее тестирование.
* Каждому этапу разработки соответствует этап тестирования.

5.3. Можно ли интегрировать Agile в Waterfall?

Да, например:

* Гибридная модель (верхнеуровневое планирование — Waterfall, разработка — Scrum).
* Agile Stage-Gate — контрольные точки (гейты) между этапами.

6. Кейсы и примеры

6.1. Успешные проекты на Waterfall

* Boeing 787 Dreamliner (хотя были задержки, проект завершён).
* Космические миссии NASA (Apollo, Mars Rover).

6.2. Провальные кейсы

* Healthcare.gov (2013) — крах из-за плохого ТЗ.
* UK NHS IT Programme (2000-е) — провал на £10 млрд.

6.3. Как крупные корпорации используют Waterfall?

* NASA — строгий контроль этапов.
* Boeing — жёсткие стандарты безопасности.

**Методологии Agile**

1. Общие вопросы по Agile

1.1. Объяснение Agile простыми словами

Agile — это гибкий подход к управлению проектами, где работа делится на короткие итерации (спринты), а требования могут меняться.

Основные ценности Agile (Манифест Agile, 2001):

* Люди и взаимодействие важнее процессов и инструментов.
* Работающий продукт важнее документации.
* Сотрудничество с заказчиком важнее переговоров по контракту.
* Готовность к изменениям важнее следования плану.

12 принципов Agile (ключевые идеи):

* Удовлетворять клиента через ранние поставки.
* Изменения приветствуются даже на поздних этапах.
* Работающий продукт — главный показатель прогресса.
* Команда самоорганизуется.

Аналогия:

Представьте, что вы строите не дом сразу, а собираете его по комнатам:

* Сначала делаете кухню и проверяете, удобно ли.
* Потом гостиную — и снова тестируете.
* Корректируете планы на ходу.

1.2. Разница между Agile, Scrum и Kanban

| Критерий | Agile (философия) | Scrum (фреймворк) | Kanban (метод) |
| --- | --- | --- | --- |
| Суть | Гибкий подход | Работа в спринтах (2-4 нед.) | Непрерывный поток задач |
| Роли | Нет строгих ролей | Scrum Master, Product Owner | Нет ролей, только доска |
| Планирование | Итеративное | Планирование спринтов | Планирование по мере поступления задач |
| Гибкость | Высокая | Средняя (фиксированные спринты) | Очень высокая (нет спринтов) |
| Инструменты | Любые (Jira, Trello) | Scrum-доски, бэклоги | Kanban-доски (Trello, Jira) |

1.3. Популярные фреймворки Agile (кроме Scrum и Kanban)

* Lean — устранение потерь (бережливое производство).
* XP (Extreme Programming) — частые релизы, парное программирование.
* SAFe (Scaled Agile Framework) — Agile для крупных компаний.
* Crystal — адаптация под размер команды.
* DSDM (Dynamic Systems Development Method) — фокус на бизнес-потребности

2. Практическое применение Agile

2.1. Как внедрить Agile в команде с нуля? Пошаговый план

* Обучение команды (тренинги, книги, Agile-коучи).
* Выбор фреймворка (Scrum для IT, Kanban для поддержки).
* Создание бэклога (список задач с приоритетами).
* Планирование первого спринта (2-4 недели).
* Ежедневные стендапы (15 мин., 3 вопроса: что сделал? что будешь делать? какие препятствия?).
* Ретроспектива (анализ спринта: что улучшить?).
* Постепенная адаптация (гибкость к изменениям).

2.2. Лучшие инструменты для Agile

* Jira — для Scrum/Kanban, сложных проектов.
* Trello — для простых досок (Kanban).
* Notion — для документации + гибких досок.
* Azure DevOps — для IT-команд (интеграция с кодом).
* Miro — для планирования и ретроспектив.

2.3. Как проводить эффективные стендапы?

* Формат Daily Stand-up (15 мин.):
* Что сделал вчера?
* Что сделаешь сегодня?
* Какие блокеры?

Лайфхаки:

* Стоять (не сидеть!) — чтобы не затягивать.
* Не решать проблемы на месте — выносить в отдельные обсуждения.
* Фиксировать блокеры в чате/доске

3. Agile в разных сферах

3.1. Agile в маркетинге

Примеры:

* Sprint-кампании — тестируем гипотезы за 2 недели.
* Гибкое планирование контента (Kanban-доска для SMM).
* A/B-тесты вместо долгого анализа.

Кейс: Coca-Cola использует Agile для быстрого запуска рекламных идей.

3.2. Agile в нетехнических проектах

* HR: Гибкий подбор (быстрые итерации с кандидатами).
* Продажи: Scrum для управления воронкой.
* Event-менеджмент: Планирование ивентов по спринтам.

3.3. Agile в образовании

* Микрообучение (короткие модули вместо лекций).
* Гибкие учебные планы (адаптация под студентов).
* Ретроспективы после курсов (что улучшить?).

4. Проблемы и решения в Agile

4.1. Основные ошибки при переходе на Agile

* Формальное внедрение (Agile ≠ просто доски).
* Отсутствие поддержки руководства.
* Игнорирование ретроспектив (нет улучшений).
* Слишком длинные спринты (теряется гибкость).

4.2. Как мотивировать команду?

* Показать выгоды (меньше рутины, больше креатива).
* Начать с пилотного проекта (чтобы увидеть результат).
* Ввести геймификацию (баллы за завершение задач).

4.3. Что делать, если спринты срываются?

* Уменьшить объем задач в спринте.
* Разбивать задачи на более мелкие.
* Анализировать причины на ретроспективе.

5. Agile-артефакты и церемонии

5.1. Как проводить ретроспективу?

Форматы:

1. Start/Stop/Continue — что начать, прекратить, продолжить?
2. Mad/Sad/Glad — эмоции от спринта.
3. 4 вопроса:

* Что прошло хорошо?
* Что пошло не так?
* Что узнали нового?
* Что улучшить?

5.2. Как составить бэклог продукта?

* User Stories (как пользователь, я хочу…).
* Приоритизация (MoSCoW: Must have, Should have, Could have, Won’t have).
* Оценка сложности (стори поинты: 1, 2, 3, 5, 8…).

5.3. Ключевые метрики Agile

* Velocity — сколько стори поинтов завершено за спринт.
* Lead Time — время от создания задачи до завершения.
* Cycle Time — время работы над задачей.

6. Сравнение Agile с другими методологиями

6.1. Совмещение Agile и DevOps

* CI/CD (непрерывная интеграция/доставка) — автоматизация.
* Общие инструменты (Jira + Docker/Kubernetes).
* Культура сотрудничества (разработчики + эксплуатация).

6.2. Где Agile не работает?

* Жёстко регламентированные проекты (авиация, медицина).
* Команды без самоорганизации (нужна сильная дисциплина).
* Очень большие команды (требуются фреймворки типа SAFe).