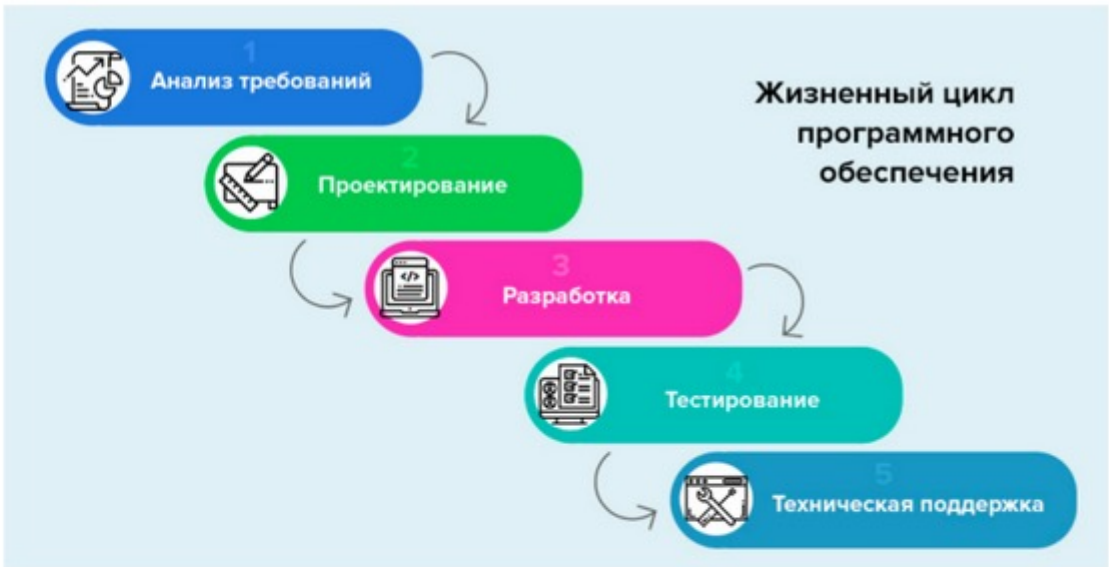
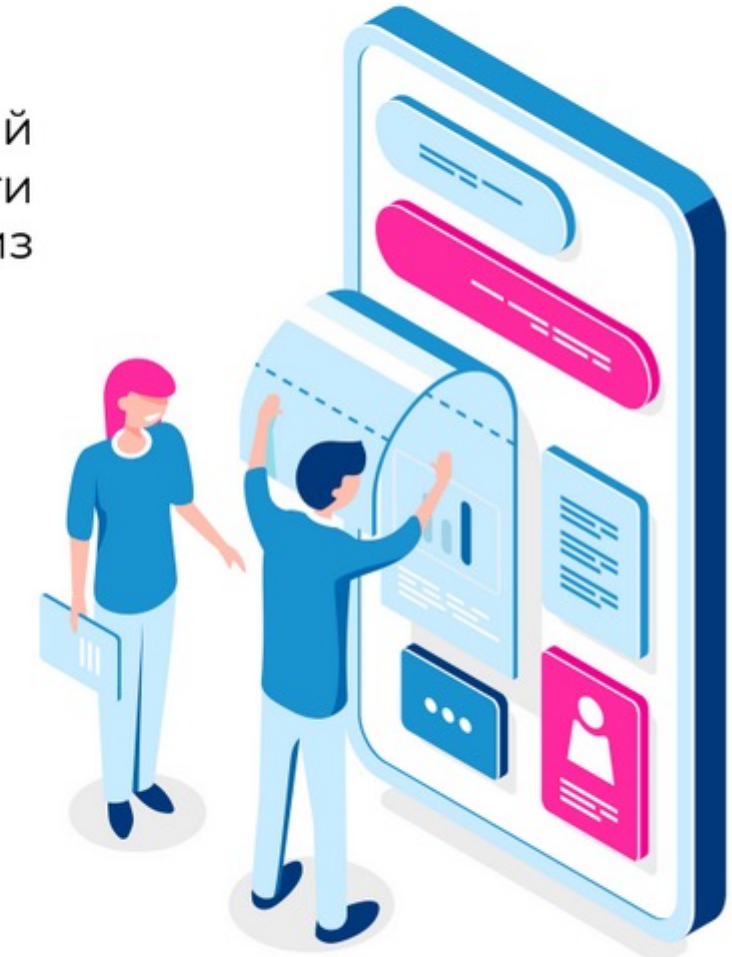
Место тестирования в процессе разработки ПО

Цикл разработки ПО.
Цикл тестирования ПО.
Классификация видов и направлений тестирования.

Цикл разработки ПО

Жизненный цикл ПО – это непрерывный процесс, который начинается с момента принятия решения о необходимости его продукта и заканчивается в момент его полного изъятия из эксплуатации.





Как вы думаете, закончился ли жизненный цикл операционной системы Windows XP?

Методологии разработки ПО тестирование

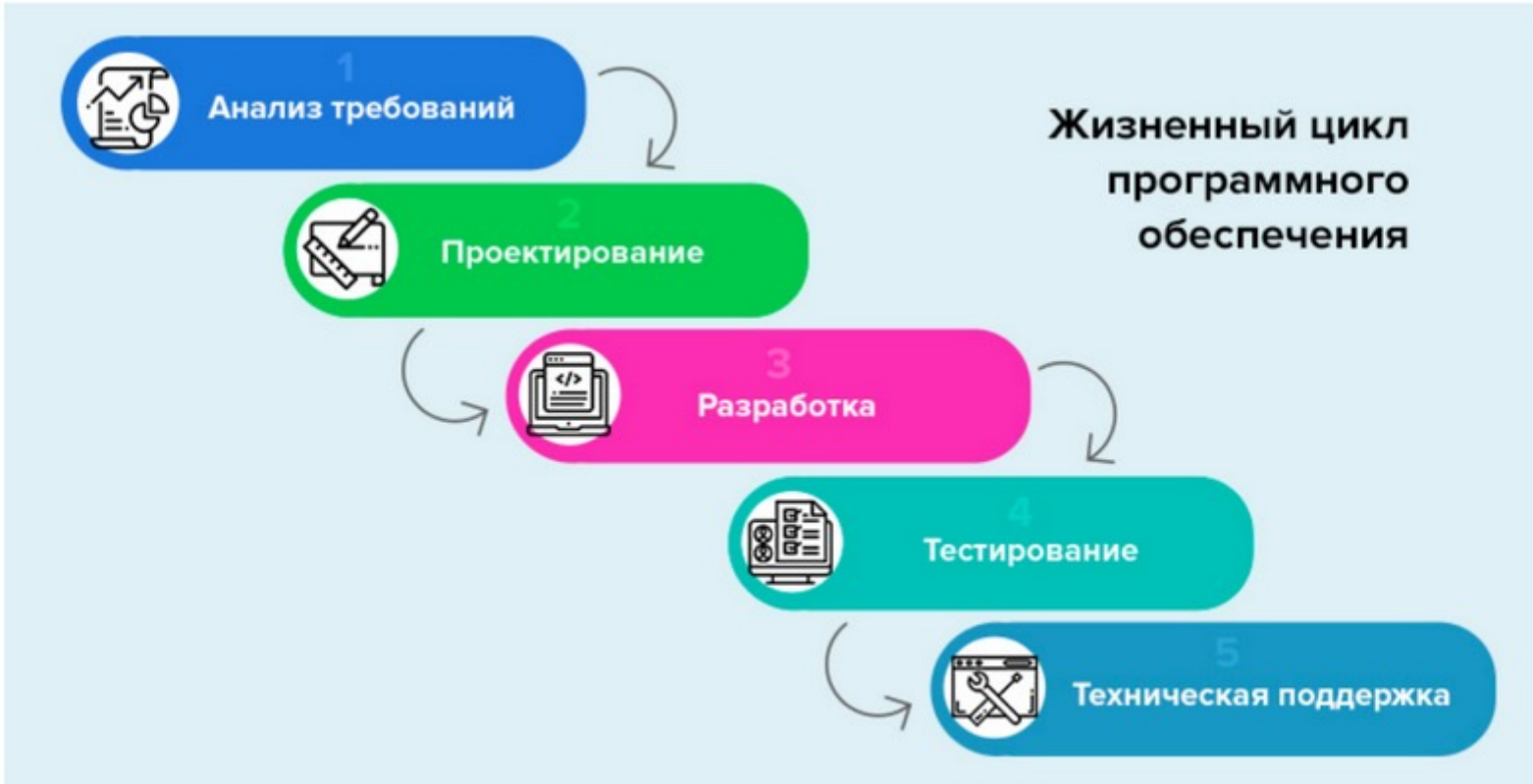
Методология – это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки ПО.

Методология определяет, как будет выполняться разработка. Существует много успешных методологий создания ПО.

Выбор конкретной **методологии зависит** от размера команды, от специфики и сложности проекта, от стабильности и зрелости процессов в компании, от личных качеств сотрудников и от предпочтений заказчика.

Каскадная модель (Waterfall model, модель «Водопад»)

При такой модели каждая из фаз проекта проводится единожды, следуя одна за другой. Для того, чтобы начать следующую стадию, необходимо полное завершение предыдущей.



Каскадная модель (Waterfall model, модель «Водопад»)

Плюсы:

- все стадии проекта выполняются в строгой последовательности;
- строгость этапов позволяет планировать сроки завершения всех работ и соответствующие ресурсы (денежные и человеческие);
- требования остаются неизменными в течение всего цикла.

Минусы:

- сложности при формулировке четких требований и невозможность их изменения;
- тестирование начинается только с середины развития проекта;
- до завершения процесса разработки пользователи не могут убедиться, качествен ли разрабатываемый продукт.

V-модель (разработка через тестирование)

Основной принцип V-образной

Суть этой модели состоит в том, что процессы на всех этапах контролируются, чтобы убедиться в возможности перехода на следующий уровень. Уже на стадии написания требований начинается процесс тестирования.

V-модель (разработка через тестирование)

Плюсы:

- строгая этапизация;
- минимизация рисков и устранение потенциальных проблем за счет того, что тестирование появляется на самых ранних стадиях;
- усовершенствованный тайм-менеджмент.

Минусы:

- невозможность адаптироваться к измененным требованиям заказчика;
- длительное время разработки (иногда длится до нескольких лет) приводит к тому, что продукт может быть уже не нужен заказчику, поскольку его потребности меняются;
- нет действий, направленных на анализ рисков.

Инкрементная модель (Incremental model)

При инкрементной модели (англ. increment – увеличение, приращение) программное обеспечение разрабатывается с линейной последовательностью стадий, но в несколько инкрементов (версий). Таким образом улучшение продукта проходит запланировано все время пока жизненный цикл разработки ПО не завершится.

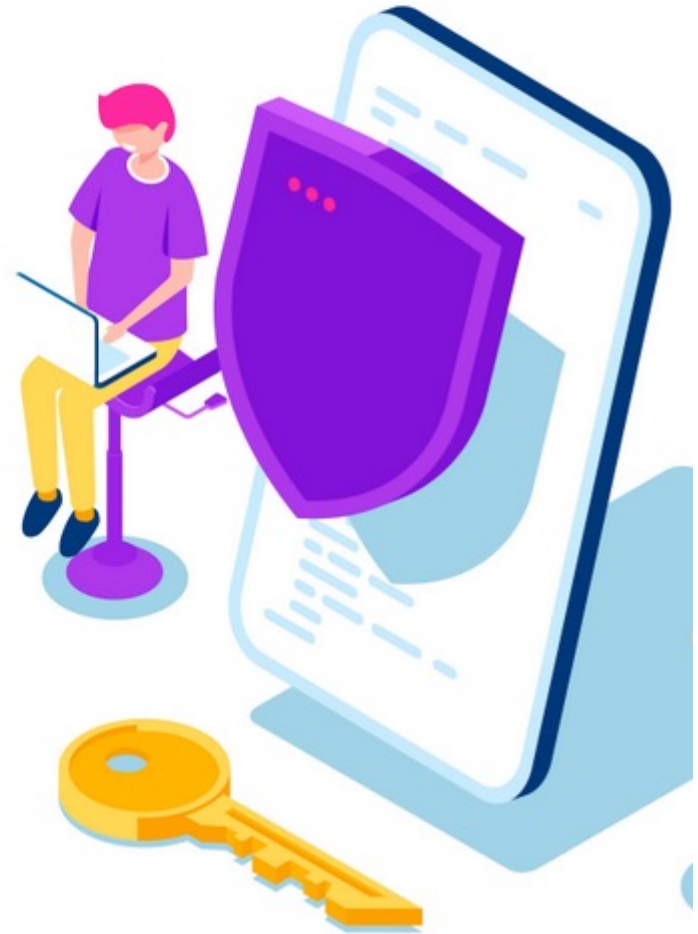
Инкрементная модель (Incremental model)

Плюсы:

- заказчик может дать свой отзыв касательно каждой версии продукта;
- есть возможность пересмотреть риски, которые связаны с затратами и соблюдением графика;
- привыкание заказчика к новой технологии происходит постепенно.

Минусы:

- функциональная система должна быть полностью определена в начале жизненного цикла для выделения итераций;
- при постоянных изменениях структура системы может быть нарушена;
- сроки сдачи системы могут быть затянуты из-за ограниченности ресурсов (исполнители, финансы).



Agile-подходы (гибкая методология разработки)

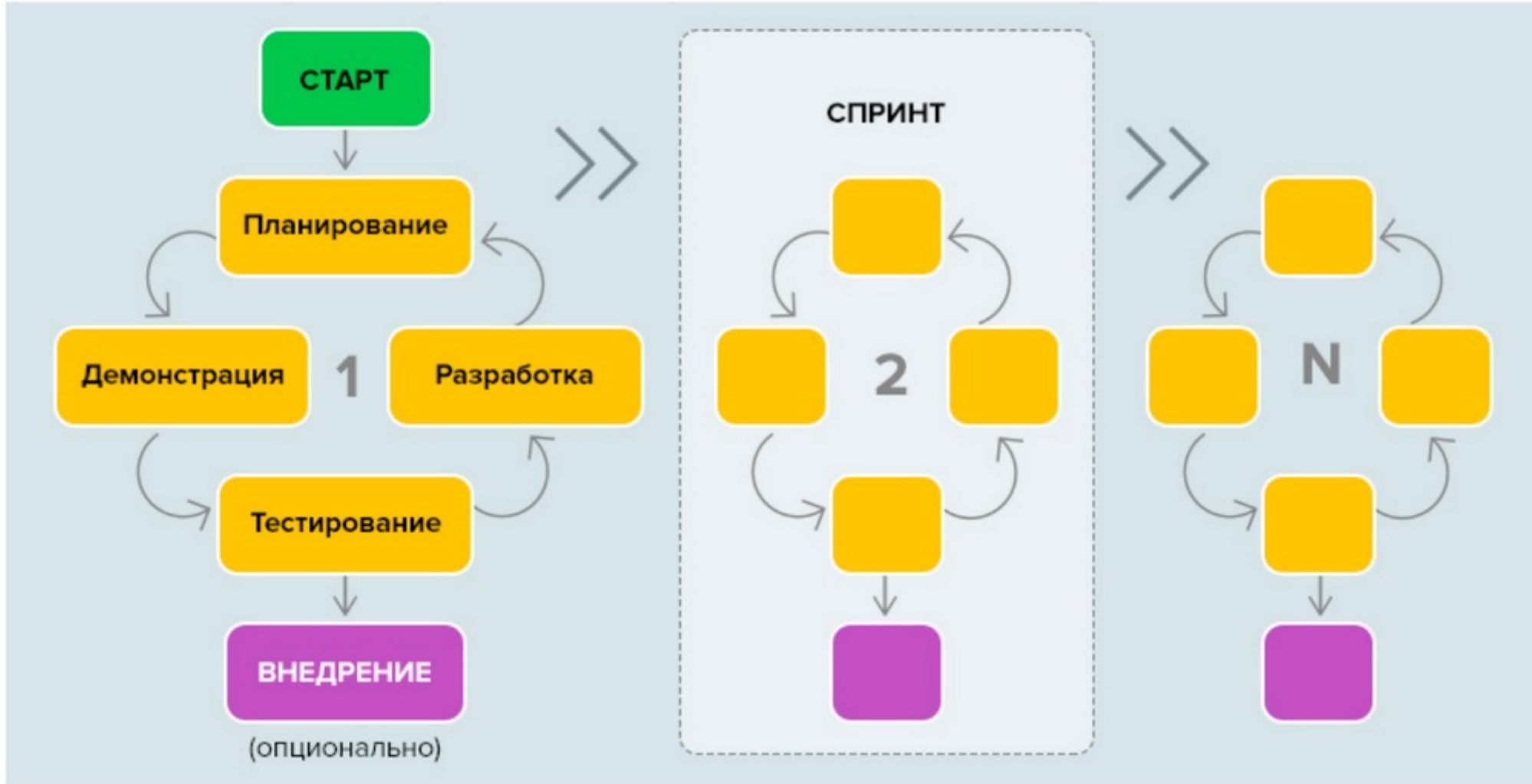
Agile — это семейство «гибких» подходов к разработке программного обеспечения. Такие подходы также иногда называют фреймворками или agile-методологиями.

В гибкой методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из Ее преимуществ.

К недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатратность и стоимость разработки.

Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

Agile-подходы (гибкая методология разработки)



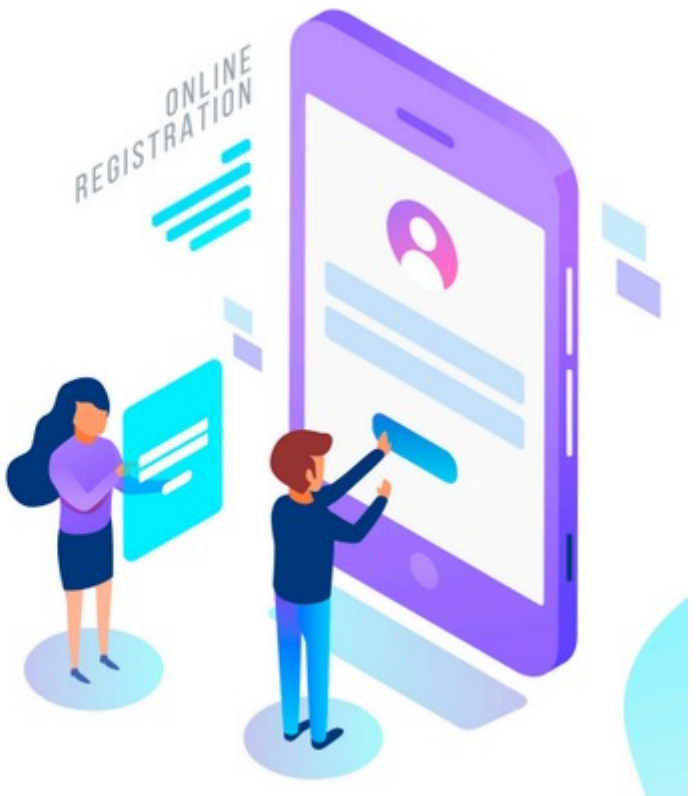
Agile-подходы (гибкая методология разработки)

Плюсы:

- быстрое принятие решений за счет постоянных коммуникаций;
- минимизация рисков;
- облегченная работа с документацией.

Минусы:

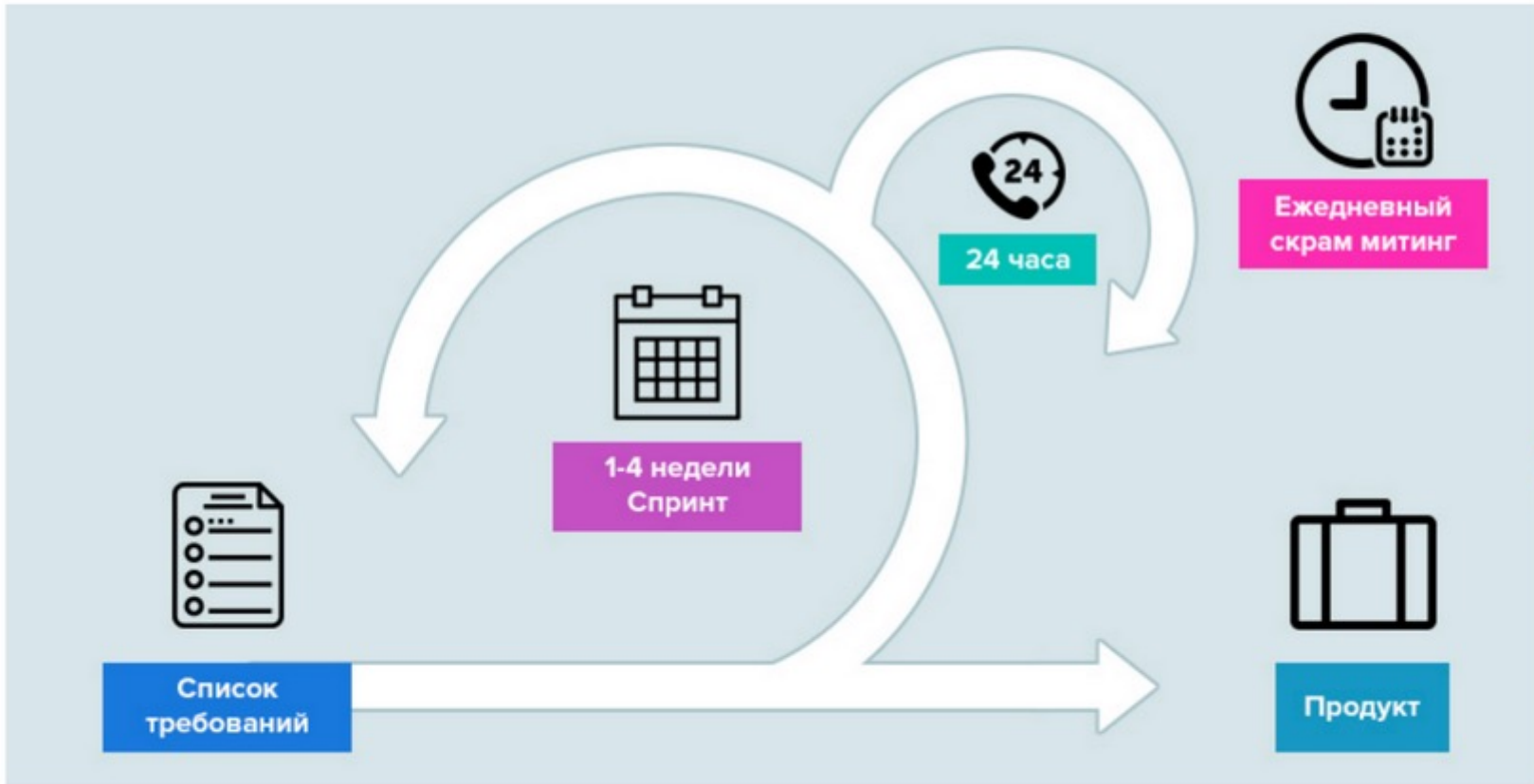
- большое количество митингов и бесед, что может увеличить время разработки продукта;
- сложно планировать процессы, так как требования постоянно меняются;
- редко используется для реализации больших проектов.



Скрам (Scrum)

Скрам – это гибкая модель разработки ПО, в которой делается акцент на качественном контроле процесса разработки.
Роли в методологии (Scrum Master, Product Owner, Team) позволяют четко распределить обязанности в процессе разработки. За успех Scrum в проекте отвечает Scrum Master и является связующим звеном между менеджментом и командой. За разработку продукта отвечает Product Owner, который также ставит задачи и принимает окончательные решения для команды.
Команда – это единое целое, в ней результаты оцениваются не по каждому отдельному участнику, а по тому, что получается в итоге у всех.
Спринты в данной методологии длятся от 1 до 4 недель. После каждого спринта команда предоставляет вариант законченного продукта.

Скрам (Scrum)


24
24 часа
Ежедневный скрам митинг
1-4 недели Спринт
Список требований
Продукт

Цикл тестирования

Этапы тестирования:

- Тестирование требований
- Уточнение требований

Анализ требований (Requirement Analysis)

- Определение цели тестирования
- Составление тест-плана
- Определение трудоёмкости тестирования
- Составление расписания тестирования

Планирование тестирования (Test Planing)

- Составление тест-кейсов, чек-листов
- Создание тестовых данных

Разработка тестов (Test Development)

- Выполнение тестов
- Создание отчётов по багам (баг-репорт)

Выполнение тестов (Test Development)

- Составление отчёта по тестированию
- Определение состояния ПО
- Оповещение заинтересованных лиц о результатах тестирования

Оценка результатов тестирования (Test Reporting)

Цикл тестирования

Анализ требований

На этапе анализа требований тестировщик:

- знакомится с ПО;
- узнает, для чего оно предназначено;
- знакомится с требованиями и вникает в проект;
- выполняет тестирование и уточнение требований.

После того, как вопросы по требованиям и функциональности ПО решены, происходит переход на этап планирования тестирования.

Цикл тестирования

Планирование тестирования
Этап сфокусирован на вопросах о том, что мы будем тестировать, какой функциональностью обладает ПО.

При создании тест-плана могут использоваться существующие шаблоны (RUP, IEEE 829) или свои собственные, принятые в компании.

В любом виде тест-план должен отвечать на следующие вопросы:

- **Что надо тестировать?**
- **Что будем тестировать?**
- **Как будем тестировать?**
- **Каковы тестовые окружения**, на которых необходимо проверять программный продукт?
- **Когда будем тестировать?**
- **Каковы риски и стратегии по их разрешению?**

Разработка тестов (тест-дизайн)

Следующий важный этап – **тест-дизайн**, когда проектируются и создаются тесты в соответствии с определенными ранее критериями качества и целями тестирования.

При проектировании тестов мы анализируем проектные артефакты, которые у нас есть – это может быть ТЗ, спецификации, планы, а в «особо запущенных» случаях – письмо от заказчика и соображения программистов.

Пример: Интернет-магазины, в отличие от специализированного ПО, должны обладать интуитивной понятностью. Если пользователю не понятно, как сделать заказ, он уйдет к конкуренту.

Важный момент при проектировании тестов – их фиксирование. В различных условиях тесты
фиксируются по-разному. Некоторые используют для этого специальные инструменты (MTM, Sitechk, TestLink) или приспосабливают существующие (Excel, Mind Map).

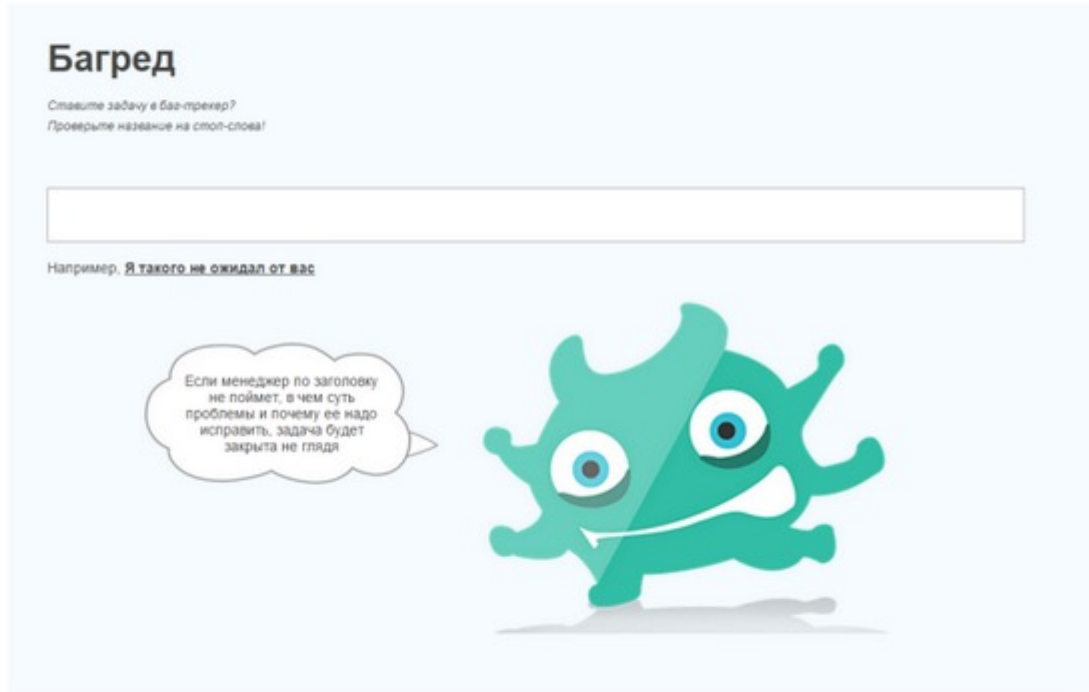
Выполнение тестов и оценка результатов

Когда тесты готовы, происходит их выполнение.
При выполнении тестов фиксируется версия ПО, на котором проводится тестирование, и результат прохождения теста.

Полученные результаты сообщаются остальным членам команды и руководителю проекта.

Если тесты провалились, то на их основе регистрируются ошибки в баг-трекере.

Выбрать баг-трекер и сделать обзор:
- JIRA
- Bontq
- YouTrack
- Redmine



Что тестируют кроме ПО?

- **Требования к ПО и документацию**;
- Дизайн;
- Код.

Что тестируют кроме ПО?

- Требования к ПО и документацию
- **Дизайн;**
- Код.

Тестирование дизайна

Тестирование дизайна ПО может проходить на разных этапах: **до, во время и после** реализации основного функционала (например, если решили полностью изменить дизайн).

При тестировании дизайна проверяется эргономичность, интуитивная понятность интерфейса.

Тестирование дизайна

Типичными проблемами при **тестировании дизайна** являются:

- **«Где я?»** Непонятно, где находится пользователь. Из-за очевидности на некоторых страницах могут убирать заголовки, названия, поясняющую информацию.
- **«Что делать дальше?»** Пользователю не очевидно, какие действия нужно совершить, чтобы достичь цели;
- **«Лавина».** На странице расположено слишком много всего, она перегружена и трудно воспринимается;

Тестирование дизайна

- **«Что я сделал?»** ПО не является интерактивным.

Пользователь должен получать от ПО ответную реакцию или корректное поведение, например:

- кнопочки визуально нажимаются;
- при изменении каких-либо данных пользователь должен видеть уведомления об успешном сохранении, удалении;
- при возвращении на предыдущее действие данные, выбранные или введенные пользователем, остаются на месте;
- Элементы интерфейса, выполняющие одинаковую функцию, должны выглядеть одинаково. Например, если есть кнопки «Отмена», «Закрыть», «Вернуть», то нужно выбрать одно название и выставить его по всему интерфейсу ПО.

Что тестируют кроме ПО?

- Требования к ПО и документацию;
- Дизайн;
- **Код.**

Тестирование программного кода

Модульное тестирование (юнит-тесты)
Интеграционное тестирование
Проверяют функциональность модуля и взаимосвязи между модулями на уровне кода. **Тестировщик не причастен.**

Ревью программного кода (Code review)
Во время такого тестирования разработчик отдает свой код на ревью (Code review) коллеге или лидеру команды разработки.
Это статический вид тестирования, тестировщик в нем не участвует.