Частное учреждение профессионального образования

«Высшая школа предпринимательства (колледж)»

(ЧУПО «ВШП»)

Кафедра информационных технологий

Внедрение и поддержка компьютерных систем

Принципы тестирования

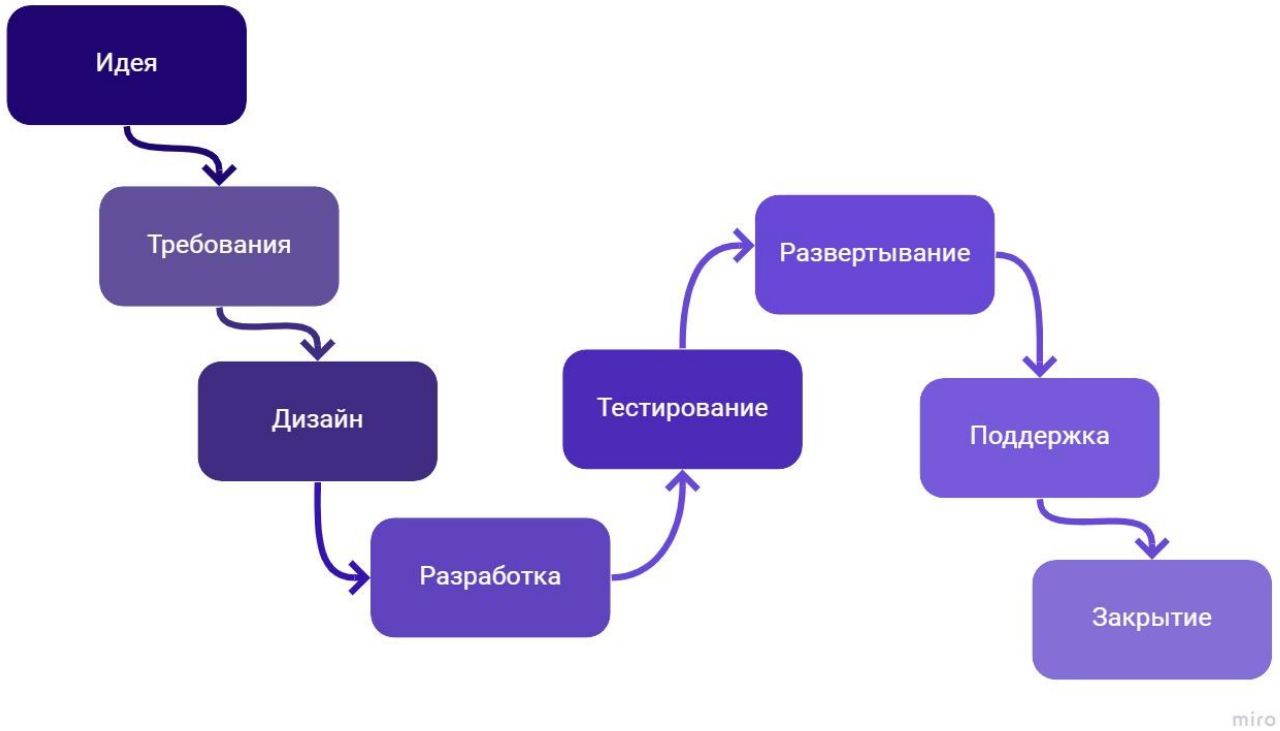
Принципы тестирования

Принцип 1 — Тестирование демонстрирует наличие дефектов
Принцип 2 — Исчерпывающее тестирование невозможно
Принцип 3 — Раннее тестирование
Принцип 4 — Скопление дефектов (принцип Парето)
Принцип 5 — Парадокс пестицида
Принцип 6 — Тестирование зависит от контекста
Принцип 7 — Заблуждение об отсутствии ошибок

Этапы жизненного цикла разработки

Этапы жизненного цикла разработки

Идея
Определение требований
Дизайн
  (архитектура) системы
Разработка
Тестирование
Развертывание
Поддержка
Закрытие



Модели и методологии разработки системы (ПО)

Модель разработки ПО описывает, какие стадии жизненного цикла проходит ПО и что происходит на каждой из них.

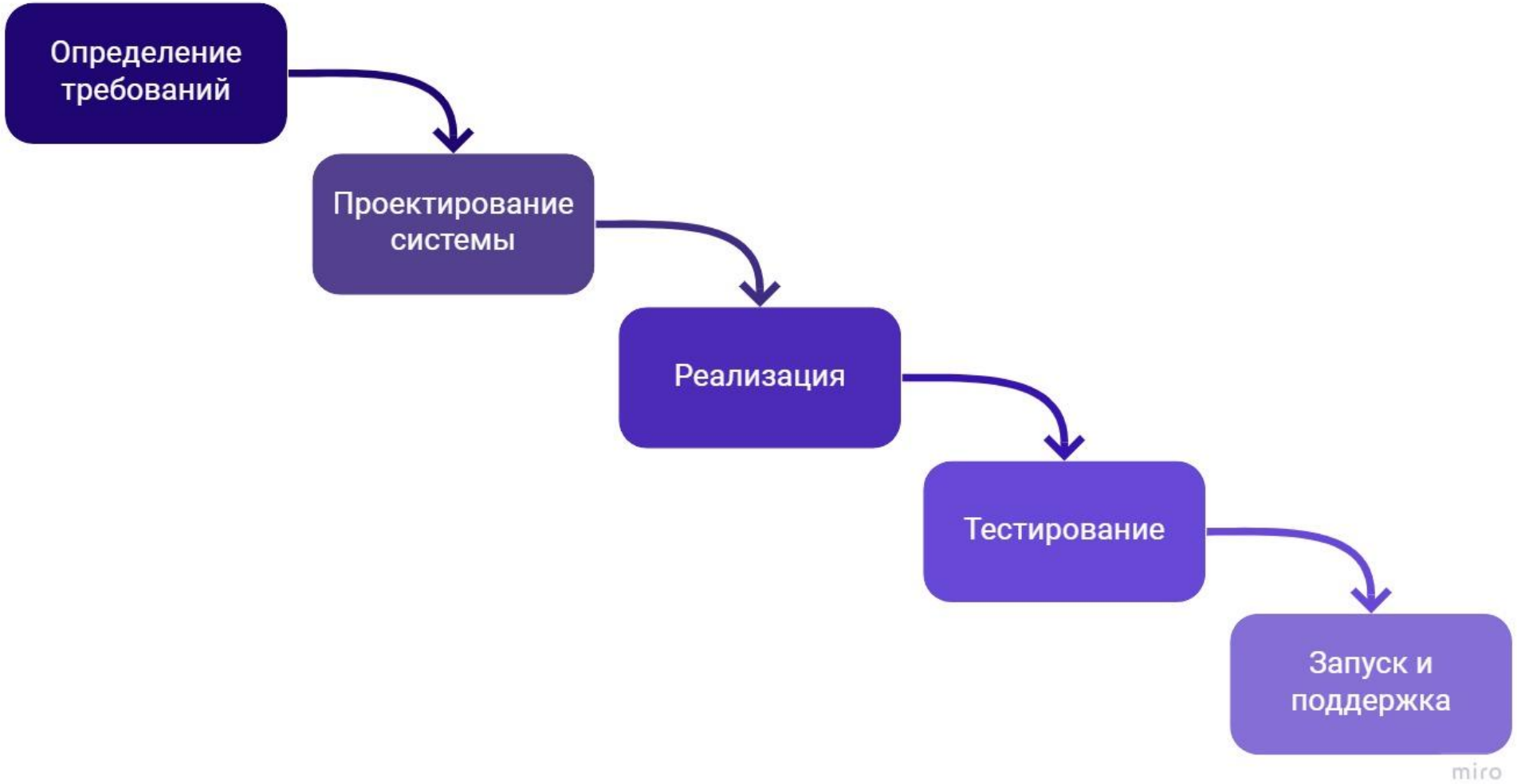
Методология разработки ПО описывает набор методов по управлению разработкой: правила, техники, принципы, которые делают разработку более эффективной.

Каскадная модель

Каскадная модель – модель, в которой процесс разработки выглядит как поток, переходящий от одной стадии к другой в строгом порядке, без возможности пропуска стадии или возврата назад.

Обычно, порядок следующий:
Определение требований
Проектирование системы
Реализация
Тестирование
Запуск и поддержка

Каскадная модель



Каскадная модель

Плюсы	1. Простая схема процесса 2. Не требует больших затрат на коммуникацию 3. Можно просчитать затраты ресурсов на реализацию
Минусы	1. Отсутствие возможности внесения изменений в проект до его окончания 2. Отсутствие возможности продумать все требования и нюансы наперед очень часто ведет к срыву сроков или уменьшению качества продукта (единственный этап, где можно “срезать” время – это тестирование) 3. Большие затраты на документацию (описание требований, дизайна системы)

Итеративная (инкрементальная) модель

Итеративная модель – модель, в которой работы выполняются параллельно с непрерывным анализом полученных результатов и корректировкой последующих этапов работы.
Начальное планирование
Разработка, которая выполняется циклически:
Планирование
Написание требований
Архитектура / дизайн
Реализация
Тестирование
Оценка
Запуск и поддержка

Итеративная (инкрементальная) модель

Итеративная (инкрементальная) модель

Плюсы	1. Меньше команда разработки (по сравнению с Waterfall) 2. Снижение рисков, снижение стоимости проекта, увеличение качества за счет раннего тестирования 3. Равномерная нагрузка на участников проекта 4. Реальная оценка текущего состояния проекта 5. Возможность быстрее принимать решения, адаптироваться под ситуацию
Минусы	1. Отсутствие возможности **точной** оценки стоимости всего проекта

Гибкая методология / Agile development

Гибкая методология / Agile development – это семейство процессов разработки, а не единственный подход в разработке программного обеспечения, который определяется Agile Manifesto.

Он не описывает конкретные практики, а определяет ценности и принципы, которыми руководствуются команды.

Гибкая методология / Agile development

Ценности гибкой методологии:

люди и взаимодействие важнее процессов и инструментов
работающий продукт важнее исчерпывающей документации
сотрудничество с заказчиком важнее согласования условий контракта
готовность к изменениям важнее следования первоначальному плану

Гибкая методология / Agile development

Плюсы	1. Все плюсы итеративной модели 2. Очень “плотное” взаимодействие команды
Минусы	1. Необходим высокий уровень квалификации команды для максимального эффекта 2. Отсутствие четкого плана, очень “поверхностное” описание требований к системе

Гибкая методология / Agile development

Множество фреймворков и методов разработки относятся к гибким методологиям, приведу самые известные:

Скрам (Scrum)
Канбан (Kanban)
Бережливая разработка (Lean development)
Экстремальное программирование (XP)

Наиболее распространенными являются Scrum и Kanban.

Скрам (Scrum)

Скрам (Scrum) – это фреймворк, предназначенный для разработки, поставки и поддержки сложных продуктов.

Основными элементами фреймворка являются:
Небольшие Скрам-команды (< 10 человек)
Четкие роли каждого участника команды (Product-owner, Scrum-master, Development Team)
События (собрания / митинги)
Артефакты (беклог продукта, беклог спринта)
Правила

Особенности Скрам:
Наличие предварительного плана, который будет изменяться по мере реализации проекта
Фиксированные по длительности итерации разработки (1-4 недели)
Фиксированные команды разработки
Постоянная работа с заинтересованными сторонами (включая клиента)

Канбан (Kanban)

В переводе с японского **Канбан** = «рекламный щит, вывеска». ☺

Канбан — это не методология. Это не замена **Scrum**. Это не метод разработки.

Канбан – это инструмент.

Канбан (Kanban)

Определение **Канбан** основано на понятии “поток” поставки ценности.

Поток – это движение ценности по всему процессу разработки продукта.

Канбан – стратегия оптимизации потока поставки ценности посредством процесса, который использует визуальную систему, имеющую ограничение незавершенной работы.

Канбан (Kanban)

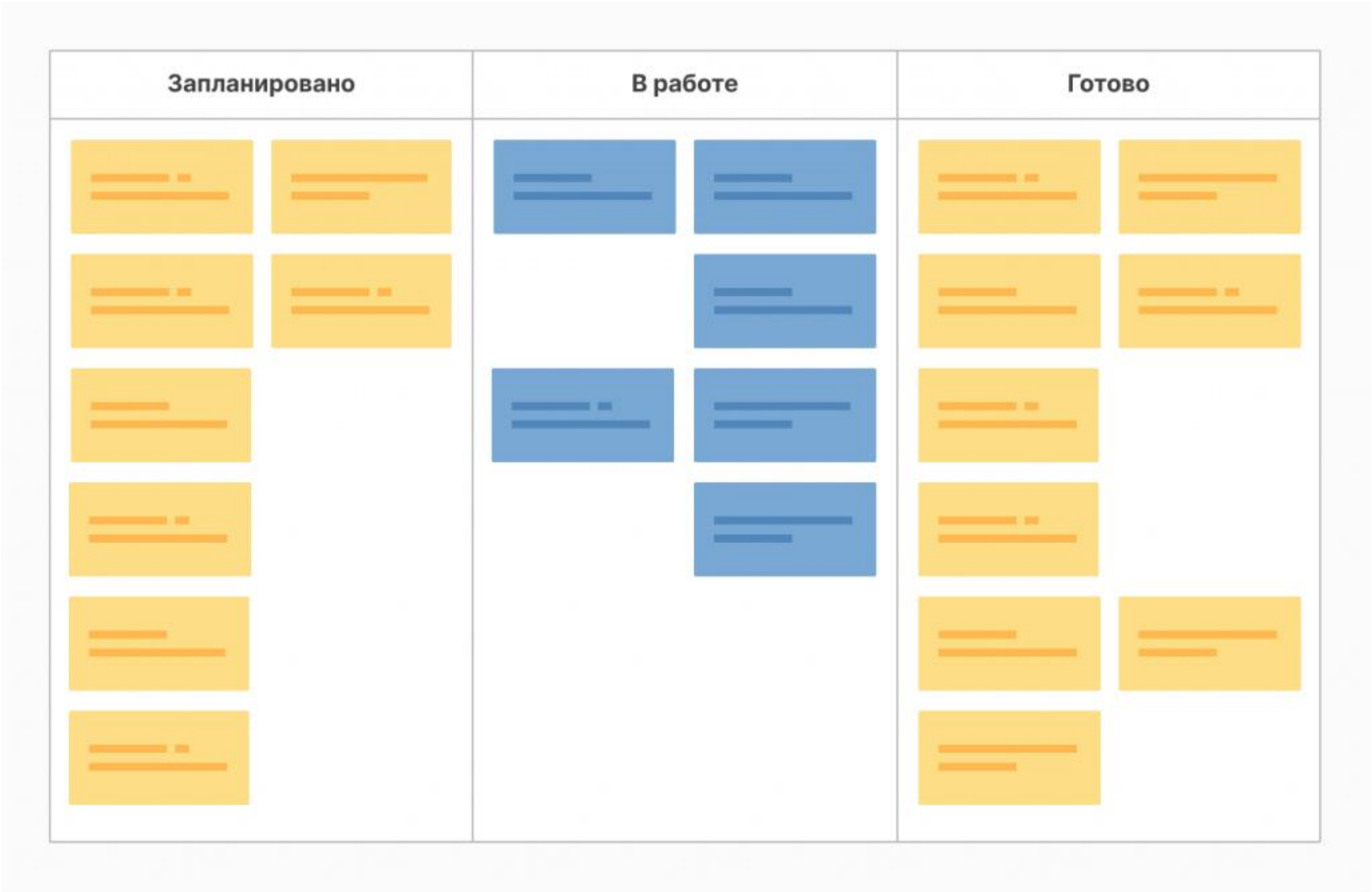
Чаще всего **Канбан** реализуется в виде “доски”, которая состоит из колонок и “задач”.

Колонки являются этапами процесса разработки.

Задачи передвигаются между этими колонками, и располагаются соответственно текущему этапу реализации.

Особенностью доски является то, что количество задач в каждой колонке — ограничено. Это условие позволяет “видеть” проблемы в процессе разработки и решать их.

Канбан (Kanban)



Фундаментальные понятия

Testing, Обеспечение качества (QA — Quality Assurance) и **контроль качества** (QC — Quality Control) — эти термины похожи на взаимозаменяемые, но разница между обеспечением качества и контролем качества все-таки есть, хоть на практике процессы и имеют некоторую схожесть.

Testing

Testing, либо тестирование - это самый нижний и первый уровень, который представляет собой проверку создаваемого программного продукта на соответствие требованиям к этому продукту.

Это рутинная работа: тебе выдали какой-то перечень

функциональности, ты их проверил, нашёл какие-то дефекты, описал их, предоставил разработчику, разработчик их пофиксил,

предоставил тебе исправленные баги на проверку.
Повторить.

QC (Quality Control)

Контроль качества системы (ПО) — анализ результатов тестирования и качества новых версий выпускаемого продукта.

К задачам **контроля качества** относятся:
проверка готовности ПО к релизу;
проверка соответствия требований и качества данного проекта.

QA (Quality Assurance)

Обеспечение качества системы (ПО)— изучение возможностей по изменению и улучшению процесса разработки, улучшению коммуникаций в команде, где тестирование является только одним из аспектов обеспечения качества.

К задачам обеспечения качества относятся:

проверка технических характеристик и требований к ПО;
оценка рисков;
планирование задач для улучшения качества продукции;
подготовка документации, тестового окружения и данных;
тестирование;
анализ результатов тестирования, а также составление отчетов и других документов.

Верификация и валидация

Верификация и валидация — два понятия тесно связаны с процессами тестирования и обеспечения качества. К сожалению,

их часто путают, хотя отличия между ними достаточно существенны.

Верификация

Это статическая практика проверки документов, дизайна, архитектуры, кода и так далее.
Процесс **верификации** проходит без запуска кода.

Верификация всегда отвечает на вопрос "делаем ли мы продукт правильно?".
Эта проверка связана с тем, что мы убеждаемся в том, что система хорошо спроектирована и безошибочно.
Верификация всегда происходит *ДО* **валидации**.

Валидация

Процесс оценки конечного продукта, когда необходимо проверить: соответствует ли программное обеспечение ожиданиям и требованиям клиента.
Это динамический процесс, в отличии от **верификации**. Этот процесс всегда включает в себя запуск кода программы и

отвечает на вопрос "делаем ли мы правильный продукт?".

Процесс **валидации** всегда происходит *ПОСЛЕ* **верификации**, поэтому на этапе **валидаци**и можно уловить ошибки, которые процесс **верификации** не позволил нам найти.

Этапы тестирования:

1. Анализ продукта
2. Работа с требованиями
3. Разработка стратегии тестирования и планирование процедур контроля качества
4. Создание тестовой документации
5. Тестирование прототипа
6. Основное тестирование
7. Стабилизация
8. Эксплуатация

Требования

Требования — это спецификация (описание) того, что должно быть реализовано.

Требования описывают то, что необходимо реализовать, без детализации технической стороны решения.

Атрибуты требований

Корректность — точное описание разрабатываемого функционала.
Проверяемость — формулировка требований таким образом, чтобы можно было выставить однозначный вердикт, выполнено все в соответствии с требованиями или нет.
Полнота — в требовании должна содержаться вся необходимая для реализации функциональности информация.
Недвусмысленность — требование должно содержать однозначные формулировки.

Атрибуты требований(1)

Непротиворечивость — требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.
Приоритетность — у каждого требования должен быть приоритет(количественная оценка степени значимости требования). Этот атрибут позволит грамотно управлять ресурсами на проекте.
Атомарность — требование нельзя разбить на отдельные части без потери деталей.
Модифицируемость — в каждое требование можно внести изменение.
Прослеживаемость — каждое требование должно иметь уникальный идентификатор, по которому на него можно сослаться.