

Гибкие методологии

Agile

- **Гибкая методология разработки (Agile software development)** – это манифест, содержащий основные ценности и принципы, на которых базируется разработка программного обеспечения.
- Манифест гибкой разработки программного обеспечения (**Manifesto for Agile Software development**) был разработан и принят в феврале 2001 года.
- Agile хорошо подходит для инновационных проектов, выполняемых стартапами.
- Agile также применяется и в таких крупных компаниях как **Google, Facebook, Альфа Банк, Правительство Москвы**.
- В меньшей степени он подходит для разработки больших проектов, например, банковских систем.
- Agile был придуман, т.к. водопадный подход (Waterfall) не позволял быстро разрабатывать программное обеспечение, удовлетворяющее нуждам заказчиков.

Гибкие методологии

- Существуют различные гибкие методологии, которые базируются на манифесте Agile. Например:
 - **Scrum** (термин означает схватку вокруг мяча в регби),
 - **Extreme Programming** (сокращенно **XP**) (экстремальное программирование),
 - **Lean** (бережливое программирование),
 - **Kanban** (Канбан).
- Эти методологии подразумевают итерационную разработку. В конце каждой итерации программный продукт готов к выпуску.

Ценности Agile

Манифест определяет четыре основные ценности:

- Люди и взаимодействие важнее процессов и инструментов.
- Работающий программный продукт важнее исчерпывающей документации.
- Сотрудничество с заказчиком важнее согласования условий контракта.
- Готовность к изменениям важнее следования первоначальному плану.

потребностей заказчика.

2. Изменение требований приветствуется на любой стадии разработки. Изменения обеспечивают заказчику конкурентные преимущества.
3. Работающий продукт следует выпускать как можно чаще.
4. На протяжении всего проекта разработчики и заказчик должны ежедневно работать вместе.
5. Над проектом должны работать мотивированные специалисты. Для этого необходимо создать условия, обеспечить поддержку и доверять.
6. Для эффективного обмена информацией с самой командой и внутри команды подходит непосредственное общение.
7. Основной показатель прогресса – работающий продукт.
8. Процесс разработки должен быть постоянным и устойчивым.
9. Внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
10. Минимизация лишней работы.
11. Только самоорганизующиеся команды предлагают лучшие архитектурные и технические решения. Члены команды совместными усилиями планируют проект и берут на себя коллективную ответственность за

Принципы Agile

Scram

- Scram - это одна из методологий гибкой разработки программного обеспечения.
- Scram используется для управления проектами по разработке программного обеспечения.
- Scrum может также применяться для организации работы команды, занимающейся технической поддержкой программного обеспечения (т.е. помощью в установке, настройке и использовании программного обеспечения).
- Этот подход был предложен в 1986 году авторами: Хиротака Такбэути и Икудзиро Нонака.
- Scrum обычно дополняют инженерными практиками из других гибких методологий (например, практиками разработки из экстремального программирования и практиками анализа и сбора требований из ICONIX).

- # Описание Scrum
- **Заинтересованные лица** будут использовать программный продукт или поддерживать его. Они разрабатывают **пользовательские истории (требования)**, которые описывают пожелания к продукту. Например, заказ товара должен выполняться через интернет.
 - **Владелец продукта** помогает заинтересованным лицам описывать их идеи в виде требований и упорядочивает перечень требований по очередности реализации. Этот перечень называется **бэклогом продукта**. Например, требование аутентификации пользователя в системе должно быть реализовано ранее требования наличия интерфейса для заказа продукта.
 - **Команда** (5-9 человек) реализует требования владельца продукта и работает как единая группа. Вклад в разработку каждого участника в отдельности не оценивается.
 - Раз в месяц (2-4 недели) **команда** берет верхнюю часть списка, уточняет и детализирует перечень требований, которые необходимо реализовать за месяц. Это называется **бэклогом спринта**.
 - Команда выполняет месячные **спринты**.
 - **Спринт** - это итерация в работе над проектом, в рамках которой выполняется месячный объем требований (**бэклог спринта**) по созданию, тестированию и демонстрации продукта.
 - Каждый день члены команды собираются на **ежедневный митинг** - короткую встречу (не более 15 мин), где рассказывают друг другу о состоянии дел, планах на сегодня и возникших проблемах.

Обзор спринта

- **Обзор спринта** - это демонстрация владельцу продукта и заинтересованным лицам работающего программного продукта, сделанного за спринт. Основная задача проведения обзора спринта заключается в получении обратной связи.
- По результатам демонстрации могут быть внесены изменения в бэклог продукта. Если изменения должны быть реализованы немедленно, то они вносятся в бэклог спринта.
- Демонстрация результатов работы не только мотивирует команду, но и подталкивает реализовывать задачи полностью.

Ретроспектива

- Через небольшое время после обзора спринта команда проводит **ретроспективу**, где присутствует scrum-мастер и, при необходимости, владелец продукта.
- Ретроспектива занимает от 30 минут до 4 часов.
- Каждый участник отвечает на два вопроса:
 - Что было сделано хорошо во время спринта?
 - Что можно улучшить?
- Scrum-мастер добавляет несколько улучшений (2-3), которые команда будет реализовывать, в бэклог продукта.

Практики XP

Практики экстремального программирования можно условно разделить на управленческие и инженерные.

- *Управленческие практики :*
- Частые небольшие релизы
- Заказчик всегда рядом
- 40-часовая рабочая неделя
- Коллективное владение кодом

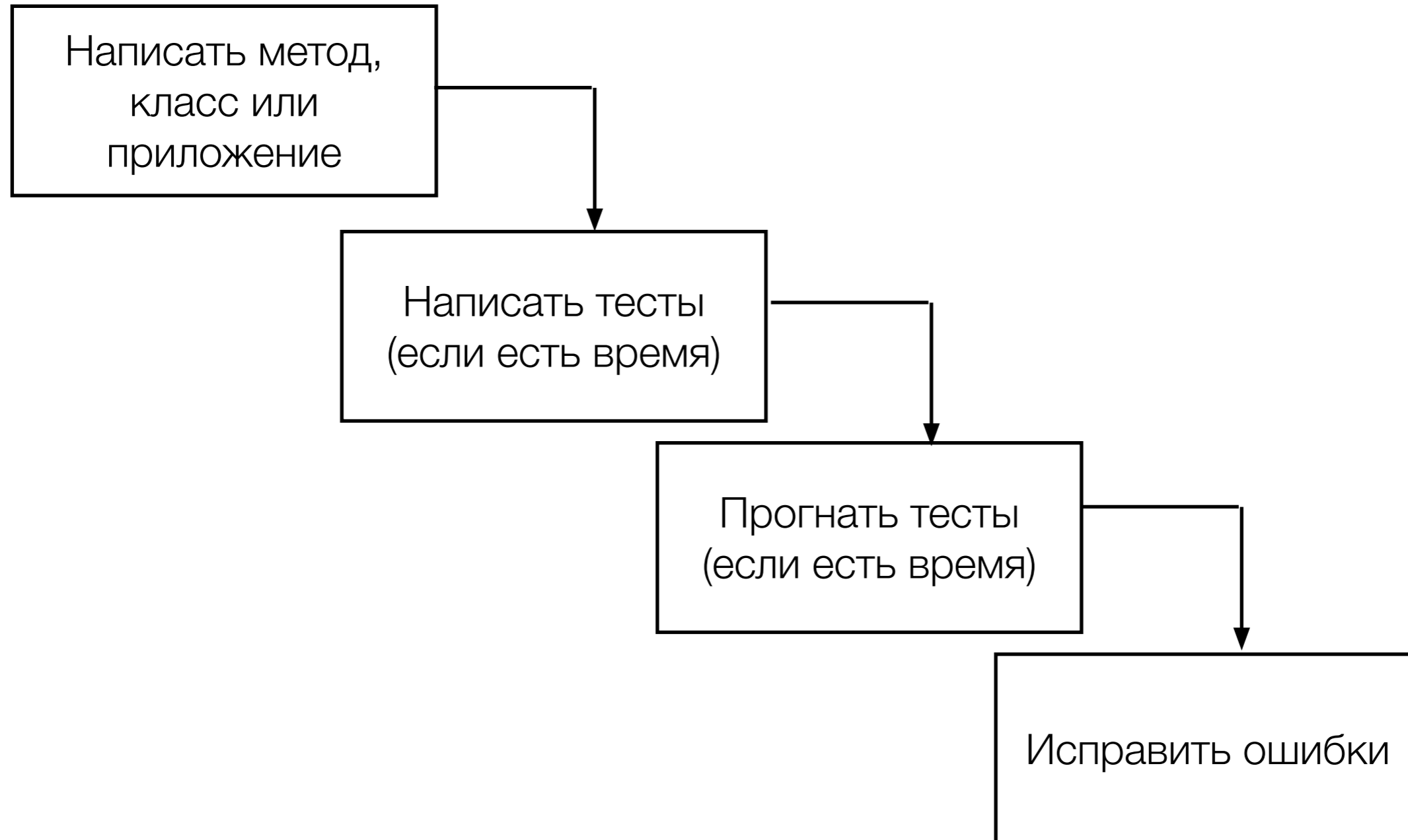
Инженерные практики .

- Разработка через тестирование
- Парное программирование
- Непрерывная интеграция
- Рефакторинг
- Простота
- Стандарт копирования

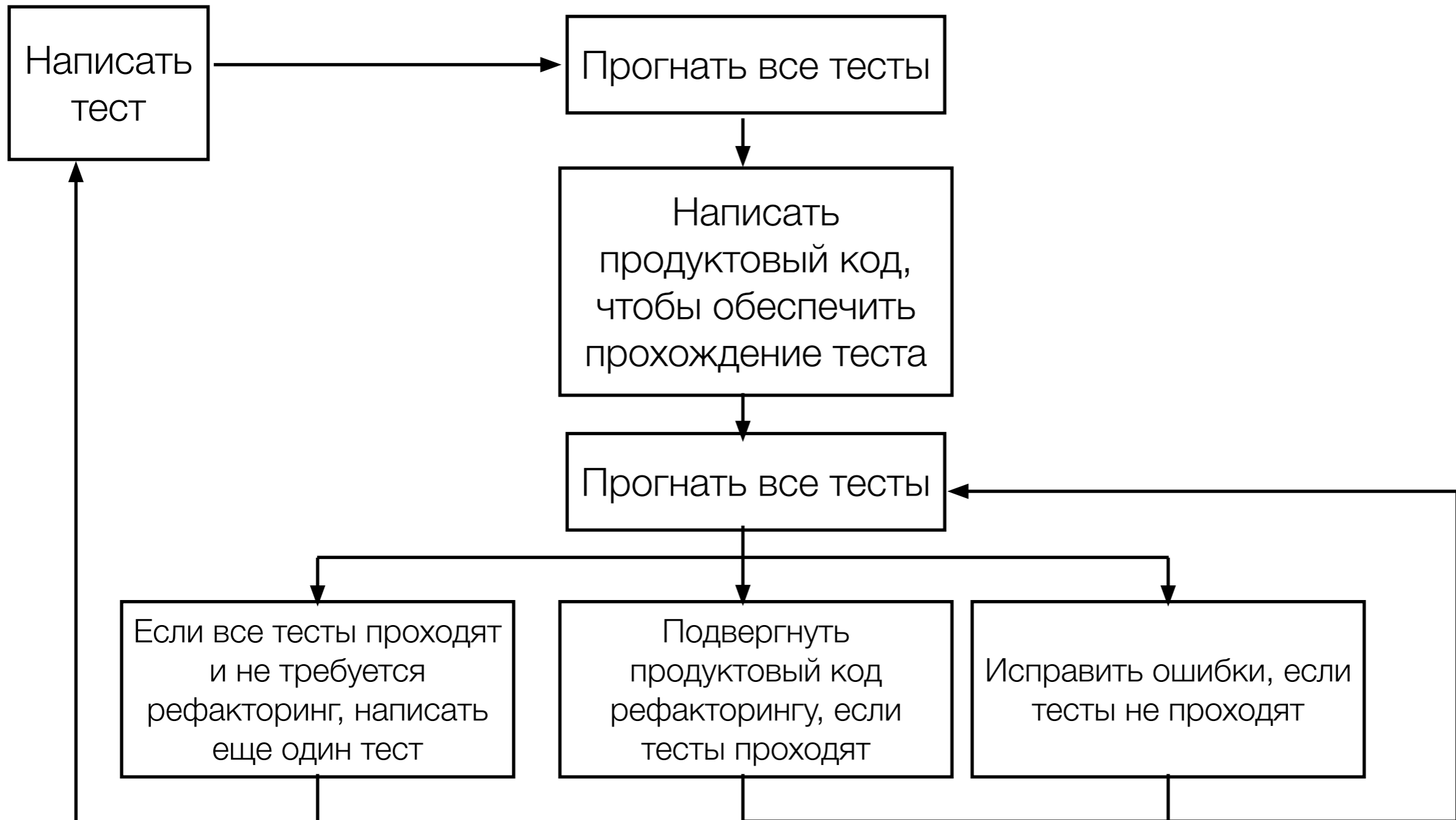
Разработка через тестирование

- TDD (test-driven development) - разработка через тестирование означает, что программист создает автоматизированный тест перед тем, как начинает писать код программы.
- Код считается работающим, когда тест проходит.
- Такие автоматизированные тесты называют "модульными", т.к. они предназначены для проверки отдельных модулей программы (классов, методов, функций, подпрограмм).
- Разработка через тестирование позволяет убедиться, что каждый отдельный модуль работает.
- Модульные тесты помогают также выявить и устранить зависимости между модулями, когда исправления в одном модуле могут вызвать ошибку в другом модуле.

Традиционный способ написания ТЕСТОВ



Разработка через тестирование



Парное программирование

- Два разработчика пишут код, находясь за одним и тем же компьютером.
- Один вводит код, а другой наблюдает.
- Оба разработчика ведут постоянное обсуждение кода.
- Преимущества парного программирования:
 - меньше ошибок, т.к. за процессом следят двое;
 - уменьшается усталость, т.к. программисты сменяют друг друга при вводе кода;
 - один постоянно наблюдает за другим, не давая ему словчить (например, упростить код, исключив из него некоторую функциональность).

Непрерывная интеграция

- Непрерывная интеграция реализуется за счет инструментов, позволяющих нескольким членам команды одновременно работать с одной версией исходного кода.
- Команда использует некоторую систему контроля версий (Version Control System, VCS), которая имеет централизованный набор файлов программы.
- Примерами систем контроля версий являются: Git, Mercurial.
- При работе с некоторым файлом один из программистов копирует его к себе (это называют помещением в **песочницу**). Программист правит файл в своей "песочнице" и по окончании работы с ним помещает его обратно в общее хранилище.
- Изменения в коде, произведенные разными программистами, могут конфликтовать друг с другом и приводить к ошибкам. Поэтому требуется постоянная интеграция (сборка) системы из отдельных частей и совместная проверка работы всей системы.