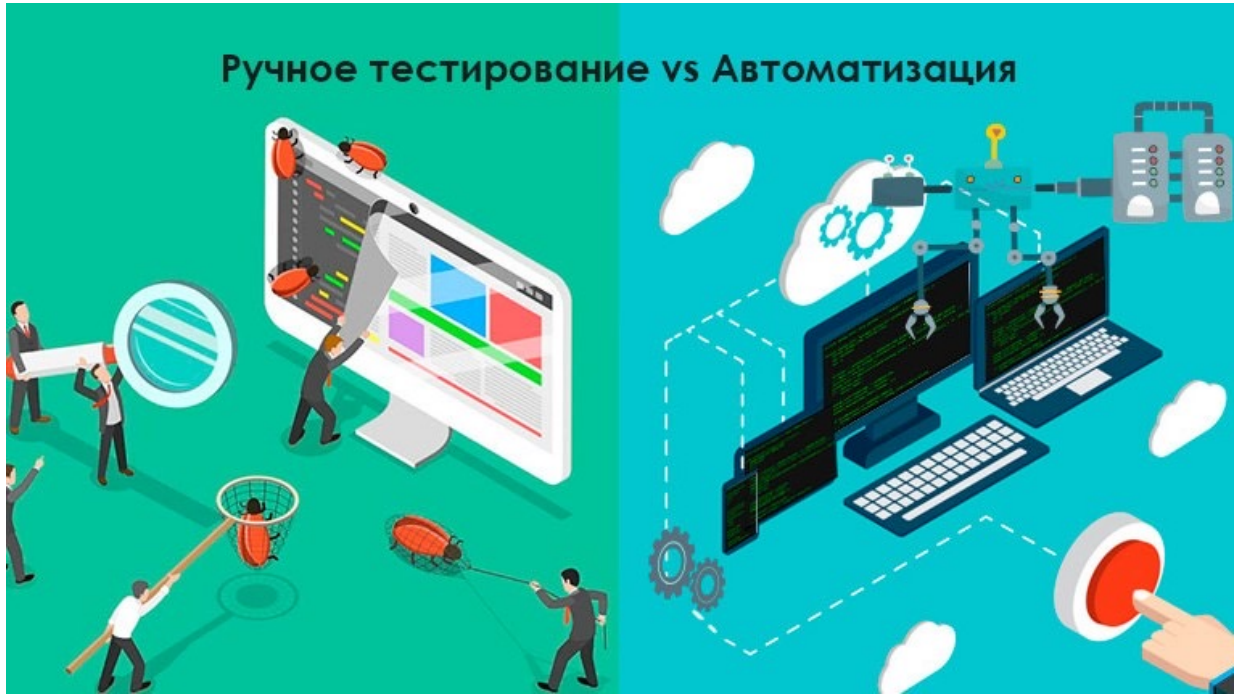
Модульное тестирование

Юнит-тестирование

Автоматизация тестирования

Существует существенная разница между ручным и автоматизированным тестированием



Важность автоматизации

- Традиционно тестирование требовало чрезмерной ручной работы через развертывание в тестовой среде, а затем тестов в стиле чёрного ящика, например, кликанием повсюду в пользовательском интерфейсе с наблюдением, появляются ли баги.

- Очевидно, что тестирование всех изменений вручную занимает очень много времени. Выход – автоматизация.

Пирамида тестирования

- Если серьёзно подходить к автоматическим тестам, то есть одна ключевая концепция: пирамида тестов. Её предоставил Майк Кон в своей книге «Scrum: гибкая разработка ПО».

- Это отличная визуальная метафора, наталкивающая на мысль о разных уровнях тестов. Она также показывает объём тестов на каждом уровне.

Пирамида тестирования

- Оригинальная пирамида состоит из трёх уровней:
  - Юнит-тесты
  - Сервисные тесты
  - Тесты UI

Что такое юнит?

- Разные программисты могут ответить на этот вопрос по разному.
- Если вы пишете на **функциональном языке**, то юнитами будут называться отдельные функции программы. Юнит-тесты вызывают определённые функции с нужными параметрами и возвращают некое значение.
- Если вы пишете на **объектно-ориентированном языке**, то юнитом может также называться в том числе и класс.

Юнит-тестирование

- Идея модульного тестирования состоит в том, что параллельно **основному компоненту** программы создаётся дополнительный **тестовый.**
- По результатам выполнения **тестового** компонента судят о правильности работы основного.

Юнит-тестирование

- Для юнит-тестирования существуют множество готовых фреймворков, которые облегчают разработку тестов и берут на себя значительную часть работы по анализу и предоставлению их результатов.

Плюсы юнит-тестирования

- Модульное тестирование позволяет быть уверенным в том, что из-за исправления одного бага у вас не появится другой.
- Юнит-тесты служат некоторым образом документацией к коду и путеводителем по нему. Тесты рассказывают о том, как та или иная функция должна реагировать на входные параметры.
- Применение юнит-тестов даёт более качественное отделение интерфейса от реализации.

Минусы юнит-тестирования

- Когда каждая функция тестируется по отдельности, нет гарантии того, что вместе они будут работать правильно.
- При плохо продуманной архитектуре проекта тесты могут мешать изменению кода.
- Тесты, успешно выполненные после внесения изменений в код, не дают гарантий, что ошибок нет.