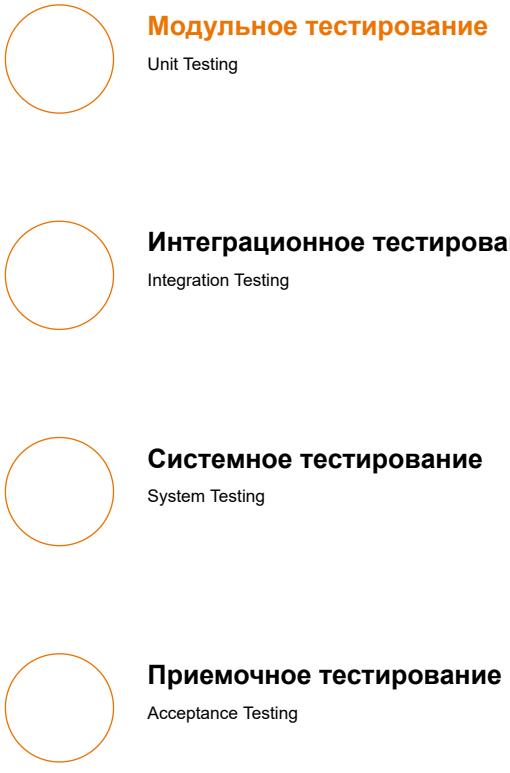
Что такое модульное тестирование?

Модульное тестирование (Unit Testing) - это тип тестирования программного обеспечения, при кото тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается проверить, что каждая единица программного кода работает должным образом. Данный вид тестир выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют част проверяют его работоспособность. Единицей для измерения может служить отдельная функция, ме процедура, модуль или объект.

В моделях разработки SDLC, STLC, V Model модульное тестирование - это первый уровень тестир выполняемый перед интеграционным тестированием. Модульное тестирование - это метод тестиро WhiteBox, который обычно выполняется разработчиком. На деле же из-за нехватки времени или ха разработчиков, иногда модульное тестирование приходится проводить QA инженерам.

Зачем нужно модульное тестирование?

Отсутствие модульного тестирования при написании кода значительно увеличивает уровень дефек дальнейшем (интеграционном, системном, и приемочном) тестировании. Качественное модульное тестирование на этапе разработки экономит время, а следовательно, в конечном итоге, и деньги.



1. Модульные тесты позволяют исправить ошибки на ранних этапах разработки и снизить затраты.
2. Это помогает разработчикам лучше понимать кодовую базу проекта и позволяет им быстрее и пр изменения в продукт.
3. Хорошие юнит-тесты служат проектной документацией.
4. Модульные тесты помогают с миграцией кода. Просто переносите код и тесты в новый проект и код, пока тесты не запустятся снова.

Как сделать модульное тестирование

Модульное тестирование делится на ручное и автоматизированное. И хотя программная инженерия
превосходство одного над другим, чаще, все-таки, используется автоматизированное.
При ручном модульном тестировании, как правило используется пошаговая инструкция.
Алгоритм же автоматизированного заключается в следующем:

- Разработчик записывает в приложение единицу кода, чтобы протестировать ее. После: они комме
  наконец, удаляют тестовый код при развертывании приложения.
- Разработчик может изолировать единицу кода для более качественного тестирования. Эта практи
  подразумевает копирование кода в собственную среду тестирования. Изоляция кода помогает вы
  ненужные зависимости между тестируемым кодом и другими модулями или пространствами дан
  продукте.
- Кодер обычно использует UnitTest Framework для разработки автоматизированных тестовых слу
  Используя инфраструктуру автоматизации, разработчик задает критерии теста для проверки корр
  выполнения кода, и в процессе выполнения тестовых случаев регистрирует неудачные. Многие ф
  автоматически отмечают и сообщают, о неудачных тестах и могут остановить последующее тест
  опираясь на серьезность сбоя.
- Алгоритм модульного тестирования:
  - Создание тестовых случаев
  - Просмотр / переработка
  - Базовая линия
  - Выполнение тестовых случаев.

Методы модульного тестирования

Ниже перечислены методы покрытия кода:

- Заявление покрытия
- Охват решений
- Охват филиала
- Состояние покрытия
- Покрытие конечного автомата

Для получения дополнительной информации см. Https://www.guru99.com/code-coverage.html.

Пример модульного тестирования: фиктивные объекты

Модульное тестирование основывается на создании фиктивных объектов для тестирования фрагме
которые еще не являются частью законченного приложения. Подставные объекты заполняют недос
части программы.

Например, у вас может быть функция, которая нуждается в переменных или объектах, которые еще
В модульном тестировании они будут учитываться в форме фиктивных объектов, созданных исклю
для целей модульного тестирования, выполненного в этом разделе кода.

Разработка через тестирование (TDD)

Модульное тестирование в TDD включает в себя широкое использование платформ тестирования. I
модульного тестирования используется для создания автоматизированных модульных тестов. Стру
модульного тестирования не являются уникальными для TDD, но они необходимы для него. Ниже
преимущества TDD:

- Тесты написаны перед кодом
- Можно положиться на тестирование фреймворков
- Все классы в приложениях протестированы
- Быстрая и простая интеграция

Преимущество модульного тестирования

- Разработчики, желающие узнать, какие функциональные возможности предоставляет модуль и к
  использовать, могут взглянуть на модульные тесты, чтобы получить общее представление об API
- Модульное тестирование позволяет программисту выполнить рефакторинг кода на этапе регресс
  тестирования и убедиться, что модуль все еще работает правильно. Процедура заключается в наб
  контрольных примеров для всех функций и методов, чтобы в случае, если изменение вызвало ош
  можно было быстро идентифицировать и исправить.

- Можем тестировать части проекта, не дожидаясь завершения других.

Недостатки модульного тестирования

- Не выявит всех ошибок. Невозможно оценить все пути выполнения даже в самых тривиальных п
- Модульное тестирование по своей природе ориентировано на единицу кода. Следовательно, он н
  отловить ошибки интеграции или ошибки системного уровня.

Используйте модульное тестирование в сочетании с другими видами тестирования.

Рекомендации по модульному тестированию

- Модульные тесты должны быть независимыми. В случае каких-либо улучшений или изменений
  требованиях, тестовые случаи не должны меняться.
- Тестируйте только один модуль за раз.
- Следуйте четким и последовательным соглашениям об именах для ваших модульных тестов
- В случае изменения кода в каком-либо модуле убедитесь, что для модуля имеется соответствующ
  пример, и модуль проходит тестирование перед изменением реализации.
- Пофиксите все выявленные баги перед переходом к следующему этапу (, как минимум в модели
  SDLC).
- Примите подход «тест, как ваш код». Чем больше кода вы пишете без тестирования, тем больше
  вам придется проверять на наличие ошибок в дальнейшем.

Резюме

Модульное тестирование может быть сложным или довольно простым, в зависимости от тестируем
приложения и используемых стратегий, инструментов и принципов тестирования. Помните, что мо
тестирование всегда становится необходимом на каком-то из уровней разработки продукта. Это уве