**Что такое TDD и как я его понимаю.**

Аббревиатура TDD расшифровывается как **Test-Driven Development (TDD)**, что переводится как «разработка через тестирование». Данный метод был введен в программирование в конце двадцатого века Кентом Беком как часть Extreme Programming.

Структура TDD, состоит из следующих пунктов, исходя из книги «*Test-Driven Development by Example»* и статьи Кента Бека «[Canon TDD](https://tidyfirst.substack.com/p/canon-tdd)»:

1. **Составление списка сценариев для новой функциональности.**

Эти сценарии описывают ожидаемые варианты поведения функциональности, которую мы хотим внести в проект.

1. **Написание самих тестов для элементов из списка сценариев.**

Данные тесты будут критерием правильности и работоспособности, вносимых нами изменений в проект.

1. **Проверка тестов.**

Запуск всех тестов, написанных в пункте №2. Причем тесты должны провалиться, чтобы исключить их неисправность.

1. **Написание простого кода, который способен пройти новые тесты.**

Данный код будет реализовывать часть функциональности, однако требует рефакторинга.

1. **Пропускание кода через набор тестов.**

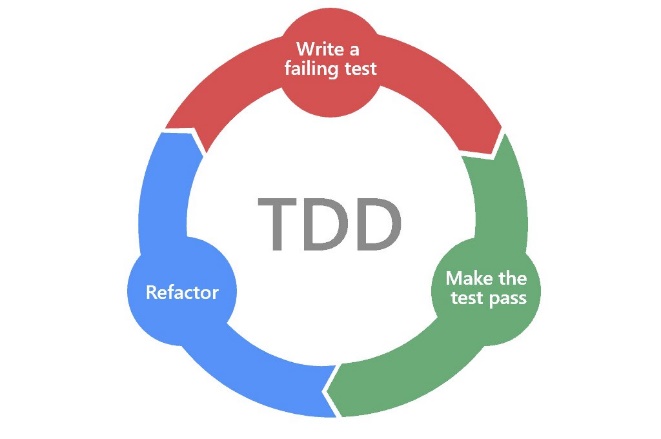
В случае полного прохождения тестов переходим к пункту №6, в противном случае вносим изменения в код до тех пор, пока пункт №5 не будет выполнен.

1. **Рефакторинг**.

Улучшение примитивного кода, сохраняя его способность полного прохождения набора тестов. В процессе рефакторинга происходит обновление набора тестов, тем самым зацикливая пункты №6 и №2.

Для понимания основной сути TDD можно рассмотреть более простую структуру, состоящую из трех пунктов:

**Красный-зеленый-рефакторинг.**

Красный – создание и проверка тестов, необходимых для реализации нового модуля.

Зеленый – реализация простейшего варианта модуля, способного пройти ранее созданный набор тестов.

Рефакторинг – улучшение модуля.

TDD можно использовать не только в разработке программного обеспечения, но и в создании различных продуктов, например, печатных плат.

По моему мнению, основной особенностью метода TDD является создание сначала наборов тестов, которые будут проверять не только работоспособность наших разработок, но и соответствие ранее поставленным требованиям. А уже потом непосредственно сама реализация.

Данная методология, как и любая другая имеет ряд своих преимуществ и недостатков. Все перечислять я не буду, но выделю те, которые считаю ключевыми.

Из плюсов могу отметить следующие:

* Повышение качества выпускаемого продукта.
* Уменьшение времени отладки и проверки внесенных изменений.
* Наличие работающего варианта продукта (демо), к которому мы можем откатиться в любой момент (У нас не возникнет ситуации, в которой выпущенный нами релиз полностью провалиться).
* Лучшее понимание самого проекта и возможных проблем, которые могут возникнуть в будущем.

К недостаткам стоит отнести:

* Увеличение объема работы и как следствие удорожание разработки.
* Появление ошибок, связанных с плохо написанными тестами и тестами с ошибками.
* Появление чувства уверенности в отсутствии ошибок в разрабатываемом продукте. Следует понимать, что TDD позволяет провести модульные тесты. Существует три вида тестов: модульные, интегрированные и системные. Модульные тесты проверяют работоспособность отдельно взятого модуля. Модулем может выступать отдельная взятая функция в программировании или периферия в электронных устройствах, например, LCD-экран, датчик влажности и т.д. Интегрированные тесты позволяют проверить связку нескольких модулей и их взаимодействие. Системные тесты с свою очередь осуществляют проверку всей системы в совокупности. И использую TDD, мы не сможем быть уверенными в отсутствии ошибок на интегрированном и системных уровнях.

В заключение, я хотел бы сказать, что TDD – это хорошая методика разработки, но в реальной жизни необходимо вносить соответствующие поправки. Например, следует понимать, что мы не сможем покрыть всё тестами. Мы можем стремиться к этому, но финансовые и временные ограничения остаются с нами и часто диктуют нам свои условия.