УДК 004.054

TEST-DRIVEN DEVELOPMENT – РАЗРАБОТКА ЧЕРЕЗ ТЕСТИРОВАНИЕ: ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Н.В. ГУРТОВЕНКО
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

В статье рассматривается метод разработки TDD. Анализируются его преимущества и недостатки.

Введение. Для создания успешного ИТ-проекта важным шагом является выбор методологии разработки, которая содержит свой подход к процессу создания программ в виде шагов, задач, действий. Таких методов существует немало, каждый имеет свои достоинства и недостатки, и выбор главным образом зависит от поставленной задачи. Один из них -метод Test-Driven Development – разработка через тестирование.

Основной раздел. TDD представляет собой процесс разработки программного обеспечения, требующий от программиста написать автоматизированный тестовый пример для требуемой функциональности (первоначально неисправного), затем добавить минимальный код, необходимый для прохождения теста и, наконец, реорганизовать код и убедиться, что автоматизированные тесты до сих пор проходят. В традиционных методах, вначале пишется код, а затем разрабатываются тестовые примеры для проверки кода. В TDD, тестовые примеры (как правило, называемые юнит-тесты), чтобы удовлетворить требование пишутся до того, как реализация началась.

TDD состоит из цикла«red–green–refactor».

ЭтапRed. Пишется тест, который представляет собой необходимое поведение системы, затем создается метод-заглушка, чтобы можно было быстро собрать проект. Тест компилируется, но не возвращает нужный результат. Появляется потребность в написании программы.

Этап Green. Пишется минимальный код, который бы позволил тесту выполниться верно. В действительности это значит, что пишется код без структуры, без дизайна, без каких-либо шаблонов проектирования. Теперь появляется потребность придать коду «красивый вид».

Этап рефакторинга. Изменяется внешняя структура кода, без изменения его внешнего поведения. Это разделение на методы, добавление элементов шаблона проектирования, создание дополнительных классов и т. д. Последовательность этапов цикла очень важна. Принцип метода «Test First» подразумевает, что пишется только код, абсолютно необходимый для успешного прохождения всех тестов.

Преимущества методологии:

- Уменьшение времени на отладку. Если не использовать методологию TDD, то потребуется меньше чистого времени на написание кода. Однако потраченное на тесты время окупается при отладке кода и отлове ошибок – а эта часть процесса разработки присутствует всегда, потому что почти невозможно написать код без единой ошибки и это нормально. Благодаря TDD не нужно гадать, где находится ошибка, протестированный код проще поддерживать.
- Удобство изменяемости. Покрытие тестами помогает избежать ситуации, когда при изменении кода в одном месте возникает ошибка в совсем другой части кода. В итоге это позволяет безопасно проводить изменение или рефакторинг кода, потому что если что-то пойдет не так, то тесты сразу обработают нештатную ситуацию.
- Тесты, особенно если они написаны в выразительном стиле, могут служить в качестве технической документации проекта, в которой описывается то, как код должен работать. Такая документация очень полезна для новых разработчиков в проекте, которые для того, чтобы включиться в работу, вначале должны разобраться с кодом. Тест в этом случае служит конкретным примером использования кода. Полное покрытие кода тестами дает огромную практическую документацию кода, отражающую реальное состояние системы.
- Модульность. Один из принципов разработки через тестирование предусматривает, что каждая функция выполняет определенную, небольшую часть работы, потому что просто невозможно протестировать «всемогущий» метод, который выполняет десяток функций в нескольких потоках. Все это вынуждает разделять программу на модули, чтобы удалось протестировать все ветви кода.
- Полноценные тесты. Существует большая разница между написанием обычных тестов и тестов по методу TDD. Когда тесты пишутся после реализации есть вероятность получить неполноценные тесты из-за того, что могут быть учтены не все сценарии работы метода. Например, может считаться, что написанный метод работает, так как он проходит тест, но на самом деле может оказаться, что в методе покрыты тестами не все условия. Использование TDD позволяет не случиться такому, потому что условие не может появиться в коде без теста.

• TDD является лучшим способом, чтобы гарантировать, что тесты на самом деле охватывают все требования, а не только код. Кроме того, TDD помогает не попасть в «ловушку»: добавление функциональности, которой клиенту на самом деле не нужно, но было бы неплохо иметь.

Недостатки:

• Возможность применить TDD имеется не всегда. Существуют задачи, которые невозможно решить только при помощи тестов. Например, это задачи в области безопасности данных и взаимодействия между процессами. Иногда бывает, что сложно сразу представить, как будет выглядеть работа модуля. Кроме того, невозможно решить с помощью TDD разработку баз данных, компиляторов и интерпретаторов языков программирования, невозможно автоматизировать тестирование графического интерфейса и распределенных объектов.

• Начальные требования не всегда могут быть понятны и правильно интерпретированы. В итоге можно получить ошибку в тесте, в коде и в понимании. Опасность ситуации заключается в том, что с виду кажется, что все работает правильно, ведь тесты проходят зеленый этап. На всем этом можно потерять массу времени.

• Необходимость поддержки тестов. База кода при стопроцентном покрытии тестами увеличивается почти в два раза. И кроме того всю эту базу необходимо документировать, поддерживать и проводить рефакторинг.

Заключение. Не смотря на существование большого количества мощных инструментов разработки, программирование по-прежнему остается сложной работой. Методология разработки через тестирование позволяет разделить работу на много маленьких частей и сконцентрировать внимание программиста на единственной задаче. Таким образом, методика TDD позволяет разделить процесс разработки на элементарные режимы, избавляя от монотонности, предлагая быстро переключаться между этими режимами, что повышает эффективность разработки.

ЛИТЕРАТУРА

1. Imprium.ru [Электронный ресурс]. – Режим доступа: https://imprium.ru/articles/test-based-development. – Дата доступа: 20.09.2020.
2. Proglib.io. Сайт о программировании [Электронный ресурс]. – Режим доступа: https://proglib.io/p/test-driven-development/. – Дата доступа: 20.09.2020.
3. Itvdn.com. Сайт о программировании [Электронный ресурс]. – Режим доступа: https://itvdn.com/ru/video/test-driven-development. – Дата доступа: 20.09.2020.