ТЕСТИРОВАНИЕ КАК СПОСОБ ПОВЫШЕНИЯ КАЧЕСТВА ПО

Юдин Дмитрий Геннадьевич

студент, Томский государственный университет систем управления и радиоэлектроники, РФ, г. Томск

Гатулин Ринат Рашидович

студент, Томский государственный университет систем управления и радиоэлектроники, РФ, г. Томск

На сегодняшний день проблема качества ПО выходит на первый план, широко обсуждается среди экспертов IT-индустрии в мире. Одним из наиболее распространенных методов улучшения качества программного обеспечения в настоящее время является тестирование.

Интерес к тестированию особенно бурно начал проявляться в 90-ые годы в США. Бурное развитие автоматизированных систем разработки программного обеспечения и сетевых технологий привело к росту рынка производства ПО и к более тщательному подходу в обеспечении качества и надежности разрабатываемых программ. Ситуация усугубилась тем фактом, что в настоящее время компьютеризация захватила многие аспекты человеческой жизни. И вопрос о качестве приобретает особую важность: сейчас это не только комфорт от использования качественных программ без дефектов, сегодня ПО управляет оборудованием в больницах, диспетчерскими системами в аэропортах, атомными реакторами, космическими кораблями и т.д. Сегодня тестирование стало обязательной частью разработки программного обеспечения, является самой популярной методикой повышения качества, подкрепленной многими исследованиями и богатым опытом разработки программного обеспечения. Тестирование направлено на обнаружение и устранение как можно большего числа ошибок. В итоге качество ПО возрастает. Тестирование улучшает не только качество ПО, но и снижает стоимость разработки, так как если в процесс разработки итеративным образом внедрить тестирование, это позволит раньше находить дефекты в коде, что в свою очередь снижает стоимость дальнейшей разработки ПО.

Существует множество видов тестирования:

1. Блочное тестирование. Тестирование полного класса, метода или небольшого приложения, выполняется отдельно от прочих частей системы.

2. Тестирование компонента. Тестирование класса, пакета, небольшого приложения или другого элемента системы, выполняемое в изоляции от остальных частей системы.

3. Интеграционное тестирование. Совместное выполнение двух или более классов, пакетов, компонентов или подсистем. Этот вид тестирования обычно начинают проводить, как только созданы два класса, которые можно протестировать, и продолжают до завершения работы над системой.

4. Регрессивное тестирование. Повторное выполнение тестов, направленное на обнаружение дефектов в программе, уже прошедшей этот набор тестов.

5. Тестирование системы. Выполнение ПО в его окончательной конфигурации, интегрированного с другими программными и аппаратными системами [2].

Тестирование обычно разделяют на две общие категории:

1. Тестирование методом черного ящика. Подразумевается, что программист не владеет сведениями о внутренней работе тестируемого элемента.

2. Тестирование методом белого ящика. Внутренняя реализация тестируемого элемента программисту известна [4, с. 494]. В результате программист получает некоторые преимущества.

В целом все виды тестирования позволяют покрыть готовый продукт тестами, но как показывают исследования тесты позволяют обнаружить около 60% ошибок в существующем коде. Что достаточно невысоко. Существует более эффективная методика по мнению экспертов в области разработки ПО [1]. Это создание программных продуктов основанное на повторении коротких циклов разработки, путем изначального написания тестов, а уже после разработка функционала программы в рамках одного цикла реализации. Эта технология называется – разработка через тестирование (Test Driven Development). В начале после написания теста, данный тест не будет проходить успешно, так как соответствующий ему код реализации еще на написан. Чтобы написать тест разработчик должен четко понимать предъявляемые к системе требования это отличает разработку через тестирование от методик, когда код уже написан. Она заставляет разработчика сфокусироваться на требованиях до написания кода. Пример цикла разработки по методологии TDD представлен на рисунке 1.

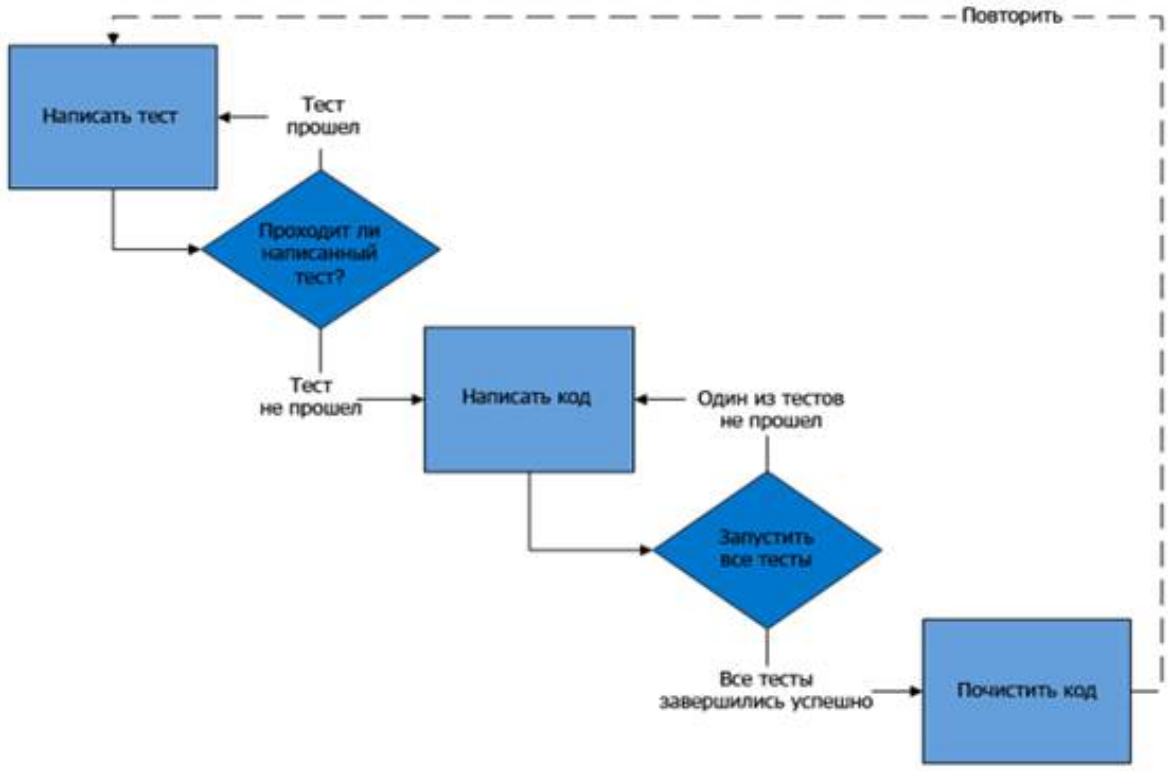

Рисунок 1. Цикл разработки по методологии TDD

Следуя методологии TDD, можно получить следующие преимущества:

1. Код полностью покрыт тестами, так как подразумевает изначальное написание теста.

2. Создавая тесты до написания кода класса, метода или чего-либо еще, программист заранее обдумывает возможности его использования, что положительно скажется на качестве

реализуемого модуля, так и на архитектуре проекта в целом.

3. Хорошие тесты позволяют заменить документацию [3, с. 114]. Пример теста можно видеть на рисунке 2.

```
describe("Video", () => {
        it("should return video title", () => expect(video.getTitle()).toBe(TEST_TITLE));
        it("should return video bitrate", () => expect(video.getBitrate()).toBe(TEST_BITRATE));
});
```

Рисунок 2. Пример теста для видео тега в веб-приложении

Существует три основных правила которым необходимо следовать при разработке по TDD методологии:

1. новый программный код реализуется только после того, как будет написан модульный тест, который не проходит;

2. пишется ровной такой объем кода модульного теста, который необходим чтобы этот тест не проходил;

3. пишется ровной такой объем нового реализуемого кода, который необходим, чтобы этот тест проходил [5, с. 186].

В целом методология TDD имеет много преимуществ. Данная технология разработки появилась относительно недавно, и поэтому ведутся дискуссии об эффективности данной методологии, но в последние годы можно увидеть положительную динамику в распространении использования TDD. Так как сообщество разработчиков ПО осознало выгоды использования данной методологии по сравнению просто написанием модульных, функциональных, интеграционных тестов.

В данной статье была рассмотрена важная часть разработки ПО – тестирование. На данный момент есть множество видов тестирования, и по оценкам экспертов покрытие кода тестами (интеграционными, модульными, функциональными и т.д.) не всегда позволяет с уверенностью говорить о высоком качестве программного продукта. Но создаются новые эффективные методики тестирования такие как TDD, которые позволяют с большей уверенностью говорить о качестве ПО, и с массовым внедрением таких методик, программные продукты, возможно, смогут стать более надежными и качественными.

Список литературы:

1. Качество программного обеспечения. Microsoft. – [Электронный ресурс] – URL: https://social.msdn.microsoft.com/Forums/ru-RU/af2ebf9f-c8f1-4062-a409-0d7819c1de8f/ (Дата обращения 30.11.2016).

2. Классификация видов тестирования. Habrahabr – [Электронный ресурс] – URL: https://habrahabr.ru/company/npo-comp/blog/223833/ (Дата обращения 28.11.2016).

3. Кент Б. Экстремальное программирование: разработка через тестирование. / Б. Кент – Издательство «Питер», 2003. – 224 с.

4. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ./ С. Макконелл – М.:

Издательство «Русская редакция», 2010. – 896 с.

5. Канер С. Тестирование программного обеспечения. / С. Канер – Издательство «ДиаСофт», 2001. – 544 с.