АНАЛИЗ МЕТОДОВ ТЕСТ-ДИЗАЙНА В ТЕСТИРОВАНИИ ПРОГРАММНОГО ПРОДУКТА

ПРОКАЗОВА Жанна Витальевна
магистрант
ФГБОУ ВО «Донской государственный технический университет»
г. Ростов-на-Дону, Россия

Проверка качества разрабатываемого программного обеспечения – это важнейшая фаза жизненного цикла разработки. В результате высокой конкуренции на рынке качественный продукт играет решающую роль при выборе клиента. Из-за этих условий в команде разработки и появилась такая роль как тестировщик программного обеспечения.
Ключевые слова: тестирование, тест-дизайн, качество, проверка, программный продукт.

В настоящее время набирает популярность такая профессия как тестировщик программного обеспечения. Это относительно молодая профессия, она появилась в процессе развития разработки программного обеспечения. Сейчас практически во всех командах есть тестировщик или отдел тестирования. Тестирование также развивалось в процессе развития информационных технологий. Из ручного тестирования появилось автоматизированное тестирование, что освободило тестировщиков от рутинной работы – проверки неизмененного функционала программного обеспечения (регрессионное тестирование). Регрессионное тестирование представляет собой вид тестирования, которое заключается в проверке того, что внесенное изменение кода программного обеспечения не оказало влияния на неизмененный функционал программного обеспечения.

В процессе работы тестировщика было выявлены 7 принципов, которые позволяют рационально отнестись к процессу тестирования. Выделяют следующие принципы:

1. Процесс проведения тестирования показывает только наличие дефектов, но не их отсутствия. Нельзя уверенно утверждать, что дефектов нет, они есть практически во всех проектах, но незначительные.

2. Невозможно провести полное тестирование, это невыполнимо. Исчерпывающее тестирование с применением всех возможных вариантов ввода и комбинаций недостижимо, но кроме тривиальных случаев.

3. Необходимо проводить тестирование на ранних этапах разработки программного обеспечения, чтобы исключить появление дефектов в технической документации до этапа кодирования.

4. Дефекты имеют свойства скапливаться в одном месте (модули). В разных модулях программного обеспечения содержатся различное количество дефектов, то есть плотность дефектов в разных местах может отличаться.

5. Парадокс пестицида. Если всегда проводить одни и те же тест-кейсы, в итоге данный набор не сможет находить новые дефекты. Во избежание данной ситуации необходимо периодически обновлять тестовые сценарии.

6. Тестирование напрямую зависит от контекста. Для тестирования разных программных обеспечений необходимо применять разные подходы в зависимости от контекста описания технической документации.

7. Заблуждение, что ошибок нет. Отсутствие дефектов при тестировании не говорит об идеальном программном обеспечении. Главным критерием готовности продукта является удобство пользования.

Для обеспечения максимального покрытия тестовыми сценариями в процессе тестирования были разработаны методы (техники тест-дизайна), чтобы при меньших затратах покрыть большую часть функционала программного обеспечения. Разберем и сравним тест-дизайн:

1. Классы эквивалентности. Данный метод дает возможность уменьшить количество тестовых сценариев. С помощью анализа предметной области объекта тестирования берется одно значение из созданного класса и подразумевается, что для всех значений в этом классе поведение будет одинаковым.

2. Анализ граничных значений. Данный метод немного похож на разделение на классы эквивалентности, но основное отличие между этими двумя методами заключается в том, что берутся значения на границе между выбранными классами.

3. Диаграмма переходов и состояний. Данный метод наглядно демонстрирует состояние программного обеспечения при разных вводных параметрах на разных этапах работы. При визуализации тестовых сценариев процесс тестирования выглядит нагляднее, воспринимать проще, чем текст.

4. Попарное тестирование. Данный метод наиболее сложный. В основе метода лежит комбинирование вводных значений. Процесс комбинирования дает возможность создать уникальные пары, тем самым позволяя тестировать программное обеспечение при наличии данных в разных сочетаниях.

По отдельности данные методы сокращают время для проведения тестирования программного обеспечения с помощью уменьшения возможных вводных данных. Наиболее лучшим вариантом будет комбинирование методов, это намного сократит время, и будет наиболее эффективно. Но нельзя забывать, что 100% покрытие тестовыми сценариями программного обеспечения практически невозможно добиться.

Но при попытке достичь 100% покрытия тестами программного обеспечения может потребоваться огромное количество времени. В большинстве случаев это не стоит потраченного времени.

Таким образом, можно сделать вывод, что тестирование увеличивает знание о качестве программного продукта, тем самым позволяя увеличить это качество в случае несоответствия. А техники тест-дизайна дают возможность при сокращении времени на проведение тестирования увеличить покрытие тестовыми сценариями, тем самым за более короткий промежуток времени проверить большую часть программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. *Короткий Д.А.* Автоматизация тестирования программного обеспечения автоматизированных комплексов скользящей опалубки // Вестник Донского государственного технического университета. – 2012. – № 12(3). – С. 37-43.

2. Немного о простом. Тест-дизайн. Часть 1. 2022. – URL: https://habr.com/ru/post/462553/ (дата обращения: 17.10.2022).

3. Теоретические основы тестирование. 2022. – URL: https://clck.ru/UMAEj (дата обращения: 17.10.2022).

4. Тестирование программного обеспечения. 2022. – URL: https://clck.ru/umaf4 (дата обращения: 17.10.2022)

5. Тест-дизайн и техники тест-дизайна (Test Design and Software Testing Techniques). 2022. – URL:h ttps://vladislaveremeev.gitbook.io/qa_bible/test-dizain/test-dizain-i-tekhniki-test-dizaina-test-design-and-software-testing-techniques (дата обращения: 17.10.2022).