МИНИСТЕРСВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(СПбГУ)

Математико-механический факультет

ЭССЕ

на тему

«Место и роль тестирования в реализации качества программного продукта на примере методологий Waterfall, Agile и V-model»

по курсу

Разработка и стандартизация программных средств
и информационных технологий

Выполнила студентка
группы 371: Долгополова Мария Андреевна

Научный руководитель: КИЯЕВ Владимир Ильич

Оценка _____

Санкт-Петербург
2019

Сложно представить себе ситуацию, когда процесс разработки обходится без какого-либо тестирования. Особенности современного ПО (сложность архитектуры и используемых алгоритмов, высокие требования к скорости, безопасности и надежности) приводят к тому, что пренебрежительное отношение к тестированию отрицательно сказывается на качестве создаваемого продукта.

В ходе тестирования специалисты по контролю качества проверяют, выполняет ли ПО все ожидаемые от него функции и удовлетворяет ли нефункциональным требованиям (требования к безопасности, эффективности по времени и используемым ресурсам, совместимости, стабильности, устойчивости к отказам, удобству использования). Поэтому для обеспечения высокого качества продукта включение тестирования в жизненный цикл разработки ПО является обязательным.

Основные преимущества внедрения тестирования в процесс создания программных продуктов:

- Тестирование на ранних стадиях разработки снижает стоимость исправления ошибок.
- Тщательное тестирование обеспечивает высокое качество продукта. Благодаря этому продукт становится конкурентоспособным и привлекательным для пользователей.
- Среда, в которой разрабатывается продукт, отличается от той, в которой он будет использоваться. Тестирование позволяет проверить ПО в условиях, схожих с реальными.

В зависимости от выбранной методологии разработки, способы организации тестирования и его место в жизненном цикле создания ПО могут отличаться. Рассмотрим процесс тестирования на примере методологий Waterfall, Agile и V-model.

Waterfall

Создание программного продукта состоит из нескольких стадий:

1) Инициализация
2) Анализ
3) Проектирование
4) Программирование
5) Системное тестирование
6) Внешнее тестирование
7) Релиз
8) Поддержка

Водопадная модель предполагает последовательное прохождение стадий, каждая из которых должна быть завершена до начала следующей. Системное тестирование продукта начинается только после того, как разработка завершена или почти завершена. Но тестирование в каком-либо виде присутствует почти на всех этапах.

Планирование

На этапе планирования происходит анализ требований, определение функциональных характеристик программного продукта.

В это время тестируются идеи. Аналитики, специалисты по маркетингу, руководители проекта, специалисты по контролю качества анализируют требования, черновики проектных документов. Результаты этой деятельности могут привести к значительному пересмотру существующих планов.

Цель тестирования требований – получить ответы на вопросы:

– Адекватны ли эти требования?

– Полны ли они?

– Совместимы ли требования между собой?

– Выполнимы ли они?

– Разумны ли они?

– Поддаются ли они тестированию?

Тестирование требований позволяет выявлять ошибки на ранних этапах, что приводит к снижению стоимости их исправления. Хорошо описанные требования позволяют лучше оценить объем работы и проработать техническое задание. За счет этого минимизируются риски и снижается общая стоимость разработки.

Тестировщики на стадии планирования начинают разрабатывать план и стратегию системного тестирования.

Проектирование

Происходит проработка внешнего дизайна и внутренней структуры (программная архитектура, организация данных, алгоритмы) продукта.

Как и на этапе планирования, тестируются идеи. Но теперь они лучше формализованы и описаны более подробно.

Цель тестирование на стадии проектирования – ответить на вопросы:

– Соответствует ли проект требованиям?

– Полон ли проект?

– Достаточно ли он реалистичен?

На этом этапе полезно создавать и тестировать прототипы. Это помогает оценить качество, удобство использования создаваемого продукта и его коммерческие перспективы. Прототип сравнивается со схожими по назначению программными продуктами, что позволяет предложить варианты по его улучшению.

Благодаря тестированию прототипа можно сэкономить время и сократить затраты, так как проект будет тщательно проработан с учетом требований еще до того, как команда разработчиков приступит к написанию кода.

На этапе проектирования тестировщики продолжают работать с тест-планом и тест-стратегией, пишут тест-кейсы, чек-листы, трассибилити матрицу.

Программирование

На этапе программирования планируется и настраивается тестовое окружение, тестировщики продолжают разработку тест-кейсов и чек-листов.

Программисты проводят юнит-тестирование и код-ревью своего кода. Юнит-тесты позволяют провести проверки на уровне самого кода, проверяя работоспособность отдельных функций и методов. Код-ревью нужно для того, чтобы определить степень соответствия кода реализации регламентированным нормам и обеспечить его прозрачную поддерживаемость в будущем.

Также на этой стадии начинается тестирование уже написанных частей системы.

Системное тестирование

Тестирование каждой версии системы состоит из нескольких этапов:

1) Smoke tests (приемочное тестирование)

   При поступлении новой версии продукта тестировщики сначала проверяют, пригодна ли она к дальнейшему тестированию. Если она сразу «падает» или не выполняет базовую функциональность, то дальше тестировать нет смысла.

   Приемочные тесты должны проверять только самые основные функции продукта и быть достаточно короткими для того, чтобы проводить их на каждой новой сборке.

   Так как одни и те же тесты нужно проводить много раз, smoke-тестирование часто автоматизируется.

2) Компонентное тестирование

   Должна быть протестирована вся функциональность (полное покрытие функциональности можно контролировать с помощью трассибилити матрицы или чек-листов), проведено тестирование пользовательского интерфейса, надежности, скорости.

   Существует 2 способа организации тестирования:

   - Восходящее тестирование – сначала тестируются отдельные элементы системы и только потом – их совместная работа. Если ошибка проявляется при совместной работе предварительно протестированных модулей, значит дело в их интерфейсе.

- Нисходящее тестирование – сначала тестируется самый верхний уровень иерархии, а от него тестировщик постепенно спускается вниз.

Для тестирования используются тест-кейсы. Их можно группировать:

По степени позитивности сценариев:

- Позитивные тест-кейсы – сценарии, предполагающие нормальное использование и работу системы.
- Негативные тест-кейсы – сценарии, проверяющие ситуации, связанные с потенциальными ошибками пользователей или дефектами в системе.

По объекту тестирования:

- Функциональные тест-кейсы – проверяют функциональность продукта.
- Нефункциональные тест-кейсы – проверяют скорость, надежность, безопасность, совместимость, пользовательский интерфейс.

3) End-to-end (сквозное тестирование)

E2E – тестирование всей системы целиком с точки зрения пользователя. То есть тестируются не отдельные функции, а весь процесс работы с системой от начала до конца. Во время E2E-тестирования нужно проверить самые важные и часто используемые сценарии.

Тест-кейсы для E2E должны согласовываться с бизнес-аналитиком или заказчиком.

4) Регрессионное тестирование

Регрессионное тестирование – это тестирование, направленное на обнаружение дефектов в уже протестированных участках системы.

Применяется для того чтобы:

- Убедиться, что выявленная ошибка полностью исправлена
- Убедиться, что исправляя одну часть программы, программист не испортил другую.

Регрессионные тесты проводятся после каждого изменения в системе. В этом случае для тестирования выбираются тест-кейсы, проверяющие часть ПО, к которой принадлежат изменения, и тест-кейсы, проверяющие старую функциональность, которая зависит от измененной части ПО.

В конце процесса системного тестирования проводится финальная регрессионная проверка – запускаются все или самые важные тест-кейсы.

Регрессионные тесты нужно проводить много раз, поэтому они часто автоматизируются для экономии времени.

5) Подготовка TSR

   После проведения всех системных тестов нужно подготовить Test Summary Report. Это документ, в котором описаны все тестовые активности и результаты, которые были получены после проделанной работы.

   TSR важен для принятия решения о продолжении тестирования или изменения его стратегии. Часто TSR используется для того, чтобы ознакомить заказчика с результатами тестирования. В этом случае для дефектов, которые пока не исправлены, нужно обязательно указать предполагаемую дату их исправления и обходной путь.

Внешнее тестирование

На этом этапе ПО тестирует заказчик. Специалисты по контролю качества помогают заказчику в тестировании, документируют и передают команде разработки найденные дефекты, объясняют заказчику степень их критичности, демонстрируют обходные пути.

Внешнее тестирование включает в себя следующие процедуры:

1) System Integration Testing
   Проверяется взаимодействие ПО с другими системами. Этот процесс помогает обнаружить дефекты программного интерфейса.
2) User Acceptance Testing

User Acceptance Testing – тестирование конечными пользователями. На этом этапе отдельно проверяются кейсы, относящиеся к разным функциональностям, разным частям системы.

3) End-to-end testing
   Тестируется весь процесс работы с системой.

Релиз

Специалисты по контролю качества продолжают поддерживать заказчика в процессе тестирования. В случае нахождения дефектов принимается решение о том, исправлять их сейчас или включить исправления в следующую версию.

Поддержка

Даже после релиза остается необходимость в тестировании, проводимом на этапе эксплуатации и поддержки. Пользователи могут использовать систему необычным способом, работать в окружении, совместимость с которым не тестировалась ранее. Поэтому всегда есть вероятность обнаружения новых дефектов и возникновения проблем у пользователей.

Agile

В Agile-методологии в приоритете не следование плану, а актуальные потребности пользователя, готовность к изменениям. Можно вносить изменения в требования в процессе разработки.

Разработка проходит через ряд циклов — спринтов. Спринт (итерация) — это фактически отдельный проект, где разрабатывают фрагмент программы, улучшают или добавляют новые функциональности. На каждой итерации происходит анализ требований, проектирование, разработка и тестирование.

Методы и инструменты для тестирования и обеспечения качества, используемые в Waterfall, применяются и в Agile. Отличие в том, что на каждой итерации присутствуют все этапы процесса разработки и тестирования. То есть функциональность тестируется в конце спринта, во время которого она был разработана, а не в конце всего проекта. Также в конце каждого спринта необходимо провести регрессионные тесты для проверки частей системы, разработанных ранее. Таким образом, с каждым спринтом объем кода, который нужно протестировать, увеличивается.

Часто в Agile-проектах применяются следующие методы контроля качества:

- Непрерывная интеграция
- Парное программирование
- Код-ревью
- Разработка через тестирование
- Рефакторинг

Тестовая документация в Agile менее формальная и подробная, чем в Waterfall. Например, вместо тест-кейсов используются чек-листы, часто отсутствует трассибилити-матрица, тестовый план пишется в упрощенном формате и может меняться в процессе разработки. После завершения процесса тестирования каждого спринта пишется краткий отчет по тестированию, а после тестирования последнего спринта подготавливается общий документ – Test Summary Report.

Несмотря на менее формальный подход к тестированию, этот процесс играет важную роль в разработке. Если на каком-то спринте не будут обнаружены присутствующие дефекты, они могут повлиять на разработку следующих

частей системы. Так, с каждым новым спринтом стоимость исправления этих дефектов будет увеличиваться.

Цели применения Agile-методологии разработки и тестирования:

- Каждый спринт – маленький этап, для которого тестирование и анализ рисков обеспечить проще, чем для всего продукта.
- Гибкость – готовность к изменению требований на любом этапе разработки. В случае обнаружения дефекта в требованиях его исправление обойдется дешевле, чем в других методологиях.
- Раннее создание работающего ПО.
- Методология Agile применима в ситуациях, когда на начальной стадии разработки известны не все требования.

V-model

Особенность V-model - направленность на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования. Эта методология применяется, если необходимо особо тщательное тестирование, например, когда предъявляются очень высокие требования к качеству продукта.

В V-методологии каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования. Для каждого уровня создается тестовый план, определяются ожидаемые результаты, а также критерии начала и окончания тестирования.

- На этапе обсуждения проекта с заказчиком разрабатывается тестовый план и тестовые сценарии для приемо-сдаточного тестирования. Цель этого вида тестирования – оценить пригодность системы для использования и степень ее соответствия ожиданиям заказчика.
- На этапе формализации и детализации требований ведется подготовка к функциональному тестированию.
- Фаза проектирования архитектуры параллельна подготовке к интеграционному тестированию (проверка взаимодействия компонентов системы друг с другом).
- Во время программирования разрабатываются модульные тесты для проверки работоспособности отдельных частей системы.

Таким образом, в V-методологии приёмо-сдаточные испытания основываются на бизнес-требованиях заказчика, функциональное тестирование – на желаниях заказчика и функциональных требованиях, интеграционное тестирование — на требованиях и архитектуре, модульное тестирование — на требованиях, архитектуре и алгоритмах.

Цели применения такого подхода к тестированию:

- Минимизация рисков **(**V-образная модель делает проект более прозрачным и повышает качество контроля проекта путём стандартизации промежуточных результатов, что позволяет выявлять отклонения в проекте и риски на ранних этапах жизненного цикла).
- Повышение и гарантии качества (промежуточные результаты могут быть проверены на ранних стадиях. Универсальное документирование облегчает понятность и проверяемость).

- Уменьшение общей стоимости проекта (ресурсы, требуемые для разработки, могут быть заранее просчитаны и проконтролированы).
- Повышение качества коммуникации между участниками проекта (уменьшаются неточности в понимании между заказчиком и командой разработки).

Заключение

Объем и время проведения тестирования отличается в разных методологиях. Однако, независимо от выбора методологии, тестирование и обеспечение качества играет важную роль в процессе разработки и должно присутствовать на протяжении всего жизненного цикла создания продукта.

От того, насколько тщательно проведено тестирование, зависит процесс и результат разработки. Если тестирование хорошо спланировано, применяется на протяжении всего жизненного цикла разработки ПО, участники процесса ответственно относятся к контролю качества, выделяется достаточное количество ресурсов на проведение тестов, то конечный продукт более стабилен, надежен и предоставляет функционал, удовлетворяющий заказчика.

Вместо того, чтобы нагружать команду тестировщиков непосредственно перед запуском продукта, нужно тщательным образом провести планирование проведения тестовых мероприятий на самых ранних стадиях работы над проектом. Проведение тестов, специфичных для определенной стадии разработки, позволяет сэкономить значительное количество времени и усилий, поскольку раннее обнаружение ошибок значительно сокращает затраты на их исправление.

Использованная литература:

Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: пер. с англ. [Текст]/ Сэм Канер, Джек Фолк, Енг Кек Нгуен - Киев: ДиаСофт, 2001. – 538 с.

Савин Р. Тестирование dot com или Пособие по жестокому обращению с багами в интернет-стартапах [Текст]/ Роман Савин – Москва: Издательские решения, 2017. – 312 с.

habr [Электронный ресурс]: коллективный блог с элементами новостного сайта, созданный для публикации статей, связанных с информационными технологиями / компания Тематические медиа, 2006 / режим доступа: https://habr.com/ru/

База знаний QALight [Электронный ресурс]: сайт, посвященный вопросам повышения качества программного обеспечения / компания «QALight / режим доступа: https://qalight.com.ua/baza-znaniy/

ПроТестинг [Электронный ресурс]: сайт, посвященный тестированию программного обеспечения / режим доступа: http://www.protesting.ru/

Software-Testing.RU [Электронный ресурс]: проект, посвященный вопросам тестирования и повышения качества программного обеспечения / режим доступа: https://software-testing.ru/

TMGURU [Электронный ресурс]: проект, посвященный вопросам планирования и управления тестированием / режим доступа: http://tmguru.ru/