Регрессионное и интеграционное тестирование

Цель регрессионного тестирования – удостовериться в том, что существующая функциональность не была затронута изменениями в коде.

Регрессионное тестирование простыми словами

Приведём пример. Представьте, что ваша команда разработала простое приложение-фонарик. В приложении есть всего две функции: включение и выключение фонарика. Вы проводите тестирование функциональности, чтобы убедиться в правильной работе этих функций и приложения в целом. Результат тестирования положительный – вам не о чем беспокоиться.

Спустя какое-то время вы решаете усовершенствовать приложение, добавив в него функцию включения от того, что пользователь встряхивает телефон. Новая функциональность добавлена, проводится очередной раунд тестирования. Однако на этот раз вам необходимо проверить работоспособность не только новой функциональности, но и включения/выключения. А вдруг изменения затронули и её каким-то образом?

Этот пример демонстрирует место регрессионного тестирования в процессе разработки ПО.

РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ И МЕТОДОЛОГИИ УПРАВЛЕНИЯ ПРОЕКТАМИ

Рассмотрим, как на проведение регрессионного тестирования влияет методология проекта. Возьмём для примера три самых популярных: гибкую, каскадную и гибридную.

Гибкая методология (Agile)

Разработка по Agile ведётся короткими итерациями (спринтами), продолжительностью 4–6 недель каждая. В конце каждой итерации заказчик получает готовый продукт, который может выполнять определённые функции. В идеале регрессионное тестирование проводится в конце каждого спринта, но на деле так происходит редко.

В чём трудность? Жёсткие дедлайны и задержки при разработке оставляют меньше времени на тестирование, чем требуется.

Решение. В конце каждого спринта проводится регрессионное тестирование с частичным покрытием, проверяются области, которые с большей долей вероятности могли быть затронуты разработкой. Перед крупными релизами проводится регрессионное тестирование всего продукта.

Каскадная методология (Waterfall)

Каскадная методология предполагает переход к следующему этапу только после полного завершения предыдущего. Таким образом, тестирование начинается после окончания разработки.

В чём трудность? Вносить изменения в готовый продукт – трудоёмкий и дорогостоящий процесс. Неудивительно, что каскадная модель применяется на небольших проектах с чёткими требованиями, которые не меняются в процессе разработки.
Решение. До того как начать процесс разработки, рекомендуется провести тестирование требований и убедиться, что они обладают всеми характеристиками качественных требований: единичностью, актуальностью, выполнимостью и т. д.
Рекомендуем доверять тестирование требований независимым специалистам, которые их объективно оценят и дадут рекомендации по оптимизации. Ведь устранение ошибок в требованиях – это исправление в тексте. Исправление ошибок в готовом продукте – это часы работы разработчиков и тестировщиков.

Гибридная методология

Все преимущества гибкой и каскадной методологий объединила в себе гибридная методология управления проектами. Этапы планирования и определения требований проходят согласно каскадной методологии, а этапы проектирования, разработки, внедрения и оценки – согласно гибкому подходу.

Соответственно, на разных стадиях проекта будет выполняться полное или частичное регрессионное тестирование.

7 СТАДИЙ РЕГРЕССИОННОГО ТЕСТИРОВАНИЯ

Независимо от того, какой методологии придерживаетесь вы, изменения ПО требуют выполнения регрессионного тестирования, состоящего из следующих стадий:

1. Анализ внесённых изменений, требований и поиск областей, которые могли быть затронуты.
2. Составление набора релевантных тест-кейсов для тестирования.
3. Проведение первого раунда регрессионного тестирования.
4. Составление отчета о дефектах.

Каждый дефект вносится в баг-трекинговую систему, описываются шаги для его воспроизведения. Если возможно, описание сопровождается видео, скриншотами.

5. Устранение дефектов.
6. Верификация дефектов.

На этой стадии QA-инженеры проверяют, действительно ли дефект исправлен. Если проблема остается, создаётся новый отчет. В некоторых случаях устранение дефекта подлежит обсуждению. Все критические и значительные дефекты должны быть устранены обязательно, а вот минорные, устранение которых требует больших затрат, могут быть оставлены. Особенно в тех случаях, если пользователю они не видны.

7. Проведение второго круга регрессионного тестирования.

Исходя из опыта команды a1qa, в среднем требуется не менее трёх раундов регрессионного тестирования для устранения всех дефектов и стабилизации приложения.

РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ: РУЧНОЕ ИЛИ АВТОМАТИЗИРОВАННОЕ?

На любом проекте регрессионные тесты – первые кандидаты на автоматизацию. Они запускаются регулярно, в большом количестве. Автоматизация позволяет не только сократить сроки тестирования, но и высвобождает ресурсы для более высокоуровневых задач, исследования приложения.

Однако несмотря на востребованность автоматизации, ручное регрессионное тестирование также имеет место быть.

Ручное тестирование

Внедрять автоматизацию на небольших проектах, которые длятся всего несколько месяцев, не всегда целесообразно: затраты могут не окупиться, а разработка автотестов лишь затянет сроки реализации проекта. В таком случае тестирование проводится вручную без каких-либо особых инструментов, за исключением баг-трекинговой системы, например, JIRA.

Однако если проект разрастается, функциональность ПО увеличивается с каждым последующим релизом, это влечёт за собой увеличение объемов тестирования. В таком случае стоит задуматься о привлечении специалистов по автоматизации.

Автоматизация тестирования

Выгоду автоматизации тестирования нельзя недооценивать:

- Улучшается качество продукта.
- Ускоряется выпуск ПО на рынок.
- Оптимизируется стоимость тестирования.

Кстати, есть и такие проекты, на которых разумно совместить ручные проверки с автоматизированными. Все зависит от специфики программного продукта.

ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ на РЕАЛЬНОМ ПРОЕКТЕ от компании A1QA простыми словами

ЧТО ЖЕ ЭТО ЗА ТЕСТИРОВАНИЕ И КАК ОНО ПРОВОДИТСЯ?

Первой на ум приходит интеграция продукта с платежными сервисами. Это, безусловно, важный аспект для проверки тестировщиками, но далеко не единственный.

Бизнес сегодня опирается на множество программных решений: вебсайт, системы ERP, CRM, CMS. От интеграции всех систем зависит качество обработки запросов пользователей, скорость предоставления услуг и успешность бизнеса в целом.

На примере реального проекта a1qa покажем, интеграция каких систем может тестироваться и что требуется от команды, чтобы получить релевантные результаты проверок.

ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ: ОБЗОР ПРОЕКТА

Заказчик

В a1qa обратился представитель популярного англоязычного журнала. Издание доступно как в печатном виде, так и онлайн. На тестировании онлайн-портала и сфокусировалась команда a1qa.

Задача проекта

Помимо функциональности портала, команда должна была проверить модуль подписки, который состоял из нескольких компонентов. Данный модуль представлял особую важность, поскольку именно он отвечал за монетизацию онлайн-версии журнала.

Для реализации функции подписки и ее управления использовались:

- CMS-решение, предоставляющее любые данные о подписках с применением различных фильтров: типа подписки, ее продолжительности и так далее.
- Вебсайт, через который пользователь взаимодействует с системой.
- CRM Salesforce. Функция – хранение данных о пользователях и приобретенных ими подписках. Дополнительная надстройка позволяет команде заказчика управлять приобретенными подписками, а также создавать новые и проверять старые подписки.
- SaaS-решение для выставления счетов и обработки платежей.
- Сервисная шина Mule ESB, с помощью которой осуществляется обмен данными между системами.
- База данных как инструмент Business Intelligence.
- Salesforce Marketing Cloud – инструмент рассылки корреспонденции и коммуникации с пользователями.
- Еще одна система, хранящая данные о зарегистрированных пользователях с инструментом для публикации статей, видео- и аудио-контента.

Процесс оформления подписки был построен следующим образом:

1. Подготовка набора данных, создание подписки.
2. Предоставление пользователю возможности приобретения подписки после внесения персональных и платежных данных.
3. Обработка заказа третьей стороной, предоставляющей свои услуги клиенту в данной сфере.

ЦЕЛЬ КЛИЕНТА

Клиент хотел освободить процесс от третьих сторон. Для этого требовалось убедиться, что разработанная система подписки может бесперебойно решать все задачи без участия третьих сторон.

ЗАДАЧА ТЕСТИРОВАНИЯ

Команда a1qa должна была подтвердить, что продукт способен выполнять возложенные функции. В ходе проекта некоторые компоненты разрабатывались с нуля, некоторые настраивались на базе готовых.

Важно было проверить, как они взаимодействуют между собой, и ответить на вопрос: способна ли вся система решать требуемые задачи?

СТРАТЕГИЯ ПРОВЕДЕНИЯ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ A1QA

1. Определены ключевые бизнес-процессы, которые должна выполнять система: создание, отмена, приостановка и возобновление подписки, изменение платежной информации для подписки и т.д.
2. Разработана тестовая документация с учетом всех возможных вариаций. Вариации – различные альтернативные выполнения операций (например, отмена подписки может произойти по желанию заказчика, а может быть произведена автоматически, если платежные данные были отклонены банком), а также различные параметры (например, тип продукта). В документации требовалось учесть проверку того, например, что создание подписки пройдет успешно для всех продуктов в рамках каждого бизнес-процесса.
3. Проведено тестирование, с помощью которого пошагово пройден каждый бизнес-процесс со стартового компонента (где он был инициирован) через все промежуточные и до финального (или финальных) с проверкой того, что все данные передаются правильно, а ожидаемые события на самом деле случаются.

Большинство процессов включало в себя передачу данных из одного модуля (чаще всего из Salesforce) во все остальные. Если начальной точкой был не SF, то информация из модуля поступала в MuleESB, а потом в SF, а оттуда во все остальные (опять же, через MuleESB).

На проведение тестирования интеграции было потрачено порядка 40% всех трудозатрат QA-команды.

Трудности

Большинство трудностей при проведении тестирования интеграции было вызвано недостаточной проработкой требований на начальном этапе проекта. Именно некачественные требования стали причиной множества дефектов и нестабильности системы в целом.

Поясним. Изначально требования выглядели как набор пользовательских историй (User Story) в JIRA и содержали только заголовки без какого-либо пояснения. Создавали их чаще всего разработчики.

Команда a1qa инициировала изменения в подходе написания требований: теперь для них обязательно добавляются описания и Acceptance Criteria, создаются промежуточные задачи с четким определением, кто и за что отвечает.

АВТОМАТИЗАЦИЯ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ

Автоматизация тестирования – непростой вопрос, который требует внимательного сопоставления всех «за» и «против».

Автоматизация интеграционного тестирования требует еще более внимательного подхода. С одной стороны, разработка автотестов сокращает время на выполнение тестов. С другой стороны, автотесты эффективны, когда работают с повторяющимися наборами данных или, как минимум, предсказуемыми.

А при оформлении подписки далеко не всегда так происходит: данные обновляются регулярно и хаотично. Поэтому тестирование проводилось преимущественно вручную.
Лишь на поздних стадиях проекта была внедрена автоматизация. Какие же тест-кейсы были автоматизированы? Были отобраны ключевые бизнес-процессы. Для каждого бизнес-процесса были прописаны вариации его прохождения. Автоматизированы были те тест-кейсы, которые покрывали регулярные и стабильные бизнес-процессы. Тем самым, автоматизация обеспечила максимальное покрытие при оптимальных затратах усилий.

РЕЗУЛЬТАТЫ

Работа над проектом продолжается, но уже можно сказать, что система функционирует надежно. Каждый компонент выполняет свою роль. А все вместе они помогают достичь поставленной цели – обеспечить бесперебойную работу важных для заказчика бизнес-процессов.

Подводя итоги

Тестирование интеграции обязательно, если на проекте задействованы процессы со сложной бизнес-логикой, которые задействуют различные системы.

Для проведения эффективного тестирования, обнаружения всех дефектов и недочетов команда по тестированию должна:

1. Понимать структуру продукта, знать, как между собой взаимодействуют все модули;
2. Владеть спецификой проекта. Это важно для написания хороших тест-кейсов, анализа результатов прогона тестов, выбора между ручным и автоматизированным тестированием.