АВТОМАТИЗАЦИЯ ТЕСТОВ КАК ЧАСТЬ AGILE-ПОДХОДА В ТЕСТИРОВАНИИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Нестерова Ольга Александровна

магистрант, Тольяттинского государственного университета, РФ, г. Тольятти

Аннотация. В ходе исследования была выявлена следующая **проблема**. Особенности Agile-подхода полностью соответствуют целям и задачам мобильного тестирования.

Но на данный момент описано мало практических путей внедрения гибкого тестирования программного обеспечения в целом и мобильных приложений в частности.

Ключевые слова: тестирование, quality assurance (QA), мобильные приложения, мобильное программное обеспечение (ПО), мобильные устройства, Agile-подход, гибкое тестирование, автоматизация, автотесты

Актуальность: в настоящее время концепция Agile набирает все большую популярность в сфере разработки программного обеспечения, в том числе – для мобильных устройств. Чтобы переход на гибкое тестирование принес пользу, нужно осознавать преимущества и недостатки этого подхода и постепенно вносить изменения в текущий процесс.

Идея: переход на Agile приносит наибольшую пользу на устоявшихся проектах, в рамках которых нужно регулярно и быстро проверять большой объем разработанного ранее функционала.

Пути решения проблемы: проанализировать особенности процесса тестирования «долгоиграющих» мобильных приложений, обосновать необходимость плавного перехода от традиционных методов разработки к гибким и пояснить, почему такой переход стоит начинать с внедрения простых автотестов.

Во вводной части приводится пример мобильного ПО, работа с которым подразумевает длительный цикл разработки и итеративное добавление нового функционала. Затем описываются нюансы тестирования такого приложения.

В основной части анализируются подводные камни гибкой методологии и описываются риски, возникающие при «переводе» разработки мобильного приложения на Agile.

После рассматривается, как принципы концепции Agile соотносятся со спецификой приложения с долгим циклом разработки.

Далее описывается, как применение простых автотестов позволяет совершить плавный переход к гибкому тестированию, приводятся аргументы в пользу такого решения.

В заключительной части перечисляются выводы, полученные в ходе исследования.

Вводная часть

Для примера рассмотрим VetVet – мобильное приложение для сотрудников ветеринарной клиники «Усы, лапы, хвост».

С его помощью пользователи могут создавать электронные медицинские карты животных.

Программа состоит из серверной части, iOS/ Android – совместимого мобильного интерфейса и мобильного API.

В первой версии функционал VetVet сводился к возможности создавать, редактировать и удалять учетные записи о четвероногих клиентах ветклиники.

Хотя такая реализация соответствовала пожеланиям заказчика, после релиза стабильной версии приложение не было переведено в стадию поддержки. Заказчик хорошо понимал, что растущая конкуренция в сфере ветуслуг требует постоянного обновления всех компонентов его бизнеса, в том числе – заказанного им приложения.

Поэтому VetVet стал «обрастать» новыми функциями. Так была разработана вторая часть приложения – для владельцев животного.

Теперь пользователи из этой группы могли вносить изменения в анкеты своих питомцев, а также просматривать записи, оставленные ветспециалистами.

Затем внедрили функцию «самозапись». У владельцев появилась возможность дистанционно записывать питомцев на прием, предварительно выбрав все необходимые параметры: вид услуги, дату и время, ветврача и т.д.

Одновременно с этим был реализован механизм отправки push-нотификаций.

Хозяева животных стали получать уведомления о предстоящем визите в ветклинику, переносе или отмене записи на прием.

В ходе дальнейшее эволюции в VetVet появился чат для обмена сообщениями между владельцами и ветеринарами, блок с отзывами о работе конкретных специалистов, раздел для получения индивидуальных рекомендаций по содержанию, питанию и дрессировке животного…

Таким образом, за 4 года изначальный функционал приложения вырос в несколько раз, а бизнес-логика была дополнена многими новыми деталями. Параллельно усложнялась и техническая часть приложения: в мобильное API были добавлены новые запросы и параметры, в базе данных появились новые сущности, были пересмотрены связи между ранее созданными сущностями и т.п.

Длительный цикл разработки определил специфику тестирования VetVet. При каждом внесении изменения QA-специалистам нужно подготавливать тестовую документацию для нового функционала и обновить список проверок для затронутой им существующей части ПО.

Кроме того, значительно вырос объем базовых проверок, проводимых в рамках санити-, смоук- и приемочного тестирования. В частности, выросло время на подготовку тестовых пререквизитов – создание наборов данных с разными комбинациями параметров, которые необходимы для прохождения более сложных тестовых сценариев.

Стоит добавить сюда, что все перечисленное по-прежнему надо было проверять на разных версиях iOS и Android, наиболее популярных моделях устройств, помнить о взаимном влиянии операционных систем, «железа», «оболочек» для Android и пр.

Время, выделенное на проведение тестирования, при этом не увеличилось. Ведь как мобильная разработка подразумевает частый выпуск новых версий и постоянное обновление с учетом новых идей у конкурентов и обратной связи от пользователей.

В подобной ситуации команда тестирования рискует упустить или слишком поздно

обнаружить крупные ошибки и, тем самым, многократно увеличить расходы на их исправление (см. Рисунок 1).

Чтобы не жертвовать объемом тестирования (и, как следствие, качеством продукта), было решено пересмотреть подход к работе над проектом. В ходе анализа выяснилось, что описанные выше особенности разработки полностью соотносятся с принципами методологии Agile.

Рисунок 1. Увеличение стоимости исправления ошибок в зависимости от времени их обнаружения

Основная часть

Несмотря на растущую популярность, Agile-методология не является универсальным решением для любых проектов. Рассмотрим, как отказ даже от одного из принципов гибкого подхода может стать препятствием для продуктивной работы над проектом.

Agile ставит во главу процесса разработки общение: между членами команды, между командой и проектным менеджером, между специалистами и заказчиком. Есть как минимум три популярных сценария, в которых этим моментом придется пренебречь:

1. Недавно запущенный проект, для работы над которым набрана команда из специалистов, которые ранее не работали вместе или имеют негативный опыт совместной работы.
2. Проект с ограниченным бюджетом, в котором не учтены затраты на все

коммуникационные активности. Т.е. заказчик оплачивает только анализ, разработку и тестирование будущего программного продукта.
3. Разросшийся проект, в котором участвует большое количество команд. Слаженность их работы во многом зависит от баланса коммуникационных и технических качеств у скрам-мастеров и менеджеров.

В первом случае налаживание общения потребует дополнительного времени и финансовых вложений. Нужно провести серию тимбилдингов для сплочения команды и, возможно, организовать тренинги по развитию коммуникативных навыков у будущих коллег.

Во втором примере оплачены лишь основные проектные работы, поэтому у команды просто нет возможности (и желания) тратить время на «разговоры».

А в третьем случае работа команд можно рассинхронизироваться, если процессами руководят люди, которые недооценивают роль коммуникаций и открыто пренебрегают ими в пользу «активной» деятельности.

Вернемся к рассмотренному выше приложению VetVet и обоснуем плюсы его перевода на Agile.

При внедрении гибкой методологии на устоявшемся проекте вероятность возникновения коммуникационных рисков невелика. Во-первых, ядро команды составляют люди, которые успели сработаться и наладить общение. Во-вторых, постоянный заказчик понимает пользу внутрипроектных коммуникаций и не стремится урезать бюджет разработки за их счет.

Более того, работа над мобильным ПО с длительным циклом разработки совместима и с другими базовыми принципами Agile:

1. Удовлетворение потребностей заказчика за счет регулярной поставки новых версий приложения. В случае с VetVet это оперативная реализация нового функционала по запросу заказчика.
2. Частое обновление программного продукта. Это требование автоматически выполняется как за счет появление у заказчика новых «хотелок», так и за счет необходимости поддерживать работоспособность приложения на новых моделях мобильных устройств.
3. Изменение требований допускается даже на поздних сроках разработки. Разработка новых возможностей часто требует доработки и усложнения логики уже существующего функционала.
4. Постоянное внимание к техническому совершенству – так же идеально согласуется с необходимостью не отставать от эволюции мобильных ОС и «железа».
5. Анализ способов улучшения эффективности – требование обусловлено необходимостью выполнять все больший объем работ в каждой следующей итерации.
6. Поддержка постоянно высокого темпа разработки – опять же, неизбежный момент при разработке мобильного ПО.

Как мы видим, переход к гибкой методологии – вполне органичное решение для команды, которая разрабатывает VetVet.

Но, чтобы получить все преимущества Agile, изменения нужно вводить постепенно.

Если говорить о процессе тестирования, первым шагом в сторону гибкой методологии может стать применение простых автоматизированных тестов.

В первую очередь рекомендуют автоматизировать [1]

- Тест-кейсы с высоким риском — критические для бизнеса тест-кейсы;
- Повторяющиеся тест-кейсы;
- Тест-кейсы, которые слишком сложно и неудобно выполнять вручную;
- Тест-кейсы, требующие много времени.

Первые версии VetVet тестировались только вручную. Исключение составляли юнит-тесты, подготовкой и прогоном которых занимались разработчики.

По мере наращивания функционала приложения были выделены кандидаты на автоматизацию:

1. Стабильный функционал, который сам по себе не подвергается кардинальным изменениям, но может «пострадать» после добавления новых элементов.
2. Вспомогательные кейсы, которые выполняются перед основными проверками и нужны для создания тестовых данных.
3. Кейсы, предназначенные для тестирования второстепенного или смежного функционала. Например, для обработки всевозможных «флагов» состояний объектов, автоматической генерации сущностей, связанных с создаваемыми объектами, отправки различных уведомлений о том, что был создан/ изменен тот или иной объект.

Для запуска автотестов было решено использовать программу Postman. Выбор был сделан с учетом структуры VetVet. Действия конечного пользователя, независимо от роли, легко разбиваются на последовательности запросов. Каждый из них использует один или несколько параметров, полученных в ответе на ранее выполненный запрос/ запросы (см. Рисунок 2).

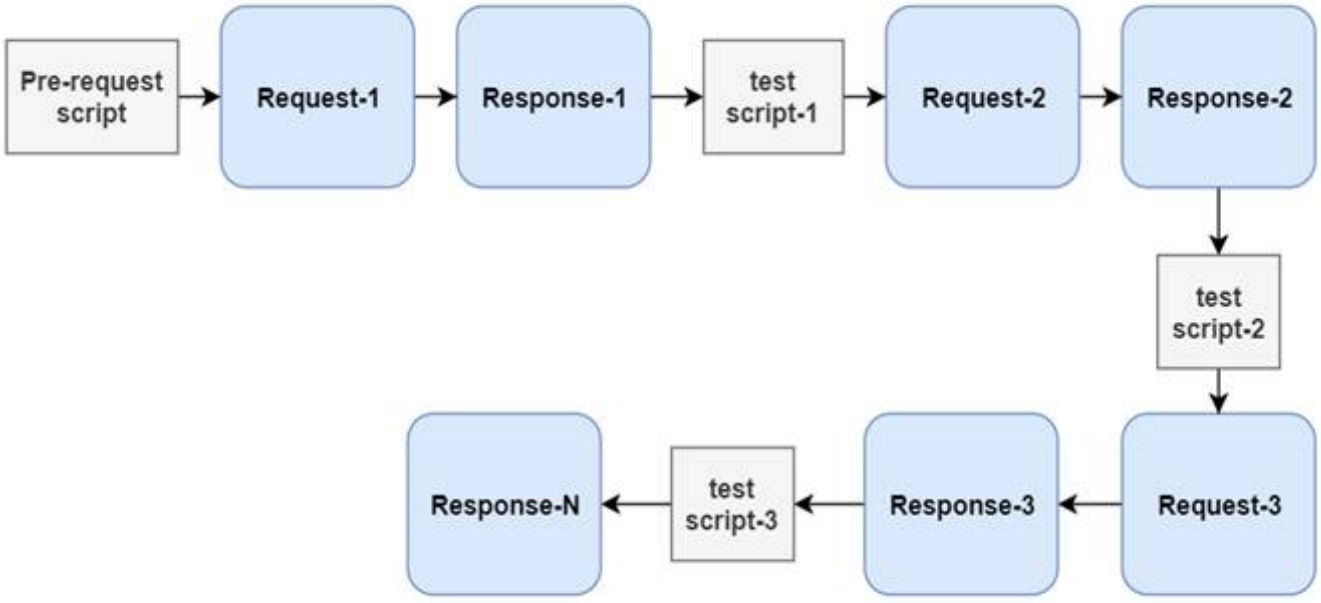

Рисунок 2. Схема отправки связанных запросов

Всего в автотесты для Postman было переведено около 25 % кейсов для функциональных проверок. Благодаря это время проведения регрессионного тестирования уменьшилось на 40 %. За счет этого длительность итерации сократилась до одного месяца – процессы выдачи приблизились к коротким циклам Agile.

Заключительная часть

На начальном этапе исследования была выдвинута идея о том, что переход на Agile приносит наибольшую пользу на устоявшихся проектах.

При каждом внесении изменения сильно возрастает объем базовых проверок, проводимых в рамках санити-, смоук- и приемочного тестирования.

В частности, увеличивается время на подготовку тестовых пререквизитов – создание наборов с разными комбинациями параметров, которые необходимы для прохождения более сложных тестовых сценариев.

При этом не растет время, выделенное на проведение тестирования.

В подобной ситуации команда тестирования рискует упустить крупные ошибки и, тем самым, многократно увеличить расходы на их исправление.

При внедрении гибкой методологии нужно убедится, что в проектной команде налажена коммуникация, а заказчик готов оплачивать время, которое уходит на обсуждение разных аспектов разработки приложения.

В ходе знакомства со специализированной литературой и работы в качестве QA-специалиста были выявлены особенности тестирования мобильных приложений с долгим циклом разработки, благодаря которым их можно легко перевести на гибкую методологию:

1. Регулярная и оперативная реализация нового функционала по запросу заказчика;
2. Частое обновление программного продукта за счет появление у заказчика новых «хотелок» и необходимости поддерживать работоспособность приложения на новых моделях мобильных устройств;
3. Разработка новых возможностей часто требует доработки и усложнения логики уже существующего функционала;
4. Необходимость не отставать от эволюции мобильных ОС и «железа» мотивирует команду стремится к техническому совершенству;
5. Увеличение объема работ в каждой следующей итерации порождает потребность в анализе способов улучшения эффективности;
6. Поддержка постоянно высокого темпа разработки, обусловленная спецификой мобильной разработки.

Первым шагом в сторону гибкой методологии может стать применение простых автоматизированных тестов, которые выполняются с помощью Postman. Начинать автоматизацию стоит с кейсов, которые

1. проверяют стабильный функционал, который не изменяется напрямую, но может быть затронут при добавлении новых элементов;
2. нужны для создания тестовых наборов данных.
3. предназначены для тестирования второстепенного или смежного функционала.

Использование автотестов для проведения функциональных проверок существенно ускоряет регрессионное тестирование. Это позволяет сократить длительность итерации до одного месяца и приблизить процессы выдачи к коротким циклам Agile.

Список литературы:

1. Автоматизация тестирования и Agile [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/otus/blog/351104/

2. Бизнес-аналитики в Agile – зачем, почему, как [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/dataart/blog/291448/

3. Криспин, Лайза Гибкое тестирование: Пер. с англ./ Лайза Криспин, Джанет Грегори. – М.: ООО «И.Д. Вильямс», 2010. – 464 с.

4. Куликов, С.С. Тестирование программного обеспечения / С. С. Соколов. – Минск: Четыре

четверти, 2015. – 294 с.

5. Agile Alliance (русскоязычная версия) [Электронный ресурс] – Режим доступа: https://www.agilealliance.org/

6. Agile Manifesto (русскоязычная версия) [Электронный ресурс] – Режим доступа: http://agilemanifesto.org/iso/ru/principles.html