AGILE-ПОДХОД В ТЕСТИРОВАНИИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ СМАРТ-УСТРОЙСТВ

Нестерова Ольга Александровна

магистрант Тольяттинского государственного университета, РФ, г. Тольятти

Аннотация. В ходе исследования была выявлена следующая **проблема**: мобильные приложения разрабатываются в сжатые сроки и нуждаются в регулярном выпуске новых версий, поэтому для создания и тестирования таких программ необходим подход, который позволит быстро создать приложение, соответствующее требованиям заказчика и ожиданиям конечных пользователей.

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

Актуальность: в настоящее время мобильная разработка является одним из ключевых направлений деятельности большинства ИТ-компаний. Качество мобильных приложений во многом зависит оттого, насколько методология разработки соответствует технической и экономической специфике продукты.

Идея: Agile-подход полностью соответствует целям и задачам тестирования мобильных приложений, которые предназначены для работы со смарт-устройствами.

Пути решения проблемы: расписать применение Agile-подхода на всех стадиях разработки мобильного приложения, выделить ключевые преимущества Agile по сравнению с традиционной методологией разработки ПО (Waterfall).

Во вводной части дается определение Agile и описываются особенности жизненного цикла мобильного ПО. Далее приводятся аргументы в пользу Agile-подхода в работе QA-команды, которая тестирует мобильные программы для управления смарт-гаджетами.

В основной части подробно разбираются перечисленные ранее преимущества Agile-подхода с опорой на источники по теме. В заключительной части описываются выводы, сделанные в ходе исследования.

Вводная часть

Гибкая методология разработки или Agile – это серия подходов к созданию программного обеспечения. Ключевые особенности методологии описаны в Agile Manifesto – программном документе сообщества «Agile Alliance» [5], разработанном в феврале 2001 года.

В основе Agile лежат 12 принципов:

1. Наивысшим приоритетом является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.
2. Изменение требований приветствуется, даже на поздних стадиях разработки.

3. Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
4. На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
5. Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
6. Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.
7. Работающий продукт — основной показатель прогресса.
8. Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно.
9. Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
10. Простота — искусство минимизации лишней работы — крайне необходима.
11. Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
12. Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы [6].

Agile-подход к тестированию подразумевает следующие изменения в работе QA-команды [1]:

- Тестирование перестает быть фазой в создании ПО и активно применяется на всех стадиях жизненного цикла продукта;
- Объем тестовой документации сокращается до минимума: на смену подробным тест-кейсам приходят более высокоуровневые и универсальные тест-планы и чек-листы;
- На всех стадиях разработки поддерживается обратная связь между специалистами по тестированию и остальными членами команды (разработчики, бизнес-аналитики, дизайнеры, проджект-менеджер);
- Быстрая отдача от тестирования [3, 44] – найденные баги подлежат оперативному исправлению, что позволяет поддерживать «чистоту кода» и избежать накопления legacy-кода;
- Тестирование – неотъемлемая часть критерия готовности: степень готовности ПО определяется с учетом количества, приоритета и серьезности обнаруженных проблем.

Разница между традиционной и гибкой методологией разработки ПО проиллюстрирована на Рисунке 1 [3, 43].

Рисунок 1. Разница между традиционной и гибкой методологией разработки ПО

Преимущества гибкой разработки – максимальное вовлечение заказчика, тщательная работа с требованиями, тесная интеграция тестирования и разработки, уменьшение объема документации [3, 29].

Перечисленные принципы и изменения хорошо сочетаются со спецификой мобильных приложений:

- короткий цикл разработки;
- необходимость часто выпускать новые версии продукта;
- необходимость оперативно вносить в ПО изменения, обусловленные эволюцией мобильных операционных систем и выпуском новых моделей мобильных устройств.

Основная часть

Agile – больше, чем просто набор методик. Это целая философия, которая сводится не столько к использованию определенных инструментов, сколько к формированию особого мировоззрения.

Лайза Криспин, автор книги «Гибкое тестирование», дает такое определение «гибкого тестировщика»: «это профессиональный тестировщик, который принимает изменения, способен к продуктивному сотрудничеству как с техническими специалистами, так и специалистами в области бизнеса, и понимает концепцию применения тестов для документирования требований и управления разработкой» [3, 48].

Рассмотрим, в чем проявляется Agile-подход к тестированию на разных стадиях создания мобильного приложения для смарт-гаджетов.

Проектирование. Задачи QA-специалиста на этапе проектирования – тесное общение с бизнес-аналитиками, изучение подготовленной ими проектной документации и написание тестовых сценариев.

В первом случае нужно убедиться, что проектная документация служит следующим целям:

- дает наглядное представление о программном продукте (кто и как его может использовать, как будут реализованы основные функции);
- способствует цельному видению ПО – не содержит двусмысленных формулировок, содержимое документов не противоречит друг другу;
- служит «мостиком» между заказчиком и членами проектной команды – использованные термины и иллюстрационные материалы понятны и заказчику, и исполнителям.

При проектировании приложений для смарт-гаджетов необходимо продумывать три отдельные схемы разворачивания процессов (см. Рисунок 2):

- взаимодействие программы с веб-сервером – обновление версии ПО, хранение и/ или обработка пользовательских данных;
- взаимодействие ПО с прошивкой смарт-девайса – получение информации о состоянии устройства, реализация целевой функция приложения путем выполнения команд;
- аналитическая работа приложения – сбор и обработка результатов выполнение команд (составление статистики, предоставление информации в удобной для пользователя форме – в виде графиков, таблиц и пр.).

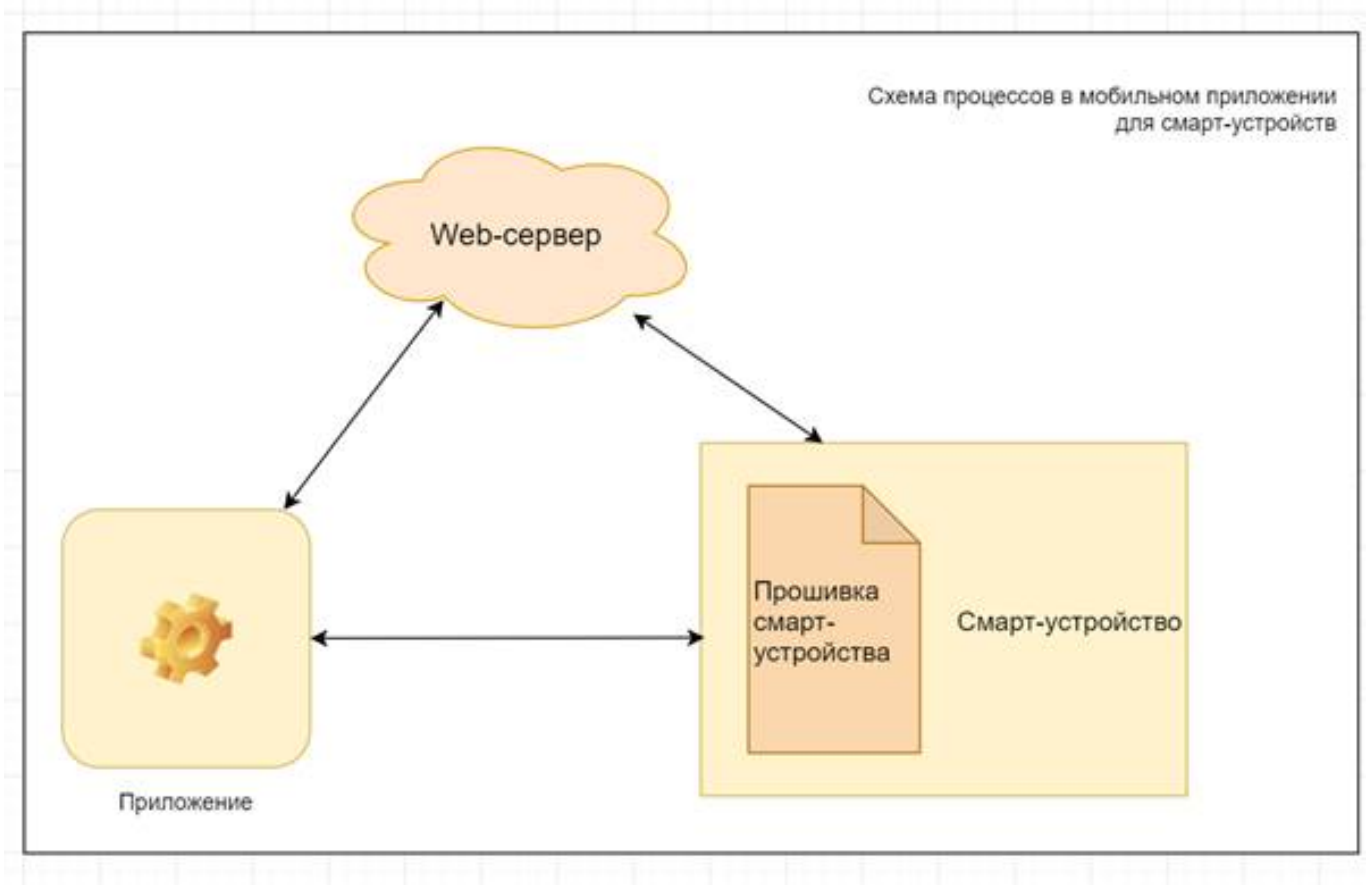

Рисунок 2. Схема процессов мобильного приложения для смарт-устройств

В требованиях должны быть описаны основные аспекты разработки и функционирования ПО – «ядро», которое в дальнейшем не будет меняться кардинальным образом. Сюда можно отнести

- Основные функции приложения (запросы для «общения» с сервером, команды для работы с целевым гаджетом, методы для отправки и получения этих команд).
- Логика описанных выше процессов (последовательность обмена запросами в треугольнике «сервер – приложение – прошивка смарт-гаджета»).
- Список экранов приложения. Обычно их количество соответствует числу основных разделов и условно делится на несколько групп – запуск и авторизация, аккаунт (личная информация пользователя), настройки (конфигурирование самого приложения), управление (работа с гаджетом), платный контент (опционально).
- Логика обработки разных типов ошибок – модальные окна с сообщениями о сетевых проблемах, прерывании отправленных на устройство команд или отрицательном исходе их выполнения.
- Определение пользовательских ролей, основные пользовательские сценарии – конечный пользователь, администратор, пользователь-«посредник» (обеспечивает доступ конечных пользователей к основному функционалу, не обладает правами для администрирования приложения, но для работы с приложением должен овладеть специализированными навыками);
- Стиль оформления программы (цветовая гамма, тип шрифтов, перечень широко используемых компонентов – таблицы, кнопки, чекбоксы, поп-апы и пр.).

Для деталей, которые часто меняются, будет достаточно более общего описания:

- текстовки (сообщения об ошибках, предупреждения, автоматически рассылаемые письма);
- способы взаимодействия с элементами (жесты, доступные для конкретной модели

мобильного устройства и мобильной ОС).

Включение схем и диаграмм делает спецификации более наглядными и простыми для изучения. Например, use-case диаграмма позволяет создать образ конечного пользователя и определить задачи, которые он сможет решить с помощью мобильного приложения (см. пример use-case диаграммы на Рисунке 3).

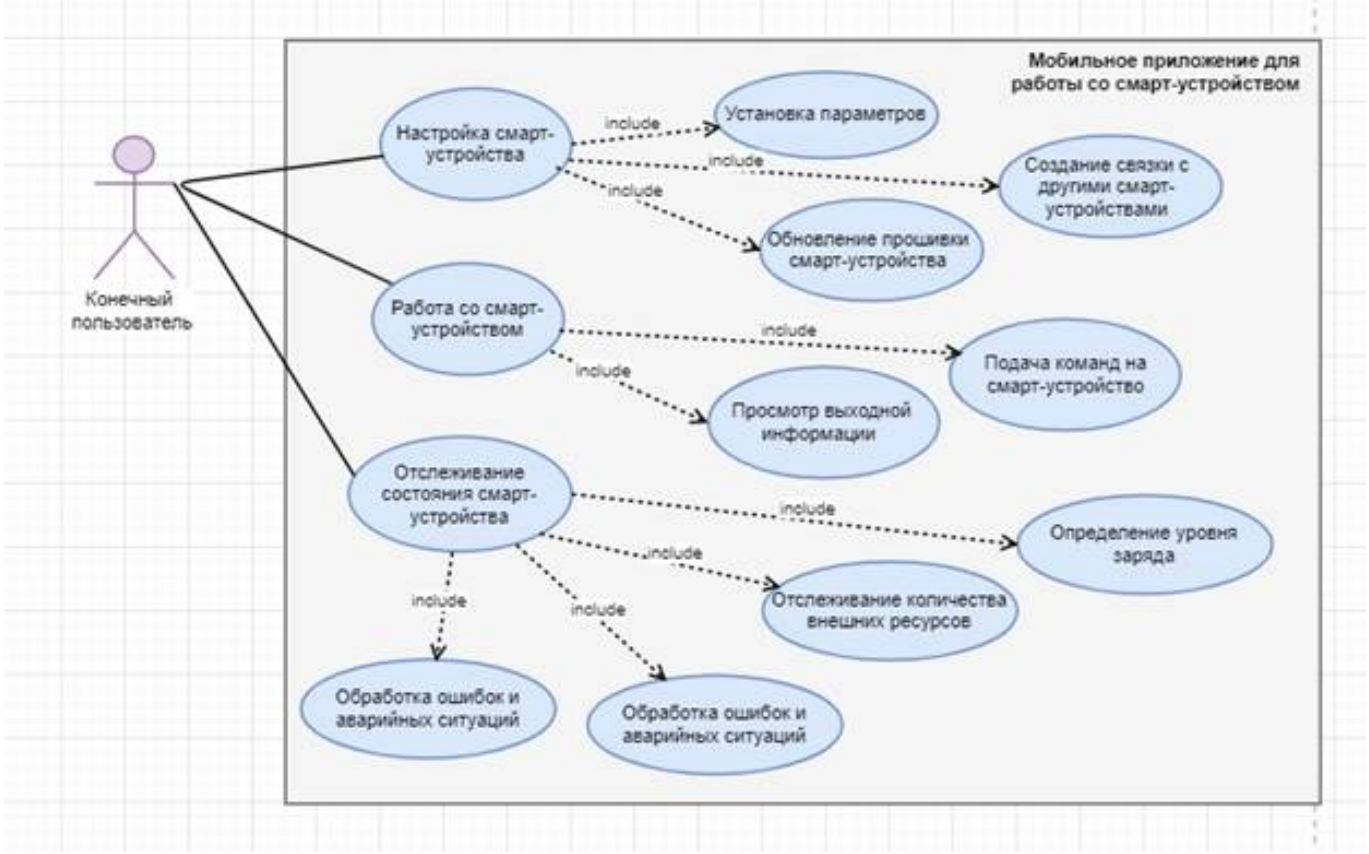

Рисунок 3. Пример Use-case диаграммы для мобильного приложения, предназначенного для работы со смарт-гаджетами

Вторым шагом будет создание тестовых сценариев. В документе этого типа описывается бизнес-идея приложения, поэтому он не зависит от деталей реализации и может быть подготовлен до написания кода [3, 44].

При подготовке сценариев следует отталкиваться от типа и условий использования «умного» устройства, для которого создается приложение. Если предназначение гаджета подразумевает его использование в спокойной домашней обстановке, можно добавить более длительные цепочки переходов от экрана к экрану, внести в UI большее количество функциональных элементов.

Если подразумевается использование устройства исключительно (или по большей части) вне дома, программа для него должна обладать противоположными чертами:

- выполнение основных действий за минимальное количество шагов;
- интерфейс с меньшим количеством деталей;
- отображение результатов работы устройства в формате, удобном для быстро просмотра.

При таком подходе к проектированию можно в краткие сроки подготовить документацию, которая создает наглядное представление о будущем продукте. Отказ от всесторонней

детализации позволяет быстро вносить изменения в соответствии с пожеланиями заказчика, требованиями рынка и т.д.

Разработка. На этой стадии QA-специалисты активно взаимодействуют с разработчиками. При Agile-подходе программный продукт создается по «кусочкам» – инкрементам. Тестировщики проверяют каждый кусочек сразу после сборки кода, только после этого разработчики переключаются на работу над следующим инкрементом [3, 43].

QA-специалисты занимаются ручным тестированием, которое направлено на поиск существенных (блокирующих, критических и важных) ошибок. Одновременно (силами квалифицированных членов команды тестирования или с помощью разработчиков) создаются автоматизированные функциональные тесты.

Чаще всего в первую очередь автоматизируются положительные тесты, направленные на проверку приложения при выполнении корректных действий. Затем добавляются проверки отрицательных сценариев и граничных значений.

Необходимо рассмотреть аспекты, которые представляют для заказчика наибольшую ценность, а также учесть контекст, в котором пребывает проект [3, 51]. Особое внимание нужно уделить следующим моментам:

- Логичность цепочек взаимодействия в описанном ранее треугольнике «сервер – приложение – прошивка устройства».
- Уровень безопасности приложения. Информация о пользователя должна быть защищена от веб-угроз, возникающих при обращении ПО к сетевым ресурсам, а также от прямых атак злоумышленников на целевое смарт-устройство и смартфон/ планшет с установленным приложением. В первую очередь нужно проверить надежность шифрования передаваемых данных, убедиться, что приложение запрашивает пароль или другой идентификатор для доступа к конфиденциальной информации, а также завершает сессию по тайм-ауту или при наступлении определенного события.
- Поддерживаемость. Следует отдельно проверить совместимость программы с популярными моделями смартфонов/ планшетов, затем совместимость последних с прошивкой «умного» устройства.
- Производительность смарт-гаджета при работе с несколькими экземплярами приложения и/ или «производственные мощности» самой программы при параллельной работе с разными смарт-устройствами.

Для удобства документирования и оперативной обработки найденных ошибок стоит использовать багтрекинговую систему. Необходимо, чтобы ею пользовались все участники процесса разработки.

В таком случае каждый член команды может в любой момент просмотреть историю приложения и узнать, какие вопросы были подняты в течение конкретной итерации, какие проблемы были найдены, как, кем и когда они были устранены.

Завершение итерации. На этой стадии происходит ввод перовой или очередной версии приложения в эксплуатацию и поддержка его стабильной работы. На этой стадии деятельность QA-специалиста сосредоточена на воспроизведении и анализе проблем, которые найдены заказчиком или конечными пользователями.

В мобильной разработке важна своевременная обработка не только критических проблем.

Причиной негативных отзывов пользователей (и, как следствие, коммерческого провала приложения) нередко служат нефункциональные баги – неудобный размер кнопок, раздражающие цвета, запутанная логика переходов между экранами и т.п.

Не меньший вред репутации мобильных программ наносят «мелкие» ошибки, которые стабильно воспроизводятся в нескольких версиях продукта. В эту же категорию стоит включить моменты, которые расходятся с общепринятой практикой.

Будет не лишним

- знакомиться с приложениями-аналогами от других производителей;
- следить за актуальными тенденциями в сфере программного и UI дизайна, открытиями в области кибернетики и психологии.

Выход новых версий приложения и прошивки смарт-устройства не должен нарушать совместимость этого устройства с клиентской и серверной частями программы.

Для этого необходимо обеспечить

- согласованность существующих и новых методов отправки команд;
- грамотное обновление запросов, которыми обмениваются мобильный API и сервер;
- сохранность пользовательских данных при апгрейде приложения, переходе на новую систему управления базами данных.

Заключительная часть

В начале данного исследования была выдвинута идея о преимуществе Agile-подхода в тестировании мобильных приложений, которые предназначены для работы со «умными» гаджетами.

В ходе знакомства со специализированной литературой, а также работы в качестве QA-специалиста был выявлен ряд плюсов гибкого подхода, которые наблюдаются на всех этапах разработки приложения.

Тестирование на стадии проектирования сводится к оценке качества спецификаций, в которых описаны техническая сторона и бизнес-логика приложения. В случае с ПО для смарт-устройств ядром спецификаций становится иллюстрация процессов взаимодействия приложения с сервером и прошивкой целевого гаджета

Большое количество диаграмм и рисунков помогает сократить объем документации, а отказ от исчерпывающего описания каждого аспекта позволяет быстро вносить изменения без ущерба для цельности общей идеи программного продукта.

Главный плюс гибкого тестирования на стадии разработки – возможность быстро выпускать новые версии программы. За счет этого программа постоянно эволюционирует и не теряет актуальности при выпуске новых моделей смартфонов/ планшетов, мобильных ОС, а также свежих версий прошивки целевого гаджета.

Agile-подход на этапе завершения итерации позволяет поддерживать обратную связь с заказчиком и конечными пользователями, своевременно устранять ошибки и обеспечивать согласованность работы прошивки смарт-устройства, клиентской и серверной частей приложения.

Таким образом, применение Agile-методологии к тестированию ПО для «умных» устройств ускоряет его разработку и одновременно повышает качество его работы. Это становится возможным благодаря тому, что гибкое тестирование не является ограниченной во времени фазой в разработке приложения, а непрерывно осуществляется в течение всего процесса.

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», 2010ю – 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