Литература

1. Bezdek J. C., Pattern Recognition with Fuzzy Objective Function Algoritms [Текст] / J. C. Bezdek// Plenum Press, – 1981.

2. Осовский, С. Нейронные сети для обработки информации [Текст] / С.Осовский – М.: Финансы и статистика, 2002 – 344 с.

А.В. Лагарникова, И.А. Сюсин

КОНЦЕПЦИЯ КОНСТРУКТОРА ПОСТРОЕНИЯ АВТОМАТИЗИРОВАННЫХ ТЕСТОВ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА УРОВНЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

(Самарский университет)

Тестирование приложений – это важный этап в процессе разработки программных систем. На сегодняшний день существует большое количество методов тестирования, каждый из которых имеет ряд достоинств и недостатков, а также зависит от конкретных целей и не всегда может быть применим. Существуют два основных вида тестирования: ручное и автоматизированное. Как правило, ручное тестирование используется для проверки функциональности приложения с точки зрения пользователя, а автоматизированное - для проверки внутренней функциональности приложения [1, 2].

Условно тестирование мобильных приложений можно разбить на 3 уровня [3]:

1) Unit Tests Layer – уровень автоматизированных модульных тестов, написанных разработчиками.

2) Functional Tests Layer (Non-UI) – уровень тестирования бизнес-логики приложения с выполнением функциональных тестов.

3) GUI Tests Layer – уровень тестирования пользовательского интерфейса приложения с возможностью тестирования функционального уровня, выполняя операции, использующие бизнес-логику приложения.

В данной статье будет рассмотрен уровень GUI Tests Layer, а также возможность применения методов его автоматизации.

Автоматизированное тестирование не может в полной мере покрыть требования к пользовательскому интерфейсу [4]. Данная проблема является наиболее актуальной для тестирования мобильных приложений. В автоматизированном тестировании нет возможности исследовательского тестирования, которая есть в ручном тестировании. Целью исследовательского ручного тестирования является проверка сложных возможностей приложения, а именно поведение приложения при мультитач – взаимодействиях (тактильные особенности приложения), при всплывающих уведомлениях (баннеры, action-меню, системные уведомления), а также при открытии другого приложения поверх тестируемого, поведение приложения при заблокированном – разблокированном экране [5, 6]. Важно, что используются не заранее составленные тест-кейсы, как

в автоматизированном тестировании, а сценарии, сформировавшиеся в процессе исследовательского ручного тестирования. Ручное тестирование мобильного приложения позволяет выявить общие коэффициенты удобства использования, такие как согласованность, точность контента, внешний вид и функционирование при разных разрешениях или аспектах вращения устройства. Проблема ручного тестирования заключается в том, что тестировщик должен выполнять один и тот же тестовый сценарий многократно в процессе разработки приложения. При этом часто возникают ситуации, когда разработчик внес правки и некоторый функционал, ранее работающий корректно, перестал отрабатывать, либо разработчик внес правки, но сценарий так и не стал отрабатываться корректно. Автоматизация тестовых сценариев ручного тестирования позволит увеличить скорость разработки приложения, путем уведомления разработчика о возникших неполадках [7].

Для решения задачи автоматизации тестовых сценариев предлагается разработать конструктор построения автотестов мобильных приложений – программный инструмент, позволяющий считывать последовательности действий ручного тестировщика (в том числе и сложные аспекты приложения) при помощи логов (списков событий) приложения, записи этой последовательности в файл, который будет являться в дальнейшем тестовым сценарием для выполнения тестировщиком или разработчиком. С одной стороны, это позволит инженеру по тестированию облегчить процесс выполнения тестового сценария, так как ему достаточно будет только запустить систему, которая сама выполнит всю последовательность действий. В случае успешного прохождения система сообщит об этом разработчику. Если же в результате выполнения теста возникла ошибка, то система должна автоматически уведомить разработчика об возникшей ситуации. С другой стороны, это позволит программисту после внесения правок автоматически проверять корректность работы текущей версии приложения без участия тестировщика.

Задача разработки конструктора заключается в снижении трудоемкости написания тестов уровня GUI Tests Layer за счет автоматизации рутинных операций. Для этого необходимо внедрить автотесты, созданные с его помощью, в слой CI (Continuous Integration), который выполняет всю работу по сборке, развертыванию и тестированию приложения в тестовую среду проекта. В этом случае у каждого участника проекта (в основном у тестировщика и разработчика) будет возможность автоматически выполнить тестовый сценарий, ранее проделанный вручную тестировщиком, чтобы убедиться в том, что приложение работает корректно. Также данные автотесты будут полезны в дальнейшем при приемочном тестировании сборки, а также для составления отчета о выполненных работах тестировщика.

Концепция конструктора построения автоматизированных тестов для мобильных приложений на уровне пользовательского интерфейса позволит усовершенствовать процесс разработки мобильного приложения путем уменьшения трудозатрат ручных инженеров-тестировщиков, а внедрение конструктора

позволит увеличить скорость разработки приложения и ведения отчетности о выявленных проблемах.

Литература

1. Anthony I. Waaserman. Software engineering issues for mobile application development [TEXT] / Anthony I. Waaserman// FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research/.New York, NY, USA: ACM, 2010, pp. 397–400 (Дата обращения: 18.03.2018)

2. Ravi ram Chandra Nimbalkar. Mobile Application Testing and challenges it [TEXT] / Ravi ram Chandra Nimbalkar //International Journal of science and research, June 2015 (Дата обращения: 18.03.2018)

3. X. Wei, L. Gomez, I. Neamtiu, and F. Michalis. Malicious android applications in the enterprise: What do they do and how do we fix it [TEXT] / X. Wei, L. Gomez, I. Neamtiu, and F. Michalis // ICDE Workshop on Secure Data Management on Smartphones and Mobiles - SDMSM 2012 (Дата обращения: 18.03.2018)

4. Mobile Test Automation Tools [электронный ресурс]. – Режим доступа: URL: https://www.testingexcellence.com/open-source-mobile-test-automation-tools/ (Дата обращения: 18.03.2018)

5. Mobile App Testing [электронный ресурс]. – Режим доступа: URL: https://www.infoq.com/articles/mobile-app-testing-the-secret-to-the-perfect-app (Дата обращения: 11.03.2018)

6. The Testing Pyramid for Mobile [электронный ресурс]. – Режим доступа: URL: https://saucelabs.com/resources/articles/whats-special-about-mobile-testing (Дата обращения: 11.03.2018)

7. Problems with Functional Test Automation [электронный ресурс]. – Режим доступа: URL: https://www.utest.com/articles/5-problems-with-functional-test-automation-today (Дата обращения: 18.03.2018)

Д.В. Лещева[1], В.А. Семенова[2], С.В. Смирнов[2]

ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ПРОГРАММНОЙ ЛАБОРАТОРИИ ДЛЯ ОНТОЛОГИЧЕСКОГО АНАЛИЗА ДАННЫХ

([1]Поволжский университет телекоммуникаций и информатики)
([2]Институт проблем управления сложными системами РАН)

Онтологический анализ данных (ОАД) представляет собой методический комплекс для выявления понятийной структуры исследуемой предметной области (ПрО) на основе эмпирических данных, представленных в виде соответствия «объекты-свойства» [1].

Для автоматизации ОАД в Институте проблем управления сложными системами РАН создается программная лаборатория OntoWorker (OW) [2] на широко распространенной платформе Excel, достоинство которой состоит не только в широчайших вычислительных возможностях, но и в уникальном качестве