УДК 004.89

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ANDROID ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Черевко Н.А., Белов Ю.С.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», филиал, Калуга, e-mail: nikikwiki@mail.ru

Мобильные приложения ежедневно используются более чем половиной населения мира для выполнения самых разных задач. По мере все более широкого использования этих приложений возникает потребность в эффективных методах их тестирования. Традиционно компании использовали методы автоматического тестирования. Однако в последнее время все большее число организаций используют автоматизированные инструменты для тестирования. Многие фреймворки позволяют автоматизировать процесс тестирования приложений, однако существующие фреймворки в основном полагаются на разработчика приложений для предоставления тестовых сценариев для каждого разработанного приложения, что предотвращает повторное использование этих тестов для аналогичных приложений. В этой статье представляется новый подход к автоматизации тестирования приложений Android, используя методы машинного обучения и повторное использование популярных тестовых сценариев. Он основан на предположении, что различные компоненты в Android приложениях имеют схожую структуру интерфейса. Чтобы использовать это сходство, используются методы машинного обучения, которые классифицируют каждый компонент как один из семи типов заранее заготовленных компонентов, определённых в данной работе. Демонстрируются потенциальные преимущества нового подхода в эмпирическом исследовании, где показывается, что разработанный инструмент тестирования, основанный на предложенном подходе, превосходит стандартные методы в реальных условиях.

Ключевые слова: тестирование Android-приложений, автоматизация мобильного тестирования, классификация компонентов

AUTOMATION OF ANDROID APPLICATIONS TESTING USING MACHINE LEARNING ACTIVITIES CLASSIFICATION

Cherevko N.A., Belov Yu.S.

Bauman Moscow State Technical University, branch, Kaluga, e-mail: nikikwiki@mail.ru

Mobile applications are being used every day by more than half of the world's population to perform a great variety of tasks. With the increasingly widespread usage of these applications, the need arises for efficient techniques to test them. Traditionally, companies used manual testing methods for testing their applications. However, recently, a growing number of companies and organizations use automated tools for their testings. Many frameworks allow automating the process of application testing, however existing frameworks mainly rely on the application developer for providing testing scripts for each developed application, thus preventing reuse of these tests for similar applications. The problem with this approach is that these tests are hand-coded for specific applications and specific scenarios, and each new application requires spending many resources to reuse these tests. In this paper, we present a novel approach for the automation of testing Android applications by leveraging machine learning techniques and reusing popular test scenarios. We discuss and demonstrate the potential benefits of our approach in an empirical study where we show that our developed testing tool, based on the proposed approach, outperforms standard methods in realistic settings.

Keywords: Android application testing, mobile testing automation, activities classification

Мобильные устройства становятся ключевым компонентом в нашей жизни, и в настоящее время ими владеет более половины населения мира. На сегодня разработано более пяти миллионов приложений. По мере того, как мобильные устройства становятся все более популярными, возникает потребность в эффективных методах тестирования такого программного обеспечения. Исследование [1] обнаружило, что мобильные приложения склонны к значительному количеству дефектов и ошибок. Традиционно компании использовали методы автоматического тестирования. Однако в последнее время все большее число организаций используют автоматизированные инструменты для тестирования [2]. Автоматизация тестирования стала стандартом с множеством решений и фреймворков, позволяющих автоматизировать процесс тестирования приложений. Проблема данного подхода заключается в том, что тесты написаны вручную, для конкретных приложений и сценариев, и каждое новое приложение потребует затрат большого количества ресурсов для повторного использования этих тестов. Кроме того, каждое изменение в приложении приводит к доработке тестовых сценариев. В данной статье представлен новый подход к автоматическому тестированию Android-приложений, который позволяет найти наибольшее число функциональных ошибок. Он основан на предположении, что различные компоненты в Android приложени-

ях имеют схожую структуру интерфейса. Чтобы использовать это сходство, используются методы машинного обучения, которые классифицируют каждый компонент как один из семи типов заранее заготовленных компонентов, определённых в данной работе. Затем для каждого компонента запускаются специализированные тесты пользовательского интерфейса, которые были созданы с учетом структуры определённого типа компонента.

Цель исследования – изучить способы автоматизации тестирования программного обеспечения с помощью методов машинного обучения.

Материалы и методы исследования

Автоматическое тестирование. Создание инструмента, который позволит автоматически тестировать произвольное мобильное приложение, является чрезвычайно сложной задачей [3]. Многие из ошибок в мобильных приложениях отличаются от ошибок, представленных в традиционных настольных приложениях, в основном из-за неотъемлемого различия в архитектуре и методологиях разработки. Таким образом, привычные подходы к тестированию программного обеспечения не могут быть естественным образом перенесены на мобильные приложения. Чтобы уменьшить потребность в написании избыточных сценариев тестирования для мобильных приложений, которые имеют много общих характеристик, большое количество исследовательских усилий в последнее время было сосредоточено на разработке методов и алгоритмов автоматизированного тестирования, позволяющих автоматически тестировать приложения. Тестирование приложения Android с использованием методов автоматического тестирования обычно выполняется путем запуска назначенного приложения с генерацией событий пользовательского интерфейса, которые имитируют поведение пользователя, включая такие действия, как щелчки, прокрутка и пролистывание [4]. Это нужно для генерации как можно большего количества релевантных входных данных, чтобы изучить приложение с максимальным охватом.

На данный момент существующие методы тестирования Android-приложений можно разделить на три категории:

– Метод случайного тестирования: программы, использующие этот подход, генерируют события пользовательского интерфейса случайным образом, выполняя их одно за другим. В основном этот подход используется для проверки надежности приложения, так как большинство этих событий маловероятны для обычного пользователя.

– Метод исследований на основе модели: программы, использующие данный подход, создают модель графического пользовательского интерфейса приложения, чтобы использовать его для создания событий пользовательского интерфейса. Модель представляет собой конечный автомат, состояния которого являются различными компонентами приложения, а его переходы – различные события пользовательского интерфейса.

– Систематический метод тестирования: такие программы генерируют уникальное поведение пользователей путем динамического анализа исходного кода приложения.

Все вышеперечисленные подходы имеют одно заметное ограничение: эти методы направлены на поиск технических ошибок и дефектов в приложении, то есть сбоев, вызванных необработанными исключениями [5]. Однако многие из ошибок, присутствующих в современных приложениях, связаны с логикой программы.

Классификация типов компонентов. После изучения более сотни мобильных приложений были выведены семь видов компонентов.

1. Загрузочный экран. Это экран, отображаемый при открытии компонентов, на котором обычно изображены картинка или текст. Пока мы видим данный компонент, приложение загружается в фоновом режиме. Чаще всего заставка является первым экраном в приложении.

2. Компонент-реклама. Большинство Android-приложений можно загрузить бесплатно. Поэтому для получения прибыли разработчики включают в продукт рекламу. Рекламные объявления могут появиться в любое время в любом месте приложения, что усложняет их классификацию. Идентификация таких действий важна, так как нажатие объявления во время тестирования приведет к выходу из приложения.

3. Компонент входа. Множество современных приложений требуют имени пользователя и пароля для использования служб приложения.

4. Компонент соединения: У многих современных средств коммуникации, таких как веб-сайты, новостные порталы, онлайн-телевидение, есть специальное мобильное приложение, позволяющее пользователю получить доступ к медиаконтенту на своем телефоне. Компонент соединения – это экран, содержащий такие приложения.

5. Почтовый компонент. Это компонент, агрегирующий почтовые приложения с такими функциями, как управление входящей

почтой и отправкой новых писем. Такие приложения имеют четко определенную цель с ограниченным количеством возможных действий.

6. Компонент-браузер. Браузеры – это одни из наиболее важных приложений, поскольку они позволяют пользователю получать доступ к веб-сайтам на своем мобильном устройстве.

7. Компонент-список. Приложения со списками обычно имеют общую цель – отслеживать список задач пользователя.

Построение вектора признаков. Каждый компонент может быть наполнен большим количеством функций, связанных с содержащимися в них элементами пользовательского интерфейса, такими как классы элементов, атрибуты, различное их положение в компоненте, а также навигационная панель [6]. При построении признакового описания объектов необходимо решить, какие элементы являются наиболее информативными и могут различаться для разных компонентов. Такие элементы являются интерактивными. Кроме того, изучив базовые шаблоны компонентов из руководства по дизайну компонентов Android Studio, можно сделать вывод, что каждый компонент можно разделить на три части: верхнюю, среднюю и нижнюю. Используется следующее деление экрана: 20 % – 60 % – 20 %, как показано на рис. 1.

Это позволяет выделить следующие группы интерактивных элементов:

– Элементы, которые можно нажимать.

– Горизонтально пролистываемые элементы.

– Вертикальные элементы, которые можно смахивать.

– Элементы текстового поля, в которые можно вводить текст.

Классификатор. Для создания классификатора используется специальный набор программного обеспечения для машинного обучения [7]. Требуется провести десятикратный процесс классификации набора данных с различными алгоритмами классификации, измеряя точность каждого из них. На рис. 2 показана точность, усредненная по предсказанию типа компонента с использованием различных моделей классификации.

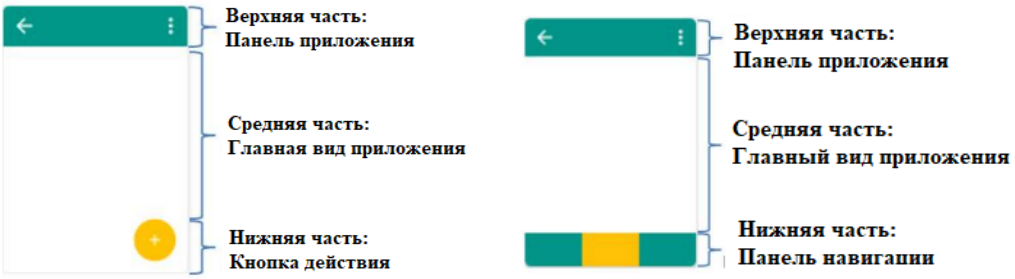

Рис. 1. Стандартные шаблоны Android-приложений

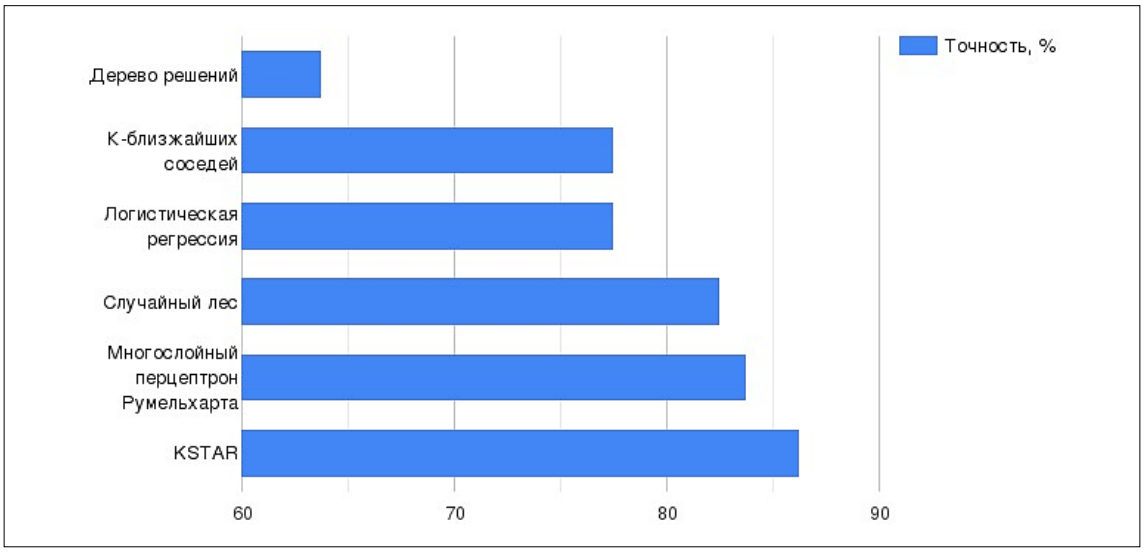

Рис. 2. Точность предсказания типа компонента с использованием различных моделей классификации

Как видно из рис. 2, с помощью классификатора KSTAR удалось достичь наивысшей точности классификации. Таким образом, моделью выбора для данной работы становится KSTAR. Данный классификатор использует меру энтропийного расстояния в качестве функции сходства, чтобы определить, какие из экземпляров больше всего похожи на исходный [8]. Кроме того, требуется оценить, какие функции вносят больше информации в модель исследования. Для этого нужно провести отбор признаков на основе коэффициента прироста информации, который оценивает «ценность» признака путем измерения прироста информации по отношению к классу.

Функции, связанные с количеством нажимаемых элементов в разных частях экрана, так же как и количество общих элементов, показывают высокий коэффициент прироста информации при классификации компонента по типу. Однако функции, связанные со смахиваемыми элементами, и текстовые поля в верхней и нижней частях экрана не вносят информации в модель.

Практическая оценка метода в реальном приложении. В качестве эксперимента используются два мобильных приложения с открытым исходным кодом. Чтобы протестировать эффективность выбранного метода, требуется искусственно добавить ошибки.

– *«K-9Mail»* – приложение электронной почты. Рассмотрены следующие компоненты:

о *«Список сообщений»* – компонент почты, содержащий следующие ошибки:

- Пользователь не может открыть содержимое письма из списка входящих.

- Пользователь может отправить электронное письмо без адреса получателя.

- Пользователь не может отправить письмо на действительный адрес электронной почты.

- Пользователь может отправить электронное письмо с недопустимым адресом получателя.

о *«Вход»* – компонент входа содержит следующие ошибки:

- Пользователь может войти в систему без ввода имени пользователя и пароля.

- Пользователь может войти с неверным именем пользователя и неверным паролем.

- Пользователь не может войти в систему с действительным именем пользователя и действительным паролем.

– *«CrimeTalk Reader»* – приложение для просмотра статей. Рассмотрены следующие компоненты:

о *«Главный компонент»* – компонент содержит следующие ошибки:

- Пользователь не может провести пальцем по экрану влево и вправо, чтобы просмотреть различные разделы портала.

- Пользователь не может нажимать на разные вкладки меню для просмотра различных разделов.

- Пользователь не может открыть статью.

Затем специальное программное обеспечение в течение двух минут совершало 50,000 псевдослучайных действий в каждом компоненте. Параллельно выполнялись обычные автоматические тесты. В результате были получены отчеты о количестве обнаруженных логических ошибок, а также классификация каждого компонента.

Результаты исследования и их обсуждение

В таблице представлены результаты эксперимента. Из нее можно выделить три основные тенденции: 1) тестируемый метод смог правильно классифицировать три «невидимых» действия; 2) удалось обнаружить все «заложенные» ошибки, в то время как стандартный тест не обнаружил ни одной. Кроме того, удалось найти логическую ошибку, которая уже была частью исходной версии приложения, без вмешательства в код; 3) стандартным методом удалось обнаружить один сбой в реальном времени, который не был обнаружен тестируемым методом.

Ошибки, обнаруженные стандартным методом автоматического тестирования

Приложение/компонент	Встроенный/добавленный (тип ошибки)	Критическая ошибка в реальном времени	Логические ошибки
K-9Mail: Список сообщений	Встроенный	0 критических ошибок	0 логических ошибок
	Добавленный	1 критическая ошибка	
K-9Mail: Вход	Встроенный	0 критических ошибок	
	Добавленный	0 критических ошибок	
CrimeTalk Reader: Главный компонент	Встроенный	0 критических ошибок	
	Добавленный	0 критических ошибок	

У данного подхода есть ограничения. Поскольку система может иметь бесконечное количество запусков, проверка поведения приложения ограничивается запусками, которые фактически выполняются. Таким образом, тестируемый метод ограничен типами компонентов и функциями, которые были определены заранее. Определенные в данной работе семь типов компонентов были разработаны для доказательства эффективности выбранного подхода, в то же время настоящий продукт будет содержать больше видов компонентов и функций. Кроме того, стоит учитывать объем эксперимента (два приложения), а также тот факт, что некоторые ошибки были заложены в приложения заранее.

Заключение

В данной статье представлен новый подход к тестированию Android-приложений с использованием методов машинного обучения. Использование таких методов позволило классифицировать каждое действие пользователя по определенному типу, что в свою очередь позволяет протестировать различные ожидаемые варианты поведения различных компонентов. Кроме того, данный метод протестирован на разных приложениях и продемонстрировал преимущество перед популярными методами автоматического тестирования. Наглядно продемонстрировано, что тестируемый метод обнаруживает логические ошибки приложения, а не только критические ошибки в реальном времени, что открывает возможность разработки более сложных инструментов тестирования.

Список литературы

1. Takeuchi Mi., Gustiar Mu. Development Money Diary Application Models on Android. International Conference on Information Management and Technology. 2020. P. 1–4.

2. Naja Fa., Mansur Sy., Wibawanto Ad. Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform. 20th International Conference on Advances in ICT for Emerging Regions. 2020. P. 4–6.

3. Pan M., Xu To., Pei Yu. GUI-Guided Test Script Repair for Mobile Apps. IEEE Transactions on Software Engineering. 2022. Vol. 48. No. 3. P. 3–5.

4. Motan Ma., Zein Sa. Android App Testing: A Model for Generating Automated Lifecycle Tests. 4th International Symposium on Multidisciplinary Studies and Innovative Technologies. 2020. P. 2–3.

5. Md Faiz., Anwar Md. Hybrid Classification Model to Detect Android Application-Collusion. 43rd International Conference on Telecommunications and Signal Processing. 2020. P. 1–2.

6. Mahmud Ta. API Compatibility Issue Detection, Testing and Analysis for Android App. 36th IEEE/ACM International Conference on Automated Software Engineering. 2021. P. 6–8.

7. Jha K., Nadi S. Annotation practices in Android apps. IEEE 20th International Working Conference on Source Code Analysis and Manipulation. 2020. P. 2–3.

8. Черевко Н.А., Белов Ю.С. Применение технологий машинного обучения в тестировании программного обеспечения // Высокие технологии и инновации в науке. Международной научной конференции. СПб., 2021. С. 127–129.