DOI 10.37882/2223-2966.2023.07.26

АВТОМАТИЗИРОВАННЫЙ СПОСОБ ГЕНЕРАЦИИ ТЕСТОВЫХ СЦЕНАРИЕВ ДЛЯ ПРОВЕРКИ КОРРЕКТНОСТИ ПЕРЕХОДОВ МЕЖДУ ЭЛЕМЕНТАМИ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

AN AUTOMATED WAY TO GENERATE TEST SCENARIOS TO CHECK THE CORRECTNESS OF TRANSITIONS BETWEEN ELEMENTS OF A MOBILE APPLICATION

N. Naumova
I. Barsukov
E. Mashina
D. Bostrikova

Summary: The article describes the method proposed by the authors for solving the task of automating the testing of the user interface of a mobile application.

The presented solution is designed to test the functionality of the mobile application under test and includes checking the capabilities of the user interface, including interaction with controls, navigation, and response to user input. The novelty of the proposed method consists of the developed universal method of testing the user interface of mobile applications, organised using the PageObject pattern.

The proposed approach allows you to automatically generate test scenarios based on the analysis of transitions between application pages while providing the greatest coverage of transitions and methods associated with them, which allows you to effectively investigate the functionality and behaviour of the user interface of the mobile application under test.

Keywords: system testing, test automation, user interface testing, code generation, mobile testing, PageObject pattern.

Наумова Надежда Александровна
Преподаватель, Национальный исследовательский университет ИТМО (г. Санкт-Петербург)
nnaumova@niuitmo.ru
Барсуков Илья Александрович
Преподаватель, Национальный исследовательский университет ИТМО (г. Санкт-Петербург)
265478@niuitmo.ru
Машина Екатерина Алексеевна
Преподаватель, Национальный исследовательский университет ИТМО (г. Санкт-Петербург)
mashina.katherina@niuitmo.ru
Бострикова Дарья Константиновна
Преподаватель, Национальный исследовательский университет ИТМО (г. Санкт-Петербург)
265065@niuitmo.ru

Аннотация. В статье описывается предложенный авторами способ решения задачи автоматизации тестирования пользовательского интерфейса мобильного приложения.

Представленное решение предназначено для проверки функциональности тестируемого мобильного приложения и включает в себя проверку возможностей пользовательского интерфейса, включая взаимодействие с элементами управления, навигацию, и реакцию на ввод пользователя. Новизна предлагаемого способа состоит в разработанном универсальном способе тестирования пользовательского интерфейса мобильных приложений, организованного с использованием паттерна PageObject.

Предлагаемый подход позволяет автоматически генерировать тестовые сценарии, основанные на анализе переходов между страницами приложений, обеспечивая при этом наибольшее покрытие переходов и методов, связанных с ними, что позволяет эффективно исследовать функциональность и поведение пользовательского интерфейса тестируемого мобильного приложения.

Ключевые слова: системное тестирование, автоматизация тестирования, тестирование пользовательских интерфейсов, генерация кода, мобильное тестирование, паттерн PageObject

Введение

Системное тестирование является одним из видов испытаний программного обеспечения и охватывает широкий спектр проверок системы в целом [1]. Этот вид тестирования выполняется на завершенной системе с целью определения ее работоспособности и соответствия полученного результата заданным ожиданиям.

В общем случае осуществление системного тестирования требует разработки плана тестирования, создания тестовых сценариев, выполнения тестов и анализа результатов, что в большинстве случае представляет собой достаточно трудоемкий процесс при тестировании мобильных приложений с единой логикой, но различной реализацией, вызванной различиями между платформами, для которых предназначено решение [2].

Поскольку бурный рост создаваемых мобильных приложений сегодня является одним из основных трендов развития информационных технологий во всех сферах, создание универсальных автоматизированных процессов тестирования таких решений является одним из способов повышения эффективности всей IT-отрасли [3].

В данной работе описывается предложенный авторами способ решения задачи автоматизации проверки пользовательского интерфейса мобильного приложения в рамках системного тестирования [4]. Представленное решение направлено на проверку функциональности тестируемого мобильного приложения и включает в себя тестирование возможностей пользовательского интерфейса, включая взаимодействие с элементами управления, навигацию реакцию на ввод пользователя [5].

Новизна предлагаемого решения состоит в разработанном универсальном способе тестирования пользовательского интерфейса мобильных приложений, организованного с использованием паттерна PageObject, который широко применяется для управления и организации элементов пользовательского интерфейса на отдельных страницах и экранах приложения [6]. Это позволяет представлять каждую страницу приложения в виде отдельного объекта, который инкапсулирует все элементы интерфейса и операции, связанные с этой страницей, что позволяет абстрагировать тестовые сценарии от деталей реализации интерфейса и упрощает поддержку автоматизированных тестовых сценариев при изменении интерфейса. Кроме того, применение PageObject дает возможность определить единый и надежный способ перехода между страницами приложения и взаимодействия с элементами интерфейса, что позволяет более эффективно управлять элементами пользовательского интерфейса и проверять корректность переходов между различными страницами приложения, поскольку подобные переходы всегда определяются единственным образом, благодаря чему точно известно, на какую страницу будет произведен переход при какой-либо комбинации действий на текущей странице.

Таким образом, взаимодействия пользователя с элементами пользовательского интерфейса можно описать в виде графа переходов, который определяет тестовое покрытие системы, на основании чего возможно начать создавать необходимое универсальное решение для тестирования мобильных приложений.

Модификация подхода к генерации объектов страниц

В качестве основы подход к организации процесса тестирования был выбран подход генерации объектов страниц. В используемом авторами подходе к созданию тестов особый интерес представляют функции реализации методов взаимодействия с элементами страниц, состоящие из определенного доступного списка действий, таких как нажатие на элемент, ввод символов в текстовое поле, очистка текстового поля [7]. Для обеспечения корректного взаимодействия с ними необходимо определить название возвращаемого объекта страницы в блок описания интерфейса объектов страниц, а также изменить в классах-реализациях переопределяемые функции, добавив им тип возвращаемого значения в соответствии с указанным выше в описании интерфейса и возврат из метода соответствующего объекта страницы в зависимости от типа целевой платформы.

Модифицированное описание объектов страниц представлено в листинге 1.

```
type: dialog
common:
fields:
— LAYOUT_ID
functions:
— setup : SetupPasscodeFragment
name: SetupPasscodeDialog

android:
fields:
 LAYOUT_ID: confirm_dialog
 okBtnId: ok_btn
functions:
 setup:
 — click okBtnId
```

Листинг 1. Описание объекта страницы с добавлением переходов

Представленное модифицированное описание объектов страниц позволяет построить граф переходов между страницами мобильного приложения.

Построение графа переходов

Поскольку функция перехода всегда единственным образом определяет, из какого объекта страницы в какой произойдет переход, полученное в ходе разбора описание объектов страниц можно рассматривать в терминах графа, где вершинами являются объекты страниц, а ребрами — функции перехода [8, 9].

Для решения задачи создания корректного описания переходов предлагается создать сущность перехода, которая содержит информацию о методе, реализующем переход, и названии страницы, на которую был произведен переход, а также разработать словарь переходов, где ключом выступает название текущего объекта страницы, а значением — список ранее определенных сущностей переходов из данной страницы. Кроме того, в описываемом решении предлагается создать словарь циклов, где ключом является название текущего объекта страницы, а значением — список методов, осуществляющих переходы из текущего объекта страницы в этот же объект страницы.

После осуществления парсинга текстовых описаний и перед началом генерации объектов страниц произво-

дится сбор информации о переходах. Данный процесс реализован путем итерации по списку тестовых спецификаций, полученному при анализе текстовых описаний, и сбора необходимой информации в словари переходов и циклов, что позволяет получить полную карту переходов между страницами и их методами, которая будет использоваться в дальнейшем при автоматической генерации тестовых сценариев [7].

Полученная при этом информация представляет собой ориентированный мультиграф:

$$G = (V, E, I),$$

где V — множество объектов страниц, E — множество функций осуществления переходов, I — функция, ставящая любому ребру в соответствие упорядоченную пару вершин $(x, y) \in V \times V$.

При этом следует учитывать, что рассматриваемый граф, в сущности, является мультиграфом, поскольку очевидно, что существует множество функций осуществления переходов L, являющимися петлями в конечном графе из-за того, что они производят проверку функциональности самого объекта страницы, но не переход на другой объект. Такие функции не будут рассмотрены как возможные пути и будут добавлены в конечный набор тестов сразу же, как только алгоритм окажется в вершине. Таким образом, дальнейшему анализу будет подвергаться ориентированный граф $G' = (V, E\backslash L, I)$, по которому и будет производиться поиск путей [10].

Для генерации тестовых сценариев на основе переходов между страницами мобильных приложений авторами описываемого решения предлагается применить метод реберной покраски с обходом графа в глубину с использованием веса пути. Этот подход основывается на выполнении обхода графа, в случае, когда выбор следующего ребра основан на поиске минимального веса ребра $\min\{e_1, e_2, \ldots e_n\} : e \in V_i, e \in V_{i+1}$ до следующей непосещенной вершины V_{i+1} на пути к достижению последней вершины, которая, в данном случае, представляет последний объект страницы, все пути из которого ведут в уже посещенные объекты страниц.

В процессе такого обхода графа выбираются петли $L_i : L_i \in V_i$ и добавляются в список действий, которые будут использованы для текущего тестового сценария (при этом, из набора петель подобные пути убираются во избежание возрастания числа одинаковых тестовых сценариев). Когда все вершины, в которые ведут рёбра, уже посещены, выбирается минимальное ребро, после чего маршрут считается законченным, в результате чего будет получен новый тестовый сценарий. Следующим шагом алгоритм убирает последнее ребро, после чего запускает поиск маршрута заново. Если все рёбра из $E\backslash L$ были посещены хотя бы раз, считается, что был построен набор необходимых тестовых сценариев, и алгоритм заканчивает работу.

Таким образом, получается определить путь по вышеописанному графу, состоящий из функций и объектов страниц, представленный в виде начальной страницы и списка пар функции перехода и страницы, на которую будет осуществлен переход. Созданный таким образом список преобразовывается в тестовый сценарий. После этого ребро, соединяющее текущую и предыдущую вершины, удаляется, так как считается, что этот вариант уже проверен. Если после удаления ребра пути в данную вершину больше не остается, то эта вершина также удаляется.

После завершения процедуры удаления вершин осуществляется построение пути от начальной вершины с повторением описанных выше шагов.

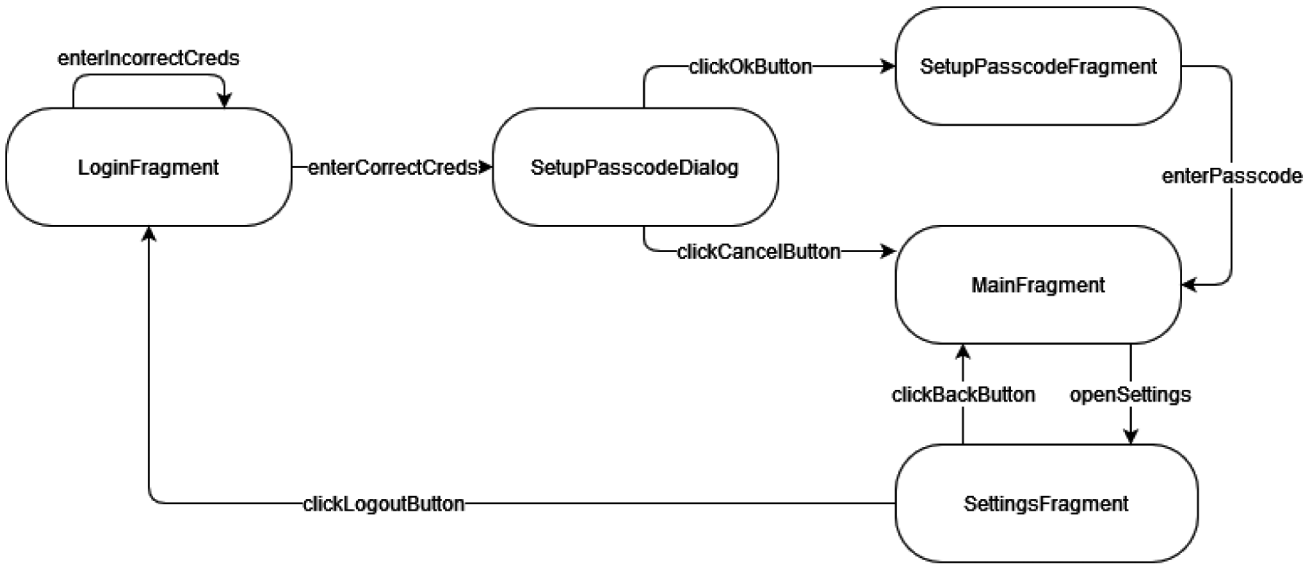

Рис. 1. Граф переходов между страницами приложения

Таблица 1.

Таблица переходов между состояниями приложения

From \ To	LoginFragment	SetupPasscodeDialog	SetupPasscodeFragment	MainFragment	SettingsFragment
LoginFragment	enterIncorrectCreds	enterCorrectCreds	–	–	–
SetupPasscodeDialog	–	–	clickOkButton	clickCancelButton	–
SetupPasscodeFragment	–	–	–	enterPasscode	–
MainFragment	–	–	–	–	openSettings
SettingsFragment	clickLogoutButton	–	–	clickBackButton	–

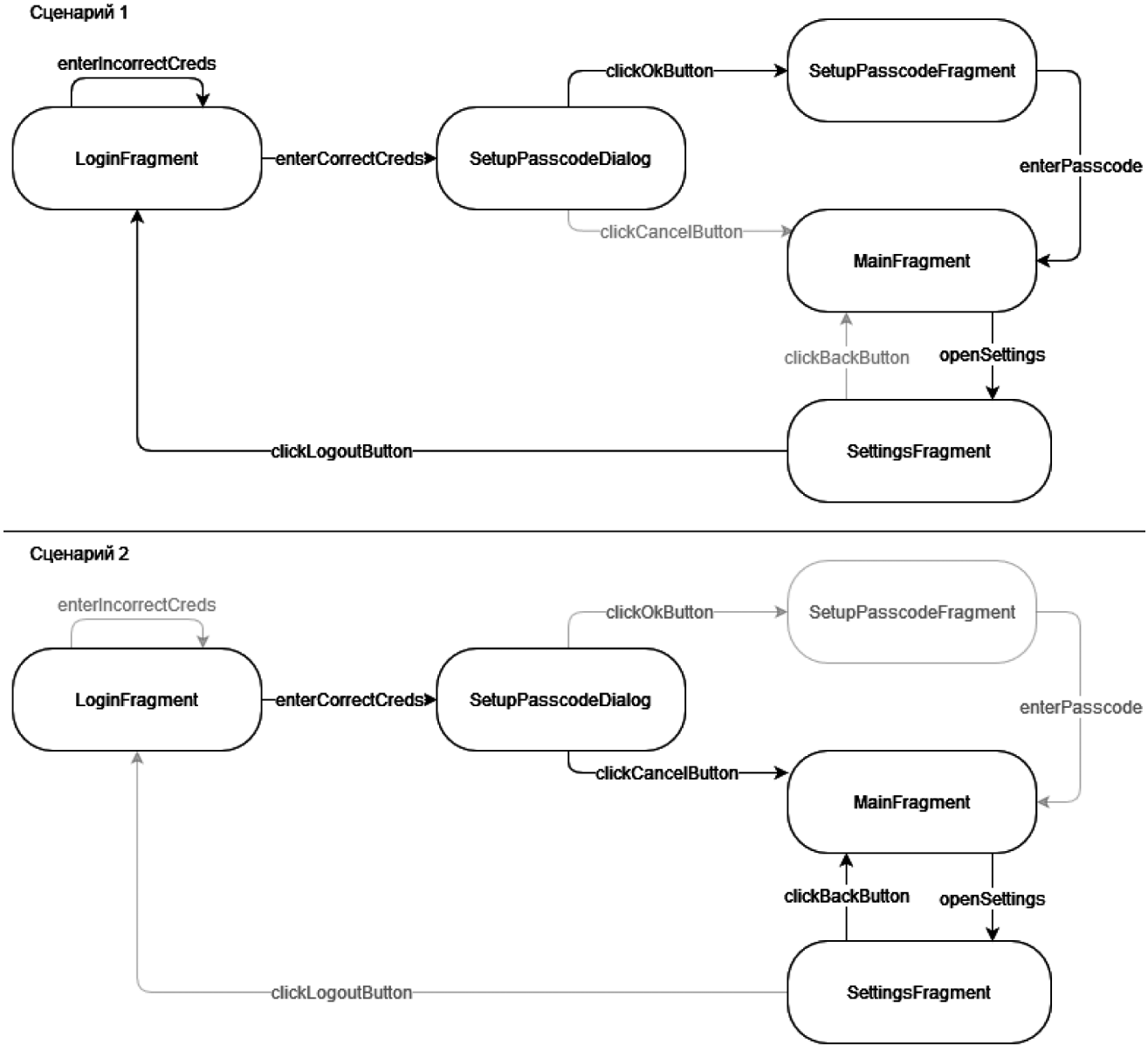

Рис. 2. Визуализация тестовых сценариев, полученных в результате работы алгоритма обхода графа

Реализация тестового сценария

Для генерации тестовых сценариев используется подход, основанный на использовании заранее созданных тестовых сценариев. В этом подходе класс теста служит шаблоном, а метод без тела выступает в качестве шаблона для тестового метода. Далее производится проход по списку пар функции перехода и страницы вида (f_i, p_i),

где f_i — функция перехода на i-том шаге, а p_i — страница, на которую будет осуществлен переход при помощи f_i. Для этой пары выполняется следующая последовательность действий: выбирается переменная типа p_{i-1}, созданная на i – 1 шаге, у него вызывается метод f_i. Результат выполнения записывается в переменную с типом p_i. Данный алгоритм выполняется для всего списка пар, получая, тем самым, тестовый сценарий.

Далее рассмотрим упрощенный сценарий взаимодействия пользователя с мобильным приложением.

На первом шаге взаимодействия с приложением пользователь попадает на страницу логина, где имеется возможность осуществить следующие переходы: остаться на текущей странице логина в случае ввода неправильной пары логин-пароль и нажатия кнопки входа в аккаунт, или перейти в диалоговое окно, предлагающее создать короткий пин-код для аккаунта. В случае отказа от создания пин-кода пользователь будет перенаправлен на главную страницу приложения, а если пользователь решит создать пин-код, он будет перенаправлен на страницу ввода пин-кода. После успешного ввода пин-кода на странице ввода, пользователь будет перенаправлен на главную страницу приложения. На главной странице приложения есть возможность осуществить переход на страницу настроек. На странице настроек пользователь может вернуться на главную страницу приложения или выйти из аккаунта и попасть на страницу логина. Взаимодействие пользователя с приложением можно представить в виде графа переходов, представленного на рисунке 1.

На основании этого графа составлена таблица переходов, представленная в таблице 1, а также текстовое описание страниц приложения. Пример описания интерфейса страницы логина представлен в листинге 2.

```
type: fragment
common:
fields:
— LAYOUT_ID
functions:
— enterCorrectCreds : SetupPasscodeDialog
— enterIncorrectCreds : LoginFragment
name: LoginFragment
```

Листинг 2. Пример описания интерфейса страницы входа в аккаунт

На следующем шаге текстовые описания используются в усовершенствованной системе генерации; таким образом, в ходе работы алгоритма обхода графа получены следующие переходы (рисунок 2), на основании которых сгенерирован исходный код, представленный в листинге 3.

```
class Test {
private lateinit var driverEntity: DriverEntity

@Test(dataProvider = «ProvideDriver», dataProviderClass = DriversUtils::class)
fun test1(driverEntity: DriverEntity) {
this.driverEntity = driverEntity
val wait = WebDriverWait(driverEntity.driver, 60)
val platformName = driverEntity.platform
val driver = driverEntity.driver

val page1 = LoginPageObjectProvider.provide(driver, platformName)
val page2 = page1.enterIncorrectCreds()
val page3 = page2.enterCorrectCreds()
val page4 = page3.clickOkButton()
val page5 = page4.enterPasscode()
val page6 = page5.openSettings()
val page7 = page6.clickLogoutButton()
}

@Test(dataProvider = «ProvideDriver», dataProviderClass = DriversUtils::class)
fun test2(driverEntity: DriverEntity) {
this.driverEntity = driverEntity
val wait = WebDriverWait(driverEntity.driver, 60)
val platformName = driverEntity.platform
val driver = driverEntity.driver

val page1 = LoginPageObjectProvider.provide(driver, platformName)
val page2 = page1.enterCorrectCreds()
val page3 = page2.clickCancelButton()
val page4 = page3.openSettings()
val page5 = page4.clickBackButton()
}

@AfterMethod
fun tearDown() {
try {
when (driverEntity.platform) {
is ANDROID -> driverEntity.driver.resetApp()
is IOS -> driverEntity.driver.removeApp(bundleId)
}
} catch (ignored: Exception) {}
}
}
}
```

Листинг 3. Результат работы генератора исходного кода тестовых сценариев

Отметим, что в контексте данной работы тестовые сценарии выполняют функцию проверки корректности переходов между страницами мобильного приложения. Это достигается за счет особенности инициализации каждого объекта страницы, который проверяет текущую страницу путем сравнения идентификатора фрагмента [7]. Идентификатор фрагмента является уникальным идентификатором, определяющим текущий открытый

фрагмент в приложении. Таким образом, проверка идентификатора фрагмента позволяет убедиться в корректности произведенного перехода.

Проверка полноты тестового покрытия

Для проведения измерений полноты тестового покрытия в данном исследовании был использован следующий методологический подход. В первую очередь был определен набор сценариев использования, которые охватывали функциональные возможности приложения [11]. Затем были разработаны текстовые описания на основе указанных сценариев использования. После этого были сгенерированы тестовые сценарии на основе представленных текстовых описаний.

После выполнения всех тестовых сценариев был проведен анализ результатов для определения количества компонентов пользовательского интерфейса, покрытых тестами. Затем, основываясь на общем количестве компонентов и количестве покрытых тестами компонентов в каждом из определенных сценариев, был рассчитан процентный показатель полноты тестового покрытия приложения.

Такой подход позволяет объективно измерить степень покрытия приложения тестами, исходя из набора сценариев использования и соответствующих им тестовых сценариев [12].

Это позволяет оценить эффективность применяемых методов тестирования и обеспечить достаточный охват функциональных возможностей приложения тестами. Результаты измерения представлены на рисунке 3.

В ходе анализа результатов измерений было обнаружено, что использование графа переходов в процессе тестирования приложений существенно снижает скорость уменьшения процента тестового покрытия при внедрении новой функциональности, по сравнению с применением метода сценариев использования. Полученные результаты показывают, что граф переходов демонстрирует высокую эффективность в поддержании высокого уровня тестового покрытия при внедрении новых возможностей. Они демонстрируют устойчивую тенденцию к росту с каждым последующим выпуском приложения, что приводит к увеличению общего объема протестированных компонентов приложения. Таким образом, использование графов перехода позволяет эффективно управлять тестовым покрытием при добавлении новой функциональности и обеспечивает стабильный и контролируемый рост покрытия приложения в сравнении с использованием сценариев использования.

Специалист по контролю качества, проводящий ручную проверку, ограничивается только стандартными пользовательскими сценариями, не уделяя должного внимания более сложным взаимодействиям с большим числом страниц приложения. В отличие от этого, граф переходов охватывает не только все стандартные про-

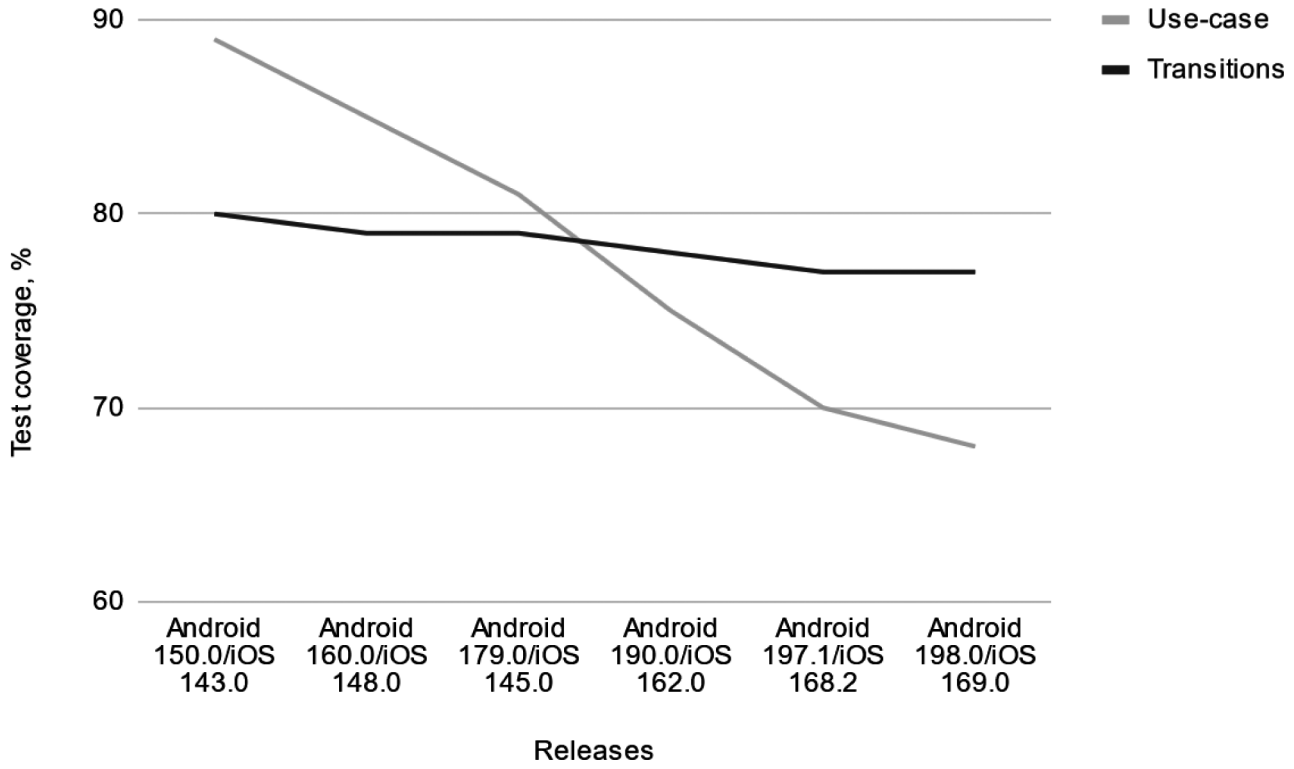

Рис. 3. График изменения меры полноты тестового покрытия при увеличении количества функциональности

верки, но и менее очевидные переходы, что способствует достижению более полного тестового покрытия и, следовательно, выпуску более стабильных версий приложений. Кроме того, такой подход является эффективным для регрессионного тестирования и со временем становится все более экономически оправданным, поскольку инженер по контролю качества не тратит время на повторную проверку ранее проведенных тестов.

Более того, в контексте новой функциональности, генератор исходного кода для проведения тестирования демонстрирует преимущество по сравнению с полной ручной проверкой. Благодаря системе генерации, тестирование может быть проведено более эффективно и быстро, поскольку он позволяет избежать полного ручного тестирования с самого начала.

Дефекты с переменными условиями либо дефекты с неясной процедурой воспроизведения представляют собой сложные сценарии, возникающие в результате непредсказуемых действий пользователей, которые инженер по контролю качества не в состоянии предвидеть и покрыть базовыми проверками функциональности. Подобные дефекты проявляются только в рабочей среде в виде стека вызовов. Однако применение графа переходов в контексте тестирования масштабных приложений позволяет раньше обнаруживать подобные дефекты, поскольку он охватывает не только очевидные сценарии использования, но и общую структуру приложения в целом. За счет выполнения комплексного набора действий в процессе тестирования обнаружение и локализация дефектов с неясной процедурой воспроизведения упрощаются до момента развертывания приложения в среде пользователя.

Заключение

Как следует из приведенных выше результатов сравнения, предложенный авторами подход к генерации тестовых сценариев для проверки корректности переходов между объектами мобильного приложения позволяет автоматически генерировать тестовые сценарии, основанные на переходах между страницами приложений, обеспечивая наибольшее по сравнению со своими аналогами, применяемыми на сегодняшний день при тестировании мобильных приложений, покрытие переходов и методов, связанных с ними, и позволяя эффективно исследовать функциональность и поведение пользовательского интерфейса мобильных приложений.

В связи с этим можно утверждать, что использование предлагаемого подхода к построению тестовых сценариев для проверки работоспособности мобильных приложений существенно повышает как качество тестирования, так и общую производительность разработки.

ЛИТЕРАТУРА

1. Sommerville I. Engineering Software Products: An Introduction to Modern Software Engineering: Pearson, 2019. 352 p.
2. Linares-Vásquez M., Moran K., Poshyvanyk D. Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing // 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). — IEEE. 2017. — С. 399–410.
3. Husmann M., Spiegel M., Murolo A., Norrie M. C. UI testing cross-device applications // Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces. — 2016. — С. 179–188.
4. Ubaidilah M.F., Permatasari D.I., Hardiansyah F.F. Automated Test on Multiple Platform Framework Development // 7th International Conference on Sustainable Information Engineering and Technology 2022. — 2022. — С. 316–319.
5. Abuaddous H.Y., Saleh A.M., Enaizan O., Ghabban F., AlBadareen A.B. Automated User Experience (UX) Testing for Mobile Application: Strengths and Limitations. // International Journal of Interactive Mobile Technologies. — 2022. — Т. 16, No 4.
6. Morgado I.C., Paiva A.C., Faria J.P. Automated pattern-based testing of mobile applications // 2014 9th International Conference on the Quality of Information and Communications Technology. — IEEE. 2014. — С. 294–299.
7. Барсуков И.А., Наумова Н.А., Бострикова Д.К., Машина Е.А. Создание унифицированных механизмов автоматизированного тестирования приложений для мобильных устройств // Инженерный вестник Дона. 2023. №5.
8. Karlsson S., Čaušević A., Sundmark D., Larsson M. Model-based automated testing of mobile applications: an industrial case study // 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). — IEEE. 2021. — С. 130–137.
9. Keng J.C.J., Jiang L., Wee T.K., Balan R.K. Graph-aided directed testing of android applications for checking runtime privacy behaviours // Proceedings of the 11th International Workshop on Automation of Software Test. — 2016. — С. 57–63.
10. Epp S. S. Discrete mathematics with applications. 4th edition: Cengage learning, 2010. 984 p.
11. Vilkomir S. Multi-device coverage testing of mobile applications // Software quality journal. — 2018. — Т. 26, No 2. — С. 197–215.
12. Ivanković M., Petrović G., Just R., Fraser G. Code coverage at Google // Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. — 2019. — С. 955–963.

© Наумова Надежда Александровна (nnaumova@niuitmo.ru); Барсуков Илья Александрович (265478@niuitmo.ru);
Машина Екатерина Алексеевна (mashina.katherina@niuitmo.ru); Бострикова Дарья Константиновна (265065@niuitmo.ru)
Журнал «Современная наука: актуальные проблемы теории и практики»