УДК 33 **DOI: 10.34670/AR.2021.47.34.043**

Экономические аспекты конвертирования кода мобильных приложений Java-Swift с использованием транс-компиляторов

Хрулёва Наталия Дмитриевна

Ведущий программист 1С,
ООО Сатурн-центр,
129344, Российская Федерация, Москва, ул. Искры, 3;
e-mail: nataliia@khruleva.ru

Аннотация

Нативное кроссплатформенное мобильное приложение имеет несколько специфичных для платформы реализаций. Как правило, приложение разрабатывается для одной платформы, а затем переносится на остальные. Перевод приложения с одного языка (например, Java) на другой (например, Swift) вручную утомителен и подвержен ошибкам, в то время как автоматизированные переводчики либо требуют определенных вручную правил перевода, либо сосредотачиваются на переводе API. Чтобы автоматизировать перевод собственных кроссплатформенных приложений, мы предлагаем подход Java2Swift, который итеративно выводит правила синтаксического преобразования и сопоставления API из Java в Swift. Учитывая корпус программного обеспечения на обоих языках, Java2Swift сначала идентифицирует синтаксически эквивалентный код на основе фигурных скобок и сходства строк. Для каждой пары похожих сегментов кода Java2Swift затем создает синтаксические деревья обоих языков, используя минималистские знания предметной области о языковом соответствии (например, операторы и маркеры) для итеративного выравнивания узлов синтаксического дерева и для вывода правил сопоставления синтаксиса и API. Java2Swift представляет выведенные правила в виде шаблонов строк, хранящихся в базе данных, для перевода кода с Java на Swift.

Для цитирования в научных исследованиях

Хрулёва Н.Д. Экономические аспекты конвертирования кода мобильных приложений Java-Swift с использованием транс-компиляторов// Экономика: вчера, сегодня, завтра. 2021. Том 11. № 9А. С. 342-346. DOI: 10.34670/AR.2021.47.34.043

Ключевые слова

Java, Swift, Swift Framework, мобильное приложение, интегрированная среда разработки, Xcode.

Введение

Чтобы увеличить долю рынка, компании-разработчики программного обеспечения и организации с открытым исходным кодом выпускают различные версии своих мобильных приложений для нескольких мобильных платформ. Чтобы обеспечить удовлетворительный пользовательский интерфейс, разработчикам мобильных устройств приходится создавать специализированные платформы.

Мы называем такие приложения нативными кроссплатформенными мобильными приложениями. В качестве конкретного примера нативное кроссплатформенное мобильное приложение может иметь три разные версии: одну – реализованную на Java для Android, другую – реализованную в Swift для iOS, и еще одну – реализованную на C# или C++ для Windows Phone. При разработке кроссплатформенных приложений программисты обычно сначала сосредотачиваются на одной платформе. Как только приложение, разработанное для этой платформы, созреет, оно будет перенесено на остальные платформы. Поскольку исходный и целевой языки программирования следуют разным грамматикам и имеют разные библиотеки программного обеспечения, разработчики нативных мобильных приложений должны одинаково хорошо разбираться в обоих языках и их API, чтобы правильно переносить код. Перевод кода приложения вручную может быть довольно утомительным и подверженным ошибкам, что мотивирует необходимость в подходах, которые могут автоматизировать процесс перевода.

Основная часть

Существующие инструменты переноса кода требуют, чтобы пользователи вручную определяли правила преобразования [Новиков, Соломатин, 2018; Валуцкая, 2017; Лебедев, 2019; Sneed, Verhoef, 2020; Karampatsis et al., 2020; Lamothe, Shang, 2018]. Однако определение этих правил вручную по-прежнему является трудоемким и подверженным ошибкам процессом, так как необходимо указать множество правил сопоставления API и синтаксиса. Например, Java2CSharp [Java2CSharp, www] и j2swift [j2swift, www] могут преобразовывать структуры программ на основе предопределенных правил перевода, но не могут переводить многие API из-за большого объема библиотек, доступных для разных языков.

MppSMT [Nguyen, Nguyen, Nguyen, 2015] – это современный подход, который автоматически переносит код Java в C# с использованием статистического машинного перевода на основе фраз. Некоторые существующие фреймворки HTML5 (например, Sencha [Sencha, www], PhoneGap [PhoneGap, www], Appcelerator [Appcelerator, www], React Native [React Native, www]) могут автоматически переводить Javascript/HTML5 на Java или Swift. Однако эти инструменты требуют, чтобы разработчики следовали определенным API JavaScript, а не поддерживали общий перевод с языка на язык.

В работе предлагается автоматизированный подход переноса программного кода Java в Swift (Java2Swift). Стоит отметить, что исходная и целевая кодовые базы нативных кроссплатформенных мобильных приложений кодируют правила перевода на различных уровнях синтаксиса.

Предлагаемый подход учитывает эти правила и автоматически применяет их для руководства процессом переноса. В частности, Java2Swift осуществляет перенос кода в два

этапа. На первом этапе Java2Swift выводит правила, сравнивая существующие реализации на обоих языках; он итеративно выравнивает и сопоставляет код Java и Swift на основе фигурных скобок (т.е. {и}) и сходства строк. Поскольку оба языка являются объектно-ориентированными, они разделяют основные синтаксические компоненты, включая объявления классов, объявления методов, структуры циклов и условные операторы.

Фигурные скобки, разделяющие эти компоненты, используются одинаково в обоих языках и, таким образом, могут служить в качестве якорей для выравнивания областей кода, которые потенциально синтаксически эквивалентны. Java2Swift – это подход, который итеративно выводит и применяет как синтаксис, так и правила миграции API с Java на Swift на основе минималистичного знания разработчиком предметной области соответствия языков. Java2Swift использует два основных понятия. Во-первых, поскольку значение ключевых арифметических и логических операторов усваивается как часть начального математического образования, разработчики языков, как правило, избегают переопределения этого значения. Во-вторых, другие операторы и маркеры часто также имеют исторически сложившуюся семантику в современных объектно-ориентированных языках программирования. Например, оператор точки (т.е. .) обращается к элементам объекта, скобки (т.е. ()) разделяют выражения, а фигурные скобки (т.е. {}) помечают блоки кода. Поскольку эти операторы и маркеры имеют одинаковую семантику в разных языках, они могут служить якорями для Java2Swift, который выравнивает и сравнивает эквивалентные идиомы кодирования. Кроме того, Java2Swift использует синтаксические деревья Java и Swift согласованного кода, чтобы определить, в каком порядке несколько операторов выражения должны служить якорями выравнивания. Используя общность приоритета операторов между языками, Java2Swift может итеративно находить оператор наивысшего соответствия в синтаксических деревьях для разделения кода различными способами и может выводить несколько правил переноса кода на разных уровнях из одной пары кодов. Полагаясь на разделение и сопоставление строк на основе дерева, Java2Swift может выводить больше сопоставлений шаблонов и аргументов, чем чистые, основанные на разделителях, неитеративные подходы. Подход Java2Swift работает, не требуя от разработчиков вручную указывать соответствие между синтаксическими компонентами разных языков или кодировать синтаксические структуры в виде последовательностей синтаксических символов, тем самым повышая уровень автоматизации, который он обеспечивает.

Заключение

Поскольку собственные кроссплатформенные мобильные приложения стали отраслевым стандартом, их разработка остается сложной задачей. Нами представляется подход Java2Swift, который облегчает перенос таких приложений между различными платформами. Управляемый данными характер Java2Swift может привести к росту его эффективности с увеличением числа кодовых баз, доступных для вывода правил. В дальнейшем планируется расширение Java2Swift для поддержки перевода, включающего рефакторинг кода, для дальнейшего повышения точности перевода кода и расширение его для решения других задач перевода между языками. Поскольку основные мобильные платформы продолжают конкурировать за доминирование на рынке, мобильные разработчики будут продолжать переводить свои приложения на разные языки, и наш подход может упростить этот нетривиальный процесс.

Библиография

1. Валуцкая Э.А. Инструментальная поддержка разработки трансляторов языков программирования с общей семантической базой. Красноярск: Сибирский федеральный университет, 2017.
2. Лебедев Р.К. Перенос и оптимизация компилятора языка C++ и библиотек для мобильной среды разработки для устройств под управлением Android // Информационные технологии. 2019. С. 104-104.
3. Новиков В.А., Соломатин Д.И. Разработка транслятора кода с языка Matlab в код на языке Python // Сборник студенческих научных работ факультета компьютерных наук Воронежского государственного университета. 2018. С. 207-215.
4. Appcelerator. URL: http://www.appcelerator.com.
5. j2swift. URL: https://github.com/patniemeyer/j2swift.
6. Java2CSharp. URL: http://sourceforge.net/projects/j2cstranslator.
7. Karampatsis R.M. et al. Big code!= big vocabulary: Open-vocabulary models for source code // 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 2020. С. 1073-1085.
8. Lamothe M., Shang W. Exploring the use of automated API migrating techniques in practice: an experience report on android // Proceedings of the 15th International Conference on Mining Software Repositories. 2018. С. 503-514.
9. Nguyen A.T., Nguyen T.T., Nguyen T.N. Divide-and-conquer approach for multi-phase statistical migration for source code (t) // In 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2015.
10. PhoneGap. URL: https://build.phonegap.com.
11. React Native. URL: https://facebook.github.io/react-native.
12. Sencha. URL: https://www.sencha.com.
13. Sneed H.M., Verhoef C. Cost driven software migration: An experience report // Journal of Software: Evolution and Process. 2020. Vol. 32. No. 7. P. 22-36.

Economic aspects of converting Java-Swift mobile application code using Trans compilers

Nataliya D. Khruleva

Lead 1C Programmer,
LLC "Saturn-Center",
129344, 3 Iskry str., Moscow, Russian Federation;
e-mail: nataliia@khruleva.ru

Abstract

A native cross-platform mobile app has several platform-specific implementations. Typically, an application is developed for one platform and then ported to others. Translating an application from one language (like Java) to another (like Swift) manually is tedious and error-prone, while automated translators either require manually defined translation rules or focus on translating the API. To automate the translation of native cross-platform applications, we propose the Java2Swift approach, which iteratively infers the API parsing and mapping rules from Java to Swift. Given the software corpus in both languages, Java2Swift first identifies syntactically equivalent code based on curly braces and string similarity. For each pair of similar code segments, Java2Swift then creates syntax trees for both languages, using minimalist domain knowledge of language matching (such as operators and markers) to iteratively align the syntax tree nodes and derive syntax and API matching rules. Java2Swift exposes inferred rules as string patterns stored in a database to translate code from Java to Swift.

For citation

Khruleva N.D. (2021) Ekonomicheskie aspekty konvertirovaniya koda mobil'nykh prilozhenii Java-Swift s ispol'zovaniem trans-kompilyatorov [One approach to converting Java-Swift mobile application code using trans-compilers]. *Ekonomika: vchera, segodnya, zavtra* [Economics: Yesterday, Today and Tomorrow], 11 (9A), pp. 342-346. DOI: 10.34670/AR.2021.47.34.043

Keywords

Java, Swift, Swift Framework, mobile application, integrated development environment, Xcode.

References

1. *Appcelerator*. Available at: http://www.appcelerator.com [Accessed 14/09/2021].
2. *j2swift*. Available at: https://github.com/patniemeyer/j2swift [Accessed 23/09/2021].
3. *Java2CSharp*. Available at: http://sourceforge.net/projects/j2cstranslator.
4. Karampatsis R.M. et al. (2020) Big code!= big vocabulary: Open-vocabulary models for source code. In: *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, pp. 1073-1085.
5. Lamothe M., Shang W. (2018) Exploring the use of automated API migrating techniques in practice: an experience report on android. In: *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 503-514.
6. Lebedev R.K. (2019) Perenos i optimizatsiya kompilyatora yazyka C++ i bibliotek dlya mobil'noi sredy razrabotki dlya ustroistv pod upravleniem Android [Transfer and optimization of the C ++ language compiler and libraries for a mobile development environment for devices running Android]. *Informatsionnye tekhnologii* [Information technologies], pp. 104-104.
7. Nguyen A.T., Nguyen T.T., Nguyen T.N. (2015) Divide-and-conquer approach for multi-phase statistical migration for source code (t). In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*.
8. Novikov V.A., Solomatin D.I. (2018) Razrabotka translyatora koda c yazyka Matlab v kod na yazyke Python [Development of a translator of Matlab code into Python code]. In: *Sbornik studencheskikh nauchnykh rabot fakul'teta komp'yuternykh nauk Voronezhskogo gosudarstvennogo universiteta* [Collection of student scientific papers of the Faculty of Computer Science of Voronezh State University], pp. 207-215.
9. *PhoneGap*. Available at: https://build.phonegap.com [Accessed 22/09/2021].
10. *React Native*. Available at: https://facebook.github.io/react-native [Accessed 17/09/2021].
11. *Sencha*. Available at: https://www.sencha.com [Accessed 14/09/2021].
12. Sneed H.M., Verhoef C. (2020) Cost driven software migration: An experience report. *Journal of Software: Evolution and Process*, 32 (7), pp. 22-36.
13. Valutskaya E.A. (2017) *Instrumental'naya podderzhka razrabotki translyatorov yazykov programmirovaniya s obshchei semanticheskoi bazoi* [Instrumental support for the development of translators of programming languages with a common semantic base]. Krasnoyarsk: Siberian Federal University.