Санкт-Петербургский государственный университет

Программная инженерия

Поляков Александр Романович

Декларативный предметно-ориентированный язык разработки интерфейсов мобильных приложений

Отчёт по учебной (ознакомительной) практике

Научный руководитель:
старший преподаватель Дмитрий Луцив

Консультант:
исследователь в ООО Хуавей Алексей Недоря

Санкт-Петербург
2020

Оглавление

Введение

Жизнь человека в настоящее время тяжело представить без носимых устройств, проникших практически во все сферы деятельности оного. Человеко-машинное взаимодействие в данном случае в большинстве своём осуществляется посредством мобильных приложений – программного обеспечения, специально разработанного для запуска на мобильных устройствах, таких как смартфоны, планшеты, умные часы, автомобили. Изобилие устройств повлекло за собой разнообразие архитектур процессоров [1, 5] и операционных систем, которые необходимо учитывать при разработке приложения.

Несмотря на то, что в создании современных мобильных приложений явно прослеживается тренд на унификацию методологий, интерфейсов, компонентов и других атрибутов разработки программного обеспечения, своеобразными аттракторами данной унификации стали пара наиболее популярных мобильных операционных систем вкупе с несколькими схожими архитектурами процессоров. Перенос программного обеспечения на другие платформы до сих пор остаётся одним из основных подходов к разработке мультиплатформенных мобильных приложений [7]. В последнее время набирают популярность средства разработки программного обеспечения [2, 6, 9, 11, 12], позволяющие разработчикам работать над единой кодовой базой приложения, предназначенной для работы с несколькими конфигурациями пользовательских устройств.

Одним из важных требований к современным средствам разработки мультиплатформенных мобильных приложений является возможность декларативного описания пользовательского интерфейса. Такая возможность позволяет ускорить и удешевить разработку мобильных приложений за счёт разделения труда между программистами логики приложения и дизайнерами пользовательского интерфейса, сохранив при этом единство окружения разработки и исполнения. Современным и популярным подходом к предоставлению пользователям данной функциональности является использование декларативных предметно-

ориентированных языков – языков программирования с высоким уровнем абстракции, отражающих специфику решаемых с их помощью задач, оперируя понятиями и правилами из определённой области.

Accord – язык программирования общего назначения, зародившийся и разрабатываемый в российском научно-исследовательском институте компании Huawei и находящийся на данный момент в стадии ранней разработки. Одной из наиболее перспективных ниш для данного языка является разработка мобильных приложений, требующая от языка, как было описано выше, возможности декларативного описания интерфейсов мультиплатформенных приложений.

1. Постановка задачи

Целью данной работы является создание декларативного предметно-ориентированного языка описания интерфейсов мобильных приложений, разрабатываемых на языке программирования общего назначения Accord. Для её достижения были поставлены следующие задачи:

- выполнить обзор предметной области и существующих решений;
- предложить подход к созданию декларативного предметно-ориентированного языка описания интерфейсов мобильных приложений;
- реализовать данный язык;
- провести апробацию реализованного языка.

2. Обзор

В данном разделе представлен обзор предметной области: преимуществ и основных способов реализации предметно-ориентированных языков; существующих языков программирования общего назначения, предоставляющих пользователям возможность декларативного описания пользовательских интерфейсов с помощью предметно-ориентированных языков; процесс отображения пользовательских интерфейсов.

2.1. Предметная область

2.1.1. Предметно-ориентированные языки

Предметно-ориентированный язык (domain-specific language, DSL) – это язык программирования с более высоким уровнем абстракции, отражающий специфику решаемых с его помощью задач. Такой язык оперирует понятиями и правилами из определенной предметной области [4].

В отличие от языков программирования общего назначения, таких как C, Python, Java, предметно-ориентированные языки предоставляют абстракции, адекватные решаемой проблеме, позволяя выражать решения, написанные с их помощью, кратко и ёмко; причём в некоторых случаях использование DSL не требует квалификации программиста. В качестве примера DSL можно привести SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных. Основным недостатком применения предметно-ориентированных языков является стоимость их разработки, требующая экспертизы как в области разработки языков программирования, так и в целевой предметной области. Это является одной из причин того, что предметные языки редко применяются для решения задач программной инженерии, в отличие от языков программирования общего назначения. Другой причиной отказа от обособленных предметных языков является тот факт, что сочетание программной библиотеки и языка программирования общего назначения может заменять DSL. Программный интерфейс (API) библиотеки

содержит специфичный для определённой области словарь, образованный именами классов, методов и функций, доступный всем пользователям языков программирования общего назначения, подключившим библиотеку. Однако, вышеприведённый подход проигрывает предметным языкам в следующих аспектах [8, 13]:

- устоявшаяся в области нотация, как правило, выходит за рамки ограниченных механизмов определения пользовательских операторов, предоставляемых языками общего назначения;
- абстракции определённой области не всегда могут быть просто отображены в конструкции языков общего назначения [3];
- использование предметно-ориентированного языка сохраняет возможность анализа, верификации, оптимизации, параллелизации и трансформации в рамках конкретной области, что, в случае работы с исходным текстом языка программирования общего назначения, является более сложной задачей.

2.1.2. Подходы к реализации предметно-ориентированных языков

В последнее время всё больше исследований в области предметно-ориентированных языков направлены на категоризацию предметных языков, а также выработку советов и лучших практик, отвечающих на вопросы "когда и как?" создавать DSL для конкретной области [8, 10, 14].

Препроцессинг DSL-конструкции транслируются в более низкоуровневый программный код базового языка программирования общего назначения.

- *Макрокоманда.* Конструкции предметного языка представлены символьными именами, заменяемыми при обработке препроцессором на последовательность программных инструкций базового языка.
- *Транспиляция.* Исходный код предметного языка транслируется в исходный код языка общего назначения.

- *Лексическая обработка.* Трансформация предметного языка в язык общего назначения осуществляется на уровне лексем.

Преимуществом данного подхода является простота реализации DSL, поскольку большая часть семантического анализа выполняется средствами базового языка. В то же время, это является и недостатком данного подхода ввиду отсутствия предметно-ориентированного статического анализа, оптимизаций и сообщений об ошибках.

Встраивание в базовый язык В данном подходе конструкции базового языка используются для построения библиотеки предметно-ориентированных операций. С помощью синтаксиса базового языка задаётся диалект, максимально приближенный к определённой предметной области.

Преимуществом данного подхода является полное переиспользование компилятора или интерпретатора базового языка для построения DSL. Основными недостатками являются сообщения об ошибках, соответствующие спецификации базового языка, и ограниченная синтаксическая выразительность, обусловленная существующим синтаксисом базового языка.

Самостоятельный компилятор В данном подходе для создания DSL используются методы построения компиляторов или интерпретаторов. В случае компилятора, конструкции предметного языка транслируются во внутреннее представление компилятора, а статический анализ производится над спецификацией DSL. В случае интерпретатора, конструкции предметного языка распознаются и выполняются в ходе цикла выборки-распознавания-исполнения (fetch-decode-execute cycle).

Преимуществами данного подхода являются приближенные к предметной области синтаксис языка и сообщения об ошибках. Серьёзным недостатком является необходимость создания нового компилятора или интерпретатора предметного языка.

Компилятор компиляторов Данный подход схож с предыдущим за исключением того, что все или некоторые стадии компиляции выполняются с использованием *компилятора компиляторов* – программы, воспринимающей синтаксическое или семантическое описание языка программирования и генерирующей компилятор для этого языка.

Преимуществом подхода является снижение расходов на создание компилятора предметного языка. Ограниченность итогового DSL возможностями используемого компилятора компиляторов, а также сложность проработки предметного языка в деталях, что может быть критично для достижения определённого уровня производительности и близости сообщений об ошибках к предметной области, составляют недостатки данного подхода.

Расширение существующего компилятора Компилятор языка программирования общего назначения расширяется предметно-ориентированными правилами оптимизации и/или генерации кода.

В сравнении с предыдущим, данный подход менее трудоёмок из-за возможности переиспользования частей существующего компилятора. Однако, стоит отметить, что расширение существующего компилятора может оказаться сложной задачей, для выполнения которой необходима поддержка расширений со стороны компилятора языка общего назначения, а также минимизация пересечений синтаксиса и семантики базового и предметного языков.

Использование готовых инструментов Существующие инструменты и нотации адаптируются под конкретную предметную область. Примером такого подхода являются DSL, основанные на нотации XML. В большинстве случаев предметные языки, полученные данным способом, плохо подходят для их использования людьми в ручном режиме.

Вывод

2.1.3. Процесс отображения пользовательского интерфейса

2.2. Существующие решения

A. Рекомендации по выбору подхода к созданию предметно-ориентированного языка

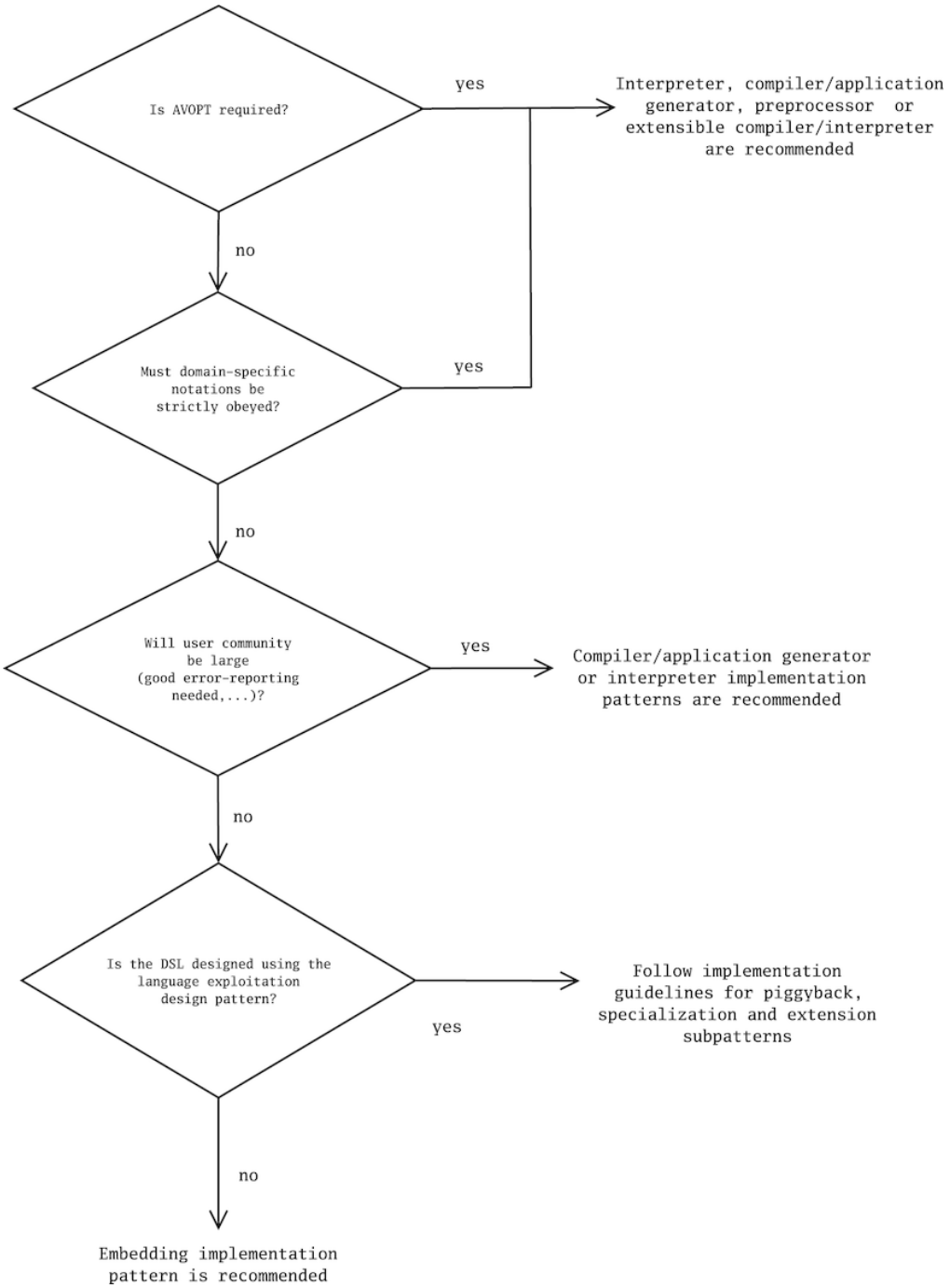

Рис. 1: Алгоритм выбора метода разработки DSL [8]

Список литературы

[1] Advaitha B Mopuru Lahari Gopalakrishnan T. Trends in Processor Architecture of Mobile Phones: A Survey // International Journal of Advanced Science and Technology. — 2020. — May. — Vol. 29, no. 05. — P. 6265 – 6274. — Access mode: http://sersc.org/journals/index.php/IJAST/article/view/15631.

[2] Flutter Homepage. — Access mode: https://flutter.dev/ (online; accessed: 02.12.2020).

[3] Gray Jeff, Karsai Gabor. An Examination of DSLs for Concisely Representing Model Traversals and Transformations. — 2003. — 01. — P. 325.

[4] Kelly Steven, Tolvanen Juha-pekka. Domain-Specific Modeling: Enabling Full Code Generation. — 2008. — 04. — ISBN: 978-0-470-03666-2.

[5] Kolawole Emmanuel Olawale, Lofinmakin Damilola Ayomiposi, Nwidobie Gabriel. Trends in Mobile Phones Processor Architecture, Academia. — Access mode: https://www.academia.edu/38755927/Trends_in_Mobile_Phones_Processor_Architecture (online; accessed: 02.12.2020).

[6] Kotlin Homepage. — Access mode: https://kotlinlang.org/ (online; accessed: 02.12.2020).

[7] Kramer D., Clark T., Oussena S. MobDSL: A Domain Specific Language for multiple mobile platform deployment // 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications. — 2010. — P. 1–7.

[8] Mernik Marjan, Heering Jan, Sloane Anthony. When and How to Develop Domain-Specific Languages // ACM Comput. Surv. — 2005. — 12. — Vol. 37. — P. 316–.

[9] React Native Homepage. — Access mode: https://reactnative.dev/ (online; accessed: 02.12.2020).

[10] Spinellis Diomidis. Notable design patterns for domain-specific languages // Journal of Systems and Software. — 2001. — Vol. 56, no. 1. — P. 91 – 99. — Access mode: http://www.sciencedirect.com/science/article/pii/S0164121200000893.

[11] Swift HomePage. — Access mode: https://developer.apple.com/swift/ (online; accessed: 02.12.2020).

[12] Vue Native HomePage. — Access mode: https://vue-native.io/ (online; accessed: 02.12.2020).

[13] Wile D. S. Supporting the DSL Spectrum // Journal of computing and information technology. — 2001. — Vol. 9, no. 4. — P. 263 – 287.

[14] A preliminary study on various implementation approaches of domain-specific language / Tomaž Kosar, Pablo E. Martı´nez López, Pablo A. Barrientos, Marjan Mernik // Information and Software Technology. — 2008. — Vol. 50, no. 5. — P. 390 – 405. — Access mode: http://www.sciencedirect.com/science/article/pii/S0950584907000419.