Санкт-Петербургский государственный университет

Программная инженерия

Поляков Александр Романович

Декларативный предметно-ориентированный язык разработки мобильных приложений

Отчёт о научно-исследовательской практике

Научный руководитель:
старший преподаватель Дмитрий Вадимович Луцив

Консультант:
руководитель направления разработки языков
Huawei R&D Алексей Евгеньевич 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.

Огромный размер рынка мобильных устройств [10] и приложений повлёк за собой увеличение количества используемых в данной области архитектур процессоров [1, 5] и операционных систем. Подобное разнообразие, а также желание бизнеса оптимизировать процесс разработки мобильных приложений, в свою очередь, стали причиной роста количества средств создания мобильных приложений, предоставляющих своим пользователям функциональность, упрощающую разработку приложений под конкретную платформу и, в последнее время, разработку кроссплатформенных приложений [2, 6, 7, 11, 13, 14].

Высокая конкуренция и технический прогресс неизменно добавляют и усиливают требования, предъявляемые к средствам разработки мобильных приложений текущего и следующего поколений.

Современное средство разработки мобильных приложений представляет собой многокомпонентный программный или программно-аппаратный комплекс, состоящий из языка программирования, его компилятора, отладчика, интегрированной среды разработки, окружения исполнения, отладочных плат целевых устройств или их эмуляторов и прочих инструментов разработки программного обеспечения. Таким образом, занимаясь созданием нового средства разработки мобильных приложений, его авторам необходимо спроектировать и создать вышеописанные компоненты таким образом, чтобы их композиция отвечала как можно большему числу требований.

Accord — язык программирования общего назначения, вбирающий в себя элементы объектно-ориентированного, функционального и компонентного программирования, зародившийся и разрабатываемый в

российском научно-исследовательском институте компании Huawei. На текущий момент язык находится в стадии ранней разработки. Одно из перспективных направлений применения данного языка в будущем — разработка мобильных приложений, что требует проектирования полноценного средства разработки приложений, на двух компонентах которого фокусируется данная работа: языке программирования Accord и его компиляторе. Эти компоненты в дальнейшем будем объединять термином "язык разработки приложений". Оставшиеся компоненты, входящие в состав средства разработки мобильных приложений, в работе затрагиваются в меньшей степени.

1. Постановка задачи

Целью данной работы является создание языка разработки мобильных приложений, основанного на языке программирования общего назначения Accord. Для её достижения были поставлены следующие задачи:

- выполнить сбор и анализ требований к современному языку разработки мобильных приложений;
- выполнить обзор предметной области и существующих решений;
- предложить подход к созданию языка разработки мобильных приложений, удовлетворяющему собранным требованиям;
- реализовать данный язык;
- провести его апробацию.

2. Требования к языку разработки мобильных приложений

В этом разделе представлены требования к языку разработки мобильных приложений наряду с кратким разъяснением причин, по которым они были выдвинуты. Предъявляемые требования были разделены на три группы: функциональные, нефункциональные и бизнес-требования.

2.1. Функциональные требования

Бесшовные реактивные обновления пользовательского интерфейса

"Горячая замена" (*hot-reload*)

Кроссплатформенная разработка

Предоставление отладочных возможностей

2.2. Нефункциональные требования

Декларативность описания пользовательского интерфейса мобильного приложения

Поддержка интегрированной средой разработки

2.3. Бизнес-требования

Поддержка устройств с высоким показателем плотности пикселей на дюйм

Поддержка устройств с высоким показателем кадровой частоты

3. Обзор

В данном разделе представлен обзор предметной области: основные способы реализации предметно-ориентированных языков; процесс отображения пользовательских интерфейсов; существующие языки программирования общего назначения, предоставляющие пользователям возможность декларативного описания пользовательских интерфейсов с помощью предметно-ориентированных языков.

3.1. Предметная область

3.1.1. 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 на-

значения может заменять DSL. Программный интерфейс (*Application Programming Interface*, 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, 15]:

- 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.

3.1.2. 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, 12, 16].

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.

3.1.3. Отображение пользовательского интерфейса

В данном разделе представлен один из способов отображения пользовательских интерфейсов, использующийся в популярных средствах разработки приложений [2, 11, 13, 14]. Графическая подсистема языка Accord основана на схожем подходе. Стоит также отметить, что далее будут рассмотрены лишь базовые принципы построения и отображения интерфейсов, которых должно быть достаточно для объяснения тех или иных оптимизационных решений, принятых в данной работе.

Задачей построения и отображения пользовательского интерфейса занимается так называемый движок рендеринга или отрисовки (*rendering engine*) — программное обеспечение, получающее изображение по какой-либо модели. Здесь *модель* — это описание любых объектов или явлений на строго определённом языке или в виде структуры данных.

На рисунке 1 представлен схематичный шаг алгоритма рендеринга пользовательского интерфейса. Так, отрисовка кадра N состоит из нескольких последовательных этапов:

1. построение дерева компонентов по текущему состоянию пользовательского интерфейса. *Дерево компонентов* — древовидная структура данных, элементами которой являются высокоуровневые компоненты интерфейса. Примером такого дерева может являться DOM (от англ. *Document Object Model* — объектная модель документа), использующийся в современных веб-браузерах.

2. построение дерева элементов по дереву компонентов. *Дерево элементов* — древовидная структура данных, изоморфная дереву компонентов, элементами которой являются структуры данных движка рендеринга.

3. построение дерева рендеринга по дереву элементов. *Дерево рендеринга* — древовидная структура данных, содержащая в себе только необходимые для непосредственно рендеринга элементы интерфейса. Говоря о переходе между деревом элементов и деревом рендеринга как об отображении, можно сказать, что не все узлы

дерева элементов имеют свой образ в дереве рендеринга. Так же стоит отметить, что узлы дерева рендеринга содержат в себе низкоуровневую информацию, такую как относительные координаты элемента в виртуальном пространстве движка отрисовки.

4. размещение и отрисовка: перевод всех относительных свойств элементов в абсолютные, например, перевод относительных координат элемента в виртуальном пространстве движка отрисовки в координаты элемента на экране реального устройства; формирование запроса к аппаратуре для отрисовки интерфейса.

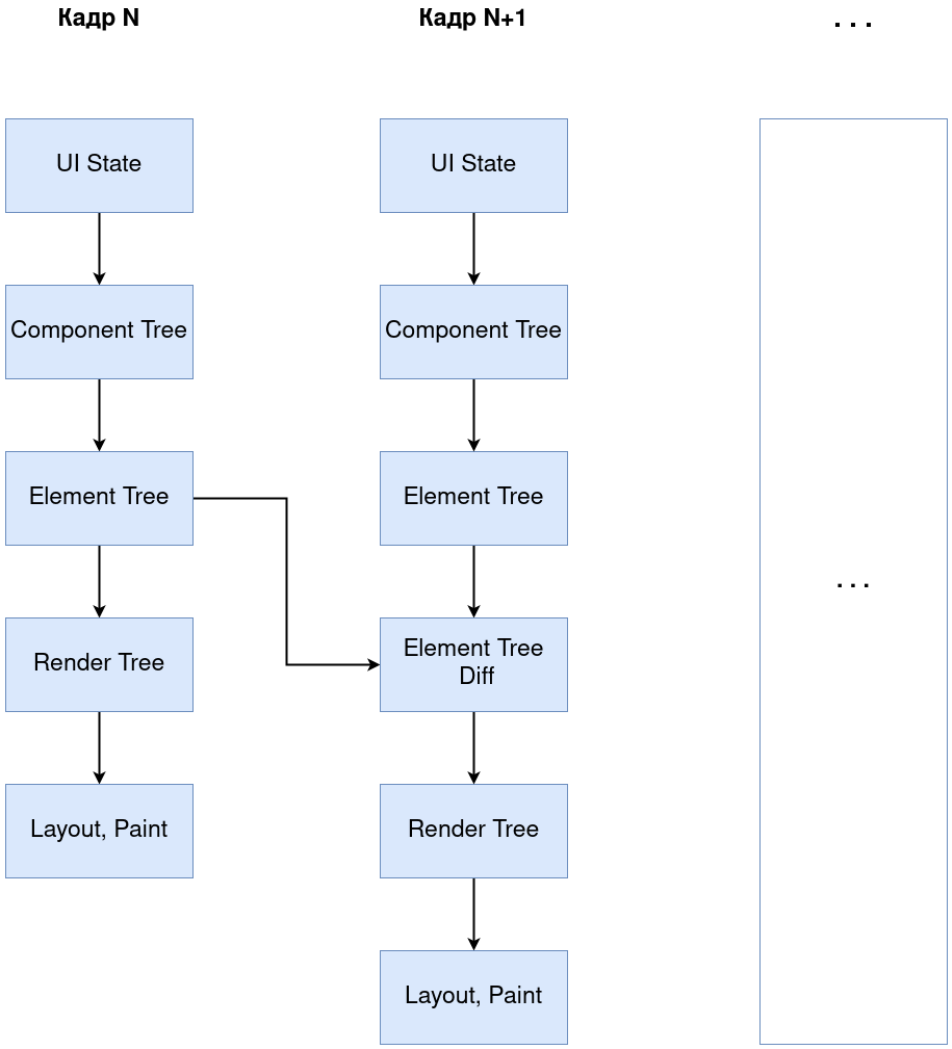

Рис. 1: Схематичный шаг алгоритма рендеринга пользовательского интерфейса

Процесс отрисовки кадра *N+1*, в свою очередь, повторяет вышеописанный алгоритм за исключением следующего: дерево рендеринга перестраивается на основании разницы между деревьями элементов текущего и предыдущего кадров. Подобное дополнение позволяет значительно повысить производительность всего графического конвейера за счёт переиспользования вычислений, хранящихся в дереве рендеринга предыдущего кадра.

3.2. Существующие решения

	VueJS	*SwiftUI*	*Flutter*
Бесшовные реактивные обновления пользовательского интерфейса	+	+	−
"Горячая замена" (*hot-reload*)	+	+	+
Кроссплатформенная разработка	+	−	+
Отладочные возможности	+	+	+
Декларативность описания пользовательского интерфейса	+	+	−
Интегрированная среда разработки	+	+	+
Устройства с высоким показателем плотности пикселей на дюйм	+	?	+
Устройства с высоким показателем кадровой частоты	−	?	+

Таблица 1: Сводная таблица удовлетворения существующих решений собранным требованиям

Как видно из таблицы 1, на данный момент наиболее популярные языки разработки мобильных приложений не удовлетворяют всем требованиям, собранным в данной работе.

4. Архитектура и особенности реализации

В данном разделе представлена общая архитектура решения и особенности реализации, позволившие решению удовлетворить всем требованиям к современному языку разработки мобильных приложений, указанным в разделе 2.

4.1. Архитектура решения

4.2. Особенности реализации

Встраивание предметного языка разработки мобильных приложений в язык программирования *Accord*

Реализация предметно-ориентированного языка разработки мобильных приложений методом встраивания данного предметного языка в базовый язык, которым в данном случае выступает язык программирования *Accord* сохраняет преимущества и недостатки данного подхода, описанные в разделе 3.1.1. Однако, предметно-специфичные анализ, оптимизации и трансформации были учтены на этапе проектирования данного решения, что нивелировало часть недостатков подхода.

Основная функциональность языка *Accord*, использованная для создания предметного языка разработки приложений:

- Легковесные структуры
- Статическая типизация
- Семантический анализ
- Реактивность на уровне языка
- Кроссплатформенность

Разделение дерева компонент на статическую и динамическую части

Благодаря статической типизации, на стадии компиляции возможно разделить пользовательский интерфейс, представленный деревом компонент, на две обособленные в памяти приложения составляющие.

- Статическая часть дерева: отсутствие изменения структуры дерева компонент во время исполнения мобильного приложения позволяет полностью исключить узлы такого дерева из дорогостоящего алгоритма нахождения разницы между деревьями элементов интерфейса.

- Динамическая часть дерева: знание о том, какие элементы интерфейса могут стать точкой начала изменения структуры дерева компонент позволяет компилятору вызывать алгоритм нахождения разницы между деревьями элементов точечно, минимизируя количество входных данных этого алгоритма.

Выделение данных состояния приложения в памяти

Декларативность

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

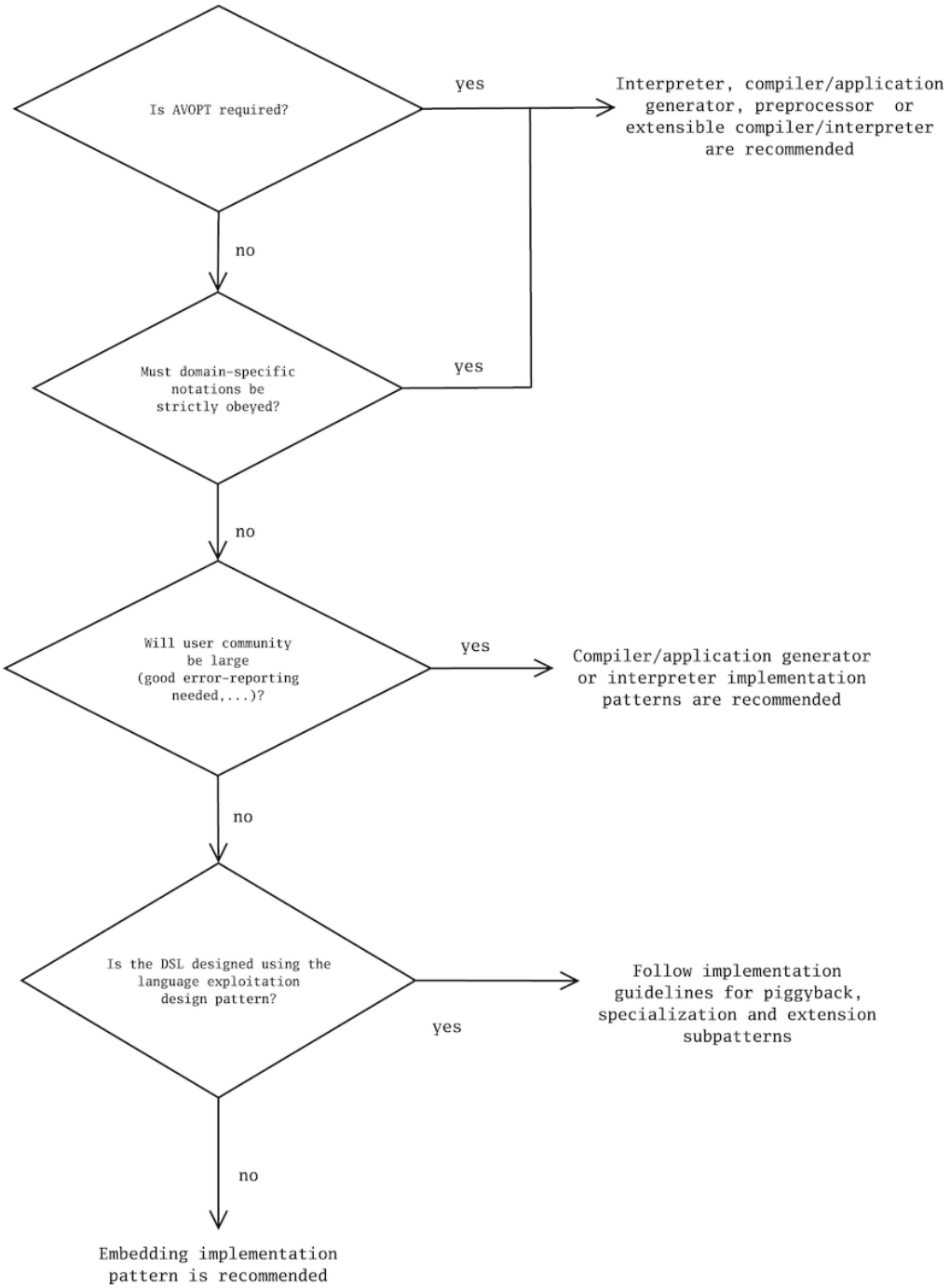

Рис. 2: Алгоритм выбора метода разработки DSL [8]

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] Mobile Devices Pixel Density Statistics. — Access mode: https://pixensity.com/list/phone/ (online; accessed: 20.12.2020).

[10] Mobile Devices Sales Statistics. — Access mode: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/ (online; accessed: 02.12.2020).

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

[12] 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.

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

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

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

[16] 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.