Информатика и вычислительная техника

УДК 004.43 DOI: 10.14529/ctcr150301

ПРОБЛЕМАТИКА ИСПОЛЬЗОВАНИЯ ТЕКСТОВЫХ DSL В ИНФОРМАЦИОННЫХ СИСТЕМАХ

А.А. Кирсанова

Южно-Уральский государственный университет, г. Челябинск

Рассматривается современное положение дел в среде разработки информационных систем с точки зрения внедрения и использования предметно-ориентированных языков в системе для предоставления всего комплекса функционала пользователю. На сегодняшний день существует несколько методик по внедрению предметно-ориентированных языков в информационную систему. В первую очередь эти подходы различаются по типу внедряемого DSL: внутренний или внешний, API-подобный или полностью интегрированный. Если существующие подходы по внедрению предметно-ориентированных языков в информационные системы проанализировать на предмет выявления недостатков, то данные недостатки позволят определиться с кругом проблем, который возникает при использовании текстовых предметно-ориентированных языков в информационных системах. Подобный анализ позволит сформулировать задачи исследования, решение которых поможет преодолеть сложившуюся ситуацию с использованием текстовых предметно-ориентированных языков и соответствующих проблем, которые возникают при интеграции подсистемы интерпретации DSL в информационную систему.

Ключевые слова: доменно-специфичные языки, проектирование DSL, инженерия программных языков.

Введение

Доменно-специфичные языки (или предметно-ориентированные языки) DSL, становятся все более и более популярными в области разработки программного обеспечения информационных систем. Инструментарий для работы с ними также развивается и становится все лучше, позволяя разрабатывать новые DSL каждый раз с все меньшими затратами усилий.

В данной статье рассмотрено современное положение дел в среде разработки информационных систем с точки зрения внедрения и использования предметно-ориентированных языков в системе для предоставления всего комплекса функционала пользователю. Рассмотрены недостатки существующих подходов интеграции DSL в информационные системы и сформулированы задачи, выполнение которых позволит преодолеть выявленные проблемы.

1. Предпосылки

Строго говоря, деление языков программирования на языки общего назначения и предметно-ориентированные весьма условно, особенно, если учесть, что формально любой протокол или, например, формат файлов является языком. Действительно, как известно, формат файла – это спецификация структуры данных, записанных в компьютерном файле [1]. В вычислительной технике общепринятая концепция файла заключается в хранении неструктурированной последовательности байт. Компьютерные программы, сохраняющие в файлах структурированные данные, должны самостоятельно как-то преобразовывать их в последовательность байтов и наоборот. Иначе говоря, языком объектно-ориентированного программирования (ООП) данные необходимо «сериализовывать» и «десериализовывать». Если же говорить в терминологии формальных языков [2, 3], то компьютерные программы для каждого файла выполняют процесс «разбора» и «парсинга», что, по сути, выполняет каждый компилятор с языковыми конструкциями, подаваемыми ему на вход.

Так существует масса языков общего назначения, применяемых в качестве предметно-ориентированных для определённых задач, и наоборот, предметно-ориентированных языков, применяемых в качестве языков общего назначения [4]. Ярким примером является язык Си, разработанный в качестве кроссплатформенного ассемблера, но на практике применяемый гораздо шире. Язык ML, породивший целое семейство языков общего назначения, изначально разрабатывался в качестве DSL, так как подразумевался только для использования в системе автоматического доказательства теорем.

Как в свое время заметил Мартин Уорд: «Рост сложности любой программной системы принципиально ограничен тем пределом, до которого ещё можно сохранять контроль над ней: если объём информации, требуемый для осмысления компонента этой системы, превышает «вместимость» мозга одного человека, то этот компонент не будет до конца понят» [5].

Многие подзадачи (например, задачи сложных статистических расчетов, на которые накладывается множество постоянно изменяющихся во времени требований [5, с. 3]) не удавалось решить посредством классического подхода из-за превышения общей сложности решения возможностей человека по восприятию и переработке информации. Решение же на DSL оказывается не просто возможным, а очень простым и интуитивным, к тому же исключающим размножение ошибок, так как доступ к очень сложным функциям системы осуществляется через примитивы DSL [5].

Именно поэтому DSL применяют в информационных системах как инструментарий для более полного использования предоставляемых системой функций.

2. Область применения

Стоит отметить, что в данной работе мы будем рассматривать лишь текстовые DSL языки (рис. 1) [6].

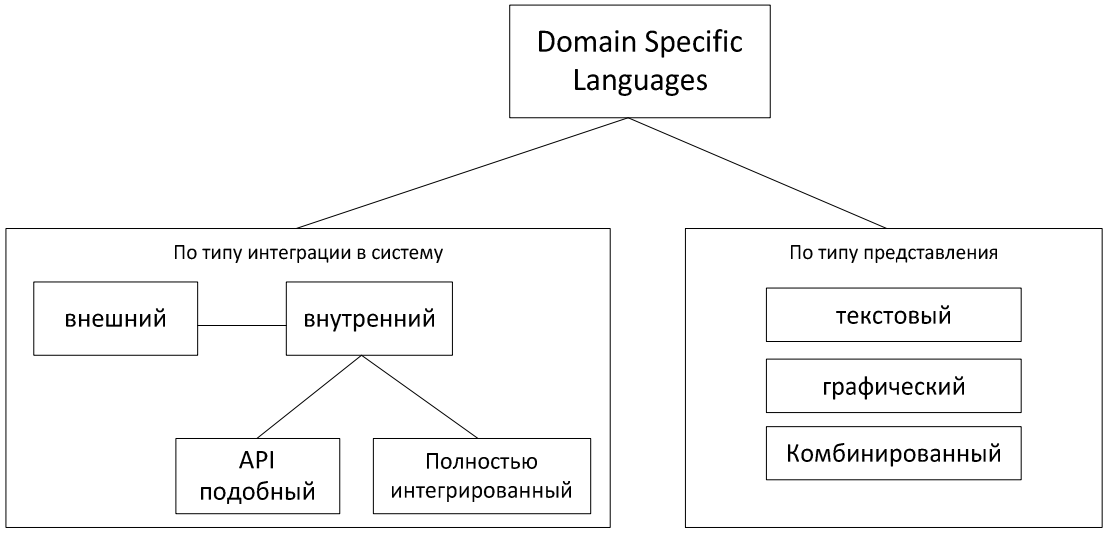

Рис. 1. Классификация DSL

Суть построения грамматики и транслятора DSL языка сводится к написанию внешнего модуля со своей системой классов [7], где описываются все правила, подправила, параметры правил, операторы, допустимые символы, семантические действия и т. д. Затем для каждого описанного правила, оператора и т. п. описываются методы взаимодействия с системой.

Можно рассмотреть следующий пример такого подхода:

```
grammarExpr;

@header {
package test;
import java.util.HashMap;
}
```

```
@lexer::header {package test;}

@members {
/** Map variable name to Integer object holding value */
HashMap memory = new HashMap();
}

prog:   stat+ ;

stat:   expr NEWLINE {System.out.println($expr.value);}
    |   ID '=' expr NEWLINE
        {memory.put($ID.text, new Integer($expr.value));}
    |   NEWLINE
    ;

exprreturns [int value]
    :   e=multExpr {$value = $e.value;}
        (   '+' e=multExpr {$value += $e.value;}
        |   '-' e=multExpr {$value -= $e.value;}
        )*
    ;

multExprreturns [int value]
    :   e=atom {$value = $e.value;} ('*' e=atom {$value *= $e.value;})*
    ;

atomreturns [int value]
    :   INT {$value = Integer.parseInt($INT.text);}
    |   ID
        {
        Integer v = (Integer)memory.get($ID.text);
if ( v!=null ) $value = v.intValue();
else System.err.println("undefined variable "+$ID.text);
        }
    |   '(' e=expr ')' {$value = $e.value;}
    ;

ID  :   ('a'..'z'|'A'..'Z')+ ;
INT :   '0'..'9'+ ;
NEWLINE:'\r'? '\n' ;
WS  :   (' '|'\t')+ {skip();} ;
```

Выглядит не слишком сложно. Однако не стоит забывать о некоторых моментах: это лишь облегченный пример DSL для простейшей программы подсчета арифметических операций, а также что это лишь описание языка. Описывать его использование понадобится в коде отдельно.

Выглядеть это будет примерно следующим образом:

```
importorg.antlr.runtime.*;

publicclassTest {
publicstatic void main(String[] args) throws Exception {
        ANTLRInputStream input = newANTLRInputStream(System.in);
        ExprLexer lexer = newExprLexer(input);
        CommonTokenStream tokens = newCommonTokenStream(lexer);
        ExprParser parser = newExprParser(tokens);
parser.prog();
    }
}
```

Бесспорно, что подобный подход максимально гибок и предоставляет практически неограниченный простор для построения DSL языка. Но что также означает, что создание и поддержание подобного модуля в системе потребует значительных усилий. Если существующая система, в которую планируется внедрить DSL, является средней по размерам (рис. 2), т. е. предоставляет существенное количество функций, но разрабатывается командой разработчиков, не позволяющей по своему количеству назначить лишь по одной роли каждому разработчику [8], подобная нагрузка будет избыточной для данной команды.

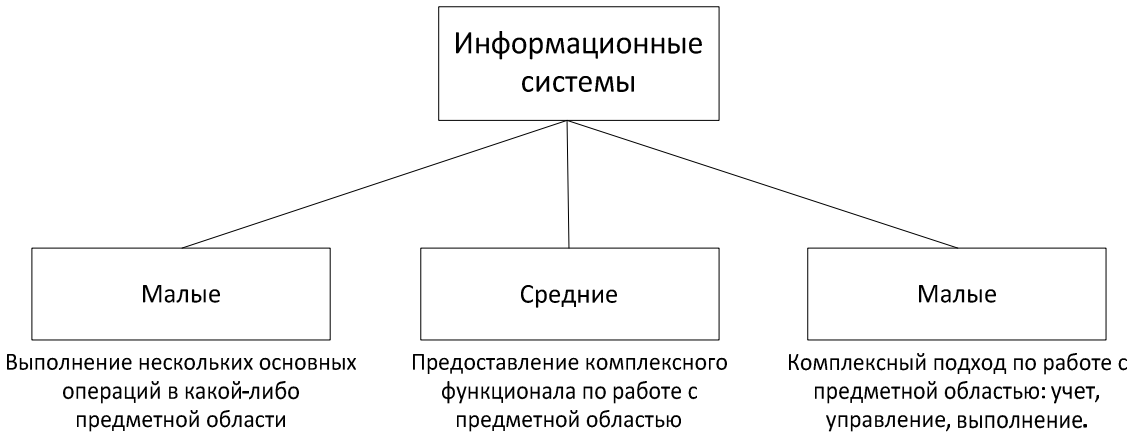

Рис. 2. Классификация информационных систем

Подобные модули в средних и крупных системах потребуют выделения отдельной рабочей силы для поддержания работоспособности модуля. Мелкие информационные системы позволяют иметь разработанный модуль внешнего DSL без поддержки отдельно выделенного разработчика, но зачастую такие системы не нуждаются в подобном функционале по причинам малого множества предоставляемых функций системы, которые можно реализовать без использования языковых средств.

Утверждение, что лучше потратить год на написание программы, которая автоматизирует ваш труд, чем «на коленке» написать частный случай решения за один день, истинно [9]. Однако также общеизвестно, что «сначала человек пишет некий частный случай, потом другой, а затем, уже набравшись опыта, он начинает выполнять однотипные задачи все быстрее и быстрее, и в очередной момент он, найдя закономерность, уже с легкостью сможет написать средство автоматизации, в то время как на подобное решение в самом начале он бы затратил неоправданно много времени» [9, с. 14–15].

Получается, что в таком случае, первый вариант слишком долгозатратен, второй – неэффективен. До сих пор не выявлено методологического подхода, который в равной степени обеспечивал бы эффективное, быстро реализуемое, а также оптимальное по соотношению затрат программное решение по интегрированию DSL в информационную систему.

3. Постановка задач исследования

Проблемы, описанные в предыдущем разделе можно сформулировать следующим образом:

1. Недостаточное освещение проблемы исследования внедрения текстовых DSL в средние и крупные информационные системы без разработки грамматики и синтаксиса языка.

2. Отсутствие выработанной методологии по внедрению внешнего DSL в уже существующую информационную систему без использования больших трудозатрат со стороны разработчиков системы.

Исходя из рассмотренной проблематики, можно вывести задачи, которые требуется выполнить в дальнейшем:

1. Исследовать основные концепции и принципы использования DSL в информационных системах для дальнейшего формулирования абстрактного подхода к построению грамматики и синтаксиса DSL, которая бы не была бы строго зависима от используемой предметной области системы.

2. Предложить методологию по интеграции DSL в информационную систему без написания разработчиками системы отдельных модулей по интерпретации языка системой.

Заключение

Рассмотрены современные проблемы использования предметно-ориентированных языков в информационных системах. Рассмотрены типичные сценарии внедрения DSL в информационную систему на данный момент. Сформулирована проблематика и задачи исследования. Его идея сводится к тому, чтобы разработать методологию создания и поддержки модуля внешнего DSL, но который содержал бы в себе элементы внутреннего DSL, т. е. реплицировался на внутреннее API системы так, чтобы трудозатраты разработчика по работе с модулем языковой системы при разработке собственно информационной системы свелись к минимуму и не затрудняли поддержку и расширение системы в будущем.

Литература

1. Таненбаум, Э. Современные операционные системы / Э. Таненбаум. – СПб.: Питер, 2010. – 1038 с.

2. Ахо, А. Теория синтаксического анализа, перевода и компиляции: в 2 т. / А. Ахо, Дж. Ульман. – М.: Мир, 1978. – Т. 1. – 613 с.

3. Гавриков, М.М. Теоретические основы разработки и реализации языков программирования / М.М. Гавриков, А.Н. Иванченко, Д.В. Гринченков. – М.: КноРус, 2010. – 178 с.

4. Czarnecki, K. DSL implementation in metaocaml, template haskell, and C++ / K. Czarnecki, T. O'Donnell, J.J. Striegnitz, W. Taha. – Berlin, Heidelberg: Springer-Verlag, 2004. –332 p.

5. Ward, M.P. Language Oriented Programming / M.P. Ward. – 1994. – http://www.cse.dmu.ac.uk/~mward/martin/papers/middle-out-t.pdf.

6. Fowler, M. Domain-SpecificLanguages / M. Fowler. – Addison-Wesley, 2011. – 640 c.

7. Fowler, M. Language Workbenches: The Killer-App for Domain Specific Languages? / M. Fowler. – 2005. – http://martinfowler.com/articles/languageWorkbench.html.

8. Братищенко, В.В. Проектирование информационных систем: учеб. пособие / В.В. Братищенко. – Иркутск: Изд-во БГУЭП, 2004. – 84 с.

9. Parr, T. The Definitive ANTLR Reference Building Domain-Specific Languages / T. Parr. – Pragmatic Bookshelf, 2013. – 369 p.

Кирсанова Александра Александровна, программист ВЦ ЮУрГУ, преподаватель кафедры электронных вычислительных машин, Южно-Уральский государственный университет, г. Челябинск; alexander.a.kirsanov@gmail.com.

Поступила в редакцию 2 марта 2015 г.

DOI: 10.14529/ctcr150301

PROBLEMS OF USING TEXTUAL DSL IN INFORMATION SYSTEMS

***A.A. Kirsanova**, South Ural State University, Chelyabinsk, Russian Federation, alexander.a.kirsanov@gmail.com*

Modern situation in information systems design from the aspect of domain-specific languages implementation and using is discussed. Today there are several ways of integrating DSL into information system. Firstly all these methods differ in DSL type being used: internal or external, API-like or fully integrated. If all these methods are analyzed to find out all their disadvantages those ones will help to state problems which can occur while using textual DSL in information system. Such

analysis will make possible to formulate research tasks which after solving will help to create new more efficient method of implemention and integration DSL into information system.

Keywords: domain-specific languages, DSL design, program languages engineering.

References

1. Tanenbaum A. *Sovremennye operatsionnye sistemy* [Modern Operating Systems]. St. Petersburg, Piter Publ., 2010. 1038 p.

2. Aho A., Ullman J. *Teoriya sintaksicheskogo analiza, perevoda i kompilyatsii v 2 tomah*. [The Theory of Parsing, Translation and Compiling in 2 vol.]. Moscow, Mir Publ., 1978, vol. 1, 613 p.

3. Gavrikov M. M., Ivanchenko A. N., Grinchenkov D. V. *Teoreticheskie osnovy razrabotki i realizatsii yazykov programmirovaniya* [Theoretical Basis for the Development and Implementation of Program Languages]. Moscow, KnoRus Publ., 2010. 178 p.

4. Czarnecki K., O'Donnell T., Striegnitz J.J., Taha W. DSL implementation in metaocaml, template haskell, and C++. *Springer-Verlag*, 2004. 332 p.

5. Ward M.P. Language Oriented Programming. 1994. Available at: http://www.cse.dmu.ac.uk/~mward/martin/papers/middle-out-t.pdf.

6. Fowler M. Domain-Specific Languages. *Addison-Wesley*, 2011. 640 p.

7. Fowler M. Language Workbenches: The Killer-App for Domain Specific Languages? 2005. Available at: http://martinfowler.com/articles/languageWorkbench.html.

8. Bratishchenko V.V. *Proektirovanie informatsionnykh system* [Designing Information Systems]. Irkutsk, Baikal State University of Economy and Law Publ., 2004. 84 p.

9. Parr T. The Definitive ANTLR Reference Building Domain-Specific Languages. *Pragmatic Bookshelf*, 2013. 369 p.

Received 2 March 2015

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Кирсанова, А.А. Проблематика использования текстовых DSL в информационных системах / А.А. Кирсанова // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2015. – Т. 15, № 3. – С. 5–10. DOI: 10.14529/ctcr150301

FOR CITATION

Kirsanova A.A. Problems of Using Textual DSL in Information Systems. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2015, vol. 15, no. 3, pp. 5–10. (in Russ.) DOI: 10.14529/ctcr150301