УДК 004.94

О.Л. Викентьева[1], Д.А. Селуков[2]

[1]Национальный исследовательский университет «Высшая школа экономики», Пермь, Россия
[2]Пермский национальный исследовательский политехнический университет, Пермь, Россия

ИСПОЛЬЗОВАНИЕ ПРЕДМЕТНО-ОРИЕНТИРОВАННОГО ЯЗЫКА ДЛЯ РАЗРАБОТКИ ТРЕНАЖЕРА ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ ДЛЯ СБОРКИ ДЕТАЛЕЙ

Программные продукты, разрабатываемые в настоящее время, представляют собой сложные и высоконагруженные системы, затрагивающие различные предметные области. Для создания сложного программного обеспечения применяются различные методы, выбор которых зависит от конечных целей, а также заданного набора ресурсов. На ранних стадиях разработки программного обеспечения, в частности, при формировании концепции приложения, часто возникают проблемы, связанные со сложностью восприятия экспертами языков, используемых для моделирования предметной области, что тормозит процесс разработки. В работе рассматриваются вопросы, связанные с разработкой модуля тренажера виртуальной реальности для сборки деталей. Тренажер виртуальной реальности представляет собой аппаратно-программный комплекс, состоящий из очков виртуальной реальности, устройства Kinect, предназначенного для отслеживания движений пользователя в пространстве, и программы, содержащей различные сборки деталей, которые и предлагается собрать пользователю. Рассматриваемый модуль предназначен для конвертирования деталей из одного формата в другой. Рассматриваются два подхода к разработке: с использованием предметно-ориентированного подхода и с использованием методов объектно-ориентированного программирования. Представлены реализация модуля в среде программирования Visual Studio и алгоритм разработки на основе предметно-ориентированного языка. Также в работе представлена архитектура модуля, разработанная на основе объектно-ориентированного подхода. Выполнено сравнение рассмотренных подходов. Подход на основе DSL позволит легко добавить новые форматы представления деталей в модуль конвертации за счет небольшого изменения DSL. При использовании ООП потребуется переписывать алгоритмы, позволяющие выполнить преобразования из одного формата в другой.

Ключевые слова: предметно-ориентированный язык, объектно-ориентированный язык, генерация кода, языково-ориентированное программирование, тренажер виртуальной реальности.

O.L. Vikenteva, D.A. Selukov[2]

[1]National Research University «Higher School of Economics»,
Perm, Russian Federation
[2]Perm National Research Polytechnic University, Perm, Russian Federation

EMPLOYMENT OF DOMAIN SPECIFIC LANGUAGE IN DEVELOPMENT OF VIRTUAL REALITY SIMULATOR OF PARTS ASSEMBLY

Software products, currently under development, are complex and highly loaded system, covering various subject areas. To build complex software by using different methods, the choice of which depends on the final goals and also given set of resources. In the early stages of software development, in particular, the formation of the concept of the application, there are often problems associated with the complexity of the perception of the experts of the languages used for domain modeling, which hampers the development process. The paper discusses the issues related to the development of a module of the simulator of virtual reality for Assembly details. The virtual reality simulator is a hardware-software complex, consisting of points of virtual reality, the device Kinect, is designed to track the movements of the user in space and programs, containing a variety of Assembly parts, which are collected to the user. The module is designed to convert items from one format to another. Two approaches to development: using object-oriented approach and using methods of object-oriented programming. Are the module implementation in the programming environment Visual Studio and algorithm development based on object-oriented language. Also, the paper presents the architecture of the module developed on the basis of the object-oriented approach. The comparison of the considered approaches. The approach is based on DSL will enable you to easily add new formats for the presentation of parts in the module of conversion by small changes of the DSL. When using OOP you will need to rewrite algorithms to convert from one format to another.

Keywords: domain-specific language, object-oriented language, code generation, language-oriented programming, simulator of virtual reality.

Введение. На сегодняшний день существует огромное количество языков программирования, а также множество подходов и методов для написания программ. Отличие современных методов разработки программных систем от тех, которые были созданы несколько десятилетий назад, заключается в разработке программных средств в комплексе, т.е. помимо организации программного кода рассматриваются также его защита, документирование, использование сторонних сервисов для контроля версий, тестирование и т.д. Различные методы, используемые для разработки сложного программного обеспечения, применяются для различных конечных целей и для определенного набора ресурсов.

Программные продукты, разрабатываемые в настоящее время, представляют собой сложные и высоконагруженные системы, затрагивающие различные предметные области. На ранних стадиях разработки

ПО, в частности, при формировании концепции приложения, часто возникают проблемы, связанные со сложностью восприятия экспертами языков, используемых для моделирования предметной области, что тормозит процесс разработки. Также бывает сложно объяснить понятия, которыми оперируют пользователи информационной системы, с помощью языков общего назначения. Поэтому растет популярность предметно-ориентированных языков программирования (DSL), предназначенных для определенного спектра задач в рамках конкретной предметной области [1]. Предметно-ориентированные языки понятны определенным категориям специалистов, так как они оперируют терминами близкой для них предметной области [2].

Тренажер виртуальной реальности представляет собой аппаратно-программный комплекс, состоящий из очков виртуальной реальности, надетых на голову пользователя, в которые вставлен смартфон, демонстрирующий главное окно программы, устройства Kinect, предназначенного для отслеживания движений пользователя в пространстве, и программы, содержащей различные сборки деталей, которые и предлагается собрать пользователю. Главное окно программы представляет собой набор отдельных деталей, находящихся на определенном расстоянии друг от друга. Двигая руками, пользователь может брать детали и производить над ними определенные операции (закручивание, вставка), тем самым собирая единую деталь. После окончания сборки пользователь может посмотреть статистику, показывающую, за какое время была собрана деталь, какое количество ошибок было совершено, а также сравнить свои результаты с результатами других пользователей тренажера или со своими результатами, полученными ранее. Архитектура программной части тренажера представлена на рис. 1.

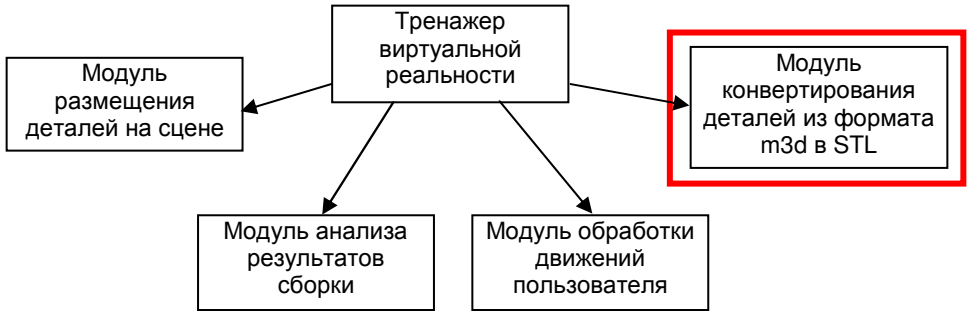

Рис. 1. Архитектура тренажера виртуальной реальности

Рассмотрим модуль конвертирования деталей из формата m3d в формат STL. Детали и сборки в тренажер загружаются из программы «КОМПАС 3D», где изначально создаются. Однако данная программа имеет закрытый файловый формат (m3d), что не позволяет в дальнейшем работать с деталями, созданными в этой программе, в других средах. Для решения этой проблемы был разработан модуль, который представляет собой конвертер, позволяющий переводить детали из данного формата в открытый и часто используемый для хранения 3D-моделей формат STL. Формат STL был выбран потому, что время генерации такого файла относительно мало и он имеет простую структуру. Контекстная диаграмма процесса конвертирования изображена на рис. 2.

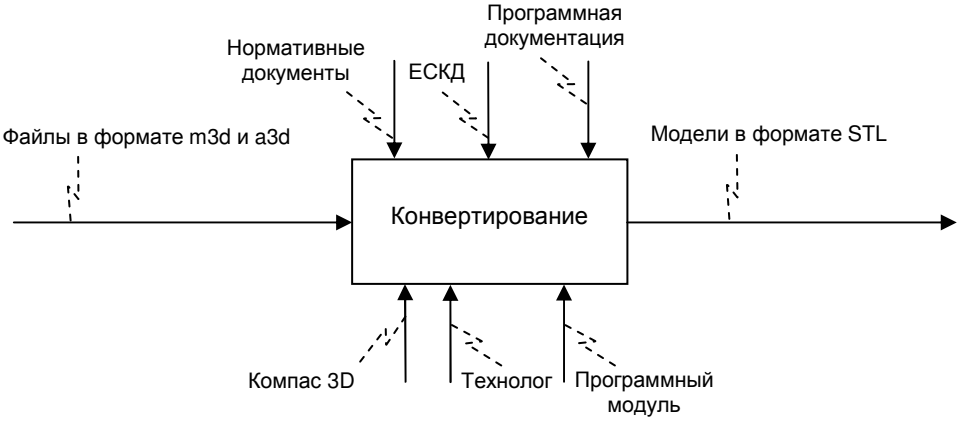

Рис. 2. IDEF0-диаграмма процесса конвертирования

В качестве основной методики для реализации модуля был выбран подход языково-ориентированного программирования (ЯОП). ЯОП представляет собой парадигму программирования, смысл которой состоит в том, что процесс разработки программного обеспечения делится на стадии разработки предметно-ориентированных языков и описания решения задачи с использованием этих языков [3].

Язык DSL представляет собой язык программирования, который применим только для конкретной области. Построение такого языка отражает специфику решаемых с его помощью задач. Актуальность использования DSL на сегодняшний день подтверждается многими известными программными продуктами, использующими или включающими такие языки (Oracle, AutoCAD, Microsoft Office и др.) [4].

Удобство применения DSL также отмечается и на ранних стадиях разработки крупного ПО, когда для проектирования архитектуры ПО разработчикам необходимо понять все тонкости предметной области, которые им могут рассказать эксперты. В итоге получившаяся схема будет понятна как экспертам, поскольку будет содержать понятные им слова, так и разработчикам, так как им будет ясна дальнейшая архитектура приложения. Предметно-ориентированные языки также можно использовать для создания программ для автоматизированного тестирования [5], для рисования блок-схем на основе псевдокода, который создается автоматически при добавлении элементов на лист [6], также для описания документов, которые используются при проектировании информационных систем [7]. Однако язык DSL не является универсальным, а имеет конкретную специфику. Зачастую при разработке синтаксического дерева языка могут возникнуть трудности с созданием зарезервированных слов или с иерархией понятий, поэтому рекомендуется сначала построить онтологию предметной области, затем на ее основании, соблюдая иерархию, выделять слова, наиболее четко характеризующие содержимое каждого класса [8].

В качестве примера можно рассматривать средство для разработки мобильных приложений Ubiq Mobile [9], которое позволяет создавать приложения, используя набор специализированных графических языков, ориентированных на разные предметные области. Это, с одной стороны, понижает сложность разработки, с другой – расширяет области использования мобильных устройств. В некоторых источниках [10] были выделены шаги, которые выполняются при разработке DSL, а также описана архитектура построения конечного программного кода. Не исключено, что язык DSL может использоваться в программах, использующих другой подход – объектно-ориентированный [11], однако рекомендуется придерживаться какого-либо одного подхода, чтобы не было путаницы в структуре программы. Таким образом, средства на основе DSL используются для создания различных программных приложений, в том числе и мобильных. Продукты, содержащие в себе множество DSL для разных сфер, облегчают разработку сложных программных систем.

Разработка предметно-ориентированного языка для модуля тренажера виртуальной реальности. Предметно-ориентированный язык для модуля конвертирования деталей должен описывать детали

в формате САПР «КОМПАС 3D», детали в формате STL и шаги процесса конвертирования (см. рис. 2). Для описания деталей из формата «КОМПАС 3D» было принято решение выделять основные характеристики, а именно: наименование, цвет, позицию в локальных координатах сцены, длину, ширину и высоту. Для описания деталей в формате STL использовались: имя, координаты вершин треугольников, на которые разбивается модель при сохранении, координаты нормалей треугольников. Сам процесс конвертирования является достаточно простой процедурой, поэтому нет смысла его разбивать на шаги, а достаточно просто описать в отдельной функции. Зарезервированные слова получились соответственно следующие: KompasFormat (деталь в формате «КОМПАС 3D»), STLFormat (деталь в формате STL) и Convert для обозначения процесса конвертирования. Поскольку весь DSL небольшой и не прослеживается явного наследования между компонентами, то строить иерархическую модель языка не требуется. После разработки синтаксиса языка его необходимо перенести в среду разработки.

Для реализации DSL была выбрана среда разработки Visual Studio 2015 Community. Так как по умолчанию средств для работы с DSL там нет, то необходимо дополнительно установить Visual Studio SDK и Visual Studio Visualization And Modeling SDK. Данные пакеты содержат необходимые средства для разработки компонентов под саму среду Visual Studio и ЯОП [12].

В Visual Studio процесс создания языка представлен в виде диаграмм, похожих на нотацию UML. Также существуют классы и различные связи между ними. После создания диаграммы на ее основе автоматически генерируется код, который впоследствии можно внедрить в проект и пользоваться уже непосредственно словами DSL, предварительно описав в коде класса и связях, что конкретно они будут делать [13]. При создании проекта DSL автоматически создастся тестовая диаграмма, демонстрирующая возможности языка (рис. 3). После удаления ненужных классов и связей необходимо создать свою диаграмму, описывающую язык [14].

Вначале для удобства будет стоять главный абстрактный класс, характеризующий любую деталь, он создается автоматически по умолчанию, и его нельзя удалить. Данный класс по определению не будет иметь полей [15]. Далее от него наследуется класс KompasFormat, описанный выше со всеми необходимыми полями. Класс KompasFormat будет связан с классом STLFormat связью Convert. Такая небольшая

диаграмма будет удовлетворять нашим требованиям. При создании более сложного языка DSL следует обращать внимание на тип связей между классами, правильно его определить поможет диаграмма, отражающая иерархию связей между синтаксисом языка (рис. 4).

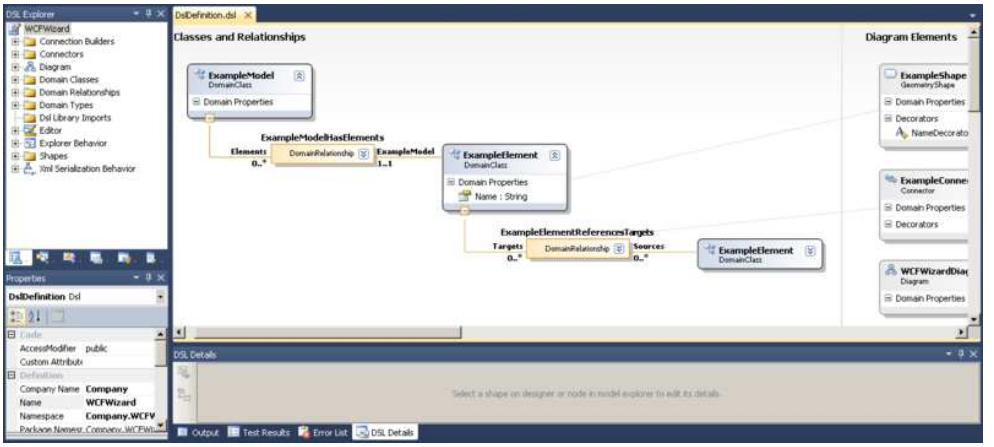

Рис. 3. Начальная диаграмма при создании проекта DSL в Visual Studio

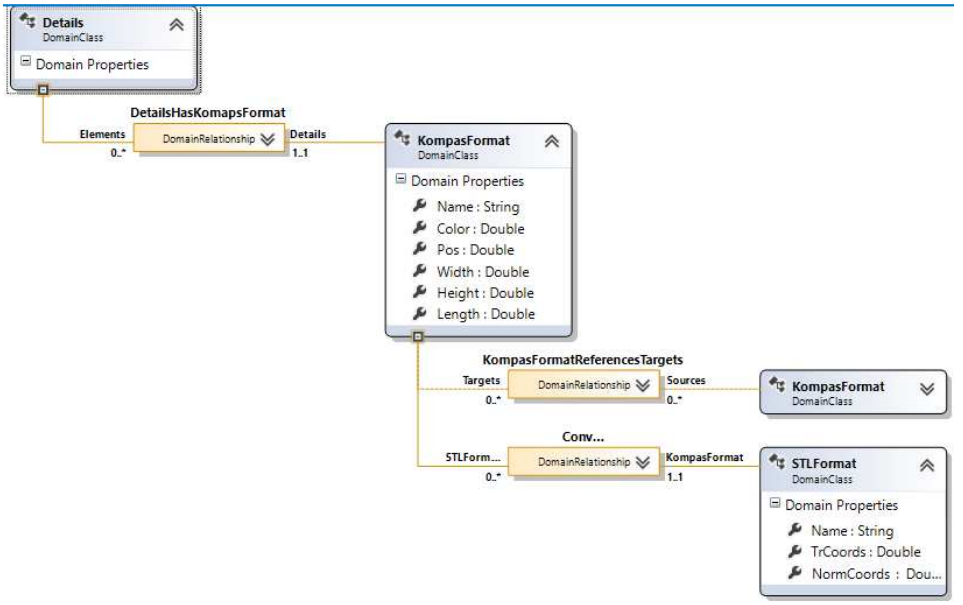

Рис. 4. Диаграмма языка DSL

На основе разработанной диаграммы языка необходимо получить программный код для того, чтобы зарезервированные слова в дальнейшем можно было использовать в модуле конвертации. Для генера-

ции кода использована технология T4, предоставляющая все необходимые инструменты для создания кода с помощью шаблонов. Данная технология позволяет генерировать программный код на языках C#, VB.Net, а также SQL-запросы, HTML-коды и множество других видов текстовой информации [16]. Файлы, являющиеся результатом работы T4, будут иметь расширение *.tt, и именно в них и будут находиться необходимые коды для дальнейшего внедрения в модуль (рис. 5).

После генерации необходимых файлов следует дополнить код классов указанием, какие именно действия необходимо производить при вызове того или иного зарезервированного слова [17]. Например, при вызове KompasFormat нужно считывать параметры детали, указанные в полях данного класса и сохранять их в отдельный файл, при вызове Convert находить файл, который был создан при вызове KompasFormat, брать из него необходимые данные и создавать новый файл с параметрами, необходимыми для сборки STL-файла.

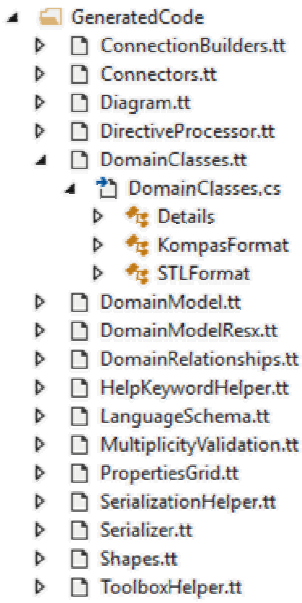

Рис. 5. Сгенерированные классы

Написав все необходимые инструкции, можно начинать внедрение DSL в модуль конвертации. Внедрение подразумевает перенос всех сгенерированных файлов в проект, создание ссылок на них и затем дальнейшее использование. Получившаяся архитектура модуля конвертации представлена на рис. 6.

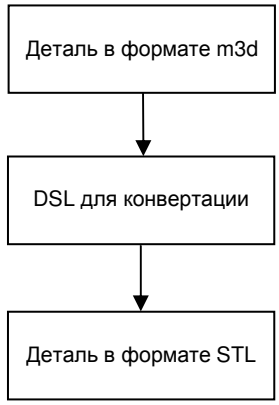

Рис. 6. Архитектура модуля конвертации

Модуль конвертации можно было также реализовать, используя объектно-ориентированный подход. В этом случае нужно было бы написать программу для формата сохранения деталей программой «КОМПАС 3D», создать классы, описывающие детали, сборки, записи данных в файлы. Архитектура модуля представлена на рис. 7.

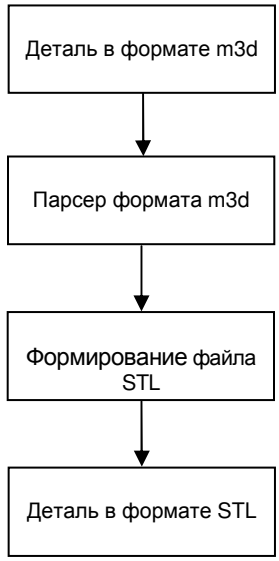

Рис. 7. Архитектура модуля при применении объектно-ориентированного подхода

При сравнении предметно-ориентированного и объектно-ориентированного подходов в каждом можно выделить свои плюсы и минусы. Например, для высоконагруженной системы с большими объемами

данных и сложной архитектурой лучше использовать объектно-ориентированный подход, для небольшой, прозрачно контролируемой и гибкой системы лучше использовать DSL [18]. Также можно отметить, что DSL на стадии проектирования использовать лучше, так как при разработке решения на основе терминов предметной области в нем может разобраться и человек, не являющийся программистом, однако хорошо понимающий данную сферу.

При выборе подхода для разработки необходимо руководствоваться не только целями и задачами приложения, но и его дальнейшим развитием, а также предпочтениями участников команды разработки (таблица). Например, подход на основе DSL позволит легко добавить новые форматы представления деталей в модуль конвертации за счет небольшого изменения DSL. При использовании ООП потребуется переписывать алгоритмы, позволяющие выполнить преобразования из одного формата в другой.

Сравнение предметно-ориентированного и объектно-ориентированного подходов

Критерии сравнения	Предметно-ориентированный подход	Объектно-ориентированный подход
Большая прозрачность и понятность на этапе проектирования	+	–
Гибкость в расширении и минимум репликации кода	+	–
Удобство моделирования компонентов приложения между собой	–	+
Лаконичность написания кода	+	–
Возможность быстрого добавления других форматов представления деталей	+	–

Таким образом, для реализации задачи, описанной в данной работе, лучше использовать метод DSL.

Выводы. Предметно-ориентированный подход позволяет организовать внутренний механизм работы программы таким образом, чтобы он был понятным и хорошо контролируемым. На примере работы модуля конвертации данный подход был продемонстрирован и показал хорошие результаты, немного уступая объектно-ориентированному подходу в производительности из-за большого объема сгенерированного

кода. Следует отметить, что сейчас часто практикуется метод генерации кода по схемам. Например, если имеется диаграмма классов, то для того, чтобы не тратить время на написание самих классов, среда программирования автоматически генерирует необходимую оболочку, а разработчику остается лишь добавить функционал [19]. Данные генераторы в средах, разработанных компанией Microsoft, используют ту же самую технологию T4, описанную выше, а для интерпретирования схем в код используется DSL, описывающий различные диаграммы [20]. Это свидетельствует о том, что данный подход совершенствуется и в дальнейшем, возможно, появятся системы, способные полностью взять на себя задачу кодирования, а разработчикам придется лишь составлять и проектировать визуальные модели.

Библиографический список

1. Лубягина А.О., Лядова Л.Н., Сухов А.О. Проектирование интегрированных информационных систем с использованием DSM-платформы // Известия Южн. федерал. ун-та. – 2014. – № 6(155). – С. 1–12.

2. Воробьев А.Ю. Предметно-ориентированный язык программирования для разработки программного обеспечения для тестирования электронных устройств // Математические структуры и моделирование. – 2014. – № 4(32). – С. 198–205.

3. Журов Д.П., Пятых С.О. Использование внешнего DSL ANTLR для разработки программного обеспечения построения блок-схем по псевдокоду // Austrian Journal of Technical and Natural Sciences. – 2015. – № 10(24). – С. 49–51.

4. Жигалова М.А., Сухов А.О. Предметно-ориентированный язык описания документов, используемых при проектировании информационных систем // Известия Южн. федерал. ун-та. – 2014. – С. 126–134.

5. Кирсанова А.А., Беляков А.Е. Возможности использования онтологии предметной области для определения семантической модели DSL // Вестник Южн.-Урал. гос. ун-та. – 2016. – № 2(16). – С. 154–159.

6. Терехов А.Н., Оносовский В.В. Платформа для разработки мобильных приложений UNIQ MOBILE // Вестник НГУ. – 2011. – № 4(9). – С. 60–70.

7. Григоров А.С. Предметно-ориентированный язык программирования для разработки информационных систем для муниципальных образований // Объектные системы. – 2010. – № 4(20). – С. 56–59.

8. Ботов Д.С. Возможности и проблемы использования современного языкового инструментария в разработке на предметно-ориентированных языках программирования // Вестник Южн.-Урал. гос. ун-та. – 2013. – № 2(13). – С. 128–130.

9. Kohlhauf Lucia, Rutke Ulrike, Neuhaus Birgit. Influence of previous knowledge, language skills and domain-specific interest on observation competency // Journal of Science Education and Technology. – 2011. – Vol. 10. – № 3. – P. 148–180.

10. Примеры задач, в которых имеет смысл использовать DSL [Электронный ресурс]. – URL: http://eax.me/dsl-tasks/ (дата обращения: 05.07.2017).

11. Building domain specific languages on the CLR [Электронный ресурс]. – URL: https://www.infoq.com/articles/dsl-on-the-clr (дата обращения: 05.07.2017).

12. Ward M.P. Language-oriented programming // Software-Concepts and Tools. – 1994. – Vol. 15. – № 4. – P. 147–161.

13. Dmitriev S. Language oriented programming: The next programming paradigm // JetBrains onBoard. – 2004. – Vol. 1. – № 2. – P. 1–13.

14. Building domain specific languages in C# [Электронный ресурс]. – URL: http://www.codemag.com/article/0902041 (дата обращения: 05.07.2017).

15. Domain-specific languages: an introductory example [Электронный ресурс]. – URL: http://www.informit.com/articles/article.aspx?p=1592379 (дата обращения: 05.07.2017).

16. Ribeiro A., Rodrigues da Silva A. Evaluation of XIS-Mobile, a domain specific language for mobile application development // Journal of Software Engineering and Applications. – 2014. – Vol. 7. – № 11. – P. 906–919.

17. Building a productive domain-specific cloud for big data processing and analytics service / Y. Yan, M. Hanifi, L. Yi, L. Huang // Journal of Computer and Communications. – 2015. – Vol. 3. – № 5. – P. 107–117.

18. Bani-Ahmad S. Bibliometry-aware and domain-specific features for discovering publication hierarchically-ordered contexts and scholarly-communication structures // Social Networking. – 2017. – Vol. 6. – № 1. – P. 61–79.

19. A domain specific language for enterprise grade cloud-mobile hybrid applications / A.H. Ranabahu, E.M. Maximilien, A.P. Sheth,

K. Thirunarayan // Proceedings of the Compilation of the Co-Located Workshops on SPLASH'11. – 2011. – Vol. 5. – № 1. – P. 77–84.

20. Simon B., Goldschmidt B., Kondorosi K. A human readable platform independent domain specific language for BPEL // Networked Digital Technologies. – 2010. – Vol. 87. – № 1. – P. 537–544.

References

1. Lubiagina A.O., Liadova L.N., Sukhov A.O. Proektirovanie integrirovannykh informatsionnykh sistem s ispol'zovaniem DSM-platformy [Design of integrated information systems using the DSM-Platform]. *Izvestiia Iuzhnogo federal'nogo universiteta*, 2014, no. 6(155), pp. 1-12.

2. Vorob'ev A.Iu. Predmetno-orientirovannyi iazyk programmirovaniia dlia razrabotki programmnogo obespecheniia dlia testirovaniia elektronnykh ustroistv [Object-oriented programming language for developing software for testing electronic devices]. *Matematicheskie struktury i modelirovanie*, 2014, no. 4(32), pp. 198-205.

3. Zhurov D.P., Piatykh S.O. Ispol'zovanie vneshnego DSL ANTLR dlia razrabotki programmnogo obespecheniia postroeniia blok-skhem po psevdokodu [Using an external DSL ANTLR to develop a pseudo code flowchart software]. *Austrian Journal of Technical and Natural Sciences*, 2015, no. 10(24), pp. 49-51.

4. Zhigalova M.A., Sukhov A.O. Predmetno-orientirovannyi iazyk opisaniia dokumentov, ispol'zuemykh pri proektirovanii informatsionnykh sistem [Object-oriented language for the description of documents used in the design of information systems]. *Izvestiia Iuzhnogo federal'nogo universiteta*, 2014, pp. 126-134.

5. Kirsanova A.A., Beliakov A.E. Vozmozhnosti ispol'zovaniia ontologii predmetnoi oblasti dlia opredeleniia semanticheskoi modeli DSL [Possibilities of using the domain ontology to determine the semantic model of DSL]. *Vestnik Iuzhno-Ural'skogo gosudarstvennogo universiteta*, 2016, no. 2(16), pp. 154-159.

6. Terekhov A.N., Onosovskii V.V. Platforma dlia razrabotki mobil'nykh prilozhenii UNIQ MOBILE [The platform for the development of mobile applications UNIQ MOBILE]. Vestnik Novosibirskogo gosudarstvennogo universiteta, 2011, no. 4(9), pp. 60-70.

7. Grigorov A.S. Predmetno-orientirovannyi iazyk programmirovaniia dlia razrabotki informatsionnykh sistem dlia munitsipal'nykh obrazovanii [Ob-

ject-oriented programming language for the development of information systems for municipal entities]. *Ob"ektnye sistemy*, 2010, no. 4(20). pp. 56-59.

8. Botov D.S. Vozmozhnosti i problemy ispol'zovaniia sovremennogo iazykovogo instrumentariia v razrabotke na predmetno-orientirovannykh iazykakh programmirovaniia [Possibilities and problems of using modern language tools in development on domain-specific programming languages]. *Vestnik Iuzhno-Ural'skogo gosudarstvennogo universiteta*, 2013, no. 2(13), pp. 128-130.

9. Kohlhauf Lucia, Rutke Ulrike, Neuhaus Birgit. Influence of previous knowledge, language skills and domain-specific interest on observation competency. *Journal of Science Education and Technology*, 2011, vol. 10, no. 3, pp. 148-180.

10. Primery zadach, v kotorykh imeet smysl ispol'zovat' DSL [Examples of tasks in which it makes sense to use DSL], available at: http://eax.me/dsl-tasks/ (accessed 05 July 2017).

11. Building domain specific languages on the CLR, available at: https://www.infoq.com/articles/dsl-on-the-clr (accessed 05 July 2017).

12. Ward M.P. Language-oriented programming. *Software-Concepts and Tools*, 1994, vol. 15, no. 4, pp. 147-161.

13. Dmitriev S. Language oriented programming: The next programming paradigm. *JetBrains onboard*, 2004, vol. 1, no. 2, pp. 1-13.

14. Building domain specific languages in C#, available at: http://www.codemag.com/article/0902041 (accessed 05 July 2017).

15. Domain-specific languages: an introductory example, available at: http://www.informit.com/articles/article.aspx?p=1592379 (accessed 05 July 2017).

16. Ribeiro A., Rodrigues da Silva A. Evaluation of XIS-Mobile, a domain specific language for mobile application development. *Journal of Software Engineering and Applications***,** 2014, vol. 7, no. 11, pp. 906-919.

17. Yan Y., Hanifi M., Yi L., Huang L. Building a productive domain-specific cloud for big data processing and analytics service. *Journal of Computer and Communications*, 2015, vol. 3, no. 5, pp. 107-117.

18. Bani-Ahmad S. Bibliometry-aware and domain-specific features for discovering publication hierarchically-ordered contexts and scholarly-communication structures. *Social Networking*, 2017, vol. 6, no. 1, pp. 61-79.

19. Ranabahu A.H., Maximilien E.M., Sheth A.P., Thirunarayan K. A domain specific language for enterprise grade cloud-mobile hybrid applications. *Proceedings of the Compilation of the Co-Located Workshops on SPLASH'11*, 2011, vol. 5, no. 1, pp. 77-84.

20. Simon B., Goldschmidt B., Kondorosi K. A human readable platform independent domain specific language for BPEL. *Networked Digital Technologies*, 2010, vol. 87, no. 1, pp. 537-544.

Сведения об авторах

Викентьева Ольга Леонидовна (Пермь, Россия) – кандидат технических наук, доцент кафедры «Информационные технологии в бизнесе» Национального исследовательского университета «Высшая школа экономики» (614070, Пермь, ул. Студенческая, 38, e-mail: oleovic@rambler.ru).

Селуков Дмитрий Александрович (Пермь, Россия) – магистрант Пермского национального исследовательского политехнического университета (614990, Пермь, Комсомольский пр., 29, e-mail: selukoff2012@gmail.com).

About the authors

Vikentyeva Olga Leonidovna (Perm, Russian Federation) is a Ph.D. in Technical Sciences at Department of Information Technologies in Business Higher School of Economics National Research University (614070, Perm, 38, Studencheskaya str., e-mail: oleovic@rambler.ru).

Selukov Dmitriy Aleksandrovich (Perm, Russian Federation) is a Master Student Perm National Research Polytechnic University (614990, Perm, 29, Komsomolsky pr., e-mail: selukoff2012@gmail.com).

Получено 09.10.2017