Степулёнок Денис Олегович

УДК 004.434

МОДЕЛЬ И МЕТОДЫ РЕАЛИЗАЦИИ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ

Аннотация

На основе анализа истории развития языков программирования делается вывод, что предметно-ориентированный подход может стать следующим шагом эволюции языков программирования и существенно повысить продуктивность и качество разработки программных продуктов. В статье выявляются общие черты существующих методик применения предметно-ориентированного подхода, и на основе анализа их недостатков формулируется новая методика его применения. Приведен пример системы, использующей предметно-ориентированный язык, позволяющий преподавателям, не имеющим опыта программирования, формализовывать сложные условия геометрических задач.

Ключевые слова: предметно-ориентированные языки, дистанционное обучение, использование компьютера в обучении, языково-ориентированное программирование, динамическая геометрия.

ВВЕДЕНИЕ

Одним из классических способов разработки программных продуктов является составление документации, основанной на требованиях к системе, и написание программного кода только после фиксации устойчивого состояния проекта. Проблема состоит в том, что требования к системе не являются неизменными, они развиваются по мере развития представлений пользователей о работе системы, постоянно уточняются, дополняются, некоторые требования перестают быть актуальными. Изменение требований ведет к соответствующему изменению проекта, которое часто влечет серьёзные изменения исходного кода. Для проверки корректности внесённых изменений необходимо повторное тестирование.

По мере развития системы «стоимость» внесения архитектурных изменений непрерывно и экспоненциально растет. Возможное решение проблемы состоит в том, чтобы сразу излагать бизнес-требования (высокоуровневые абстракции предметной области, из которых состоит проект системы) на формальном языке, доступном компьютеру. Как создать подходящий язык для этого?

Универсальные языки программирования для этой цели непригодны. Универсальные языки моделирования UML и XML тоже не подходят, так как XML-конструкции громоздки (обратная сторона самоописательности XML). Описание поведения системы в виде UML-диаграмм будет избыточным и также громоздким.

Получается, что для реализации сложных программных систем необходимо создавать языки, на которых можно запи-

сывать бизнес-требования данной предметной области. Причем для каждого типа проектируемых систем необходим особенный набор языков, поскольку понятия, на которых основана, например, бухгалтерская программа, сильно отличаются от понятий системы для автоматического управления подводным аппаратом.

Для таких языков существует устоявшийся термин Domain-Specific Language (DSL), и русский аналог – предметно-ориентированный язык (ПОЯ), которым мы и будем пользоваться в дальнейшем. Методику программирования с использованием DSL называют предметно-ориентированным или языково-ориентированным программированием [3, 5].

ПРОЕКТИРОВАНИЕ ПОЯ

Проектирование формальных языков «с нуля» – сложная задача. Действительно, при классической схеме разработки любого языка необходимо реализовать лексический, синтаксический анализатор и транслятор, преобразующий программу в исполняемый код.

При проектировании ПОЯ актуален следующий вопрос: нужно ли вносить в него императивные конструкции (условные операторы и циклы)? Не потеряем ли мы в результате то, к чему стремимся: получение простого языка, на котором записана лишь самая суть проекта, без утомительных деталей реализации? Можно сказать, что императивные конструкции целесообразно использовать лишь тогда, когда это продиктовано спецификой предметной области задачи (то есть в предметной области используются списки действий). Декларативные конструкции проще моделировать, потому что они сокращают затраты на реализацию DSL, а также легче воспринимаются и несут больше информации о проекте. В [1] утверждается, что для отбора языковых конструкций для предметно-ориентированного языка следует обратить внимание на термины и понятия, которые используются для формулировки требований к системе.

В [2] проблема традиционного программирования формулируется так: большие временные задержки в реализации идей. Суть состоит в том, что от того момента, когда точно известно, как решить проблему, и до того, когда программист «объяснит» это компьютеру, написав программу, проходит огромный промежуток времени. Сейчас большую часть времени, которое тратится на программирование, занимают попытки выразить естественно-языковые концепции в терминах абстракций языка программирования – процесс сложный, не особо творческий и не эффективный.

Модель применения предметно-ориентированного подхода такова. На каждом этапе своего жизненного цикла программный продукт выражен в объективной форме на некотором языке. Причём сначала он выражен на естественном языке предметной области в виде требований. В конце программный продукт представлен предельно конкретным программным кодом (это формальный язык). В этой модели процесс разработки – это последовательный перевод с одного языка на другой (с более абстрактного на более конкретный на каждом этапе). На ранних стадиях такой перевод осуществляет человек (с естественного языка на высокоуровневый формальный), а для последних этапов реализуется некоторый автоматический переводчик (транслятор предметно-ориентированного языка).

При создании ПОЯ возникает проблема, которая состоит в том, что если задать слишком мало элементов языка, то многие задачи будут описываться длинными и неудобными конструкциями, например, сложно выглядят формальные описания теорем для систем автоматического доказательства. Если же, напротив, задать слишком много синтаксических конструкций, то такой язык будет сложен для реализации. Получается, что при создании языка нужно найти приемлемый баланс количества языковых конструкций. При выборе необходимого и достаточного набора конструкций языка нужно обращать

внимание на язык данной предметной области. Он наиболее понятен экспертам, постоянно оптимизировался в процессе эксплуатации, для сокращения длинных и чрезмерно сложных понятий вводились новые короткие слова-понятия или аббревиатуры, постоянно уточнялся смысл существующих слов, отмирали слова, в которых отпала необходимость.

При использовании ПОЯ упрощается переход между используемыми технологиями реализации программного продукта. Например, если сайт создается на PHP и при проектировании было принято решение об использовании базы данных MySQL, а впоследствии принято решение о переходе на другую базу данных, например, на PostgresSQL или на MSSQL, то приходится изменять многие SQL-запросы, и при этом велика вероятность допустить ошибку. Но если бы эти запросы были написаны на предметно-ориентированном языке, мы бы изменили только транслятор языка без изменения кода на ПОЯ, тем самым мы бы перенесли программный продукт на другую технологию.

МЕТОДИКА ПРИМЕНЕНИЯ ПРЕДМЕТНО-ОРИЕНТИРОВАННОГО ПОДХОДА

1. Первый шаг – сбор требований. Он универсален для любой методологии создания программного обеспечения, и проводится либо в начале процесса разработки, либо на всем его протяжении. Сбор требований к будущей системе включает: опрос заказчиков, экспертов в предметной области, будущих пользователей системы, наблюдение и протоколирование процессов, которые предполагается автоматизировать, формулировка и согласование требований к будущей системе в виде текстов, схем, диаграмм. Поиск и анализ необходимых справочных сведений о предметной области (сведений, которые при формулировке задачи эксперты считают само собой подразумевающимися) [6].

2. Создание *глоссария* проекта (словаря слов или устойчивых словосочетаний, имеющих специфическое или более точное, конкретное значение в контексте данного проекта или данной предметной области). В идеале в глоссарий должны войти все понятия, которые могут по-разному пониматься экспертами в предметной области, пользователями и программистами. Глоссарий помогает более кратко и точно формализовать требования к системе, им пользуются программисты для выделения элементов предметно-ориентированного языка, а эксперты и пользователи – для понимания интерфейса системы.

3. Отделение бизнес-политики (требований к системе, зависящих от текущей ситуации) от постоянных требований (тех, которые не будут меняться на протяжении всего времени эксплуатации системы) [7].

4. Выделение из постановки задачи на естественном языке элементов предметно-ориентированного языка, то есть понятий предметной области и их параметров. При этом действия и операции с элементами предметной области также сами являются понятиями предметной области. Понятие соответствует одному слову или устойчивому словосочетанию. Если какие-то два понятия всегда соответствуют друг другу, например, человек и его дата рождения, то одно становится атрибутом другого.

5. Выбор базового синтаксиса предметно-ориентированного языка. Предлагается использовать в качестве базы синтаксис существующего языка программирования. Это облегчит задачу создания лексического и синтаксического анализатора ПОЯ, так как можно будет использовать уже существующие анализаторы. Кроме того, использование для записи, например, формул в устоявшейся нотации будет привычным для участников проекта. Выбор базового синтаксиса определяет общие правила оформления программ на ПОЯ, которым надо будет в дальнейшем следовать (определяется, каким правилам должны следовать идентификаторы, объявления новых функций, формул, условий, комбинирования понятий предмет-

ной области). Программу на предметно-ориентированном языке можно представить в виде дерева понятий и их параметров, а базовый синтаксис нужен для выделения из текста узлов этого дерева. Кроме того, базовый синтаксис задает:

a) правила различения понятий предметной области (может ли один и тот же идентификатор обозначать различные понятия предметной области в зависимости от окружающего контекста программы);

b) правила проверки грамматической правильности предметно-ориентированной программы; выявление неопределенных идентификаторов.

6. Перевод постановки задачи с естественного языка на предметно-ориентированный язык. Поскольку базовый синтаксис уже выбран, в тексте можно использовать функции, классы, объекты, переменные, еще не реализованные в языке. На предметно-ориентированный язык нужно перевести всю постановку задачи целиком. Если не удается перевести какую-либо часть постановки задачи на предметно-ориентированный язык из-за синтаксических ограничений базового синтаксиса, значит нужно вернуться на предыдущий шаг и изменить базовый синтаксис. Если же оказывается, что двум одинаковым (неразличимым) формулировкам на предметно-ориентированном языке должны соответствовать различные (и детерминированные, не случайные) конечные результаты, то нужно уточнить предметно-ориентированное описание.

7. Реализация сущностей предметной области. Лексический, синтаксический анализ с последующим исполнением или генерацией кода. Правила генерации программного кода могут задаваться в виде правил компиляции на один из существующих языков программирования или напрямую в машинный код целевой платформы. В процессе генерации кода можно: применить различные приемы оптимизации по скорости работы, по количеству потребляемой памяти и т.д.; применить схемы интернационализации (чтобы готовую систему можно было использовать на нескольких языках).

В процессе реализации появляется возможность тестирования получаемого программного продукта. В общем случае программу можно тестировать, только когда она корректно компилируется. Получается, что этап тестирования откладывается до реализации всех понятий ПОЯ. Для устранения этой проблемы предлагается автоматически создавать пустые реализации (не порождающие кода) для всех элементов предметно-ориентированного языка и отмечать их как нереализованные. Затем начинать реализацию с самых вложенных элементов, не требующих для своей работы, чтобы другие элементы уже работали. Если выделение таких групп элементов вызывает затруднения, следует применить методику модульных (unit) тестов [4]. При этом выделяются небольшие кусочки кода на предметно-ориентированном языке. Эти кусочки упрощаются до такой степени, чтобы программист четко понимал, что должно в итоге получиться (мог сам наглядно посчитать ответ), и по каждому из них пишется модульный тест. При реализации такого подхода повторяются 3 шага:

1. Создается программный код модульного теста, в котором на вход генератора программного кода передается исходный текст на ПОЯ, а на выходе ожидается сгенерированный код (который программист и так может себе представить).

2. Дальше все модульные тесты запускаются на выполнение и успешно выполняется весь набор тестов, кроме нового теста, который выполняется не успешно (показывает ошибку). Этот шаг необходим для проверки самого теста – включен ли он в общую систему тестирования и правильно ли отражает новое требование к системе, которому она, естественно, еще не удовлетворяет.

3. Генератор кода изменяется, с тем чтобы *как можно скорее* выполнялись все тесты. Нужно добавить самое простое решение, удовлетворяющее новому тесту и одновременно с этим не испортить существующие тесты. Большая часть нежелательных побочных и отдаленных эффектов от вносимых в программу изме-

нений отслеживается именно на этом этапе, с помощью достаточно полного набора тестов.

4. Весь набор тестов выполняется успешно. Если же нет, то мы возвращаемся на шаг 3 и исправляем ошибки в коде генератора.

5. Теперь, когда требуемая в этом цикле функциональность достигнута самым простым способом, программа подвергается рефакторингу для улучшения структуры и устранения избыточного, дублированного кода (происходит процесс обобщения кода), можно выполнить оптимизацию кода генератора. При этом в любой момент можно выполнить набор тестов и убедиться, что они все еще срабатывают.

6. Весь набор тестов выполняется успешно. Генератор кода снова находится в согласованном состоянии и содержит четко осязаемое улучшение по сравнению с предыдущим состоянием.

Далее будет рассмотрен пример использования (применения) предметно-ориентированного подхода к разработке программ учебного назначения.

СОЗДАНИЕ СИСТЕМЫ ДЛЯ АВТОМАТИЧЕСКОЙ ПРОВЕРКИ РЕШЕНИЙ ГЕОМЕТРИЧЕСКИХ ЗАДАЧ НА ПОСТРОЕНИЕ

Динамической геометрией называют программную среду, которая позволяет делать геометрические построения на компьютере таким образом, что при движении исходных объектов чертеж перестраивается. Среды динамической геометрии можно использовать как основу для системы, позволяющей учителю формулировать геометрические задачи на построение и автоматически проверять правильность выполнения задачи учеником. В системе необходим ПОЯ, который позволяет описывать построение и набор предикатов, определяющих требуемое построение. Проиллюстрируем на этом примере шаги применения описанной выше методики.

Собираем требования к будущей системе

Для формулировки геометрических задач в языке должны присутствовать геометрические примитивы и отношения между ними. С точки зрения аналитической геометрии все геометрические объекты задаются одним или несколькими уравнениями и, возможно, каким-то количеством неравенств. Конкретный объект определяется его типом (например, точка, прямая, отрезок, луч) и значениями коэффициентов, соответствующих типу объекта, уравнений и неравенств.

Условие задачи должно быть написано как на предметно-ориентированном языке для автоматической проверки решения, так и на естественном языке для предъявления учащемуся (восприятие формального условия неподготовленным человеком даже в предметно-ориентированном языке затруднено).

Примеры геометрических задач на построение: «построить биссектрису угла при помощи циркуля и линейки»; «дан треугольник, построить вписанную в него окружность»; «дан треугольник, построить описанную вокруг него окружность»; «даны 3 касающиеся друг друга окружности, найти вписанную окружность, касающуюся 3-х данных».

Создание глоссария проекта

Общие понятия:

Задача на построение – задача имеет атрибуты:

– *условие для ученика* – текст условия, который будет показан ученику (генерируется полуавтоматически на основе формального условия задачи);

– *дано* – чертеж с начальным построением, на котором все объекты поименованы, чтобы на них можно было ссылаться в формальном условии;

– *построить* – перечислены объекты, которые должен по условию задачи построить ученик;

– *такие что* – перечислены свойства, которыми эти объекты должны обладать;

– *используя* – перечислены инструменты (возможности программы), которые может использовать ученик при решении задачи.

Объекты для задач, связанных с геометрией на плоскости (планиметрией):

Точка – задается двумя координатами x_P и y_P.

Прямая – задается уравнением $Ax + By + C = 0$, конкретная прямая определяется тремя коэффициентами (A, B, C). Нормаль к данной прямой $a = (A, B)$.

Отрезок – задается координатами двух вершин.

Луч – задается координатами точки начала и координатами одной точки лежащей на луче.

Окружность – задается уравнением $(x - x_C)^2 + (y - y_C)^2 = R^2$, конкретная окружность определяется тремя коэффициентами $(x_C; y_C; R)$. Две окружности одинаковы при совпадении всех коэффициентов.

Круг – задается неравенством $(x - x_C)^2 + (y - y_C)^2 \le R^2$, определяется и сравнивается как окружность.

Треугольник – задается координатами 3 вершин.

Многоугольник – задается координатами n вершин.

Предикаты:

«Равенство» – для некоторых объектов (например, для точек) проверяется сравнением коэффициентов, для других же объектов (например, прямых), нужна специальная процедура сравнения.

«Перпендикулярность» – отношение проверяемое для прямых, лучей, отрезков как равенство скалярного произведения нормальных векторов нулю.

«Параллельность» – применяется к прямым, лучам отрезкам, проверяется как коллинеарность нормальных векторов.

«Пересечение» – два геометрических объекта имеют хотя бы одну общую точку.

«Касательная» – прямая является касательной к прямой, лучу, отрезку, если он в ней содержится. Прямая является касательной к кругу или окружности, если пересекает ее ровно в одной точке.

«Касание» – два геометрических объекта имеют одну общую точку, в которой у них совпадают касательные.

«Вписанная» (окружность) – окружность вписана в многоугольник, если она касается всех его сторон.

«Описанная» (окружность) – окружность описана вокруг многоугольника, если содержит все его вершины.

«Прямоугольный» (треугольник) – треугольника, у которого один из углов прямой.

Инструменты

Циркуль – можно использовать для построения окружностей, а также, чтобы отмерить заданное отрезком расстояние.

Линейка без делений – можно строить прямые, проходящие через 2 точки, лучи, а также отрезки от точки до точки.

Линейка с делениями – можно измерять расстояния и откладывать расстояния, заданные числами).

Угольник – можно построить перпендикуляр к прямой или отрезку в заданной точке.

Выделение из постановки задачи на естественном языке элементов предметно-ориентированного языка

Прежде всего, в предметно ориентированном языке будут использоваться понятия из глоссария, основным понятием языка является задача. Она агрегирует в себе остальные:

Условие для ученика: Дан треугольник, построить вписанную в него окружность, пользуясь только циркулем и линейкой.

Дано: точка A, точка B, точка C, отрезок AB, отрезок AC, отрезок BC.

Построить: окружность P.

Такие что: A «лежит на» P и B «лежит на» P и C «лежит на» P.

Инструменты: циркуль, линейка без делений.

Заметим, что «Дано» можно было бы записать как «Треугольник ABC», если такое понятие определено в системе.

Выбор базового синтаксиса предметно-ориентированного языка

Текст на ПОЯ может вводиться так, что этап лексического и синтаксического анализа не требуется вообще. Для этого при вводе текста система сразу строит синтаксическое дерево. При редактировании предметно-ориентированная программа выглядит как набор вложенных прямоугольных ячеек для редактирования. Ячейка, которая включает в себя все остальные, соответствует задаче. Внутрь ячейки может быть вложен список ячеек определенного типа и дерево ячеек (в дереве родительский узел контролирует возможные типы вложенных ячеек).

Внутрь ячейки «Задача» вложены следующие ячейки:

1. «Условие для ученика», в которую вложены две ячейки: с константной строкой «Условие для ученика», строкой для редактирования условия задачи.

2. «Дано» с вложенным списком ячеек с начальным построением. Эта ячейка заполняется при помощи графического редактора, в котором можно сделать начальное геометрическое построение, текст в ячейке появляется автоматически при построении чертежа. Прямо на чертеже можно давать объектам имена.

3. «Построить», которая позволяет добавлять объекты из глоссария (Точка, Прямая и т.д.) и при этом давать им имена, чтобы можно было ссылаться на них из условий.

4. «Такие что», в которой можно выбирать предикаты из глоссария и объекты (либо на чертеже, либо имя объекта из выпадающего списка), к которым надо применить данный предикат.

5. «Инструменты», в которые можно добавлять и удалять элементы из глоссария «инструменты».

Описание задачи хранится в древовидной структуре данных в памяти и преобразуется в XML-формат при сохранении на диск (один из стандартных подходов).

Названия и свойства конкретных понятий будут получаться из иерархии классов, задающих глоссарии предметной области и описанных на языке Java. Например, название «Прямоугольник» определяется аннотацией соответствующего Java-класса **`Rectangle`**. Для описания названия понятия из глоссария введем специальную аннотацию **`@GlossaryName`**. Для определения принадлежности к определенному типу понятий вводится несколько базовых абстрактных классов: **`GeomObject`** – геометрический объект на плоскости; **`Predicate`** – базовый класс для всех предикатов, каждый предикат должен сообщать, к каким классам объектов он применим для описания применимости того или иного предиката по отношению к определенному геометрическому объекту; **`Tool`** – базовый класс для всех инструментов.

При вводе предикатов проверяются типы передающихся в него объектов.

Создание транслятора, обрабатывающего предметно-ориентированный язык

На этом этапе создаются пустые объявления (так называемые «заглушки») для всех необходимых классов и методов. Затем происходит реализация общего механизма редактирования, показывающего окно для ввода задачи и позволяющего вводить на первом этапе строку с условием задачи. Затем реализуется механизм связи с динамическим чертежом, чтобы при выборе фигуры на чертеже ее имя добавлялось в текущую активную ячейку в условии задачи. Этот механизм используется при добавлении фигур в ячейку «дано» и при составлении предикатов.

Во всех ячейках, устроенных как вложенные клетки, показывается специальная пустая ячейка, в которую можно добавить дополнительные элементы.

Дальше для реализации применяется метод «разработка, управляемая тестами» рассмотренный в разделе методика.

Интерпретация предиката при проверке решения: при вводе решения ученик создаёт построение, состоящие из геометрических фигур. Перед началом построения задан начальный чертёж (из условия задачи). Если в условии задачи требуется построить несколько объектов, ученику

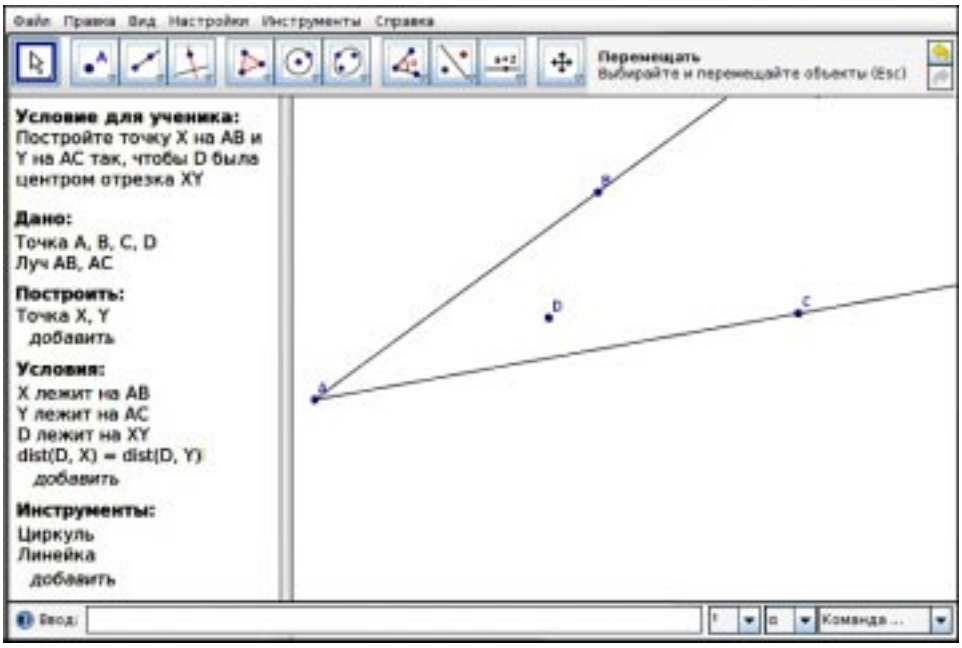

Рис. 1. Внешний вид интерфейса системы

предлагается выбрать названия для построенных им объектов из условия задачи (например, требуется построить две окружности p и q, ученик выбирает какая из окружностей p, а какая q).

После выбора всех обозначенных в условии задачи объектов для построения ученик нажимает кнопку «Проверить» и система проверяет выполнение предикатов условия задачи. Для проверки предикатов для самого внешнего уровня вызывается метод check, этот предикат вызывает по необходимости метод check для вложенных предикатов.

На рис. 1 изображен внешний вид интерфейса системы, используемого учителем для составления условий задач. Весь интерфейс является редактором предметно-ориентированного языка, он разделен на две части. Справа редактируется чертеж с помощью инструмента динамической геометрии, основанного на системе GeoGebra, слева вводится формальное условие задачи на описанном выше языке. Напрмер (рис. 1), по условию задачи требуется построить отрезок с концами на сторонах угла, так чтобы заданная точка делила его пополам. При вводе условия, как было предложено выше, преподавателю не требуется знать синтаксис языка, он может заполнять данные по ячейкам. Для добавления ячеек используются активные элементы. На рисунке они подписаны как «добавить» и способны добавлять предикаты, инструменты, объекты для построения.

ЗАКЛЮЧЕНИЕ

В статье приведен анализ преимуществ предметно-ориентированного подхода, представлена модель разработки с использованием ПОЯ, методика разработки ПОЯ и приведен пример применения описанной методики. Приведенный пример демонстрирует возможности применения предметно-ориентированных языков к созданию программ образовательного назначения. С использованием ПОЯ появляется возможность подключить к процессу программирования (составления задач) специалистов предметной области (учителей), которые не имеют опыта программирования.

Литература

1. *Кириллов Д.* Ориентация на язык // Компьютерра, 2006. № 10(630). С. 9.

2. *Дмитриев С.* Языково-ориентированное программирование: следующая парадигма // RSDN Magazine, 2005. № 5.

3. M. Ward. Language Oriented Programming // Software – Concepts and Tools, 1994. № 15. С. 147–161.

4. *Бек К.* Экстремальное программирование: разработка через тестирование. СПб.: Питер, 2003.

5. *Фаулер М.* Языковой инструментарий: новая жизнь языков предметной области // http://www.maxkir.com/sd/languageWorkbenches.html. [Электронный ресурс] / MAXKIR.com. Самое интересное о разработке программного обеспечения. Электрон. дан. 27.12.2005 Режим доступа: http://www.maxkir.com/sd/languageWorkbenches.html свободный. Загл. с экрана. Яз. рус. / (последнее обращение 06.10.2010).

6. *Кобёрн А.* Современные методы описания функциональных требований к системам, М.: Лори, 2002.

7. *Хант Э., Томас Д.* Программист-прагматик. Путь от подмастерья к мастеру, М.: Лори, 2007.

Abstract

Based on analysis of the history of programming languages papers [3,5] conclude that language-oriented approach may become the next step in the evolution of programming languages, and significantly improve productivity and quality of software development. The paper identifies common features of existing methods of language-oriented approach usage, and based on an analysis of their shortcomings presents a new method of application of the approach. An example of a system that uses a domain-specific language is presented. The language is used to enable teachers that are not skilled in programming to formalize complex statements of geometric problems.

Keywords: Domain Specific Languages, Distance Learning, computer aided education, Language Oriented Programming, Dynamic geometry.

Степулёнок Денис Олегович,
ассистент кафедры АСОИУ
СПбГЭТУ «ЛЭТИ»,
super.denis@gmail.com