УДК 004.434

ИНКРЕМЕНТАЛЬНЫЙ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫЙ ПРОЦЕСС РАЗРАБОТКИ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Н. Д. Андреев,
начальник отдела разработки программного обеспечения
ООО «Джи Джи Эй Софтвэр Сервисес», г. Санкт-Петербург
Ф. А. Новиков,
доктор физ.-мат. наук, доцент
Санкт-Петербургский государственный политехнический университет

Рассматриваются причины ограниченного применения проблемно- и предметно-ориентированных языков и предлагается инкрементальный и итеративный предметно-ориентированный процесс разработки прикладного программного обеспечения, призванный расширить сферу применения предметно-ориентированного подхода на практике. Раскрываются преимущества использования этого подхода для разработки заказного прикладного программного обеспечения, ориентированного на определенную предметную область.

Ключевые слова *— программное обеспечение, процесс разработки, предметно-ориентированные языки, аутсорсинг в информационных технологиях.*

Введение

В настоящее время эффективность и результативность используемых процессов для разработки прикладного программного обеспечения (ППО) оставляют желать лучшего. Несмотря на заметное развитие технологий программирования, принципы подходов и процессов разработки остаются в сущности все теми же. Вначале заказчики пытаются объяснить, а разработчики пытаются понять цели и задачи проекта в терминах соответствующей предметной области. Затем разработчики приступают к разработке программы с помощью языка программирования, в котором нет этих терминов предметной области, а есть термины систем программирования — переменные, процедуры, поля и их видимость. Такой «коммуникационный разрыв» негативно сказывается на эффективности разработки. Модные в последние годы гибкие процессы разработки [1] позволяют решить часть проблем, но не могут полностью гарантировать, что будет разработано именно то, что ожидает заказчик. Действительно, ведь само программирование в этом случае обычно осуществляется без какого-либо участия заказчика на чуждом ему языке. Даже если представитель заказчика сидит в одной комнате с разработчиками, он не принимает участие в написании кода и в большинстве случаев не может оценить и понять промежуточные артефакты разработки.

Подход, при котором разработка ППО ведется на предметно- и проблемно-ориентированных языках, призван изменить ситуацию в лучшую сторону, поскольку трудозатраты на разработку ППО существенно сокращаются за счет небольшого увеличения трудозатрат на разработку инструментального ПО [2]. Суть подхода — конструирование программ на предметном уровне. Под конструированием мы понимаем процесс создания визуальных и текстовых описаний программы в различных представлениях [3].

В настоящее время активно исследуются вопросы создания и использования проблемно- и предметно-ориентированных языков [4], однако в силу ряда причин на практике такой подход применяется достаточно ограниченно.

Применение проблемно- и предметно-ориентированных языков

При проблемно- или предметно-ориентированном подходе для определенных целей или предметной области обычно создается специа-

лизированный язык прикладного программирования.

Проблемно-ориентированные (PO — Problem Oriented) языки создаются для решения конкретной проблемы или ряда сходных проблем в различных предметных областях. Например, язык, определяющий сущности и связи между ними, может иметь семантику системы с базой данных (БД), набором форм пользовательского интерфейса и логикой работы с БД [5]. Предметно-ориентированные (DS — Domain Specific) языки имеют специфику конкретной предметной области — например, язык для создания информационных систем в области химии [6].

Преимущества разработки в терминах предметной области очевидны. Действительно, решать определенную проблему на специализированном языке проще и быстрее. При использовании специализированного высокоуровневого языка часто используется идея порождающего программирования [7–9], и это помогает избежать повторной разработки однотипного кода. А если создан специализированный предметно-ориентированный язык, то в нем используются термины предметной области, и значит можно с большим успехом привлекать к разработке «сильных пользователей» — специалистов в этой предметной области. При этом риск появления неправильно понятых требований к продукту будет минимальным. Также дизайнерские решения можно с легкостью модифицировать даже на последних стадиях разработки — ведь дублирование шаблонов и кода будет ниже, чем при традиционном подходе. Кроме того, при поддержке системы обновление технологий тоже будет проходить менее болезненно — необходимо лишь изменить существующий или реализовать новый генератор с языка высокого уровня.

Этот подход не является новым; такие проблемно-ориентированные языки, как SQL, TEX появились достаточно давно. Однако толчок к потенциально массовому использованию подходов дало появление методологии языково-ориентированного программирования [1] и инструментов, его поддерживающих [2]. Также свое место в подходах к разработке ПО и даже модели CMMI заняла методология разработки фабрик ПО [10]. Суть ее заключается в обобщении множества программных продуктов, выделении и разработке общих частей и построении, таким образом, линии по созданию серии программных продуктов. Часто способом обобщения служит предметно-ориентированный язык, а процесс его создания обычно характеризуется следующим.

1. Язык создается с нуля — не расширяет какие-то другие проблемно- или предметно-ориентированные языки.

2. Язык проектируется и реализуется так, чтобы его можно было сразу использовать для разработки различных продуктов линии.

3. Получаемый исходный код на языке более низкого уровня сложно поддерживать в дальнейшем — не предполагается, что можно частично или полностью продолжить разработку на языке более низкого уровня.

В целом подход оказывается не гибким [11], и для разработки *нескольких* продуктов процесс выглядит следующим образом (рис. 1). На первом шаге разрабатывается предметно-ориентированный язык *A*. Далее этот язык используется для разработки ряда продуктов, например #1, #2 и #3. Производятся изменения языка *A* по результатам его использования для разработки продуктов. Новая версия языка *A* используется для разработки новой версии продукта #2 и нового продукта #4.

Недостатки такого процесса состоят в следующем.

1. Высокая стоимость и длительность шага #1 — анализ требований, обобщение продуктов и разработка соответствующего языка потребуют существенных времени и ресурсов.

2. Первые результаты работы над продуктом #1 достигаются значительно позже, чем при использовании обычных методологий. На практике это вызывает недовольство и недоверие заказчика, а значит, требует дополнительных усилий от команды разработки. Например, приходится вести практически параллельную разработку для демонстрации прогресса заказчику. Заказчика редко удается убедить в необходимости соз-

■ ***Рис. 1.*** Традиционный предметно-ориентированный процесс разработки ППО

дания специализированного языка до создания этого языка.

3. Существуют риски, характерные для водопадных процессов разработки, например, неправильное понимание требований к языкам и продуктам линии. Выявление непонимания на поздних фазах разработки ведет к большим потерям.

Кроме того, иногда в силу различных причин поддерживать разработанный язык гораздо дороже, чем вручную изменять сгенерированный код на языке общего назначения. К сожалению, создатели генератора кода с предметно-ориентированного языка часто не предполагают, что получаемый код можно дорабатывать или поддерживать вручную. В таком случае мы становимся «заложниками» разработанного языка, и риски разработки в целом повышаются. Авторы неоднократно сталкивались на практике с подобными ситуациями, и обычно руководство тогда принимало решение о разработке на языке низкого уровня с «нуля» и без какого-либо использования предметно-ориентированного подхода.

Вышеперечисленные свойства и недостатки процесса, по нашему мнению, являются причинами ограниченного использования предметно-ориентированного подхода на практике.

Инкрементальный и итеративный предметно-ориентированные процессы разработки

На практике необходим более гибкий предметно-ориентированный процесс разработки, при котором мы предлагаем использовать следующие подходы.

1. Не ставить перед собой цель создать «предельно» общий язык, который сразу окажется применимым для большого количества различных продуктов предметной области.

2. Начинать с создания частных проблемно-ориентированных языков. При этом время, потраченное на создание подобных языков, не должно сильно замедлить создание продукта. Мы рекомендуем применять тут правило «трех ударов» [12] при нахождении однотипного программного кода.

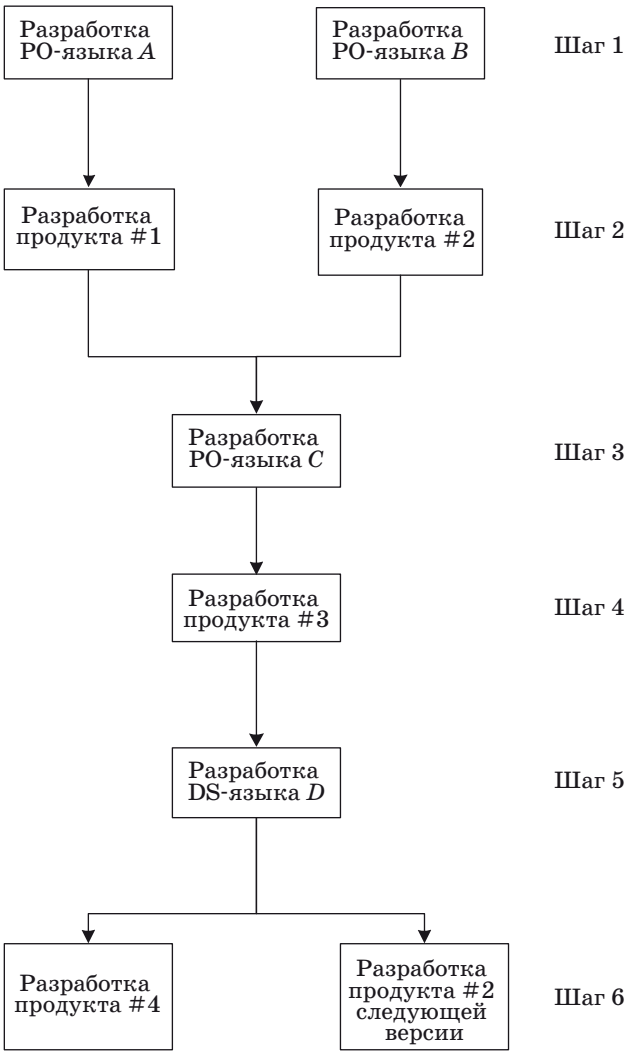
3. Повторно использовать существующие проблемно- и предметно-ориентированные языки — создавать новые языки на основе уже существующих.

4. Не исключать полностью дописывание кода на языке обычного на сегодняшний день уровня (например, Java или C#). Важно, чтобы код, генерируемый из проблемно- и предметно-ориентированных языков, был поддерживаемым.

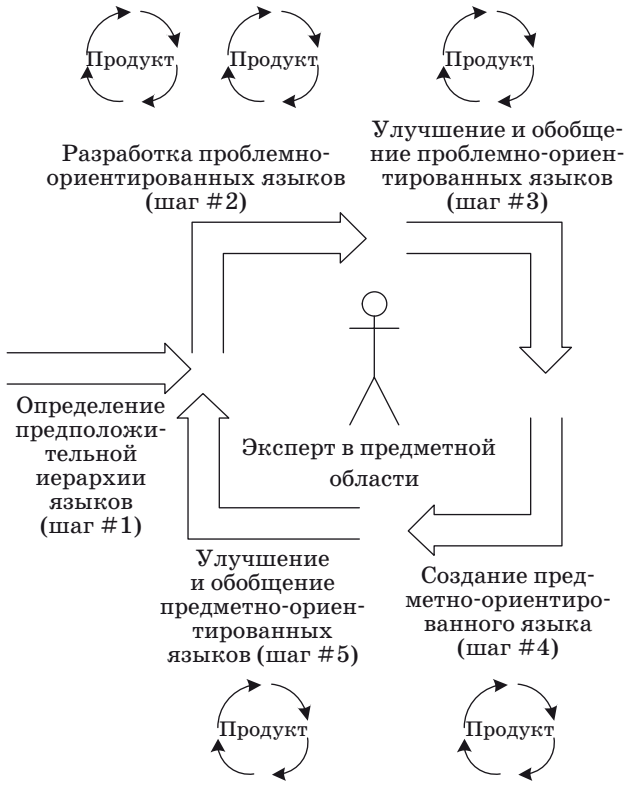
5. При наличии одновременно генерируемого и рукописного кода использовать модель частично генерируемого кода [13].

Предлагаемый процесс на основе этих принципов является инкрементальным (рис. 2) и итеративным (рис. 3).

На первом шаге разрабатываются проблемно-ориентированные языки *A* и *B*, которые используются на шаге #2 для создания продуктов. На шаге #3 создается новый проблемно-ориентированный язык *C*, основанный на созданных ранее языках. На шаге #4 разрабатывается новый продукт при помощи языка *C*. На шаге #5 создается предметно-ориентированный язык *D* на основе ранее созданных проблемно-ориентированных языков. На последнем шаге язык *D* используется для создания нового и доработки старого продуктов. Таким образом обеспечивается инкрементальность процесса.



■ ***Рис. 2.*** Пример инкрементальной предметно-ориентированной разработки ПО



■ ***Рис. 3.*** Пример итеративной и инкрементальной предметно-ориентированной разработки ПО

Ключевая часть процесса — создание эффективной иерархии обобщения проблемно- и предметно-ориентированных языков (рис. 4).

В целом предлагаемый процесс выглядит следующим образом (см. рис. 3).

1. Определение предположительной иерархии проблемно- и предметно-ориентированных языков.

2. Реализация одного или нескольких проблемно-ориентированных языков в рамках бюджета одного из продуктов.

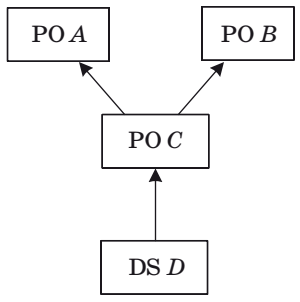
3. Улучшение и обобщение созданных проблемно-ориентированных языков в рамках одного или нескольких продуктов.

4. Создание предметно-ориентированного языка на основе существующих проблемно-ориентированных языков и его использование для создания одного из продуктов.

5. Улучшение и обобщение созданных предметно-ориентированных языков в рамках создания одного или нескольких продуктов.

6. Переход к шагу #2.

Важно отметить, что разработанные проблемно-ориентированные языки могут быть повторно использованы для создания других предметно-ориентированных языков, в том числе в других предметных областях. Итеративность процесса состоит в продолжающемся процессе создания новых проблемно- и предметно-ориентированных языков и их обобщении.



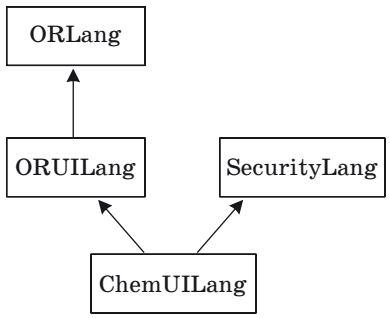
■ ***Рис. 4.*** Иерархия наследования проблемно- и предметно-ориентированных языков

Пример иерархии проблемно- и предметно-ориентированных языков

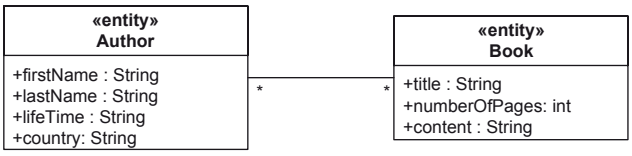
Иерархия разработанных проблемно- и предметно-ориентированных языков представлена на рис. 5.

Проблемно-ориентированный язык ORLang определяет сущности, их связи и логику сохранения/получения в БД. В результате работы транслятора с этого языка сгенерируются скрипты создания схемы БД (например, Oracle или MS SQL Server), объектная модель и код DAL (Data Access Layer) для работы с этой БД (например, на Java или C#). Этот язык — графический и является подмножеством нотации диаграмм классов UML (рис. 6).

Проблемно-ориентированный язык ORUILang расширяет синтаксис и семантику языка ORLang. Дополнительная семантика — пользовательский интерфейс для работы с БД. На уровне



■ ***Рис. 5.*** Иерархия разработанных проблемно- и предметно-ориентированных языков



■ ***Рис. 6.*** Пример описания связи двух сущностей на языке ORLang

синтаксиса дополнительно настраиваются свойства получаемого пользовательского интерфейса (рис. 7). На стороне пользовательского интерфейса возможно несколько видов форм — «Search» (Поиск), «Search Results» (Результаты поиска) и «Input Data» (Ввод данных). На расширенной диаграмме сущностей языка ORLang определяются свойства форм с помощью следующих стереотипов:

«search» — поле может быть использовано как критерий поиска и выводится в таблицу результатов поиска;

«search result» — поле выводится в таблицу результатов поиска;

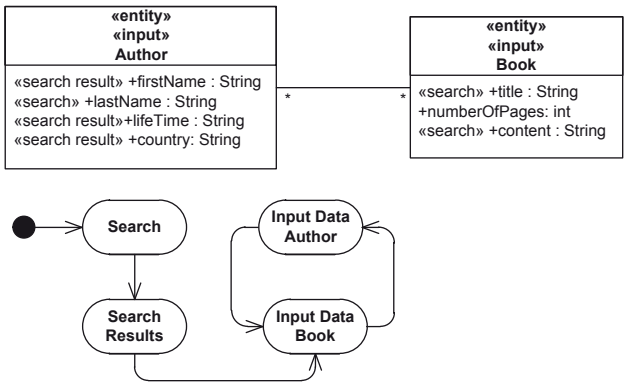
«input» — сущность может быть отредактирована или введена новая.

На дополнительной диаграмме состояний задается взаимодействие этих форм.

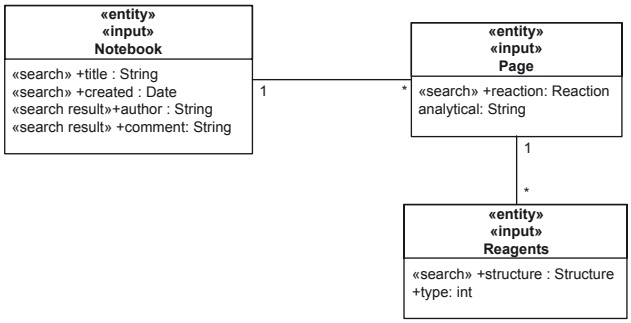
Проблемно-ориентированный язык SecurityLang определяет, как приложение работает с точки зрения безопасности: задаются параметры аутентификации, роли — и какие части приложения (пакеты) доступны для каких ролей. Синтаксис языка основан на XML:

```
<security>
  <authentication>
    <ldap url="ldap://ldap.xyz.com"
      username="ldap" password="ldappas">
  </authentication>
  <authorization>
    <role name="readonly">
      <package>com.xyz.view</package>
    </role>
    <role name="write">
      <package>com.xyz.view</package>
      <package>com.xyz.create</package>
    </role>
  </authorization>
</security>
```

Предметно-ориентированный язык для создания информационных систем в области химии ChemUILang расширяет два проблемно-ориентированных языка — ORUILang и SecurityLang.



■ *Рис. 7.* Пример описания взаимодействия форм на языке ORUILang



■ *Рис. 8.* Пример описания свойств пользовательского интерфейса на языке ChemUILang

В этом языке можно дополнительно определять свойства специальных типов (рис. 8) — химическая структура (Structure) и химическая реакция (Reaction). Семантика таких типов — отображение соответствующих свойств на пользовательском интерфейсе в виде химических структур или реакций. Кроме того, в системе автоматически поддерживается подструктурный поиск по соответствующим полям в БД.

Применение предметно-ориентированного процесса при аутсорсинге

Теоретики создания линий разработки ПО обычно приводят в качестве примера компанию, работающую в определенной предметной области, которой нужно разрабатывать ПО в этой области: например, автоконцерну необходимо создавать ПО для различных моделей автомобилей [14]. Мы считаем, что при некоторых условиях предметно-ориентированный процесс может не менее эффективно применяться и при аутсорсинге — разработке ПО на заказ.

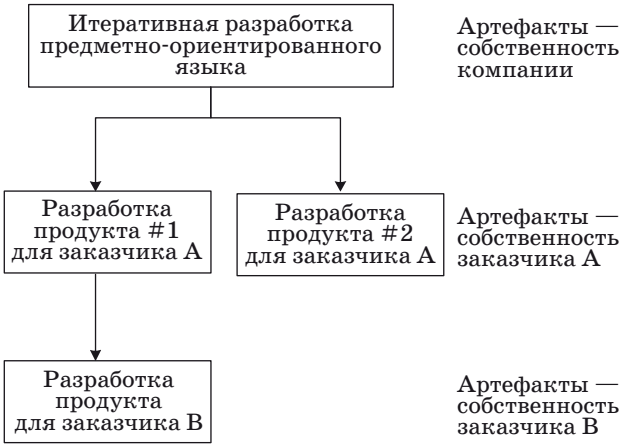
Несмотря на бурное развитие в последние годы, аутсорсинг разработки ПО в России развит не очень сильно по сравнению с аналогичным рынком, например, в Индии. Основные проблемы — «нефтяная» экономика, завышенные зарплатные ожидания и гораздо меньшее количество профессиональных разработчиков ПО. Следовательно, для того чтобы победить в конкурентной борьбе, разработку надо вести более эффективно.

Прежде всего, компании необходимо сфокусироваться на одной из предметных областей и достичь в этой области признанной экспертизы. Даже для больших аутсорсинговых компаний сложно иметь экспертизу и наработки в различных предметных областях. При наличии же только одной предметной области может эффективно применяться предложенный итеративный предметно-ориентированный подход разработки.

При использовании предметно-ориентированного подхода разработка становится более интеллектуальной. Это позволяет не только повысить эффективность и превзойти конкурентов, программирующих в стиле copy-paste (дублирование кода и подходов), но и привлекать и удерживать талантливых молодых разработчиков.

Когда у компании несколько заказчиков в одной предметной области, то встает проблема повторного использования кода и знаний. Во-первых, люди обычно продолжают разрабатывать или поддерживать существующую систему, и переключить их на другого заказчика крайне сложно. Во-вторых, обычно заказчики очень строго относятся к разделению ресурсов и почти наверняка не позволят использовать разработанный код для других компаний. Если необходимо создать систему для одного заказчика, очень похожую на систему, разработанную для другого заказчика, то проблематично повторно использовать на уровне компании даже знания, не говоря о программном коде.

При использовании предложенного предметно-ориентированного процесса знания могут постепенно накапливаться в созданных специализированных языках. Если в результате разработки системы для одного заказчика будет создан предметно-ориентированный язык, то разработка аналогичной системы для другого заказчика будет существенно дешевле (возможно, даже дешевле, чем разработка в Индии или Китае). При этом совсем не обязательно говорить заказчику о том, что был создан какой-то предметно-ориентированный язык. Если, как писалось выше, результирующий код будет поддерживаемым и не будет восприниматься как генерируемый, то условия стандартного контракта на разработку ПО будут выполнены — весь результирующий (сгенерированный) программный код будет принадлежать заказчику. При этом код генератора и созданные языки будут принадлежать компании-разработчику.



■ ***Рис. 9.*** Процесс разработки для нескольких заказчиков

Применение предложенного процесса разработки для нескольких заказчиков показано на рис. 9. Ключевая особенность такого процесса — использование разработанного предметно-ориентированного языка при создании различных программных продуктов для нескольких заказчиков.

Заключение

В работе предложен итеративный процесс разработки ППО в форме линейки продуктов на основе иерархии предметно- и проблемно-ориентированных языков в конкретных предметных областях, который позволяет использовать предметно-ориентированный подход на практике. Кроме того, рассмотрены вопросы использования предметно-ориентированного процесса при аутсорсинге.

Дальнейшее развитие работы мы видим в разработке подходов повторного использования и построения эффективной иерархии проблемно- и предметно-ориентированных языков.

Литература

1. **Бек К.** Экстремальное программирование. — СПб.: Питер, 2003. — 224 с.
2. **Ward M.** Language Oriented Programming // Software — Concepts and Tools. 1994. Vol. 15. N 4. P. 147–161.
3. **Новиков Ф. А.** Визуальное конструирование программ // Информационно-управляющие системы. 2005. № 6. С. 9–22.
4. **Dmitriev S.** Language Oriented Programming: The Next Programming Paradigm. http://www.onboard.jetbrains.com/articles/04/10/lop/mps.pdf (дата обращения: 04.12.2011).
5. **Андреев Н. Д.** Предметно-ориентированный язык моделирования, основанный на UML // Формирование технической политики инновационных наукоемких технологий: материалы конф. и школы-семинара СПбГПУ. СПб., 2004. С. 75–82.
6. **Cheng Y., Zhang R., Hu C., Li S., Zhang M.** Design and implementation of UDLC: Unified Job-Description Language on Chemical-Grid // ICECE-2011. P. 2967–

2971. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6057124 (дата обращения: 04.12.2011).
7. **Czarnecki K., Ulrich W.** Generative Programming — Methods, Tools, and Applications. — Addison-Wesley, 2000. — 864 p.
8. **Новиков Ф. А., Степанян К. Б.** Использование порождающего программирования при реализации языка описания диаграмм // Информационно-управляющие системы. 2008. № 6. С. 32–35.
9. **Новиков Ф. А., Новосельцев В. Б.** Язык исполняемых программных спецификаций // Программирование. 2010. № 1. С. 66–78.
10. **Jones L. G., Soule A. L.** Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice. http://www.sei.cmu.edu/library/abstracts/reports/02tn012.cfm (дата обращения: 04.12.2011).
11. **Rumpe B., Schindler M., Völkel S., Weisemöller I.** Agile Development with Domain Specific Languages // Proc. of the 7th European Conf. on Modelling Foundations and Applications (ECMFA 2011). Springer-Verlag, June 2011. P. 387–388.
12. **Fowler M.** et al. Refactoring: Improving the Design of Existing Code. — Addison-Wesley Professional, 1999. — 464 p.
13. **Herrington J.** Code Generation in Action. — Manning, 2003. — 350 p.
14. **Grandy H., Benz S.** Specification based testing of automotive human machine interfaces // GI Jahrestagung. 2009. Vol. 154GI. P. 2720–2727.

Международная выставка «Охрана, безопасность, противопожарная защита — SIPS 2012»
16–18 октября 2012

Место проведения: г. Краснодар, ВЦ «КраснодарЭКСПО»

Организатор
Выставочная компания ITE Moscow

При поддержке
МВД России
Министерства транспорта РФ
Федеральной таможенной службы
МЧС Краснодарского края
Администрации Краснодарского края
ВДПО

Основные разделы выставки
Технические средства обеспечения безопасности
Системы охранного телевидения и наблюдения
Защита информации, смарт-карты
Современные системы связи, оповещения и телекоммуникаций
Пожарная безопасность
Таможенный контроль
Пограничный контроль

Выставка SIPS 2012 по традиции будет проходить в рамках проекта IDES, объединившего на одной выставочной площадке 7 специализированных выставок. По признанию специалистов IDES, это уникальный отраслевой коммуникационный проект, сумевший охватить основные направления экономики края и объединить вокруг себя профессионалов.

Выставка представляет полный спектр продукции и услуг для комплексного обеспечения безопасности во всех сферах современной жизни.

Важным стимулом для участия и посещения выставки является насыщенная деловая программа, представленная конференциями, семинарами, круглыми столами. Участники — специалисты строительных компаний, проектно-монтажных организаций, торговли, органов управления, банков, армии, НИИ, представители отраслей промышленности.

Участникам выставки SIPS 2012 предоставится уникальная возможность встретиться на одной площадке с представителями отраслевых министерств и департаментов Южного федерального округа, специалистами ведущих нефтегазовых и энергетических компаний края.

Формирование выставочной экспозиции и деловой программы продолжается.

Дополнительная информация и справки
Тел.: +7 (495) 935-73-50, доб. 4242
Эл. адрес: ides@ite-expo.ru
Сайт: http://www.ides-expo.ru/home/exhibitions/sips.aspx