инженерия программного обеспечения

Сафонов Владимир Олегович

УДК 004.4'2

МНОГОЯЗЫКОВОЕ АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СИСТЕМЕ ASPECT.NET

Аннотация

В статье рассмотрены возможности, реализованные в новой версии системы Aspect.NET, – многоязыковое аспектно-ориентированное программирование (АОП), то есть возможность реализации целевых программ и аспектов на различных языках. Анализируются роль и принципы реализации многоязыкового АОП, рассматриваются другие подходы к данной проблеме, приводятся примеры использования многоязыкового АОП в системе Aspect.NET, намечаются дальнейшие перспективы развития.

Ключевые слова: аспектно-ориентированное программирование (АОП), Aspect.NET, многоязыковое программирование, C#, Visual Basic.

ВВЕДЕНИЕ

Аспектно-ориентированное программирование (АОП) [1–4] – перспективный подход к разработке программного обеспечения, основанный на модуляризации сквозной функциональности (cross-cutting concerns) в виде особого рода модулей – *аспектов*, содержащих код сквозной функциональности и применяемых к целевым программам путем *внедрения (weaving)* – автоматизированной вставки кода *действий аспекта*, в соответствии с *правилами внедрения аспекта*, специфицированными в его определении. Принципы, преимущества, инструменты и области применения АОП подробно рассмотрены в работах [1–5].

Нашей группой сотрудников, аспирантов и студентов СПбГУ под моим руководством разработана система аспектно-ориентированного программирования Aspect.NET [1–7], используемая в настоящее время для исследовательских проектов и обучения программированию в университетах в 26 странах мира.

Автор благодарен всем участникам проекта – реализаторам системы Aspect.NET и разработчикам библиотек аспектов в системе Aspect.NET для различных предметных областей – Дмитрию Григорьеву, Михаилу Грачеву, Александру Масленникову, Руслану Муханову, Анне Когай, Доану Нгуен Вану, Михаилу Корчуну, Игорю Евдакову, Айбеку Саримбекову.

В частности, благодаря дипломным работам Игоря и Айбека, была разработана новая версия Aspect.NET 2.2 с поддержкой второго языка – Visual Basic [7], которой и посвящена данная статья.

1. РОЛЬ И ПРИНЦИПЫ РЕАЛИЗАЦИИ МНОГОЯЗЫКОВОГО АОП

В силу ряда исторических причин, большинство инструментов аспектно-ори-

ентированного программирования, начиная с наиболее известной, ставшей классической, системы AspectJ [8] реализованы как расширения какого-либо определенного языка программирования (как правило – языка Java) концепциями и конструкциями для поддержки АОП.

Преимущество подобного подхода в том, что он дает возможность пользователям привычного, постоянно используемого языка, оставаясь в рамках его хорошо освоенных понятий и обозначений, экспериментировать в изучении и использовании новой технологии АОП.

Недостаток «одноязыкового» подхода к АОП – в его ограниченности: средства поддержки АОП доступны лишь как «продолжение» конкретного языка и основанной на нем системы программирования (специфических, реализованных специально для данного АОП-расширения компилятора, редактора, отладчика, профайлера и т. д.), зависят от них и не могут быть использованы при программировании на другом языке, что неудобно, так как для многих больших программных систем принято разрабатывать модули, реализующие различного рода функциональности на различных наиболее подходящих для этого языках, например – на языках C# и Visual Basic (VB) для платформы .NET.

С методологической точки зрения, одноязыковый подход к АОП приводит также к «концептуальной путанице» (conceptual confusion), так как в одном и том же расширенном языке оказываются доступными и комбинируются понятия совершенно различной природы и разного уровня общности – например, *классы* (как основное средство ООП) и *аспекты* (как новая разновидность модулей для описания сквозной функциональности). Это подчас вызывает серьезные дополнительные проблемы в понимании и использовании АОП даже опытными программистами и существенно затрудняет его изучение и распространение.

По нашему мнению [1, 2], аспектно-ориентированное программирование – подход к разработке, модификации и модернизации программ, идеи и концепции которого являются столь общими, что не зависят (точнее, не должны зависеть) от конкретного языка программирования. АОП – методология, формулирующая общие принципы и правила автоматизированных преобразований программ, основные концепции которой – сквозная функциональность, аспект, внедрение аспекта, точка присоединения и др. – могут быть спроецированы на самые различные языки, соответствующие различным парадигмам программирования – языки ООП, процедурного программирования, функционального программирования, продукционного и даже логического программирования, поскольку для программ на каждом из этих языков имеет смысл рассматривать их преобразования, правила которых привязаны к конкретным разновидностям *модулей* (классам, процедурам, абстрактным типам данных, продукциям и др.), используемым в конкретном языке. Типичный пример правила внедрения аспекта в АОП – вставить вызов метода M перед всеми вызовами другого метода P (для языка ООП). Однако аналогичные трансформации программ могут быть выполнены и для языка процедурного программирования (вставка вызова процедуры M перед каждым вызовом процедуры P) или продукционного языка (применение продукции M непосредственно перед каждым применением продукции P).

Разумеется, возможность подобного обобщения механизмов АОП следует рассматривать в настоящий момент как рабочую гипотезу, требующую практического подтверждения (proof-of-concept) путем реализации инструментов АОП для различных языков с различными парадигмами программирования.

Первым шагом к такому подтверждению нашей концепции может стать реализация *многоязыкового аспектно-ориентированного программирования* – возможность реализации аспектов и целевых приложений на различных языках в рамках

современной платформы разработки программ. Такой платформой в настоящее время является .NET.

Платформа .NET обеспечивает общие механизмы многоязыкового программирования – *общую систему типов (CTS)*, *общую среду поддержки выполнения программ (CLR), единый промежуточный язык (CIL),* в который компилируется исходный код любого языка (например C# или VB), реализованного для платформы .NET. Цель этих механизмов – обеспечить возможность разработки многоязыковых приложений, в которых каждый модуль написан на том языке, на котором его реализация оказалась наиболее удобной (либо уже имелась в распоряжении разработчиков до начала проекта).

Вполне естественно, что при разработке инструментов АОП для платформы .NET возникает идея распространить тот же принцип многоязыкового программирования и на разработку и применение аспектов.

Рассмотрим принципы нашего подхода к многоязыковому АОП, реализованные в системе Aspect.NET. Другие подходы проанализированы в разделе 2.

1. Использование простого по структуре и семантике метаязыка спецификации аспектов, общего для всех языков их реализации (C##, VB и др.). При нашем подходе, аспект состоит из *заголовка, данных, модулей* и *действий*. Изложенная структура описания аспекта задается с помощью *АОП-аннотаций* на метаязыке спецификации аспектов *Aspect.NET.ML*. *Заголовок (%aspect*) содержит имя аспекта; *данные (%data)* – локальные (приватные) описания, используемые аспектом; модули *(%modules*) – процедуры (функции, методы) аспекта; *действия* (*%action*) – часть его модулей, вызовы которых вставляются в целевое приложение в соответствии с *правилами (%rules) внедрения аспекта (%before, %after, %instead* – перед конструкцией целевого приложения, например перед вызовом метода, после или вместо него), привязанными к каждому его действию. Данные концепции являются общими и не зависят от языка реализации аспектов. Они применимы к аспектам, написанным на любом входном языке платформы .NET, реализация которого отображает типы языка в общую систему типов (CTS) и удовлетворяет стандарту *общеязыковой спецификации (CLS)*. Соблюдение последней обеспечивает для платформы .NET совместимость в едином приложении модулей, код которых реализован на разных языках, например на языке C# или VB. Более того, предложенные нами концепции применимы и к реализации аспекта, например, на процедурном языке, хотя, на данный момент, мы пока не имеем практического опыта реализации АОП для процедурных языков – это планируется в дальнейшем. Код реализации аспекта на любом выбранном языке, таким образом, «размечается» АОП-аннотациями на метаязыке Aspect.NET.ML, причем таким образом, что если удалить или закомментировать АОП-аннотации, начинающиеся с символа % («забыть об аспектах»), то получится синтаксически правильный код единицы компиляции (класса) на языке реализации аспекта. Примеры приведены в работах [1–5] и в разделе 3 данной статьи. Отметим, что такой подход принципиально отличается, например, от подхода AspectJ, при котором код классов и аспектов перемешан между собой, и «отключение АОП» путем удаления фрагментов кода неизбежно приведет к ошибкам. Метаязык поддержан и средой разработки Aspect.NET Framework, встроенной в Visual Studio.NET как расширение (add-in), в которой пользователю предоставляются соответствующие новые типы проектов Visual Studio для АОП (например *Aspect.NET.ML module* на языке C# или VB). При создании АОП-проектов пользователь получает в текстовом редакторе интегрированной среды готовый шаблон (pattern) кода, содержащего АОП-аннотации частей аспекта на языке Aspect.NET.ML, структура которых не зависит от того, какой язык (C# или VB) выбран для реализации аспекта. После-

днее существенно облегчает для пользователя освоение и применение АОП – опытному программисту достаточно 3–5 минут и небольшого примера, чтобы начать разрабатывать аспекты на любом знакомом ему языке. Для сравнения, если бы был применен подход, аналогичный подходу AspectJ, то пользователю пришлось бы отдельно изучать методы и инструменты реализации аспектов на C#, затем – на VB, а также методы их интеграции друг с другом, если подобная интеграция вообще возможна.

2. Применение пользовательских атрибутов (custom attributes) для реализации АОП-аннотаций путем конвертирования в атрибуты. Пользовательские атрибуты – еще один многоязыковый механизм .NET, используемый в нашей системе. АОП-аннотации метаязыка Aspect.NET.ML на первом шаге сборки проекта из исходных кодов конвертируются в определения пользовательских атрибутов на языке реализации аспекта. Таким образом, код определения аспекта, содержащий АОП-аннотации, преобразуется специализированным конвертором в обычный исходный код на языке реализации, в котором действия аспектов (методы) помечены соответствующими АОП-атрибутами, содержащими правила внедрения этих действий в целевые приложения. Архитектура Aspect.NET представлена на рис. 1. Конвертор выделен точками.

В дальнейшем исходные коды реализаций аспектов обрабатываются штатными компиляторами .NET, компилирующими их в обычную бинарную сборку (assembly). При этом АОП-атрибуты хранятся в сборке вместе с бинарным кодом действий аспекта. Заметим, что бинарное представление аспектной сборки в виде кода на общем промежуточном языке CIL также не зависит от исходного языка реализации аспекта, благодаря многоязыковой природе .NET. Следовательно, для того чтобы реализовать поддержку какого-либо нового языка реализации аспектов в системе, достаточно, во-первых, реализовать конвертор метаязыка Aspect.NET.ML в определения АОП-атрибутов на новом языке их реализации, во-вторых, добавить поддержку новых типов проектов для определений аспектов на новом языке в среду Aspect.NET Framework. Первоначально, вплоть до 2010 г., все версии Aspect.NET (1.0, 1.1, 2.0, 2.1) содержали только реализацию поддержки языка C# (автор – А. Масленников). В 2010 г. студенты-дипломники И. Евдаков и А. Саримбеков реализовали поддержку языка VB для новой версии Aspect.NET – 2.2 [7], выпущенной в ноябре 2010 г. В ближайшее время планируется также реализация в системе Aspect.NET поддержки языка Managed C++, а также других языков, реализованных для .NET, включая, например, новый язык функционального программирования F#. Интересно отметить, что *компоновщик (weaver)* – компонента Aspect.NET, выполняющая внедрение кода действий аспекта в целевое приложение, вообще не потребовала никаких изменений в реализации для введения в систему нового языка, так как компоновщик работает в терминах CIL и выполняет внедрение аспектов на бинарном уровне.

2. ДРУГИЕ ПОДХОДЫ К МНОГОЯЗЫКОВОМУ АОП

По-видимому, в связи с большим влиянием системы AspectJ на исследователей и разработчиков в области АОП, вопросы многоязыкового АОП до сих пор недостаточно изучены и отражены в научной литературе – если ограничиться разработками на платформе Java, вопроса о многоязыковом АОП просто не возникает.

Однако для платформы .NET, разумеется, ввиду многоязыковой природы этой платформы, многоязыковое (cross-language) АОП рассматривалось и было реализовано в некоторых экспериментальных системах. Например в системе SourceWeave.NET [9] обеспечивается многоязыковое внедрение аспектов на уровне исходных кодов, что позволяет отлаживать результирующие приложения в терминах исходного кода. Однако, с нашей

точки зрения, данный подход не вполне концептуально чист, так как приводит к смешению исходных кодов на различных языках программирования в рамках одного модуля (класса или метода), что, на наш взгляд, отнюдь не является целью многоязыкового программирования и приводит к целому ряду дополнительных проблем совместимости и идентичности конструкций разных языков, резко усложняющему понимание процесса и результатов многоязыкового АОП. Также возможны проблемы и с эффективностью результирующих программ. Наш подход в рамках платформы .NET нам представляется более последовательным и более эффективным – внедрение аспектов происходит именно на том уровне, который является общеязыковым для платформы .NET, – на уровне единого промежуточного кода CIL и метаданных (атрибутов), что обеспечивает максимальную эффективность результирующих приложений, т.е. эффективность программы после внедрения аспектов не ухудшается, по сравнению с эффективностью той же программы после аналогичных «ручных» вставок кода.

В работе [10] анализируется многоязыковое АОП с других позиций – объединения различных инструментов АОП, поддерживающих различные языки реализации аспектов, в один инструмент многоязыкового АОП. Такая постановка задачи вынужденна, ввиду широкого использования уже существующих инструментов АОП со своими механизмами реализации аспектов, но решение подобной задачи гораздо сложнее, так как требует разработки дополнительных конверторов из одного представления аспектов в другое, либо разработки «оберток» (wrappers), позволяющих представить аспект, используемый в одном из инструментов в форме, «понятной» другому, что создает дополнительную сложность и громоздкость для использования итогового многоязыкового инструмента АОП.

Для .NET разработан также многоязыковый компоновщик (weaver) аспектов – AspectDNG [11]. Он, как и компоновщик аспектов в нашей системе Aspect.NET, выполняет внедрение аспектов на уровне единого промежуточного кода (CIL), а следовательно, благодаря многоязыковой природе .NET, «по определению» является, как и наш компоновщик аспектов, многоязыковым – аспекты могут быть разработаны на любом языке. Однако в системе AspectDNG отсутствует интеграция со средой разработки, а также какой-либо аналог единого метаязыка для спецификации аспектов – аспекты могут быть разработаны в виде обычных классов на любом языке с соблюдением определенных соглашений относительно именований, связей и сигнатуры классов и методов.

Таким образом, на данный момент система Aspect.NET является, по-видимому, первой системой многоязыкового аспектно-ориентированного программирования, обеспечивающей возможность разработки аспектов и целевых приложений на нескольких языках, реализованных в .NET, и интегрированной со средой разработки Visual Studio.

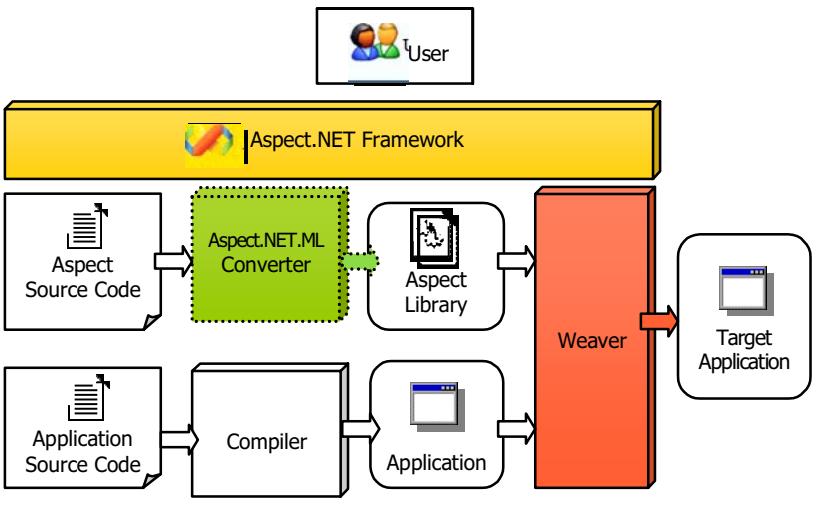

Рис. 1. Архитектура Aspect.NET

3. ВОЗМОЖНОСТИ И ПРИМЕРЫ МНОГОЯЗЫКОВОГО АОП В СИСТЕМЕ ASPECT.NET

Для иллюстрации описанных идей и принципов рассмотрим примеры разработки целевых приложений и аспектов на различных языках программирования в Aspect.NET 2.2 [7].

1) Разработка целевого приложения и аспекта на языке C# (пример Aspect2 [7]).

Рассмотрим простое консольное приложение на языке C# (см. листинг 1).

Вывод данного приложения:

```
Main
P
```

Теперь разработаем аспект протоколирования, вставляющий в целевое приложение дополнительный трассировочный вывод – перед вызовом каждого метода *M* – вывод *Hello M,* а после вызова – вывод *Bye M* (содержащий имя метода) (см. листинг 2).

Для разработки данного аспекта необходимо в среде Visual Studio выбрать тип проекта *C# / Aspect.NET ML Module*.

АОП-аннотации на метаязыке Aspect.NET.ML – заголовок аспекта, модули, правила и действия аспекта – выделены жирным шрифтом. Смысл их интуитивно совершенно очевиден благодаря простоте и наглядности метаязыка Aspect.NET.ML.

Листинг 1

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void P()
        { Console.WriteLine("P"); }

        static void Main(string[] args)
        {
            Console.WriteLine("Main");
            P();
        }
    }
}
```

Листинг 2

```
%aspect Aspect1
using System;
{
   %modules
      public static void Say (String s)
      { Console.WriteLine(s); }
   %rules
      %before %call *
      %action
        public static void SayHello ()
        { Say ("Hello " + %TargetMemberInfo.Name); }
      %after %call *
      %action
        public static void SayBye ()
        { Say ("Bye " + %TargetMemberInfo.Name); }
}
```

Аспект вставляет в целевое приложение перед вызовом любого метода (*) вызов действия *SayHello(),* выполняющего требуемый трассировочный вывод, после вызова любого метода – вызов аналогичного действия *SayBye()*. Конструкция языка Aspect.NET.ML *%TargetMemberInfo.Name* выполняет доступ к контексту точки присоединения – целевому методу – и вычисляет его имя.

Особенности практического применения системы Aspect.NET подробно описаны в работах [3, 4] , поэтому в данной статье мы их опускаем.

Вывод модифицированного целевого приложения (после применения аспекта):

```
Hello WriteLine
Main
Bye WriteLine
Hello P
Hello WriteLine
P
Bye WriteLine
Bye P
```

Данный работающий пример доступен в поставке Aspect.NET 2.2 в поддиректории *samples / Aspect2*.

2) Разработка аналогичных целевого приложения и аспекта на языке VB (пример Aspect2VB [7]).

Рассмотрим теперь аналогичный пример на языке VB, доступный в поставке Aspect.NET 2.2 в поддиректории *samples / Aspect2VB*. Функциональность целевого приложения и аспекта идентичны предыдущему примеру.

Код целевого приложения на языке VB:

```
Module Module1
    Sub P()
        Console.WriteLine("P")
    End Sub
    Sub Main()
        Console.WriteLine("Main")
        P()
    End Sub
End Module
```

Определение аспекта на языках Aspect.NET.ML и VB (см. листинг 3).

Отметим, что в определении аспекта используются в точности те же конструкции метаязыка Aspect.NET.ML, хотя, как видно из приведенного листинга кода, реализация аспекта выполнена на другом языке – Visual Basic.

3) Разработка целевого приложения на языке C#, а аспекта – на языке VB (пример TestArgsVB [7]).

Данный работающий пример доступен в поставке Aspect.NET 2.2 в поддиректории *samples / TestArgsVB*.

Листинг 3

```
%aspect Aspect1
Imports System
class Aspect1
   %modules
   %rules
      %before %call *
      %action
        public shared Sub SayHello()
           Console.WriteLine("Hello " + %TargetMemberInfo.Name)
        End Sub
      %after %call *
      %action
        public shared Sub SayBye()
            Console.WriteLine("Bye " + %TargetMemberInfo.Name)
        End Sub
End class
```

Код целевого приложения на языке C# (см. листинг 4).

Вывод данной программы:

```
Hello Method1
Hello Method2
Hello Method1
Hello Method2
Press <Enter> to exit
```

Код реализации на языке Visual Basic аспекта, выполняющего дополнительный трассировочный вывод значений аргументов методов целевого приложения (см. листинг 5).

Вывод модифицированного приложения (после применения аспекта):

```
Hello Method1
ActionMethod3: 1, 0.6
Hello Method2
ActionMethod1: 0.6
ActionMethod2: 1
Hello Method 1
ActionMethod1: 0.6
ActionMethod2: 1
Hello Method2
Press <Enter> to exit
```

Детали конструкций метаязыка АОП, выполняющих «захват» аргументов вызовов методов и выдающих информацию о них, описаны в [1].

В примере аспекта на языкеVisual Basic АОП-аннотации на языке Aspect.NET.ML выделены жирным шрифтом.

ЗАКЛЮЧЕНИЕ

В данной статье описаны принципы и первый опыт разработки многоязыковой версии системы Aspect.NET 2.2 с поддержкой языков C# и VB.

Версия системы Aspect.NET 2.2 свободно доступна для скачивания на академическом сайте Microsoft [7].

Как уже отмечалось, в дальнейшем планируется расширение числа поддерживаемых языков, выпуск новой версии Aspect.NET, совместимой с новейшей интегрированной средой Visual Studio.NET 2010, расширение функциональности си-

Листинг 4

```
using System;
namespace TestArgs
{
    class MainClass {
        static public void Run()
        {
            Program.Method1(1, 0.6);
            Program.Method2(1, 0.6);
        }
    }
    class Program {
        static void Main(string[] args)
        {
            Method1(1, 0.6);
            Method2(1, 0.6);
            MainClass.Run();
            Console.WriteLine("Press <Enter> to exit");
            Console.ReadLine();
        }
        public static void Method1(int i, double f)
          { Console.WriteLine("Hello Method1"); }
        public static int Method2(int i, double f)
          { Console.WriteLine("Hello Method2"); return 0; }
    }
}
```

Листинг 5

```
%aspect TestArgsAspect
Imports System
class TestArgsAspect
   %modules
   %rules
    %before %call *Method(int, double) && %args(arg[1]) &&
              %within(*MainClass) %action
       public shared Sub ActionMethod1(ByVal f As float)
         Console.WriteLine("ActionMethod1: {0}",f)
       End Sub
    %before %call *Method(int, ..) && %args(arg[0]) &&
              %withincode(*Run) %action
    public shared Sub ActionMethod2(ByVal i As int)
        Console.WriteLine("ActionMethod2: {0}", i)
    End Sub
    %before %call int *Method(..) && %!within(*MainClass) %action
       public shared Sub ActionMethod3
          (ByVal i As int, ByVal f As float)
          Console.WriteLine("ActionMethod3: {0}, {1}",i,f)
       End Sub
End class
```

стемы Aspect.NET, выпуск библиотек аспектов для поддержки Web-программирования и контрактного проектирования.

Автор будет признателен уважаемым коллегам за отзывы, замечания и предложения. Приглашаю присылать их по электронной почте: ***vosafonov@gmail.com***.

Литература

1. *Safonov V.O.* Using aspect-oriented programming for trustworthy software development. Wiley Interscience. John Wiley & Sons, 2008. 352 p.

2. *Сафонов В.О.* Aspect.NET – инструмент аспектно-ориентированного программирования для разработки надежных и безопасных программ // Компьютерные инструменты в образовании, 2007. № 5. С. 3–13.

3. *Сафонов В.О.* Практическое руководство по системе аспектно-ориентированного программирования Aspect.NET. Часть 1 // Компьютерные инструменты в образовании, 2008. № 3. С. 20–33.

4. *Сафонов В.О.* Практическое руководство по системе аспектно-ориентированного программирования Aspect.NET. Часть 2 // Компьютерные инструменты в образовании, 2008. № 4. С. 12–20.

5. *Сафонов В.О.* Современные технологии разработки надежных и безопасных программ (trustworthy computing) // Компьютерные инструменты в образовании, 2008. № 6. С. 25–33.

6. Web-страница проекта Aspect.NET. http://www.aspectdotnet.org

7. Aspect.NET 2.2. https://www.facultyresourcecenter.com/curriculum/pfv.aspx?ID=8658

8. Web-страница проекта AspectJ. http://www.aspectj.org

9. Jackson A., Clarke S. SourceWeave.NET: Cross-Language Aspect-Oriented Programming. Generative Programming and Component Engineering. Lecture Notes in Computer Science, 2004, Volume 3286/2004. P. 369–393.

10. *Kojarski S., Lorenz D.H.* Identifying Feature Interactions in Multi-Language Aspect-Oriented Frameworks. - ICSE '07 Proceedings of the 29th international conference on Software Engineering.

11. Web-сайт проекта AspectDNG. http://sourceforge.net/projects/aspectdng/

Abstract

The article covers novel features implemented in the new version of the aspect-oriented programming (AOP) system Aspect.NET – multi-language AOP, i.e. support of implementation of target applications and aspects to be woven in different programming languages. The role and implementation principles of multi-language AOP are analyzed; other work and approaches to the problem are considered; examples of multi-language AOP in Aspect.NET are provided; oncoming perspectives are outlined.

Keywords: aspect-oriented programming (AOP), Aspect.NET, multi-language programming, C#, Visual Basic.

Сафонов Владимир Олегович,
доктор технических наук,
профессор кафедры информатики
математико-механического
факультета СПбГУ,
vosafonov@gmail.com