Сафонов Владимир Олегович

ПРАКТИЧЕСКОЕ РУКОВОДСТВО ПО СИСТЕМЕ АСПЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ ASPECT.NET. ЧАСТЬ 1

ВВЕДЕНИЕ В АОП

Aspect.NET [1–11] – инструментарий аспектно-ориентированного программирования (АОП) [10] для платформы Microsoft.NET.

Данная статья является практическим руководством для освоения и использования новой версии Aspect.NET 2.1 [3].

Проект Aspect.NET поддержан Microsoft Research, однако уже более двух лет никем не финансируется, и вся работа, в том числе – огромная работа по сопровождению, ответам на письма пользователей, которую мы ведем, выполняется в инициативном порядке и как диссертационный проект для нескольких аспирантов. В связи с этим, разумеется, мы весьма заинтересованы в заказчиках.

АОП – перспективный подход к инженерии программ, предназначенный для разработки *сквозной функциональности (cross-cutting concerns)* – таких идей, методов, функциональных возможностей, реализуемых и модифицируемых в ходе разработки программ, которые принципиально, по своей природе, не могут быть реализованы одной *обобщенной процедурой* (*generalized procedure*) – тесно взаимосвязанной совокупностью модулей (например, иерархией классов), а требуют для своей реализации совокупности *рассредоточенных действий (tangled actions)*, которые должны быть добавлены в различные части существующего программного кода целевого приложения, для того, чтобы новая сквозная функциональность заработала.

Иными словами, сквозная функциональность – это новая функциональность, предназначенная для добавления в код существующего приложения, реализация которой «пронизывает насквозь» большинство существующих модулей программной системы.

Традиционные примеры сквозной функциональности: *безопасность (security), протоколирование (logging), безопасность исполнения программы в многопоточной вычислительной среде (MT-safety).*

Аспект – это реализация какой-либо сквозной функциональности. Суть АОП в том, что аспекты в нем рассматриваются как новая разновидность модулей, расширяющая традиционную концепцию модулей по Г. Майерсу [1]. Аспект – это модуль, применение которого осуществляется не путем вызова, как для процедуры или метода, а путем систематизированного *внедрения (weaving)* фрагментов кода аспекта в рассредоточенные модули целевой программы.

Вот реальный пример аспекта, который будет понятен разработчикам и

© В.О. Сафонов, 2008

пользователям новых платформ для разработки программ (Java, .NET) и входящих в них крупных компонент – компилятора в промежуточный код, JIT-компилятора, системы поддержки выполнения (виртуальной машины), базовой библиотеки классов. Предположим, что нам необходимо добавить в существующую версию Java Development Kit (JDK) 1.6 новую возможность – концепцию свойства (property), по аналогии с .NET, – как и планирует сделать Sun в JDK 1.7. Для этого необходимо: на лексическом уровне – добавить в Java-компилятор реализацию нового ключевого слова property; на синтаксическом – добавить реализацию синтаксической конструкции «property»; в генератор Java-байткода добавить генерацию соответствующего свойству промежуточного кода; соответственно, модифицировать и JIT-компилятор, и виртуальную машину, и базовую библиотеку классов. Весь новый код, который необходимо добавить в JDK для реализации свойств, и будет примером аспекта, хотя и довольно сложным, но вполне логичным и «близким сердцу» любого разработчика компиляторов. Он будет состоять не только из новых иерархий классов, но и из новых описаний и операторов, которые необходимо будет добавить в код существующей версии JDK.

Чтобы расширить целевую программу новой функциональностью, реализованной в виде аспекта, необходимо *внедрить* (*weave*) аспект в целевую программу. Внедрение аспекта осуществляется в соответствии с *правилами внедрения (weaving rules)*, в которых специфицированы *условия* (*conditions*) вставки *действий (actions)* аспекта в целевую программу и сами эти действия (новый вставляемый код). Инструмент АОП содержит такую важную компоненту, как *подсистема внедрения (weaver)*, которая и выполняет внедрение аспектов. При запуске подсистема внедрения аспектов, используя правила внедрения, определяет конкретные удовлетворяющие условиям *точки присоединения* аспекта (*join points*) в коде целевой программы, куда затем и добавляет действия аспекта.

Использование аспектов и применение АОП при разработке и сопровождении программ может существенно облегчить и систематизировать их, например:

– при решении ежедневных задач сопровождения программ – каждое исправление ошибки может быть представлено как аспект;

– при расширении возможностей программного обеспечения – каждая новая функциональность может быть реализована в виде аспекта;

– при улучшении надежности и безопасности программ – набор вставок кода, связанный с проверками безопасности, использованием контрактного проектирования (design by contract), безопасностью исполнения кода в многопоточной среде, протоколированием исполнения кода и т. д. – может быть также реализован в виде аспекта или библиотеки аспектов.

Более подробно об этом – в работах [1, 2].

В настоящее время наиболее известным и широко распространенным инструментом АОП является AspectJ [13], который, однако, предназначен только для платформы Java.

Мы предлагаем удобный инструмент АОП для другой современной платформы – .NET, о котором идет речь в данной статье. Aspect.NET распространяется бесплатно через академический сайт Microsoft [3, 8–10]. Aspect.NET реализован как расширение (add-in) Microsoft Visual Studio.NET 2005 и в своей реализации использует Microsoft Phoenix [14, 15] – инструментарий, предназначенный для поддержки разработки компиляторов и других средств анализа, генерации, оптимизации и модификации кода.

Aspect.NET 2.1 выпущен в апреле 2007 г. В настоящее время ведется работа над новыми версиями.

1. КРАТКИЙ ОБЗОР НОВЫХ ВОЗМОЖНОСТЕЙ ASPECT.NET 2.1

Для пользователей, которые, возможно, использовали более старые версии

Aspect.NET – 1.0, 1.1 или 2.0 [8 – 10], либо обратились к ним по ошибке, приведем для начала краткий перечень новых возможностей Aspect.NET 2.1, по сравнению с предыдущей версией Aspect.NET 2.0 [10]:

– основная новая возможность состоит в том, что ***Aspect.NET 2.1 работает с версией Phoenix Research Development Kit (RDK), выпущенной в марте 2007 г.,*** – наиболее новой версией Phoenix на момент выхода Aspect.NET 2.1;

– реализована возможность захвата возвращаемого значения целевого метода с помощью нового свойства *Aspect.RetValue* (см. пункт 7.3 с описанием примеров *RetTest* и *RetTest2*);

– улучшена производительность системы и исправлен ряд ошибок;

– добавлены новые примеры *RetTest* и *RetTest2* для практического освоения новых возможностей – захвата результата целевого метода, а также функциональности *WithinType / WithinMethod*, подробно описанной ниже.

Чтобы отличить Aspect.NET 2.1 от Aspect.NET 2.0 или более старых версий, выберите в основном меню Windows пункты *Control Panel (Панель управления)* → *Add or Remove Programs (Установка и удаление программ)*, выберите *Aspect.NET Framework* и нажмите кнопку *Click here for support information.*

Выдаваемый номер версии Aspect.NET должен быть равен 2.1.0, иначе это более старая версия системы.

2. ОКРУЖЕНИЕ ДЛЯ ИСПОЛЬЗОВАНИЯ ASPECT.NET 2.1

Aspect.NET 2.1 предназначен для использования в среде Visual Studio.NET 2005 и .NET 2.0, под управлением операционной системы ***Windows XP* с *Service Pack 2***.

Именно такое окружение было стандартным при разработке текущей версии Aspect.NET.

Данное обстоятельство следует с самого начала иметь в виду пользователям, которые надеются использовать Aspect.NET 2.1 с Windows Vista или Windows 2008, либо в среде Visual Studio.NET 2008, поэтому экспериментировать с использованием Aspect.NET 2.1 в этих более новых системах не рекомендуется, во избежание потери времени.

Безусловно, в следующих версиях Aspect.NET планируется обеспечить совместимость с Windows Vista, Windows 2008 и Visual Studio.NET 2008, однако это весьма значительная работа, требующая времени и усилий.

Aspect.NET Framework реализован как расширение (*add-in*) Visual Studio.NET 2005.

Это означает, что пользователи могут применять Aspect.NET как часть интегрированной среды Visual Studio.NET, с ее многочисленными возможностями (сборка, отладка, профилирование и др.) для разработки программного обеспечения.

На практике это означает, что не требуется какого-либо отдельного запуска Aspect.NET. Наша система запускается одновременно с Visual Studio, а ее графический пользовательский интерфейс (GUI), называемый *Aspect.NET Framework*, может рассматриваться как часть (расширенного) GUI интегрированной среды Visual Studio.NET.

Чтобы использовать Aspect.NET 2.1, необходимо предварительно инсталлировать следующие программные пакеты и инструменты:

1) ОС Windows XP с Service Pack 2,

2) Visual Studio.NET 2005 – либо Professional Edition, либо Standard Edition,

3) Microsoft Platform Research Development Kit (RDK) Codenamed «Phoenix» **March 2007 Release**, доступный для бесплатного скачивания на сайте [15].

После этого может быть инсталлирован Aspect.NET 2.1, дистрибутив которого вместе с примерами и документацией доступен в виде инсталляционного пакета .msi на сайте [3].

Пожалуйста, будьте особенно внимательны относительно используемой версии Phoenix и установок, необходимых для его работы:

• ***Используйте для Aspect.NET 2.1 только версию [15], которая на данный момент уже не является самой новой версией Phoenix*** (хотя и являлась ею на момент разработки Aspect.NET). ***Если Вы попытаетесь воспользоваться самой новой версией Phoenix SDK, апрель 2008 г. [14], Aspect.NET 2.1 с ней работать не будет.*** Перенос Aspect.NET на новую версию Phoenix – так же, как и ее перенос на новые версии Windows и Visual Studio, – значительная работа, требующая времени и заинтересованных в этом заказчиков.

• ***Для надежной работы версии Phoenix [15] обязательно добавьте к значению переменной окружения PATH следующий путь:*** **C:\Program Files\Microsoft Visual Studio 8\Common7\IDE** ***(либо замените «C» обозначением другого логического раздела, если Visual Studio на Вашем компьютере установлена на логическом диске, отличном от C:)***.

Еще раз обращаю Ваше внимание на то, что любое нарушение условий и настроек для успешной и надежной работы с Aspect.NET, полностью сформулированных выше, неизбежно приведет к ненужным проблемам. Мне приходилось многократно отвечать на однотипные письма пользователей из различных стран с одними и теми же вопросами и проблемами, вызванными лишь их же собственной невнимательностью при чтении руководства пользователя по Aspect.NET 2.1. Надеюсь, что российские пользователи окажутся более внимательными.

Если по ошибке Вы инсталлировали другую версию Phoenix и (или) другую версию Aspect.NET, настоятельно рекомендую ***деинсталлировать Aspect.NET, затем Phoenix, затем инсталлировать правильную версию Phoenix [15] и правильную версию Aspect.NET [3]***. Тогда никаких проблем с Aspect.NET возникнуть не должно.

Правильность инсталляции легко проверить путем пропуска самого простого примера, описанного ниже, входящего в поставку нашей системы.

3. ПРЕДЫДУЩИЕ ВЕРСИИ ASPECT.NET И РЕЖИМ ОБРАТНОЙ СОВМЕСТИМОСТИ

Всего до разработки текущей версии Aspect.NET 2.1 было реализовано еще три версии Aspect.NET:

• Aspect.NET 1.0 [7], выложенная на сайт в сентябре 2005. Данная версия совместима с ранней версией Phoenix RDK, датируемой февралем 2005 г., и с версией Visual Studio.NET 2005 beta 2. Мы не рекомендуем более использовать данную версию Aspect.NET, так как она зависит от устаревших версий Phoenix и Visual Studio.NET.

• Aspect.NET 1.1 [8], выложенная на сайт в марте 2006. Данная версия совместима со следующей версией Phoenix RDK, датируемой ноябрем 2005 г., и с версией Visual Stutio.NET 2005 (Professional Edition или Standard Edition). Мы не рекомендуем более использовать данную версию Aspect.NET, так как она зависит от устаревшей версии Phoenix.

• Aspect.NET 2.0 [9], выложенная на сайт в сентябре 2006. Данная версия совместима с версией Phoenix RDK, датируемой маем 2006 г., и с версией Visual Stutio.NET 2005 (Professional Edition или Standard Edition). Мы не рекомендуем более использовать данную версию Aspect.NET, так как она зависит от устаревшей версии Phoenix. Аспекты, разработанные в Aspect.NET 2.0, могут быть использованы также и в Aspect.NET 2.1, за исключением примеров, базирующихся на Phoenix API (пример *MAddNop / MAddNopCounter* в нашей поставке), так как Phoenix API существенно изменился с мая 2006 г.

Поскольку текущая версия (Aspect.NET 2.1) содержит много новых возможностей, а реализация аспектов в версии 2.1 отличается от их реализации в версиях 1.0 и 1.1, в Aspect.NET 2.1 реализован режим обратной совместимости. В данном режиме Вы можете использовать Aspect.NET 2.1 со «старыми» (разработанными в версиях 1.0 или 1.1) определениями аспектов на метаязыке

Aspect.NET.ML или непосредственно на языке C# (с использованием специализированного атрибута *AspectDef*).

Режим обратной совместимости по умолчанию отключен. Он может быть включен, используя Visual Studio.NET GUI, выбором пункта меню *Tools / Options*, затем выбором «Aspect.NET Framework» и селектора «AspectDef 1.0» в области, помеченной текстом «Aspect.NET attributes version».

Таким образом, если Вы переносите аспекты из версии Aspect.NET 1.0 или 1.1 в версию Aspect.NET 2.1, Вы можете использовать «старые» определения аспектов, специфицированные как на метаязыке АОП, так и непосредственно на языке C# со специализированным атрибутом *AspectDef*, без каких-либо изменений, со следующим исключением:

В Aspect.NET 1.0 / 1.1 была реализована ***временная*** схема связывания аспекта с точкой присоединения в целевой программе. Она была основана на своего рода «соглашении о совместимости» между Aspect.NET и автором аспекта, суть которого в том, что аргументы в действии аспекта, если они есть, использовались для передачи имени целевого метода и ссылки на целевой объект, к которому он применяется. *Для Aspect.NET 2.1 это не так.* В новой версии используются специальные конструкции метаязыка Aspect.NET.ML (например, *%TargetMemberInfo*) для явных ссылок на элементы контекста точки присоединения (например, информация о целевом методе) и специальные фильтры для точной спецификации захвата аргументов целевого метода.

Так что, если требуется перенести «старые» аспекты в Aspect.NET 2.1, не следует придерживаться описанных выше временных соглашений о связях для захвата контекста точки присоединения через аргументы действия аспекта.

Вместо этого рекомендуется использовать новые механизмы и конструкции для доступа к контексту точки присоединения, описанные ниже.

Также следует иметь в виду, что нельзя использовать в Aspect.NET 2.1 «старый» пример *MAddNop / MAddNopCounter* (из пакета поставки Aspect.NET 1.0 или 1.1 [7]), базирующийся на Phoenix, так как Phoenix API существенно изменилось, по сравнению с предыдущими версиями. Модифицированная версия примера *AddNopTool / MAddNopCounter*, работающая в Aspect.NET 2.1, включена в дистрибутив Aspect.NET 2.1 на академическом сайте Microsoft [3].

4. АРХИТЕКТУРА ASPECT.NET

Аспект в системе Aspect.NET определяется как исходный код класса (в наиболее общем виде – код *единицы компиляции*) на языке C# (в будущем – на любом другом языке, реализованном в .NET), *аннотированный* конструкциями нашего *метаязыка АОП* (называемого *Aspect.NET.ML*) с целью выделения частей определения аспекта.

Определение аспекта состоит из следующих частей:

– *заголовок аспекта (aspect header)*;

– (необязательные) *данные аспекта* (*aspect data*) – как правило, приватные поля;

– *модули аспекта (aspect modules)* – методы или функции;

– *правила внедрения аспекта (aspect weaving rules)*, которые, в свою очередь, состоят из *условий внедрения (weaving conditions)* и *действий (actions)*, которые, в соответствии с заданными правилами, должны быть *внедрены* (*woven*) в целевую сборку.

В Aspect.NET 2.1 поддерживается только возможность определения аспектов на языках Aspect.NET.ML и C#. Поддержка языка Visual Basic как второго возможного языка реализации аспектов входит в план разработки следующих версий.

Сразу отметим, что структура метаязыка Aspect.NET.ML и его семантика не привязаны к конкретному языку реализации, что позволит использовать одни и те же спецификации аспектов на Aspect.NET.ML

в сочетании с различными языками их реализации. Подобная языковая независимость метаязыка АОП – важный принцип Aspect.NET.

Правила внедрения аспекта определяют *точки присоединения (join points)* в целевой программе, в которые должны быть добавлены *действия* аспекта. Именно действия аспекта, в конечном счете, определяют его функциональность.

Препроцессор (конвертор) Aspect.NET – pre-processor (converter) – преобразует аннотации АОП в определения *специализированных атрибутов АОП (AOP custom attributes),* специально спроектированные для Aspect.NET, которые помечают классы и методы как части определения аспекта (см. рис. 1). Далее, соответствующий штатный компилятор Visual Studio.NET (для Aspect.NET 2.1 – компилятор с языка C#) преобразует специализированные атрибуты АОП в *метаданные (metadata) аспектной сборки (aspect assembly),* которые, согласно общим принципам архитектуры .NET, хранятся в сборке вместе с ее кодом на промежуточном языке MSIL.

Точки присоединения в Aspect.NET определяются по правилам внедрения, которые либо являются частью определения аспекта, либо определены в отдельном модуле – *наборе правил* (*rule set*). В Aspect.NET 2.1 наборы правил не реализованы; их реализация планируется на следующие версии.

Правила внедрения содержат:

– *условия (conditions)*, определяющие точку вызова действия аспекта относительно конкретной точки присоединения в целевой программе: *before – перед точкой присоединения; after – после точки присоединения; instead – вместо кода, являющегося точкой присоединения*;

– *контекст (context)* вызова действия аспекта: ключевое слово *call* – вызов некоторого метода в целевой программе; *assign* – *присваивание* переменной (полю) в целевой программе; *use – использование* переменной (поля); за ключевым словом следует *шаблон (wildcard)* для поиска в целевой программе контекста вызова действия аспекта.

В Aspect.NET 2.1 реализованы только точки присоединения вида *call (вызов)*. Реализация других видов точек присоединения планируется в следующих версиях.

Процесс внедрения аспекта состоит из *двух фаз: сканирования (scanning)* – поиска потенциальных точек внедрения в целевой программе – и собственно *внедрения (weaving)* вызовов действий аспекта в найденные точки присоединения.

Одним из важнейших принципов Aspect.NET, в отличие от многих других инструментов АОП, является возможность *внедрения, управляемого пользователем (user-controlled weaving)*. По завершении фазы сканирования Aspect.NET позволяет пользователю *отметить (select)* или *отменить отметку (unselect)* любой из найденных потенциальных точек внедрения, используя Aspect.NET Framework GUI. Это позволяет избежать синдрома «слепого внедрения» (blind weaving) без какого-либо контроля его результатов, что могло бы сделать результирующий код целевой программы неясным или вообще

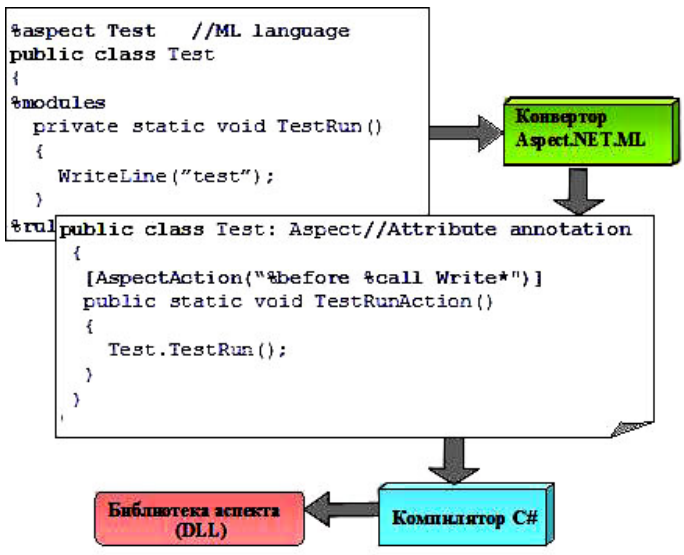

Рис. 1. Архитектура Aspect.NET

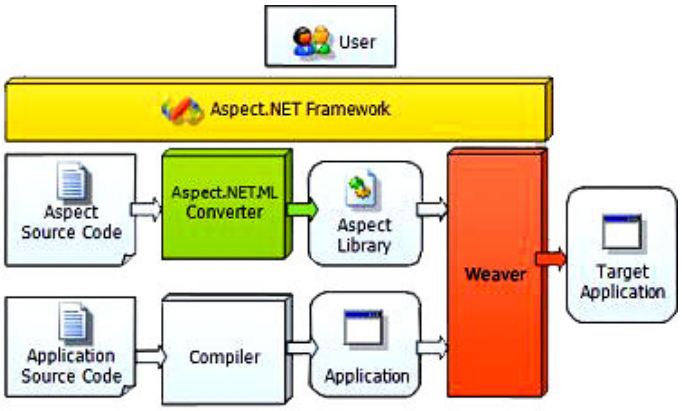

Рис. 2. Компоненты Aspect.NET

ошибочным и затруднить его отладку. Таким образом, *в Aspect.NET окончательный выбор, внедрять или не внедрять аспект в ту или иную точку присоединения, остается за пользователем.*

Пользователь может определять аспекты либо в форме Aspect.NET.ML, либо непосредственно в терминах специализированных атрибутов на языке C#, без использования метаязыка. Aspect.NET Framework поддерживает два соответствующих типа проектов для определения аспектов.

Основные компоненты Aspect.NET – *подсистема внедрения (weaver), конвертор метаязыка (meta-language converter) и Aspect.NET framework* – и их взаимосвязь изображены на рис. 2.

Листинг 1

```
using System;
using System.Collections.Generic;
using System.Text;

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

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

5. ИСПОЛЬЗОВАНИЕ ASPECT.NET FRAMEWORK НА ПРИМЕРЕ

Рассмотрим простой пример целевой программы и определения аспекта, предназначенного для внедрения в нее. Опишем последовательно все этапы определения аспекта, его внедрения с помощью Aspect.NET и тестирование функциональности измененной целевой программы после внедрения аспекта, по сравнению с ее первоначальной версией. Данного примера вполне достаточно, чтобы изучить, как работать в Aspect.NET Framework.

1. Рассмотрим и введем в Visual Studio простую консольную программу на языке C# (см. листинг 1).

Предположим, что мы создали в Visual Studio.NET 2005 *решение (solution)* для этой программы и назвали его *ConsoleApplication1*.

Выполним сборку (build) и запустим нашу программу. Ее вывод:

Main
P

2. После инсталляции Aspect.NET «поверх» Visual Studio.NET 2005, как рекомендовано выше, создадим теперь простой аспект, цель которого – вставка в целевую программу *протоколирующих сообщений* (*logging messages*) вида:

Hello P

и

Bye P

соответственно, перед вызовом каждого метода *P* и после его вызова в целевой программе (выражаясь более точно, в сообщении будем выводить имя и заголовок целевого метода).

Таким образом, данный аспект, в некотором смысле, делает целевую программу более «вежливой».

Для создания решения (solution), в котором будет определен наш аспект, вызовем Visual Studio.NET 2005 и выберем в основном меню пункт *File / New / Project*. Заметим при этом, что окно Aspect.NET Framework появляется на экране как часть GUI Visual Studio.NET, как показано на рис. 3. Выберем вид проекта «*Aspect.NET ML module*» *(модуль на метаязыке Aspect.NET.ML)*.

Заметим, что имеется и другая разновидность проекта – *Aspect.NET module (модуль Aspect.NET)*. Ее следует выбрать, если мы решим обойтись без метаязыка Aspect.NET.ML и использовать специализированные атрибуты АОП непосредственно на языке C#.

Предположим, что имя аспектного проекта – *Aspect1* (оно выбирается по умолчанию).

Aspect.NET Framework сгенерирует следующий шаблон определения аспекта (см. листинг 2), который будет записан в файл с именем (по умолчанию) *Aspect.an*.

Для имен файлов с кодом на метаязыке Aspect.NET.ML используется расширение «*.an*».

Листинг 2

```
%aspect Aspect1

using System;

class Aspect1
{
    %modules
    %rules
}
```

Теперь введем полный исходный код аспекта, приведенный в листинге 3.

Аспект *Aspect1* состоит из:

- *Заголовка* аспекта – **`%aspect Aspect1`**. Конвертор Aspect.NET преобразует его в заголовок класса с именем *Aspect1*. Допустимо также явное использование заголовка класса *Aspect1* после заголовка аспекта. В таком случае заголовок класса не будет сгенерирован.
- Раздела **`modules`**, где необходимо определить модули аспекта, которые, как правило, должны быть **`public static`** методами.
- Раздела **`rules`**, в котором указываются правила внедрения аспекта. Каждое

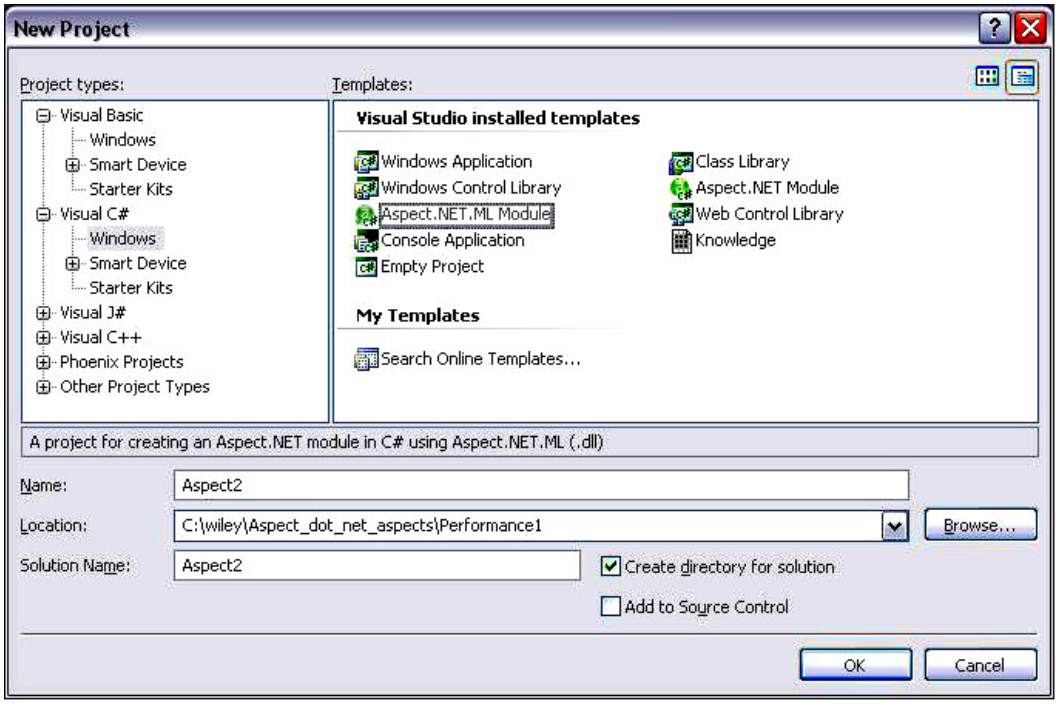

Рис. 3. Создание проекта «аспект» в Visual Studio.NET 2005

Листинг 3

```
%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.ToString()); }

      %after %call *
      %action
      public static void SayBye ()
      { Say ("Bye " + %TargetMemberInfo.ToString()); }
}
```

правило внедрения состоит из *условия* (**%before %call ***, **%after %call ***) и *действия*. Условие используется для поиска множества потенциальных точек присоединения (*join points*) в целевой программе, в которые должно быть выполнено внедрение аспекта (перед вызовом каждого метода целевой программы или после его вызова, соответственно). Множество имен методов специфицировано с помощью традиционного шаблона «*» («все»). Действие должно быть определено как **public static** метод.

Для протоколирования фактического имени и заголовка метода целевой программы в каждой точке присоединения используется **%TargetMemberInfo** – конструкция метаязыка Aspect.NET.ML. Она преобразуется конвертором в конструкцию языка C# для доступа к имени и сигнатуре целевого метода.

3. Теперь соберем (build) решение *Aspect1*. Заметим, что в ходе сборки создан новый текстовый файл *Aspect1.an.cs* с определением аспекта на языке C#, который автоматически добавлен к набору файлов решения (см. листинг 4).

Проанализируем полученный код. На уровне языка реализации C# аспект представлен определением класса, а каждое действие помечено специализированным атрибутом AspectAction, содержащим условие внедрения и используемым системой Aspect.NET.

Если пользователю так удобнее, аспект может быть определен непосредственно на C# без использования метаязыка Aspect.NET.ML. В таком случае следует выбрать другой вид проекта – *Aspect.NET module*, для которого Aspect.NET предлагает следующий шаблон исходного кода на C# (см. листинг 5).

4. Чтобы теперь внедрить аспект *Aspect1* в программу *ConsoleApplication*, откроем в Visual Studio решение *ConsoleApplication1* (см. рис. 4).

Мы увидим окно Aspect.NET Framework.

Затем выберем DLL-библиотеку аспекта Aspect1, используя иконку «*Open*», расположенную на вкладке *Aspects*. Как правило, DLL-библиотека, являющаяся представлением аспекта в виде двоичного кода, записывается системой Visual Studio.NET в поддиректорию *bin\debug* решения, представляющего аспект.

В результате увидим компоненты открытого аспекта – имя и действия, визуа-

Листинг 4

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.42
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Aspect1 {
    using System;
    using AspectDotNet;


    public class Aspect1 : Aspect {

                public static void  Say (String s) {
 Console.WriteLine(s);
        }

        [AspectAction("%before %call *")]
                public static void  SayHello () {
 Say ("Hello " + TargetMemberInfo.ToString());
        }

        [AspectAction("%after %call *")]
                public static void  SayBye () {
 Say ("Bye " + TargetMemberInfo.ToString());
        }
    }
}
```

лизируемые Aspect.NET Framework (см. рис. 5).

После этого необходимо вызвать подсистему внедрения аспектов (weaver) в режиме сканирования (поиска) точек присоединения. Это делается нажатием кнопки *Find Joinpoints*. В консоли вывода будет отображаться трассировка работы weaver'а.

Листинг 5

```
using System;
using System.Collections.Generic;
using System.Text;
using AspectDotNet;

namespace Aspect1
{
    [AspectDescription("MyAspect description")]
    public class MyAspect : Aspect
    {
    }
}
```

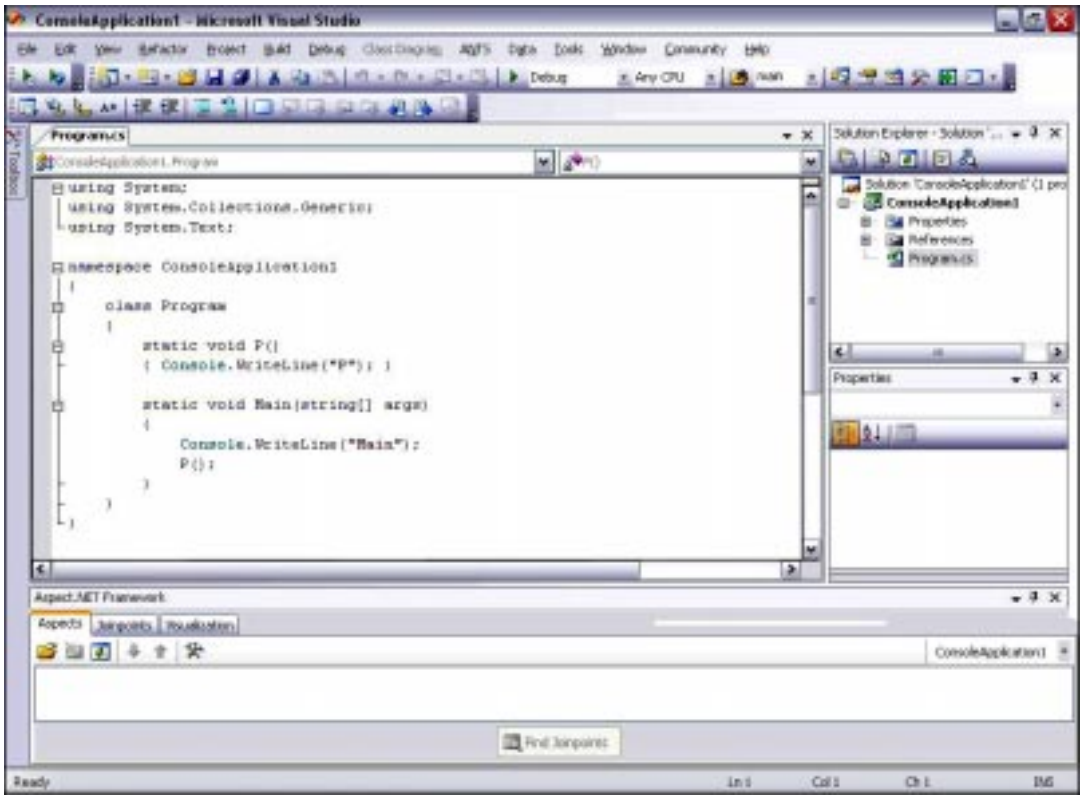

Рис. 4. Открытие целевой программы и окна Aspect.NET

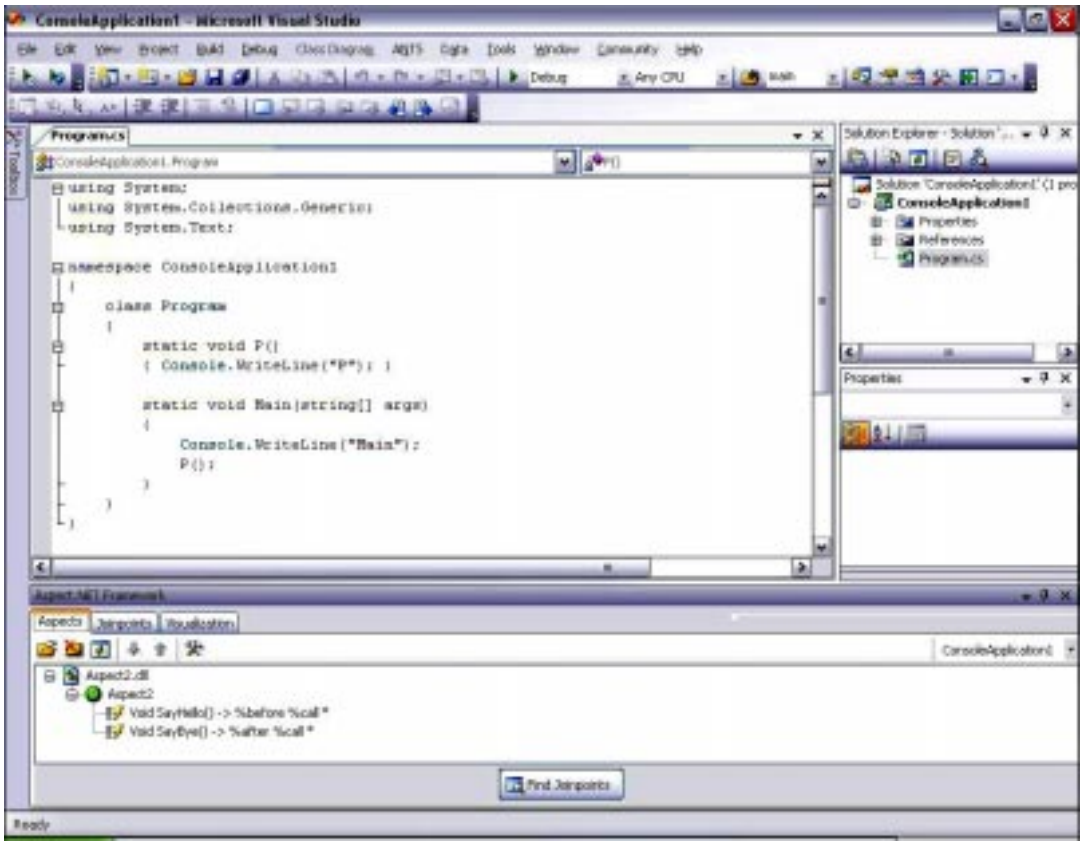

Рис. 5. Визуализация открытого аспекта

Следует отметить, что внедрение аспектов в Aspect.NET осуществляется на уровне *сборки (.NET assembly),* содержащей MSIL-код и метаданные. В этом отношении Aspect.NET отличается от многих других инструментов АОП. Результирующая сборка с внедренными аспектами может быть использована самым обычным образом, например, быть дизассемблированной утилитой *ildasm* или отлаживаться штатным отладчиком .NET.

Вернемся к нашему примеру. После завершения работы подсистемы внедрения на вкладке *Joinpoints* будут визуализированы потенциальные точки присоединения аспекта (см. рис. 6).

На данном этапе имеется возможность «вручную» отметить какую-либо потенциальную точку присоединения, либо отменить ее отметку, исключив тем самым ее из дальнейшего процесса внедрения. Это позволяет избежать «слепого», неконтролируемого внедрения аспектов – одной из самых серьезных опасностей, которые таит в себе АОП – мощное «оружие», управляющее групповыми модификациями кода. Описываемая возможность Aspect.NET – *внедрение, управляемое пользователем,* – позволяет сделать процесс внедрения аспектов более безопасным и разумным. Отметим еще раз, что подобная возможность *отсутствует* во всех известных нам остальных, даже весьма популярных, инструментах АОП.

На данном этапе имеется также возможность визуализации каждой потенциальной точки присоединения аспекта в исходном коде целевой программы. Для этого следует использовать вкладку Visualization, щелкнув на соответствующем красном отрезке, схематически изображающем точку присоединения.

Наконец, для внедрения аспекта следует нажать кнопку *Weave aspects*. При

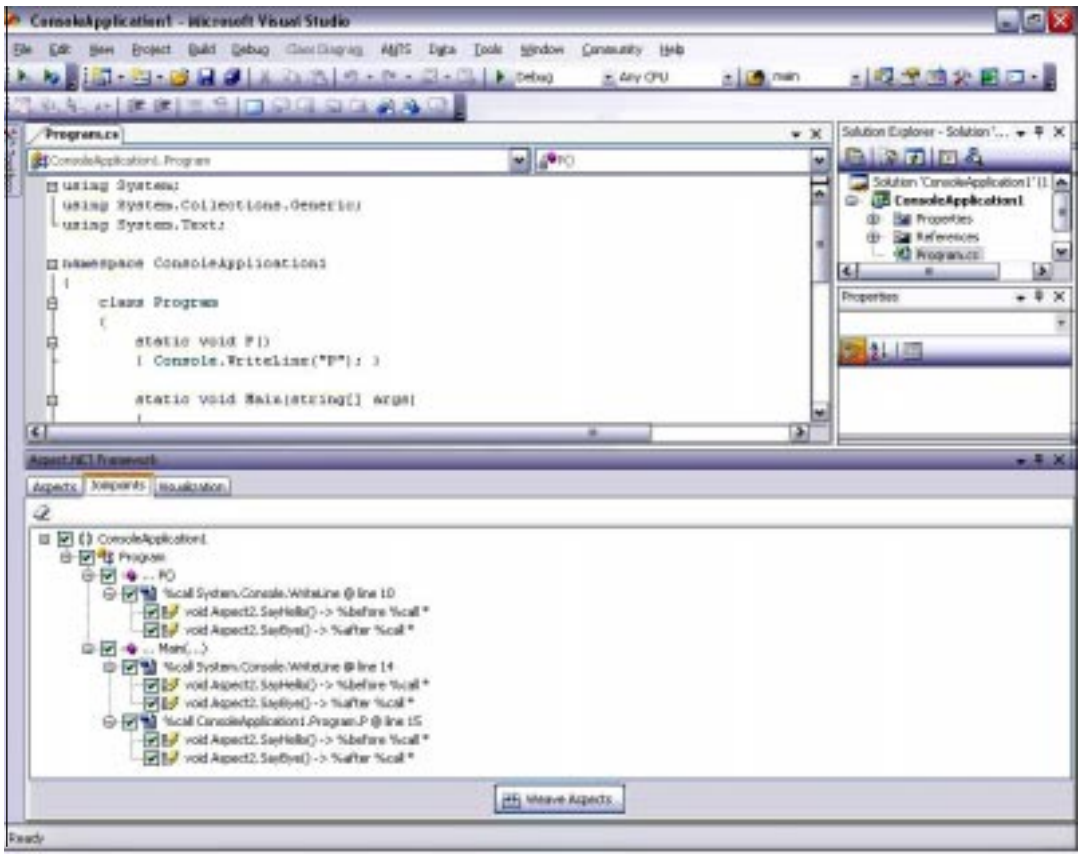

Рис. 6. Визуализация точек присоединения, найденных подсистемой ***внедрения аспектов***

этом вызовется weaver в режиме внедрения.

На консоли вывода будет выдаваться трассировка его работы.

После завершения внедрения аспекта Вам будет предложено одним нажатием кнопки сразу же запустить модифицированный код целевой программы.

Модифицированный код с внедренными аспектами будет записан системой Aspect.NET в ту же директорию, в которой расположен и исходный бинарный код целевой программы. Имя модифицированного файла кода имеет вид: *~TargetAppName,* в нашем примере: *~ConsoleApplication1*.

Вывод модифицированной программы:

Hello Void WriteLine(System.String)
Main
Bye Void WriteLine(System.String)
Hello Void P()
Hello Void WriteLine(System.String)
P
Bye Void WriteLine(System.String)
Bye Void P()

Далее Вы можете выгрузить из Aspect.NET сборку аспекта, загрузить сборку другого аспекта, загрузить заново все сборки всех аспектов, уже известных Aspect.NET, и управлять структурой списка (очереди) открытых аспектов, используя соответствующие иконки на вкладке *Aspects*.

6. ОПЦИИ СИСТЕМЫ ASPECT.NET

Опции Aspect.NET Framework могут быть установлены после выбора в главном меню пункта *Tools / Options / Aspect.NET Framework* (см. рис. 7).

Как видно из рисунка, по умолчанию Aspect.NET 2.1 генерирует *новые* специализированные атрибуты АОП – *AspectDescription* (Ваши комментарии к аспектам и их частям) и *AspectAction* (атрибут, помечающий действия аспекта).

Как уже отмечалось, для обратной совместимости с предыдущими версиями Aspect.NET (1.0, 1.1), могут быть использованы «старые» определения аспектов, созданные с помощью этих предыдущих версий. При этом они не должны использовать характерное для версий 1.0 и 1.1 временное соглашение о связях между действием аспекта и контекстом точки присоединения через аргументы действия. Если выбрать пункт *AspectDef (version 1.0),* то Aspect.NET – как конвертор, так и weaver, – будет работать в режиме совместимости с версиями 1.x и использовать «старый» специализированный атрибут *AspectDef*.

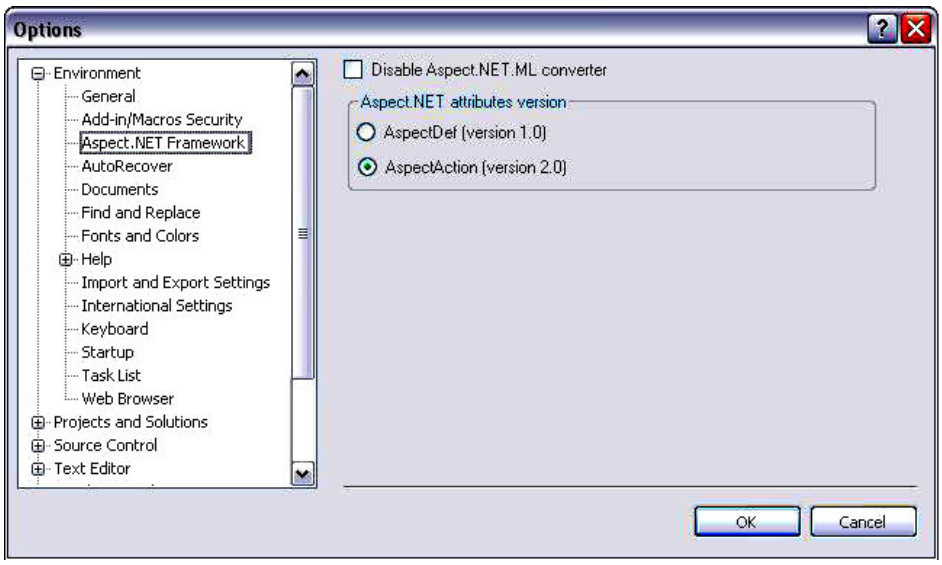

Рис. 7. Опции Aspect.NET

Наконец, фаза конвертирования из Aspect.NET.ML на C# может быть явным образом отменена с помощью опций Aspect.NET, в результате чего Вы сможете использовать непосредственно язык C# со специализированными атрибутами и соответствующий тип проекта – Aspect.NET module – для определения аспектов.

Продолжение следует.

Литература

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

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

3. Aspect.NET 2.1: http://www.msdnaa.net/curriculum/?id=6801

4. *Safonov V.O.* Aspect.NET: a new approach to aspect-oriented programming // .NET Developer's Journal, 2003, # 4.

5. *Safonov V.O.* Aspect.NET: concepts and architecture // .NET Developer's Journal, 2004, # 10.

6. *Safonov V.O., Grigoryev D.A.* Aspect.NET: aspect-oriented programming for Microsoft.NET in practice // NET Developer's Journal, 2005, # 7

7. *Safonov V.O., Gratchev M.K., Grigoryev D.A., Maslennikov A.I.* Aspect.NET – aspect-oriented toolkit for Microsoft.NET based on Phoenix and Whidbey // «.NET Technologies 2006» International Conference Proceedings, Pilsen, Czech Republic, 2006. http://dotnet.zcu.cz

8. Aspect.NET 1.0: http://www.msdnaa.net/curriculum/?id=6219

9. Aspect.NET 1.1: http://www.msdnaa.net/curriculum/?id=6334

10. Aspect.NET 2.0: http://www.msdnaa.net/curriculum/?id=6595

11. Web-сайт проекта Aspect.NET: http://www.aspectdotnet.org

12. Web-сайт, посвященный аспектно-ориентированной разработке программ: http://aosd.net

13. Web-сайт AspectJ: http://www.aspectj.org

14. Web-сайт Microsoft Phoenix: http://research.microsoft.com/phoenix

15. Web-страница Microsoft Phoenix RDK, March 2007, используемого в Aspect.NET 2.1: https://connect.microsoft.com/Downloads/DownloadDetails.aspx?SiteID=214&DownloadID=5538

Сафонов Владимир Олегович,
доктор технических наук,
профессор кафедры информатики
СПбГУ, руководитель лаборатории
Java-технологии.

© Наши авторы, 2008.
Our authors, 2008.