ПРОГРАММИРОВАНИЕ С ЯВНЫМ ВЫДЕЛЕНИЕМ СОСТОЯНИЙ НА ПЛАТФОРМЕ .NET

Д.Г.Шопырин
Санкт-Петербургский государственный университет информационных технологий, механики и оптики
Тел.: (812) 325-31-31 (доп. 30107), e-mail: sdanil@yandex.ru

В программировании с явным выделением состояний [1], также известным как SWITCH-технология [2], основными понятиями являются *автомат* и *состояние автомата*. Однако в традиционных языках программирования эти понятия отсутствуют. Поэтому, для упрощения использования SWITCH и подобных ей технологий предлагаются различные библиотеки и расширения компиляторов [3]. Ниже кратко рассматривается библиотека ViStaL, облегчающая программирование с явным выделением состояний.

Проект ViStaL (**http://vistal.sf.net/**) является развитием идей, предложенных в библиотеке STOOL [4]. В данной работе будет рассмотрена библиотека ViStaL.Net, реализованная на языке программирования C#. За основу библиотеки ViStaL.Net принят метод реализации автоматных объектов на основе виртуальных вложенных классов [5]. При реализации библиотеки ViStaL.Net широко используются технологии *отражения* (*reflection*) и *порождения (emit)* кода [6].

Основными понятиями библиотеки ViStaL.Net являются *автоматный объект* и *режим работы*.

Автоматным классом называется класс, наследующийся от библиотечного класса Machine. Автоматный класс содержит вложенные классы, описывающие режимы работы. *Автоматным объектом* называется экземпляр автоматного класса.

Режимом работы называется вложенный класс *автоматного класса*, описывающий поведение автоматного объекта в одном из его режимов. Класс режима работы наследуется от библиотечного класса Mode.

Автоматный класс не реализует какой-либо прикладной интерфейс *явно*. Вызов метода интерфейса эквивалентен отправке *сообщения объекту*. Автоматный объект предоставляет универсальный механизм обработки таких сообщений, называемый манипулятором (*handler*). Манипулятор способен обработать (успешно или безуспешно) сообщение, эквивалентное вызову *любого* метода *любого*, наперед неизвестного, интерфейса.

Методы, реализованные в классе режима работы, рассматриваются как обработчики сообщений. Манипулятор, в процессе обработки сообщения, просматривает методы текущего режима работы и сравнивает их сигнатуру с сигнатурой вызванного метода. Обработка сообщения происходит успешно, если в текущем режиме работы автоматного объекта представлен обработчик данного сообщения. Классы режима работы могут явно реализовывать прикладной интерфейс, что в значительной мере упрощает сопровождение кода.

Автоматный объект предоставляет также механизм создания *адаптеров*. *Адаптером автоматного объекта* называется объект, реализующий некий прикладной интерфейс и переадресующий все вызовы методов данного интерфейса *манипулятору*. Адаптер позволяет инкапсулировать наличие автоматной логики.

Основными высокоуровневыми средствами структурирования макрологики в библиотеке ViStaL.Net являются *вложение* и *наследование* автоматных объектов.

Вложение конечных автоматов. Возможно два уровня вложения автоматов:

- вложение автомата в автомат;
- вложение автомата в режим.

Жизненный цикл вложенного автоматного объекта эквивалентен жизненному циклу его контейнера (автомата или режима соответственно).

Вызов методов вложенного автоматного объекта происходит неявно. *Манипулятор* сначала переадресовывает вызовы вложенным автоматам, а затем осуществляет вызов метода объемлющего автомата.

Вложенный автомат декларируется посредством объявления свойства, возвращающего экземпляр автоматного объекта. Созданное свойство может быть использовано для типизированного доступа к вложенному автоматному объекту.

Наследование конечных автоматов. Модель наследования автоматных классов, используемая в библиотеке ViStaL.Net, основана на методе *виртуальных вложенных классов* [5].

Расширение поведения конечного автомата с помощью наследования принципиально отличается от наследования отдельных его частей. В работе [7] предлагается подход позволяющий организовывать иерархии режимов работы, в то время как библиотека ViStaL.Net позволяет организовывать иерархию *макрологики*, где *макрологика* трактуется как совокупность режимов работы автоматного класса и переходов между ними.

Для переопределения поведения базового автоматного класса в каком-либо из его режимов, достаточно объявить класс режима с таким же именем в производном автоматном классе. Класс режима производного объекта может являться потомком соответствующего класса режима базового объекта.

Для добавления *нового* режима работы достаточно добавить класс режима работы и обеспечить возможность перехода во вновь созданный режим как минимум из одного, уже существующего, режима работы.

Повторное использование классов режимов. Библиотека ViStaL.Net позволяет повторно использовать классы режимов в других автоматах. Класс режима M_1 автомата A_1 может быть повторно использованным в автомате A_2, не являющимся потомком A_1. Данная возможность может быть обеспечена одним из ниже перечисленных способов:

- наследование класса режима: класс режима M_2 автомата A_2 может быть унаследован от класса режима M_1 автомата A_1;
- непосредственное введение класса режима: режим M_1 может быть явно введен в автомат A_2 посредством атрибута UseMode;
- групповое введение классов режима: режим M_1, вместе со всеми остальными режимами автомата A_1, может быть введен в автомат A_2 посредством атрибута ModeBase.

Заключение

Несмотря на то, что библиотека ViStaL.Net использует только стандартные средства языка программирования C#, уровень предоставляемых библиотекой возможностей как минимум сравним с подходами, основанными на расширении семантики языка. С другой стороны, использование только стандартных средств разработки значительно упрощает использование программирования с явным выделением состояний в реальных проектах.

Литература

1. Шалыто А.А., Туккель Н.И. Программирование с явным выделением состояний // Мир ПК. 2001. №8, 9.

2. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

3. Adamczyk P. The Anthology of the Finite State Machine Design Patterns // The 10th Conference on Pattern Languages of Programs, 2003.

4. Шопырин Д.Г., Шалыто А.А. Объектно-ориентированный подход к автоматному программированию // Информационно-управляющие системы, №5, 2003, с. 29-39.

5. Шопырин Д.Г. Метод проектирования и реализации конечных автоматов на основе виртуальных вложенных классов //Информационные технологии моделирования и управления. 2005. № 1(19), с. 87-96

6. Либерти Дж. Программирование на C#. М.: Символ-Плюс, 2003, 688 с.

7. Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А. State Machine – новый паттерн объектно-ориентированного проектирования //Информационно-управляющие системы. 2004. №5, с. 13-25.