ПРИМЕНЕНИЕ КЛАССА “STATE” В ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ С ЯВНЫМ ВЫДЕЛЕНИЕМ СОСТОЯНИЙ

Д.Г.Шопырин, А.А.Шалыто
Санкт-Петербургский государственный университет точной механики и оптики (технический университет)
Тел.: (812) 325-31-31 (доп. 2271), e-mail: sdanil@land.ru

В работах [1, 2] предлагается метод построения объектно-ориентированных программ с явным выделением состояний. В этих работах подробно рассмотрены вопросы проектирования и реализации программных систем этого класса. Однако реализация программ при использовании указанного метода обладает рядом недостатков:

- не отражается совокупность состояний автоматов, последовательность их запуска и изменений состояний;
- функции протоколирования вводятся в текст программы разработчиком;
- в шаблоне функции, реализующей автомат, может быть использован один оператор *switch*.

Для устранения указанных недостатков в настоящей работе предлагается изменить структуру программ рассматриваемого класса (рис. 1) по сравнению с предложенной в работе [3].

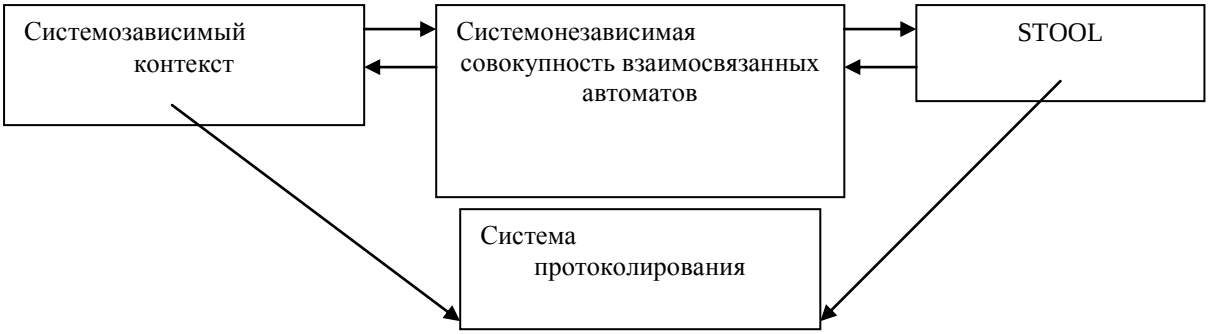

Рис. 1. Структура программы

На этой схеме блок *STOOL* (SWITCH-Technology Object Oriented Library) – библиотека классов, реализованная в настоящее время на языке C++.

Особенность предлагаемого подхода состоит в том, что в отличие от работ [1, 2], автоматы предлагается использовать не как методы классов, а как объекты, являющиеся потомками класса *Auto (Автомат)*. Это более общий подход, так как автоматы-методы можно легко свести к автоматам-объектам, но не наоборот.

Класс *State* представляет состояние автомата, а класс *Info* – описание автомата. Этот класс требуется для организации действительно автоматического протоколирования.

Как и в работах [1, 2], каждый автомат при запуске делает не более одного перехода.

Рассматриваются два варианта реализации алгоритмов с применением автоматов:

- автомат является телом цикла *while*;
- автомат используется непосредственно.

Автоматы первого типа удобны при реализации вычислительных алгоритмов [4], а автоматы второго типа – для событийных (реактивных) систем [1].

Методы *operator int()* и *operator=(int)* класса *State* позволяют пользоваться экземпляром класса *State* так, как будто он является переменной типа *int*, применяемой в работах [1, 2, 4]. Использование объекта вместо скалярной переменной позволяет вынести из оператора *switch* все функции, отличные от основных – функций переходов, входных переменных и выходных воздействий. Это позволяет также отразить "глобальное" состояние системы автоматов и одинаковым образом реализовывать действия и деятельности.

Предлагаемый подход по сравнению с применением паттерна *State* [5] в рамках объектно-ориентированного программирования обеспечивает сохранение в программах оператора *switch,* позволяющего целостно, формально и изоморфно реализовывать графы переходов автоматов.

Работа выполнена при поддержки Российского фонда фундаментальных исследований по гранту 02-07-90114 “Разработка технологии автоматного программирования”.

Литература

1. Шалыто А.А., Туккель Н.И. Танки и автоматы. //BYTE/Россия. 2003. №2. http://is.ifmo.ru (раздел «Статьи»)

2. Туккель Н.И., Шалыто А.А. Система управления танком для игры Robocode. Объектно-ориентированное программирование с явным выделением состояний. http://is.ifmo.ru (раздел «Проекты»)

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

4. Кузнецов Б.П. Психология автоматного программирования. //BYTE/Россия. 2000. № 11.

5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001.