Опубликовано в трудах научно-методической конференции "Телематика'2001", СПб.: СПбГИТМО, 2001, С.176-177

ПОВЫШЕНИЕ ЦЕНТРАЛИЗАЦИИ УПРАВЛЕНИЯ ПРИ ПРОГРАММИРОВАНИИ "РЕАКТИВНЫХ" СИСТЕМ

А.А.Шалыто, Н.И.Туккель, С.А. Ваганов
Государственное унитарное предприятие
"Научно-производственное объединение "Аврора"",
Санкт-Петербургский государственный институт
точной механики и оптики (технический университет),
Санкт-Петербург,
ООО "Compass Plus",
Магнитогорск

Для систем со сложной логикой алгоритмы функционирования могут быть разделены на две группы: функциональные и второстепенные. Функциональные алгоритмы решают общие задачи, поставленные перед системой, а второстепенные – локальные задачи.

Традиционный подход к построению "реактивных" (событийных) систем сводится к созданию обработчиков событий [1]. При этом обработчики могут либо независимо, либо совместно решать поставленную задачу. Во втором случае имеет место распределенное управление, при котором функциональная и второстепенная логика обычно перемешана и рассредоточена по обработчикам, совместное поведение которых обеспечивается за счет использования вспомогательных переменных (флагов).

Из-за этого логика программы становится крайне запутанной, и поэтому понять ее по тексту программы специалисту, не являющемуся разработчиком (как и самому разработчику через некоторое время), затруднительно даже при наличии комментариев любой степени подробности [2].

Выходом из этой ситуации является повышение централизации управления за счет выделения функциональных алгоритмов в отдельную часть программы, называемую системонезависимой. При этом в системозависимой части, которая в соответствии с предложенной авторами в [3] схемой организации событийных программ содержит функции обработчиков событий, входных переменных, выходных воздействий и вспомогательные функции, остается лишь незначительная часть логики, решающая локальные задачи.

В [4] было предложено реализовывать функциональные алгоритмы системой взаимосвязанных конечных автоматов. Это изменяет традиционный подход к построению обработчиков событий: они содержат только вызовы функций автоматов с передачей им номера произошедшего события и информации о нем.

Предлагаемый подход является составной частью технологии автоматного проектирования программ, важнейшей компонентой которой является документирование проекта.

При этом понятность каждого автомата обеспечивается четырьмя документами:

- словесным описанием, позволяющим понять назначение автомата;
- схемой связей, формально определяющей все входные и выходные воздействия автомата, а также, в какой автомат вложен рассматриваемый, и какие автоматы вложены в него;
- графом переходов, определяющим поведение автомата и оперирующим краткими формальными обозначениями входных и выходных воздействий, определенными на схеме связей;
- текстом функции, реализующей автомат, которая формально и изоморфно построена по графу переходов по шаблону на выбранном языке программирования.

В рамках предлагаемого подхода благодаря использованию автоматов, в которых центральным является понятие "состояние", удается отказаться от традиционно используемого термина "логика программы" и перейти к понятию "поведение программы".

Кроме того, благодаря использованию автоматного подхода для каждой комбинации входных воздействий обеспечивается возможность автоматического построения протокола работы программы в терминах автоматов, в котором указываются запуски и завершения работы автоматов при обработке очередного события, совершаемые автоматами переходы, опрашиваемые входные переменные и выполняемые выходные воздействия.

Сравнение традиционного и предлагаемого подходов на примере простой событийной программы выполнено авторами в [4]. В первом случае логика распределена по четырем обработчикам, а во втором – сосредоточена в одном автомате, вызываемом из четырех обработчиков, не содержащих функциональной логики.

Существенно более сложный пример сравнения традиционного и предлагаемого подходов был выполнен в ходе внедрения SWITCH-технологии [4] в объектно-ориентированную среду визуального программирования "Flora/C+" [5]. При этом был рассмотрен приведенный в дистрибутиве пример, моделирующий технологический процесс в цехе холодного проката.

Исходно этот пример реализован с помощью 393 объектов, из которых 51 объект являются обработчиками событий. Все эти обработчики в распределенной форме реализуют "смесь" из функциональной и второстепенной логики.

С целью повышения централизации управления функциональные алгоритмы были выделены и реализованы системой из четырех взаимодействующих автоматов. Кроме того, программа был преобразована в учебный пример, поясняющий механизм взаимодействия автоматов и позволяющий:

- изучать функционирование программы в различных режимах, в том числе и в пошаговых;
- автоматически строить различные протоколы работы программы в терминах автоматов.

После преобразования программы и введения указанных дополнительных возможностей, число объектов в программе увеличилось до 456, из которых 2 являются обработчиками событий и 51 – функциями. При этом все функциональные алгоритмы сосредоточены всего лишь в четырех функциях, реализующих автоматы и вызываемых из обработчиков событий, а второстепенная логика остается распределенной по функциям, реализующим входные переменные и выходные воздействия.

Программа и ее документация для описываемого примера и другие материалы по автоматному программированию приведены на сайте www.softcraft.ru (раздел "Программирование/Автоматные модели").

Литература

1. Бобровский С.
2. Безруков
3. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб.: Наука, 2000.
4. Шалыто А.А., Туккель Н.И. SWITCH-технология - автоматный поход к созданию программного обеспечения "реактивных" систем //Промышленные АСУ и контроллеры. 2000. N10.
5. http://www.compassplus.ru.