Статья опубликована в «Вестнике МГТУ «СТАНКИН». 2013. № 4, с. 71 – 75.

УДК 004.4’236

О.А. Большаков, А.В. Рыбаков

Автоматное моделирование технологических процессов на ПЛК *Siemens*

Automata modeling of technological processes on PLC *Siemens*

Данная статья является продолжением создания методик и примеров практического применения технологии автоматного программирования.

This article is continuation of creation of techniques and examples of practical application of technology of automata-based programming.

Ключевые слова: программное обеспечение (ПО), модель, разработка, требования, автоматное программирование.

Key words: software, model, development, requirements, automata-based programming.

Почему необходимо использовать модели. В процессе разработки программного обеспечения основной трудностью является естественная сложность предметной области, к которой относится решаемая задача. Всякий раз, когда при разработке программного обеспечения возникает необходимость автоматизировать созданные человеком сложные системы, избежать этой сложности нельзя – ею можно только «овладеть».

Для этого необходима хорошая предметно – ориентированная модель, проникающая значительно дальше поверхностного взгляда на проблему. Если в такой модели удастся правильно отразить внутреннюю структуру предметной области, разработчики программного обеспечения получат именно тот инструмент, в котором они нуждаются.

Модель предметной области служит единым языком, на котором могут разговаривать инженер – разработчик и специалист в этой области. Только посредством взаимопонимания инженера – разработчика и специалиста предметной области может быть разработан качественный программный продукт в поставленный срок. Источником этого взаимопонимания и служит модель.

Разработка ПО на основе моделей. В 2001 году группой *Object Management Group* (*OMG*) был запущен проект, который стал новым направлением в программной инженерии как «проектирование на базе моделей» (*Model-Driven Architecture*) [1]. Основной идеей этого подхода является независимое рассмотрение моделей, создаваемых при проектировании системы, от деталей их реализации на конкретной программно-аппаратной платформе. В

подходе *MDA* модели программных систем представляются с помощью «Унифицированного языка моделирования» (*Unified Modeling Language, UML*) [2].

Если в течение ряда лет этот язык использовался только для представления моделей, то в последнее время все большую популярность приобретает идея исполняемого *UML* [3]. Это связано с тем, что практическое использование *UML* в большинстве случаев ограничивается только моделированием статической части программ с помощью диаграмм классов и генерацией по ним каркаса кода программы, а моделирование динамических аспектов программ затруднено в связи с отсутствием в стандарте формального и однозначного описания правил интерпретации (операционной семантики) поведенческих диаграмм [4].

В качестве инструмента для моделирования динамических аспектов программ **предлагается использовать технологию автоматного программирования** [5]. Автоматная парадигма позволяет не только строить модели предметной области на основе унифицированного языка моделирования UML [6], но и заставляет эти модели исполняться. Таким образом, существует математический аппарат и инструменты для автоматического преобразования диаграмм в программный код [7, 8].

Автоматное моделирование ориентировано на разработку ПО для микропроцессоров, микроконтроллеров и ПЛК, но применяется для разных областей использования ПО – от клиент-серверных приложений, web-приложений, визуализаторов, мобильных систем, встроенных систем до систем высокой надежности (военные приложения, аэрокосмическая индустрия). Кроме того, автоматная парадигма универсальна. Модели можно реализовывать, используя разные парадигмы разработки ПО: процедурную, объектно-ориентированную, языки контроллеров (лестничные схемы, функциональные схемы). Ниже приведен пример разработки модели и ее реализации на ПЛК ведущего мирового вендора в области автоматизации – компании *Siemens*.

Методики реализации автоматных моделей. Основная цель данной статьи – познакомить читателя **с новой методикой применения автоматного подхода** для задач логического управления с использованием программного пакета *STEP 7*: расширяемого модуля *SIMATIC S7-HiGraph* [9].

Ранее К.В. Вавиловым были предложены две методики применения концепции автоматного программирования для контроллеров *Siemens*. Так в работе [10] он предложил подход к реализации автоматов в пакете *STEP 7* (стандартной среды для контроллеров *S7-200/300*) на языке инструкций *STL*. В языке *STL* нет оператора, аналогичного `switch` языка *C*, поэтому автор использовал аналог оператора `goto` – инструкцию `JMP` (переход на метку, выполняется по условию). Структура программы при применяемом подходе фиксирована,

что позволяет реализовать произвольный автомат. Однако существует несколько сложностей:

1. Нет программного средства для автоматического перехода от графов к языку инструкции *STL*, на котором пишется программа, загружаемая в контроллер. Автору пришлось **писать код вручную** по разработанной методике. Даже в этом случае программный код приобретает автоматную структуру, процесс программирования ускоряется, так как это уже больше механические действия, нежели используемые обычно.
2. Нельзя не учитывать ограниченность языка инструкции *STL*. Так как это низкоуровневый язык, то необходимо приложить массу усилий, чтобы придать программному коду автоматную структуру и выдержать семантику автоматной парадигмы (реализовать вложенные и вызываемые автоматы, групповые переходы, приоритеты и т. д.). К.В. Вавилов положил начало применения автоматного подхода на оборудовании *Siemens*, показав, что этот подход весьма гибок, и его можно реализовать почти на любом языке программирования для задач логического управления. Причем, чем сложнее логика управления, **тем больше проявляет свою ценность автоматная парадигма**, так как структура программы строго определена.

В работе К.В. Вавилова [11] подход к реализации автоматов в пакете *STEP 7* ничем не отличается от подхода, подробно изложенного в работе [10]. Единственным отличием является применение языка *SCL* вместо языка *STL*. Выбор языка *SCL* (заказываемого дополнительно к пакету *STEP 7*) обусловлен чисто практическим соображением – удобством работы с языком высокого уровня, так как язык *SCL* является Паскале-подобным. В отличие от языка *STL* в языке *SCL* есть оператор, аналогичный оператору `switch` языка *C*. Это оператор `CASE`.

Использование Паскале-подобного языка *SCL* позволило автору указанных работ генерировать программный код автоматически, используя case-средства, что значительно упрощает процесс программирования. Однако процесс генерации программного кода происходит за пределами среды разработки *STEP 7*. Возникает проблема интеграции сгенерированного программного кода с возможностями языка *SCL*.

Как было отмечено выше, К.В. Вавилов доработал свою методику, ориентируясь на возможности языка *SCL*, сохранив семантику автоматной парадигмы. Таким образом, полностью избавиться от механического ручного труда не удается, присутствует элемент ручного преобразования сгенерированного программного кода. Ниже будет показано, как от этой проблемы можно избавиться и писать программы в автоматном стиле, используя только среду разработки *STEP 7* с расширяемым модулем *SIMATIC S7-HiGraph*.

В качестве примера приведен программный блок «Клапан», который в данный момент отрабатывается на ОАО «Омутнинский металлургический завод». Блок в каждом цикле работы контроллера отслеживает текущее состояние клапана и рассчитывает процент открытия клапана. Поскольку у клапана отсутствует потенциометр, указывающий на процент открытия клапана, этот параметр рассчитывается по времени с использованием математической модели. Так как время открытия и закрытия клапана величина непостоянная (по ряду причин время, в течение которого клапан открывается или закрывается, всегда отличается), величина, относительно которой рассчитывается процент открытия клапана, должна быть динамичной.

Далее, говоря терминами автоматной парадигмы, представлены автоматы и описывающие логику их работы графы переходов.

Для удобства документирования и понимания схема автоматов выполнена в стороннем графическом редакторе в таком же виде, как выглядят программные блоки в программе ПЛК. До появления модуля *HiGraph* схема взаимодействующих автоматов представляла собой схему взаимодействующих программных блоков, таких как на рис. 1. Логику каждого автомата можно было реализовать по разработанным шаблонам на языках *STL*, *SCL*, как было сказано выше и описано в работах [10,11].

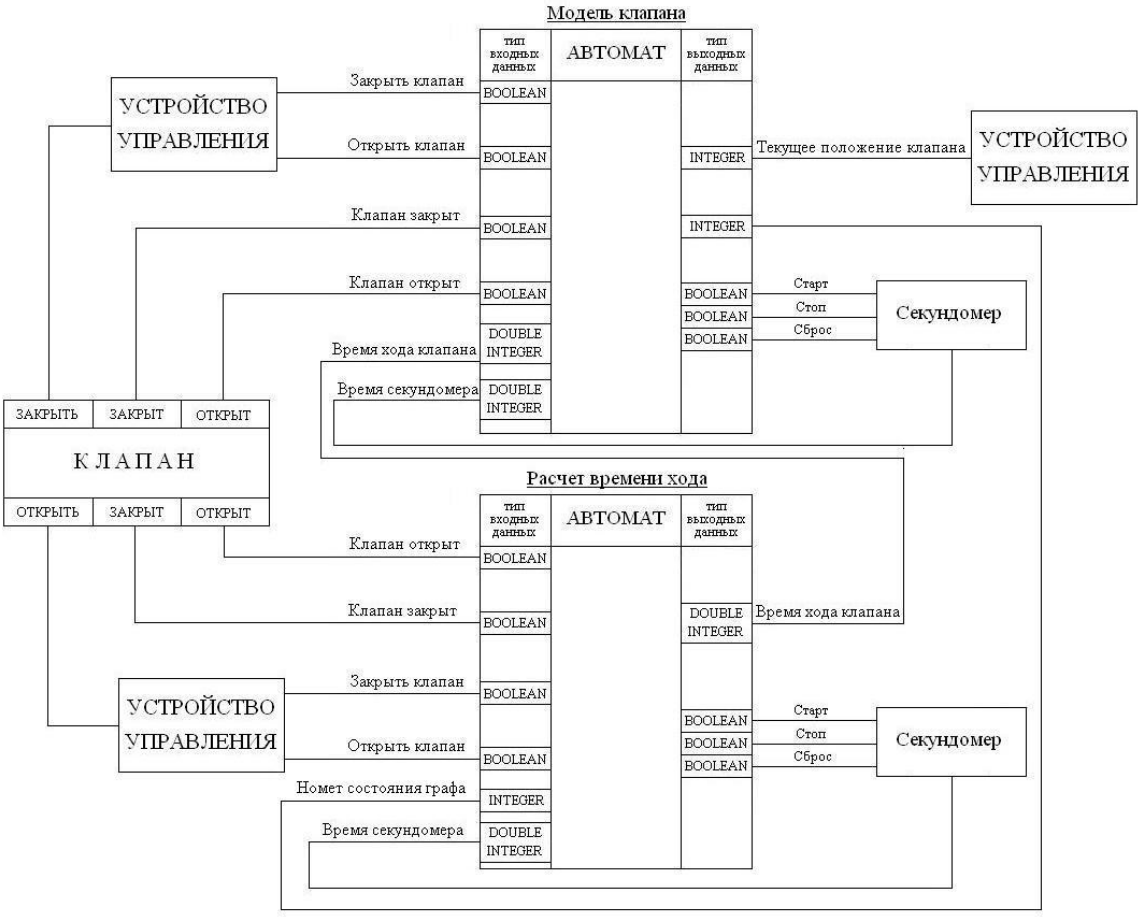

Рис. 1. Схема взаимодействия программных блоков

На данном примере можно вкратце отобразить процесс моделирования. На рис. 1 видно «Устройство управления», управляющие автоматы, объект управления «Клапан». В данном случае наличие сложной логики вызвало необходимость в разделении системы на компоненты. Сделав автоматную декомпозицию по процессам, получили два взаимодействующих автомата. Автомат «Модель клапана» имитирует поведение клапана и рассчитывает процент открытия клапана, а автомат «Расчет времени хода» – время хода клапана.

По словесному описанию поведения системы строится набор управляющих состояний (рис. 2, 3). Управляющие состояния связываются между собой переходами, добавляются входные и выходные переменные, необходимые для реализации заданного в словесном описании поведения. Так как система является событийной (поставщиком событий является клапан – «Открыт», «Закрыт»), определяется набор событий, обрабатываемых автоматом (входные переменные типа `boolean`). Они включаются в условия переходов. Верхний уровень абстракции на схеме автоматов представлен как «Устройство управления». Фактически это автоматы более высокого уровня абстракции, которые вызывают автоматы более низкого уровня абстракции рис.1.

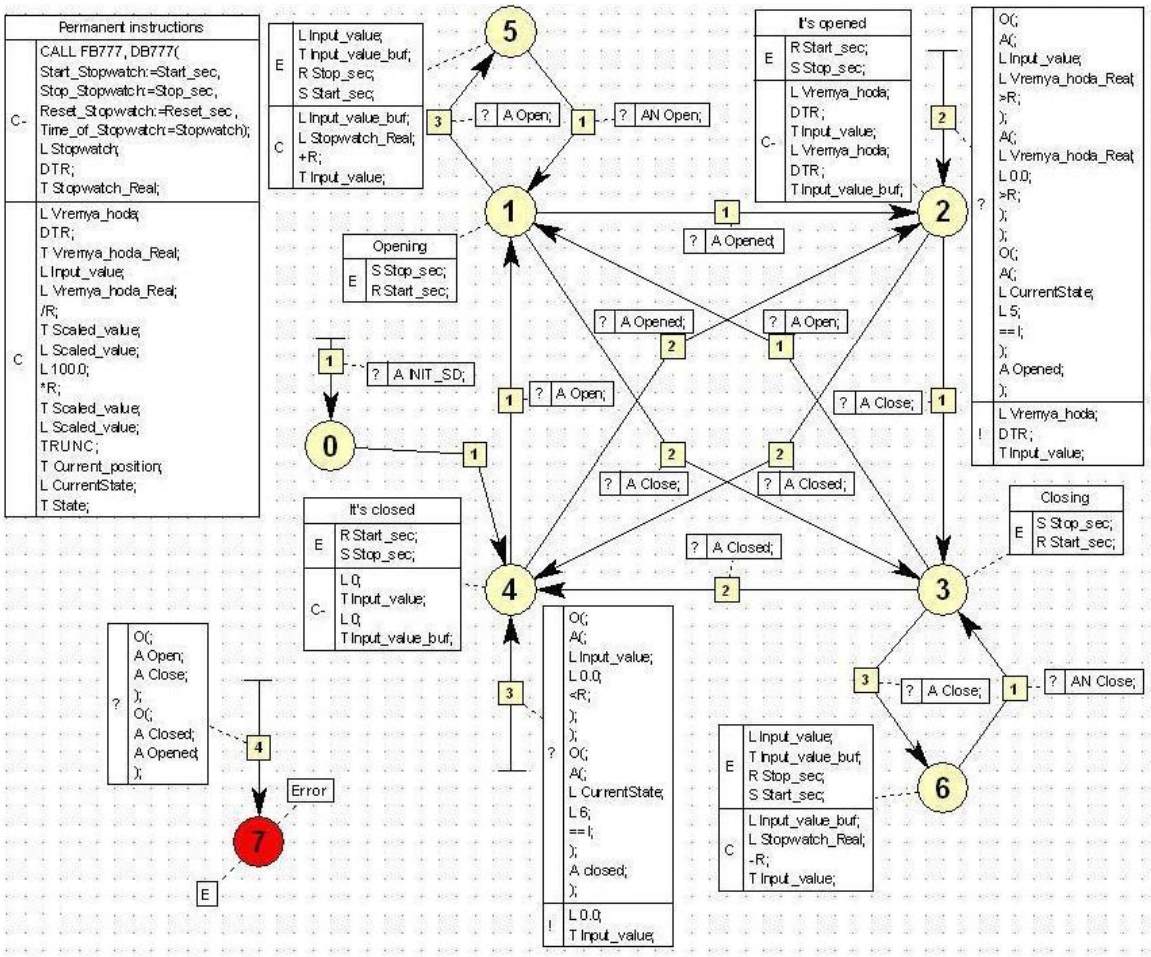

Рис. 2. Граф переходов автомата «Модель клапана»

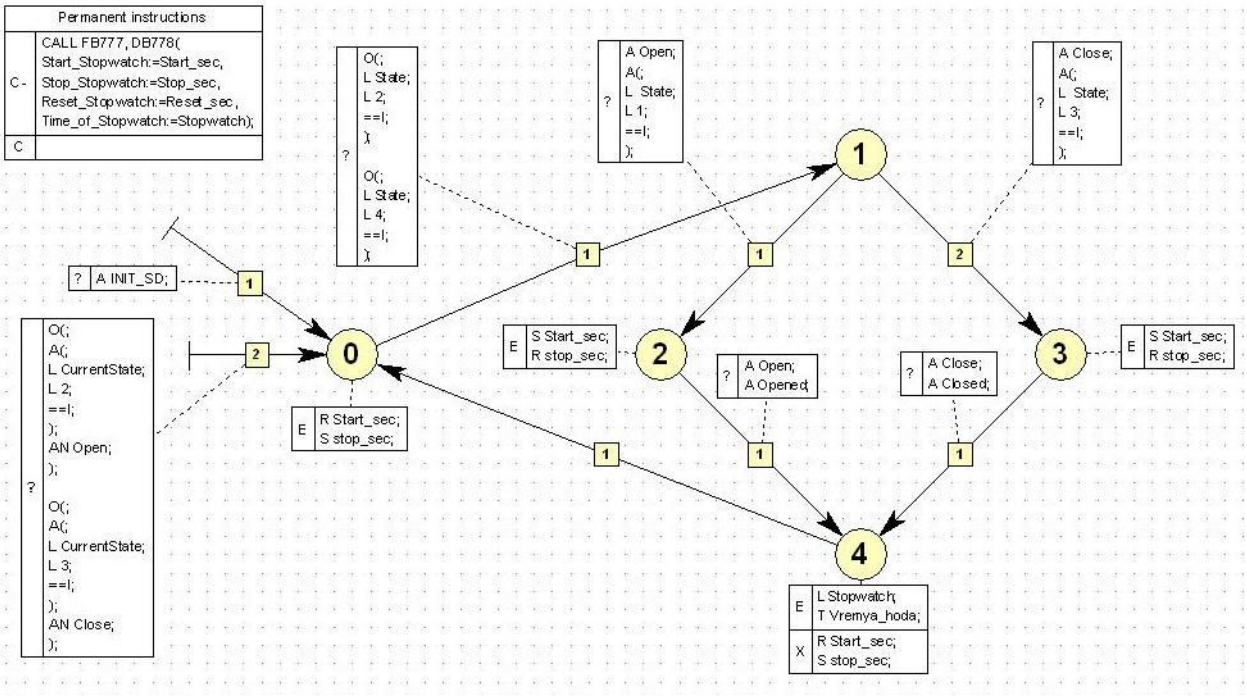

Рис. 3. Граф переходов автомата «Расчет времени хода»

Сам процесс разработки наглядно отображен на рис. 4. Специалист предметной области ставит разработчику задачу, и они вместе создают модель. В ходе разработки они видоизменяют, вносят дополнения до тех пор, пока она не будет «правильно» отображать внутреннюю структуру предметной области. Далее готовая модель на 80% автоматически и на 20% вручную преобразуется в программный код.

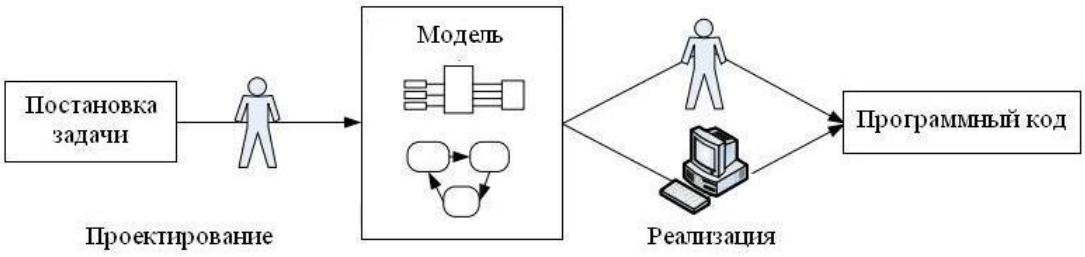

Рис. 4. Этапы разработки программной системы со сложным поведением

Модель фиксирует результат процесса проектирования. Моделями можно обмениваться с другими разработчиками, а также непосредственно использовать в качестве входных данных для следующего этапа разработки системы – реализации. Модель является математическим описанием логики поведения системы.

С появлением модуля *HiGraph* схема взаимодействующих автоматов и графы переходов полностью реализуются в графическом редакторе модуля, что избавляет программиста от необходимости использования case-средств и механической работы преобразования графов по разработанным шаблонам в программный код.

Расширяемый модуль *SIMATIC S7-HiGraph* пакета *STEP 7* полностью поддерживает семантику автоматной парадигмы. На рис. 2, 3 видно как выглядит граф в графическом редакторе *STEP 7*. Программирование действий и условий осуществляется на языке инструкции *STL*. Помимо стандартной семантики, для описания логики работы автомата в *SIMATIC S7-HiGraph* есть множество дополнительных возможностей, таких как встроенные таймеры, программирование постоянных инструкции. Вместе с приоритетами на дугах можно указать тип режима (автоматический, ручной), что позволяет в режиме реального времени видоизменять граф в заранее определенную структуру.

Приведенный в качестве примера программный блок реализован в двух автоматах и двух описывающих логику их работы графах переходов. В автомате «Модель клапана» отслеживается текущее состояние клапана и рассчитывается процент его открытия. В автомате «Расчет времени хода» рассчитывается «чистое» время хода (без остановок) клапана из состояния «открыт» в состояние «закрыт» и наоборот. Если рассчитанное время не совпадает с текущим, оно перезаписывается и передается в автомат «Модель клапана». В автомате «Модель клапана» относительно полученного значения рассчитывается степень открытия клапана.

С появлением возможности графической реализации сущностей со сложным поведением в автоматном стиле программирование ПЛК приобрело все достоинства автоматного программирования, а единая встроенная среда разработки не только ускорила процесс создания программ в автоматном стиле, лишив проблемы интеграции, но и расширила семантику автоматной парадигмы.

Заключение. Изучив работы К.В. Вавилова по созданию методик практического применения автоматного подхода на оборудовании *Siemens* [10, 11], можно заключить, что предложенный на основе модуля *SIMATIC S7-HiGraph* пакета *STEP 7* метод является **новой методикой практического применения технологии автоматного программирования**. Приведенный выше пример наглядно отображает структуру и принципы построения графов, не только полностью поддерживающих семантику автоматной парадигмы, но и расширяющих ее. Предыдущие методики не позволяют использовать автоматный подход в чистом виде в единой среде разработки, в то время как новый подход лишен этих недостатков. Кроме того, предложенная автором данной статьи методика на основе возможностей *S7-HiGraph* ставит новые задачи перед автоматным программированием, например, перестраивающиеся в зависимости от поведения среды в режиме реального времени графы, добавление новых элементов в семантику графов.

Литература

1. *OMG Model Driven Architecture*. http://www.omg.org/mda/

2. *UML – Unified Modeling Language* (www.uml.org) – универсальный язык моделирования. Методология и нотация проектирования ПО, разработанная OMG – группой (Object Management Group, www.omg.org).

3. *Mellor S., Balcer M.* Executable UML: A Foundation for Model Driven Architecture. MA: AddisonWesley, 2002. – 258 p.

4. *http://is.ifmo.ru/science/MD-Mobile.pdf*

5. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. Питер. 2011. – 165 с.

6. *Буч Г., Рамбо Г., Якобсон И.* UML. Руководство пользователя. М.: ДМК, 2000. – 358 с.

7. *Головешин А.* Использование конвертора *Visio2Switch*. http://is.ifmo.ru/progeny/visio2switch.

8. *Инструментальное средство Unimod*, http://is.ifmo.ru/unimod/.

9. *Сайт разработчика* http://www.siemens.com.

10. *Вавилов К. В.* Программируемые логические контроллеры *SIMATIC S7-200* (*SIEMENS).* Методика алгоритмизации и программирования задач логического управления. http://is.ifmo.ru/progeny/_metod065.pdf.

11. *Вавилов К. В.* Контроллеры *SIMATIC S7-300* (*SIEMENS*). Организация взаимодействия независимых локальных систем управления на основе автоматного подхода и функционального разделения автоматов управления. http://is.ifmo.ru/progeny/_s7300.pdf.

Большаков Олег Андреевич – аспирант Института конструкторско-технологической информатики Российской академии наук. Тел.: 8(917)582-63-04. OlegBolshakov@mail.ru

Рыбаков Анатолий Викторович – канд. техн. наук, доцент кафедры «Автоматизированные системы обработки информации и управления» МГТУ «СТАНКИН». Тел.: 8(916)180-82-72. avr48@rambler.ru