И.А. Баранов (ОАО «ИНЭУМ им. И.С. Брука»)

I. Baranov

МОДЕЛЬНО-ОРИЕНТИРОВАННЫЙ ПОДХОД В РАЗРАБОТКЕ ПРОГРАММНЫХ КОМПОНЕНТ ДЛЯ КОМПЛЕКСОВ СМ1820М

MODEL-DRIVEN APPROACH IN SOFTWARE COMPONENTS DEVELOPMENT FOR SM1820M SYSTEMS

Приводится анализ основных тенденций в совершенствовании методов и инструментов создания прикладных программ в области промышленной автоматизации. В качестве перспективного направления рассматривается модельно-ориентированный подход. Даётся описание основных этапов создания предметно-ориентированного языка для разработки программных компонент для комплексов СМ1820М.

This paper describes the analysis of modern approaches and tools for the development of industrial automation software. Model Driven Development is considered as an upcoming trend. The paper also describes the algorithm for Domain-Specific Language (DSL) development.

Ключевые слова: модельно-ориентированный подход, предметно-ориентированный язык, ПЛК.

Keywords: Model Driven Development, Domain-Specific Language, PLC.

Введение

Разработка современных прикладных программ для ПЛК (Программируемых Логических Контроллеров) серии СМ1820М на базе микропроцессоров «Эльбрус», SPARC [1], ARM, x86, работающих на нижнем уровне систем управления в области промышленной автоматизации, осуществляется с помощью инструментария, поддерживающего языки стандарта IEC 61131-3 (FBD, SFC, LD, ST, IL) и адаптированного под данные аппаратные

платформы. Данный подход обладает своими достоинствами и недостатками. С одной стороны, удалось достичь упрощения языков, сведения к минимуму возможности допустить ошибки и освобождения инженера-технолога от работы с низкоуровневыми элементами разработки (например, выделение и освобождение памяти, операции с указателями и ссылками и т.д.). С другой стороны, объединение в рамках одного стандарта пяти различных языков дезориентирует пользователей и затрудняет применение стандарта [2]. Но более важным недостатком можно назвать представление реализованной системы управления на низком уровне абстракции, т.к. результатом является слабо формализованная модель данной системы.

В настоящее время проводятся исследования для решения изложенных выше проблем. Некоторые производители ПЛК самостоятельно расширяют существующий стандарт IEC 61131-3, добавляя объектно-ориентированные элементы, что приветствуется инженерами-технологами [3]. Но такие изменения негативно сказываются на переносимости кода между разными инструментами, т.к. пользователь становится «привязанным» к конкретному инструменту и конкретному производителю. Развивается стандарт IEC 61499, где существенно изменён один из важнейших компонентов – функциональный блок [4]. В силу ряда причин он оказался неполным в плане определения семантики функционального блока. Одна и та же система функциональных блоков может иметь различное поведение на различных платформах, что является недопустимым [5]. Ряд работ рассматривают использование модельно-ориентированного подхода в разработке прикладных программ, как пример – адаптация UML диаграмм, языка SysML и т.д. В результате можно сделать вывод, что все современные исследования сконцентрированы на следующих аспектах:

– повышение уровня абстракции при разработке программной составляющей систем автоматизации;

– стремление найти «золотую середину» в возможностях и простоте создаваемых инструментов;

– минимизация разрыва между программистом и инженером технологом.

Анализируя данные работы и исследования, можно сделать вывод о том, что в настоящее время нет единого подхода к решению данного вопроса, и предложение различных альтернативных и новых методов и подходов является актуальным.

1. Модельно-ориентированный подход

В рамках исследования выбрано модельно-ориентированного подхода на примере создания предметно-ориентированного языка (DSL, Domain Specific Language). Использование таких языков успешно зарекомендовало себя в различных сферах разработки традиционного программного обеспечения.

Поставлена задача выработки общей методики создания предметно-ориентированного языка для формирования программных компонент, исполняемых на комплексах СМ1820М. Предлагается не изменять существующий инструмент, а сделать над ним «надстройку», позволяющую активно использовать визуальное редактирование и генерацию кода. В данном контексте под генерацией кода подразумевается генерация описания проекта для среды разработки Beremiz [1]. Схема организации такого подхода показана на рис. 1 [6]. Процесс разработки осуществляется от наиболее общих понятий принятия проектных решений с помощью представления системы управления в виде модели (Domain Specific Language Level) до генерации и сборки (компиляции) низкоуровневого кода на кроссплатформенном языке С для ПЛК (Hardware Device Level), реализующего алгоритмы, которые непосредственно будут обеспечивать выполнения всех задач автоматизации. При этом внесение изменений в алгоритмы работы данной смоделированной системы автоматизации осуществляется с помощью хорошо стандартизованной нотации технологических языков программирования стандарта IEC 61131-3 (Automation Source Code Level).

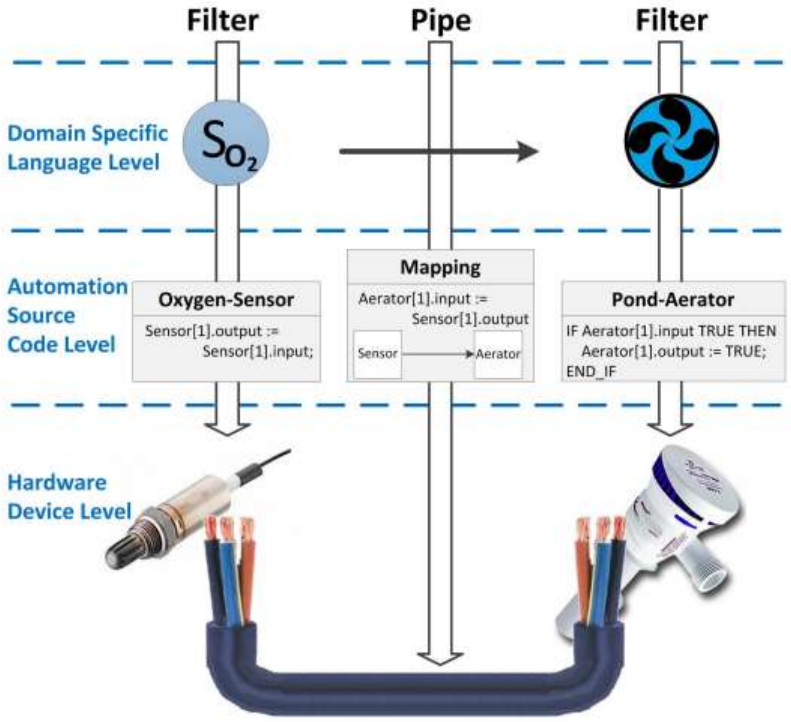

Рис. 1

Схема организации процесса разработки прикладных программ

Для дальнейшего рассмотрения модельно-ориентированного подхода и создания предметно-ориентированного языка необходимо дать некоторые определения. Модельно-ориентированный подход, или разработка, управляемая моделями (model-driven development), – это стиль разработки программного обеспечения, когда модели становятся основными артефактами разработки, из которых генерируются код и другие артефакты. Модель – это абстрактное описание программного обеспечения, которое скрывает информацию о некоторых аспектах с целью представления упрощенного описания остальных [7]. Метамодель описывает понятия, используемые в модели, и фиксирует информацию в виде метаданных, которые могут быть обработаны инструментами. Метаметамодель – это язык описания различных языков моделирования. Метаметамодели востребованы в силу разработки различных визуальных языков и необходимости стандартизации средств описания метамоделей.

В существующем инструментарии (среда разработки Beremiz) проект состоит из реализованных алгоритмов управления, пользовательских типов данных и функциональных

блоков, плагинов конфигурирования модулей ввода/вывода и интерфейсов связи с нижним и верхним уровнями и т.д. Описание всех этих компонент хранится в формате TC6 XML (рис. 2, справа), который хорошо подходит для генерации разными инструментальными средствами (рис. 2, слева) [8].

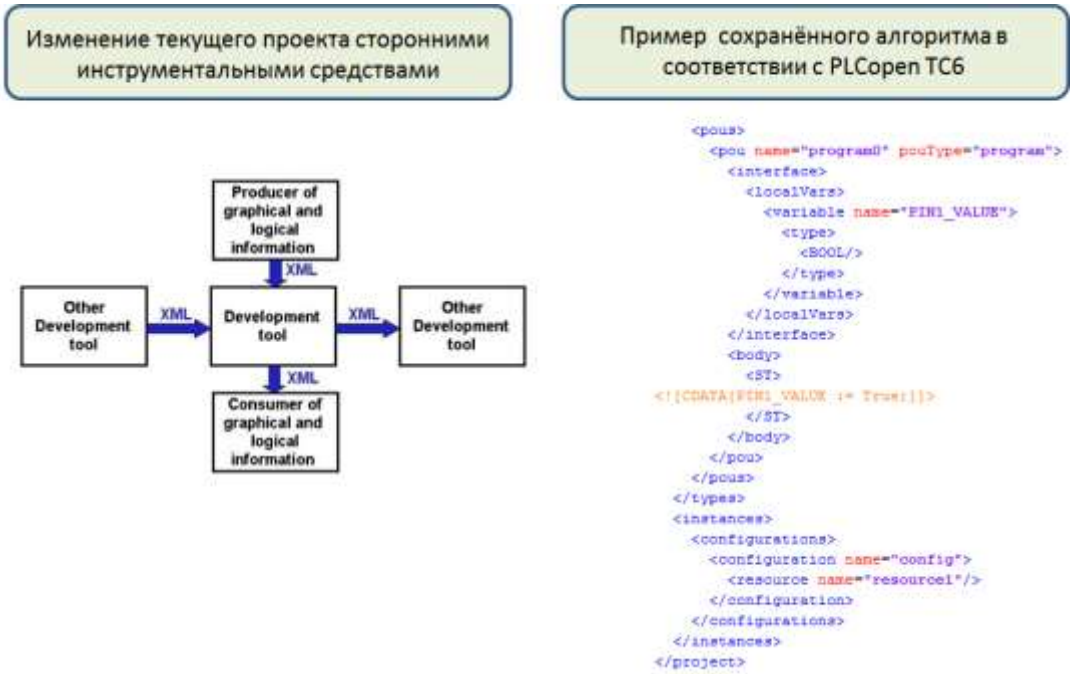

Рис. 2

Изменение текущего проекта сторонними инструментальными средствами

Следующим шагом является представление данных компонент, в частности их свойств и параметров, в виде моделей. При этом информация, описанная в модели каждого элемента, должна быть достаточной для формирования реального её представления на следующем уровне (уровне проекта в среде Beremiz).

2. Формализация требований и компонент создаваемой системы

Формализация основных требований и компонент, из которых состоит и реализуется нижний уровень системы управления, позволяет получить базовый набор элементов для создания модели данной системы и, как следствие, сгенерировать её программную составляющую. Ниже приведены примеры таких требований и компонент:

– ПЛК с определённой архитектурой, выбранные для реализации конкретной системы;

– набор модулей УСО для каждого ПЛК;

– необходимые протоколы для взаимодействия со средствами верхнего уровня (SCADA-системами, OPC-серверами и т.д.): SM1820-TCP, Modbus-TCP Slave, IEC 60870-5-104 Slave и т.д.;

– необходимые протоколы для взаимодействия со средствами нижнего уровня: Modbus-RTU Master и т.д.;

– наличие резервирования, в частности использования мажорированных систем;

– необходимость организации защиты управляющей и диагностической информации (конфигурационные скрипты пакета OpenVPN для создания туннеля типа точка-точка).

Выделив основные свойства, характеристики и функциональное назначение каждого компонента (например, скорость передачи в интерфейсе Modbus RTU Master), можно получить набор элементов (палитру моделей) с набором атрибутов (формализованных свойств компонента или требований) в предметно-ориентированном языке для моделирования системы в целом. Важным аспектом формализации является не просто определение моделей и их атрибутов, но выделение связи (отношения) между ними. Например, ПЛК «включает» модули УСО, или ПЛК «поддерживает» следующие протоколы верхнего уровня.

3. Этапы создания предметно-ориентированного языка

В настоящее время для создания предметно-ориентированных языков существует ряд бесплатных и коммерческих продуктов: Eclipse Modeling Framework, MetaEdit+, JetBrains MPS, ANTLR и т.д. Определено несколько критериев, по которым необходимо сделать выбор инструмента для данного исследования:

– возможность создавать визуальные предметно-ориентированные языки;

– удобство описания и модификации метамодели;

– гибкость в описании шаблонов генерации кода;

– бесплатность решения.

На основании анализа возможностей представленных инструментов выбран Eclipse Modeling Framework (EMF) [9], максимально отвечающий обозначенным выше критериям. Данный продукт бесплатен и распространяется под лицензией Eclipse Public License. На рис. 3 представлена обобщённая схема создания DSL в EMF.

Рис. 3

Этапы создания DSL с помощью Eclipse Modeling Framework

Первым этапом в создании DSL является описание его метамодели. Одним из вариантов её реализации в EMF является графическая нотация Ecore, которая является метаметамоделью, упомянутой выше. С её помощью определяются все компоненты метамодели DSL и связи между ними. На рис. 4 приведена часть описания данной метамодели Ecore для набора компонент, который позволяет создавать программные элементы для комплексов СМ1820М. На схеме видны связи типа «Aggregation» между ПЛК и Протоколами нижнего и верхнего уровней – 0..*, ПЛК и модулями УСО (MDV, MDVyv, MAV) – 0..*. Данный тип связи определяет, что одна модель может содержать другую.

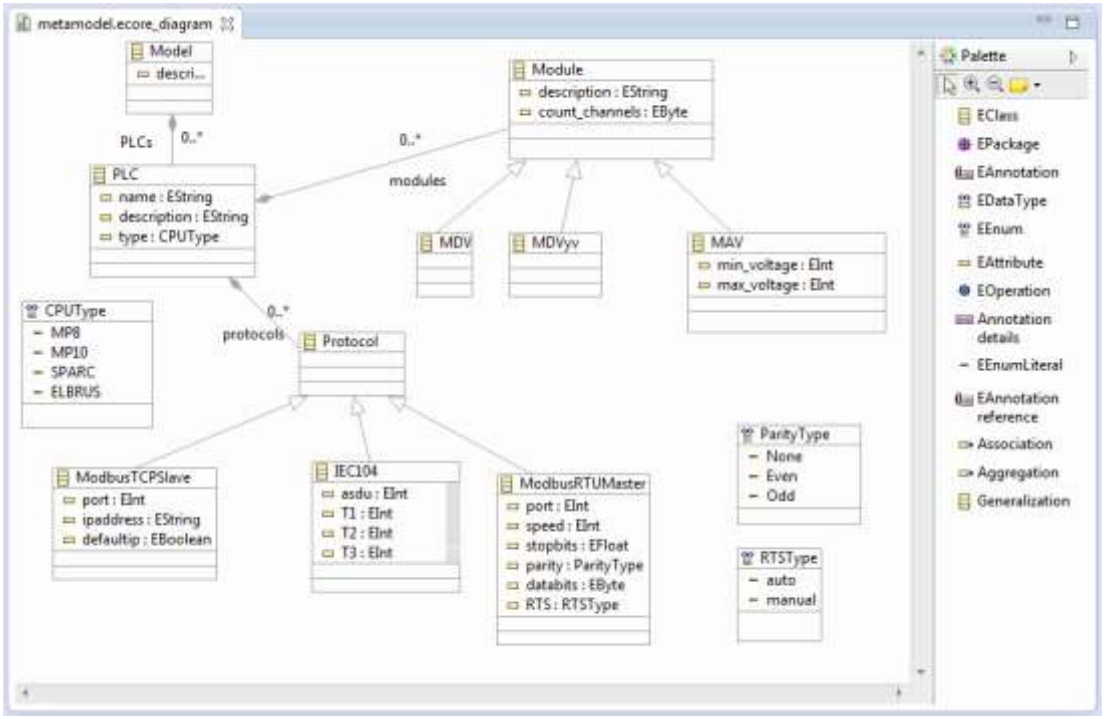

Рис. 4

Описание модели DSL языка с помощью Ecore-нотации

На втором этапе следует реализовать графическую нотацию создаваемого языка. Для этих целей в EMF существует средство Graphical Modeling Framework (GMF), которое в соответствии с существующей моделью (в данном случае описанной с помощью Ecore-нотации) генерирует графический редактор для DSL. Данная операция происходит в несколько этапов (рис. 5):

– определение графических примитивов, в т.ч. и изображений, для визуального представления компонента;

– определение элементов на палитре инструментов редактора предметно-ориентированного языка;

– создания отображения между графическими примитивами и элементами на палитре редактора.

Приведённые выше операции, как правило, выполняются с помощью соответствующих конструкторов. Организация генерации кода из модели, созданной на языке DSL, осуществляется с помощью инструмента Xpand, предоставляющего создание и редакти-

рование шаблонов, в которые подставляются данные для генерируемого XML кода. В результате сгенерированный код соответствует формату PLCopen TC6 – XML и может быть редактирован уже на более низком уровне существующими инструментами, поддерживающими стандарт IEC 61131-3. После генерации редактора средствами EMF появляется возможность конструировать программные компоненты.

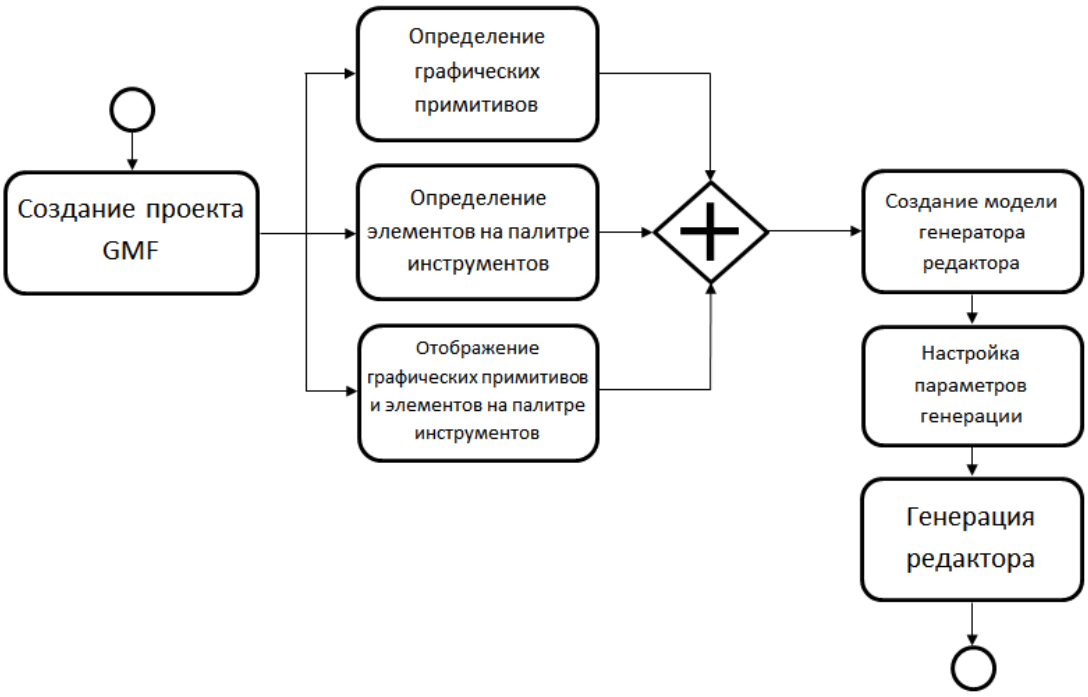

Рис. 5

Алгоритм генерации редактора DSL с помощью GMF

На рис. 6 представлен прототип редактора, где основные компоненты системы представлены с помощью геометрических фигур и определены связи между ними. После завершения формирования системы и выполнения генерации кода проект может быть открыт в среде Beremiz и изменены определённые параметры и настройки.

В целом инструмент Eclipse Modeling Framework обладает множеством возможностей и различных дополнительных компонент для расширения предметно-ориентированного языка. В ходе исследования были выявлены и некоторые недостатки. Во-первых, высокий «порог вхождения» для работы с данным инструментом из-за большого количества возможностей и инструментов. Во-вторых, полученный предметно-ориентированный язык является частью инструмента Eclipse, а не отдельным приложением.

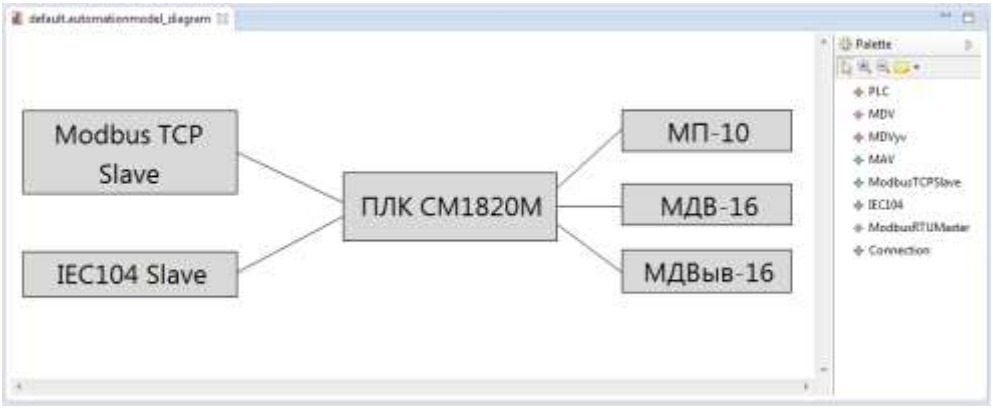

Рис. 6

Пример конфигурирования нижнего уровня системы управления с помощью сгенерированного редактора

Заключение

В статье был продемонстрирован модельно-ориентированный подход на примере разработки предметно-ориентированного языка для создания прикладных программ комплексов СМ1820М, выработана методика, основные этапы создания языка и инструментарий, необходимый для решения данной задачи. Исследование показало перспективность данного метода, в первую очередь, за счёт гибких механизмов создания метамодели и возможности её использования (расширяя и дополняя метамодели различных компонент) для решения разных задач. Планируется активное использование данного подхода в разработке программных компонент для комплексов СМ1820М с целью повышения эффективности и качества данного процесса.

Литература

1. Баранов И.А., Глухов А.В. Языки стандарта IEC-61131 для вычислительных комплексов на базе отечественных микропроцессоров с архитектурой SPARC – «Вопросы радиоэлектроники», сер. ЭВТ, 2012, вып. 3.

2. Татарчевский В.А. Проблемы применения языков стандарта IEC 61131-3 и возможные пути решения // Международная научно-техническая конференция «Информаци-

онно-математические технологии в экономике, технике и образовании». – Екатеринбург: УГТУ–УПИ, 2007, с. 239-241.

3. Хесс Д. Объектно-ориентированные расширения МЭК 61131-3 – СТА, 2006, №2, с. 90-92.

4. Thramboulidis K., Frey G. Towards a Model-Driven IEC 61131-Based Development Process in Industrial Automation, Journal of Software Engineering and Applications, 2011, 4, 217-226.

5. Дубинин В.Н. Концептуальное моделирование систем управления на основе функциональных блоков IEC 61499 – Вестник ТГТУ, 2009, т. 15, №3.

6. Preschern C., Kajtazovic N., Kreiner C. Applying Patterns to Model-Driven Development of Automation Systems: An Industrial Case Study, 17th European Conference on Pattern Languages of Programs (EuroPLoP), July 11-15, Irsee, Germany.

7. Википедия http://ru.wikipedia.org/wiki/Разработка_управляемая_моделями.

8. PLCOpen – Introduction TC6 XML http://www.plcopen.org/pages/tc6_xml/

9. www.eclipse.org/emf