УДК 681.3

ПРАКТИЧЕСКОЕ ИСПОЛЬЗОВАНИЕ МОДЕЛЬНО-ОРИЕНТИРОВАННОГО ПОДХОДА ДЛЯ РАЗРАБОТКИ ПРОГРАММНЫХ КОМПОНЕНТ КОМПЛЕКСОВ СМ1820М

Баранов И.А., аспирант МГТУ МИРЭА, E-mail: baranov_i@ineum.ru
МГТУ МИРЭА, ИНЭУМ им. И.С. Брука, Москва, Россия

Аннотация. Предлагается решение по улучшению существующего подхода к разработке программных компонент для комплексов СМ1820М. В качестве основной методики выбрано использование модельно-ориентированного похода на примере создания предметно-ориентированного языка. Представлено описание основных шагов создания предметно-ориентированного языка. Приведён пример решения практической задачи с использованием созданного инструмента.
Ключевые слова: модельно-ориентированный подход, ПЛК, промышленная автоматизация.

MODEL-DRIVEN APPROACH IN CREATING APPLIED PROGRAMS FOR SM1820M SYSTEMS

Baranov I.A., postgraduate student**,** Email: baranov_i@ineum.ru
MSTU MIREA, The Institute ECC of I.S.Bruk, Moscow, Russia

Abstract. This paper describes the solution for improving the development process of SM1820M systems. It proposes using the model-driven approach with creation of a domain specific language for solving this problem. This paper provides a description of steps for creating Domain Specific Language. It also describes the example of using created language for a real task.
Keywords: model-driven development, PLC, industrial automation.

Введение

В настоящее время в рамках совершенствования процесса разработки прикладных программ для ПЛК (Программируемых логических контроллеров) серии СМ1820М, работающих на нижнем уровне систем управления в области промышленной автоматизации, проводится ряд мероприятий по внедрению модельно-ориентированного подхода. Традиционные методы основываются на реализации конкретных алгоритмов управления и конфигурационных компонентов вручную или с использованием специализированных инструментальных средств, например, поддерживающих языки стандарта МЭК 61131-3 (FBD, SFC, LD, ST, IL).

Основным недостатком данного подхода является представление конкретного решения задачи на низком уровне абстракции, что в результате приводит к трудному восприятию всей системы управления в целом (её компонентов и взаимосвязей между ними). Также одной из современных тенденций в построении автоматизированных систем

управления является приближение интеллекта к низовому технологическому оборудованию [1], большее количество алгоритмов и логики переносится на уровень ПЛК. Следовательно, сложность программного обеспечения возрастает, и вопросы, связанные с повышением надёжности и скорости разработки, становятся еще более актуальными.

Одним из предлагаемых вариантов решения изложенных выше проблем, является использование модельно-ориентированного подхода в разработке прикладных программ и других компонентов системы управления. Выделим основные преимущества, получаемые при использовании данной методики:

- во-первых, общее представление системы на более высоком уровне абстракций, максимально приближенное к требованиям, предъявляемым заказчиком, благодаря использованию высокоуровневых моделей и соответствующих специализированных редакторов, поддерживающих работу с ними;
- во-вторых, повышение скорости разработки конкретного решения, благодаря активному использованию генераторов кода из моделей. При этом генерация кода позволяет получить не только код для программы, содержащей исполняемые управляющие алгоритмы, но и другие программные «артефакты», такие как конфигурационные файлы, визуализация системы и проектная документация. В силу растущей с каждым годом сложности программного обеспечения в области промышленной автоматизации данный аспект является важным преимуществом.
- в-третьих, снижение общих финансовых затрат на реализацию решения задачи за счёт увеличения скорости и качества выполнения проекта благодаря использованию методов, представленных выше.

На сегодняшний день одной из главных проблем модельно-ориентированного подхода в области промышленной автоматизации является малое количество публикаций с практическими примерами, связанных с данной предметной областью [2]. Подавляющее большинство работ описывают теоретические аспекты методики и основные её принципы.

Одним из ведущих сторонников модельно-ориентированного подхода в области разработки программного обеспечения является Мартин Фаулер. В его работах заметное место занимают исследования в области предметно-ориентированных языков [3]. Ему удалось структурировать и систематизировать основные паттерны, используемые для описания предметно-ориентированных языков [4].

Рассматривая область встраиваемых систем, стоит упомянуть проект mbeddr (http://mbeddr.com/), являющийся мощной платформой для программирования микро-

контроллеров, представляющий собой расширение языка Си и адаптированную среду разработки. Большое внимание уделено тестированию, верификации и поддержке создаваемых проектов.

Ряд исследований показывает пример использования языков UML и SysML в качестве средств создания программной части системы автоматизации с целью повышения уровня абстракции представления всех её компонентов. Различные типы диаграмм данных языков моделирования могут быть использованы для представления программных аспектов системы на разных уровнях (логики и поведения программ и функциональных блоков, взаимодействия между экземплярами функциональных блоков, структуры ПЛК и их взаимодействия и т.д.). Также большое значение имеет развитие стандарта МЭК 61499, основная задача которого улучшить стандарт МЭК 61131-3 в плане переносимости, конфигурируемости и т.д. Главным изменением является программный элемент – функциональный блок. 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].

Анализируя данные работы и исследования, можно сделать вывод об отсутствии единого подхода к решению вопроса максимально эффективного создания, моделирования и поддержания программных компонентов систем управления и, как следствие, об актуальности поиска новых методов и решений.

Модельно-ориентированный подход

Ярким примером использования модельно-ориентированного подхода на практике являются предметно-ориентированные языки (Domain Specific Languages) и языки моделирования (Modeling Languages). Ключевой сущностью данного подхода являются модель (главный «артефакт» процесса разработки конечного продукта), скрывающая информацию о некоторых аспектах с целью представления упрощенного описания остальных [6], и кодогенерация, основанная на трансформации моделей из одного формата представления в другой. Для реализации возможности построения модели применяется один из языков метамоделирования (графический, текстовый). Модель должна соответствовать чёткой структуре, заданной метамоделью, которая определяет абстрактный синтаксис (abstract syntax) языка моделирования [7]. Метамодель задаёт ограничения на возможные модели, которые могут быть реализованы. Как правило, языки метамоделирования основаны на аппарате объектно-ориентированного программирования и активно используют понятия классов, атрибутов, ассоциации, агрегирования и т.д.

В предметно-ориентированном языке обязательным компонентом является кон-

кретный синтаксис (concrete syntax), т.е. фактический набор сущностей, который доступен пользователю данного языка и представляет конкретные компоненты предметной области. Данный синтаксис может иметь текстовую и/или графическую форму, т.е. для одной метамодели одновременно можно определить оба данных представления [7]. В данной статье рассматривается только графический синтаксис. Автоматическая генерация кода способствуют упрощению проектирования и поддержанию больших систем, а также снижению вероятности ошибки, которая возникает при ручном выполнении данной задачи.

В целом разработка предметно-ориентированного языка, кодогенераторов и остальных компонентов, необходимых для применения модельно-ориентированной методологии, требует дополнительных усилий, которые приносят плоды в виде снижения общих затрат на разработку и поддержку нескольких систем автоматизации определённой предметной области, т.е. при активном использовании данного подхода на нескольких проектах.

Реализация предметно-ориентированного языка

Проанализировав существующие подходы к разработке прикладных программ для комплексов СМ1820М, предлагается не изменять существующий инструмент Beremiz с поддержкой языков МЭК 61131-3 [8], а выстроить над ним «надстройку», позволяющую активно использовать визуальное редактирование моделей и генерацию кода. В результате процесс разработки должен осуществляться от наиболее общих понятий принятия проектных решений (требования заказчика) с помощью представления системы управления в виде модели (Domain Specific Language Level) до генерации и сборки (компиляции) низкоуровневого кода на кроссплатформенном языке С для комплексов СМ1820М (Hardware Device Level), реализующего алгоритмы, которые непосредственно будут обеспечивать выполнение всех задач автоматизации. Внесение изменений в алгоритмы и конфигурацию данной смоделированной системы автоматизации может осуществляться с помощью хорошо стандартизованной нотации технологических языков программирования стандарта МЭК 61131-3 (Automation Source Code Level), используя среду разработки Beremiz, позволяющую также модифицировать различные конфигурационные настройки (например, параметры работы системы мажорирования). Схема предлагаемого подхода представлена на рис. 1.

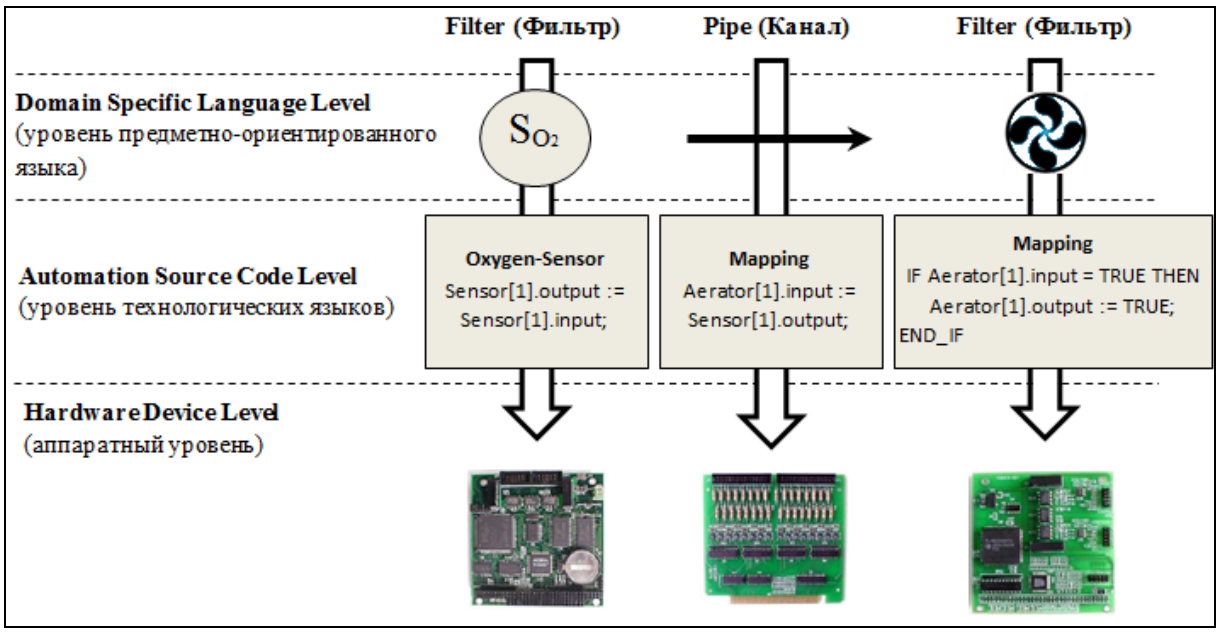

Рис. 1. **Схема уровней разработки конечного продукта с помощью предлагаемой методики**

В данном случае под генерацией кода подразумевается генерация описания проекта для среды разработки Beremiz и других программных компонентов системы управления. Структура проекта представляет собой описание в формате PLCopen TC6 – XML реализации алгоритмов управления на языках стандарта МЭК 61131-3, конфигурации и настройки модулей ввода/вывода (дискретных и аналоговых), конфигурации протокола взаимодействия с системами верхнего уровня (OPC-серверами, SCADA-системами и т.д.), конфигурации системы мажорирования (если используется) настройки процесса компиляции и т.д.

Для реализации предметно-ориентированного языка на основе проведённого анализа различных средств разработки выбрана платформа Eclipse Modeling Framework (https://www.eclipse.org/modeling/emf). Данный инструмент представляет собой свободный фреймворк, основанный на Eclipse, для генерации кода, инструментов и прочих приложений на основе структурированной модели данных (метамодели) [9]. Для реализации кодогенерации используется кодогенератор Acceleo (http://www.eclipse.org /acceleo/), интегрируемый в Eclipse Modeling Framework и обладающий возможностью трансформации моделей в текстовое представление (Model-to-text transformation) на основе специально заданных шаблонов. В качестве языка метамоделирования используется нотация Ecore, которая соответствует международному стандарту Meta Object Facility, определённому консорциумом (рабочей группой) Object Management Group, занимающимся разработкой и продвижением объектно-ориентированных технологий и стандартов. Для реализации возможности графического редактирования элементов вы-

бран фреймворк Graphical Modeling Project (http://www.eclipse.org/modeling/gmp/), который также интегрирован с Eclipse Modeling Framework.

Далее рассмотрим основные шаги по созданию предметно-ориентированного языка. Для упрощения изложения материала выделим только часть наиболее значимых компонентов нижнего и верхнего уровней системы управления.

На первом этапе создания предметно-ориентированного языка произведены систематизация, классификация и формализация основных программных компонентов, используемых инженером по автоматизации для реализации программной части, т.е. фактический набор программных компонентов, доступных в среде разработки. ПЛК серии СМ1820М представляет собой комплекс с процессорным модулем (в данном случае МП-10), модулями ввода/вывода (аналоговые, дискретные), поддержкой протоколов связи с системами верхнего уровня и т.д.

На втором этапе производится построение метамодели с помощью нотации Ecore, как показано на рис. 2. Сначала реализуются сущности в виде классов, которые представляют каждый компонент системы управления и его свойства. Каждое свойство обладает своим именем и типом, например, базовый адрес модулей ввода/вывода (класс Module) имеет тип EShort. Для отображения связи между соответствующими классами используются определённые типы отношений: агрегация (Aggregation), ассоциация (Association), наследование (Generalization). С помощью агрегации показано, что все компоненты принадлежат сущности AutomationSystem, представляющей в целом всю систему управления, и она будет являться «корневым» компонентом будущего редактора. С помощью отношения ассоциации представлен механизм взаимодействия нижнего и верхнего уровней системы управления (классы MP10 и OPCServer соответственно) через некоторый протокол (класс Protocol). Классы Protocol и SM1820MProtocol имеют отношение наследования, показывающее, что взаимодействие между нижним и верхним уровнями может осуществляться через разные протоколы (например, Modbus TCP, МЭК 60870-5 104 и т.д.). В случае необходимости, для добавления поддержки дополнительных протоколов, достаточно унаследовать новый класс от класса Protocol и указать свойства, специфичные для нового протокола.

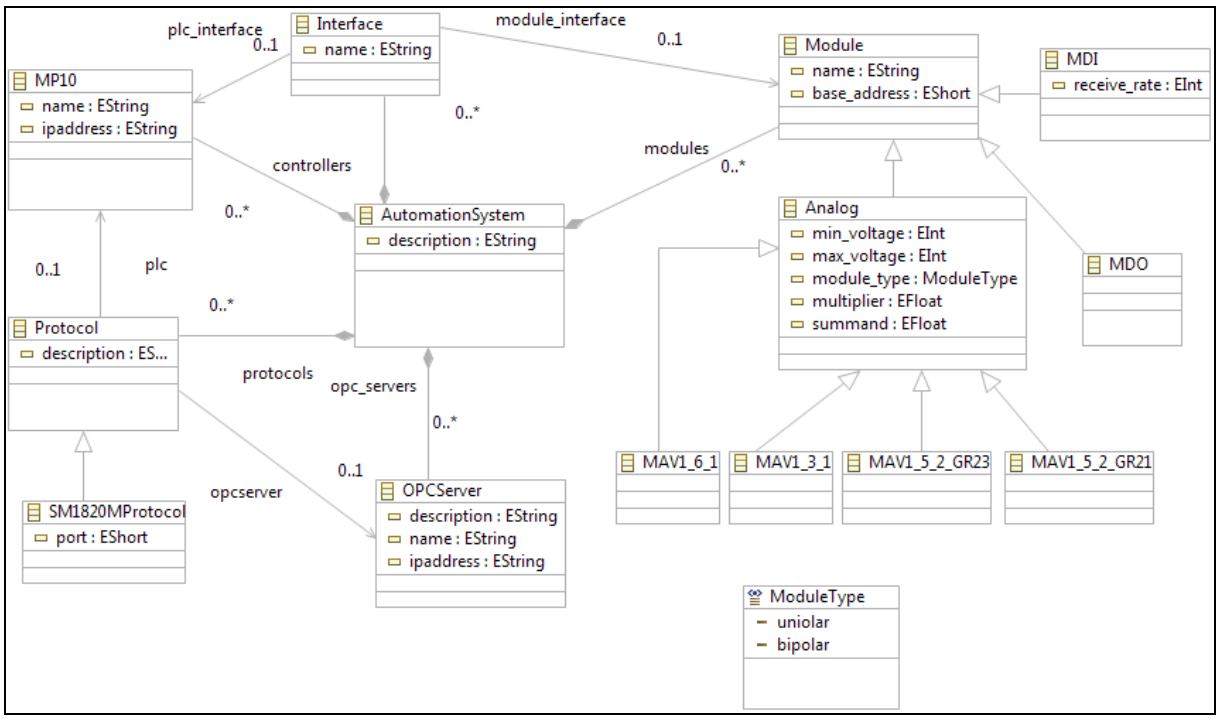

Рис. 2. **Описание модели предметно-ориентированного языка с помощью Ecore нотации**

Следующим этапом после реализации метамодели является создание редактора. Фреймворк GMF в соответствии с существующей метамоделью (в данном случае описанной с помощью Ecore нотации) генерирует графический редактор предметно-ориентированного языка и предоставляет среду выполнения для данного редактора (runtime infrastructure). Процесс создания редактора происходит в несколько этапов (рис. 3):

– 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.

Каждая формализованная ранее сущность имеет своё графическое представление, соответствующий компонент на палитре инструментов. Далее производится отображение элемента метамодели на эти две составляющие. В том числе свойства, описанные для определённой сущности, могут быть отображены на некоторый графический примитив (например, свойство «name» (имя) процессорного модуля выводится на диаграмму с помощью примитива Label, отображающего текст). В результате, например, сущности MP10, OPCServer, MAV1_6_1 и т.д. будут представлять собой узлы (Node) на

диаграмме, а Protocol и все унаследованные от него сущности – соединения (Connection). Тем самым инженер-технолог в редакторе показывает реальные взаимосвязи между компонентами системы. После проведения соответствующих настроек и всех шагов отображения примитивов создаётся модель генератора редактора и производится его генерация.

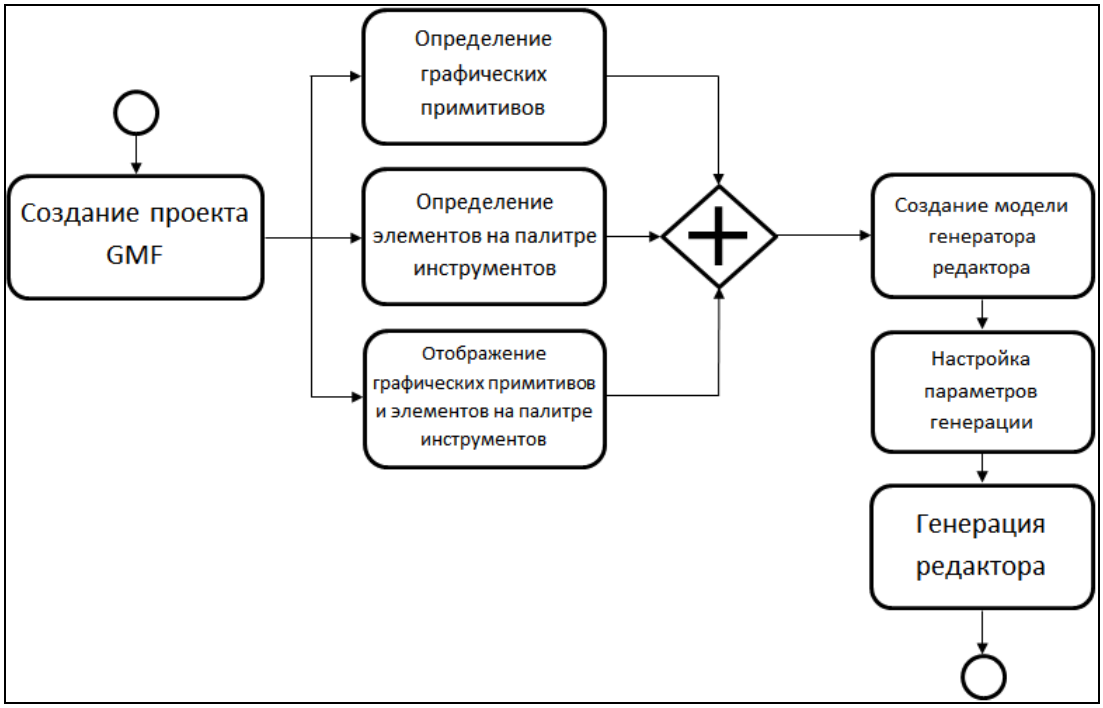

Рис. 3. **Алгоритм генерации редактора предметно-ориентированного языка с помощью GMF**

После того как произведена генерация редактора, можно выполнить его запуск в режиме исполнения и добавить механизм кодогенерации с помощью инструмента Acceleo. Генерация кода основывается на шаблонах со специальным синтаксисом, позволяющим описывать создание файлов с данными, формируемыми статически и динамически. Динамические участки создаются на основе данных из метамодели и модели. Все значения свойств сущностей, которые задаются в редакторе инженером технологом, будут отображены в код. Упрощённый пример описания шаблона приведён на рис. 4.

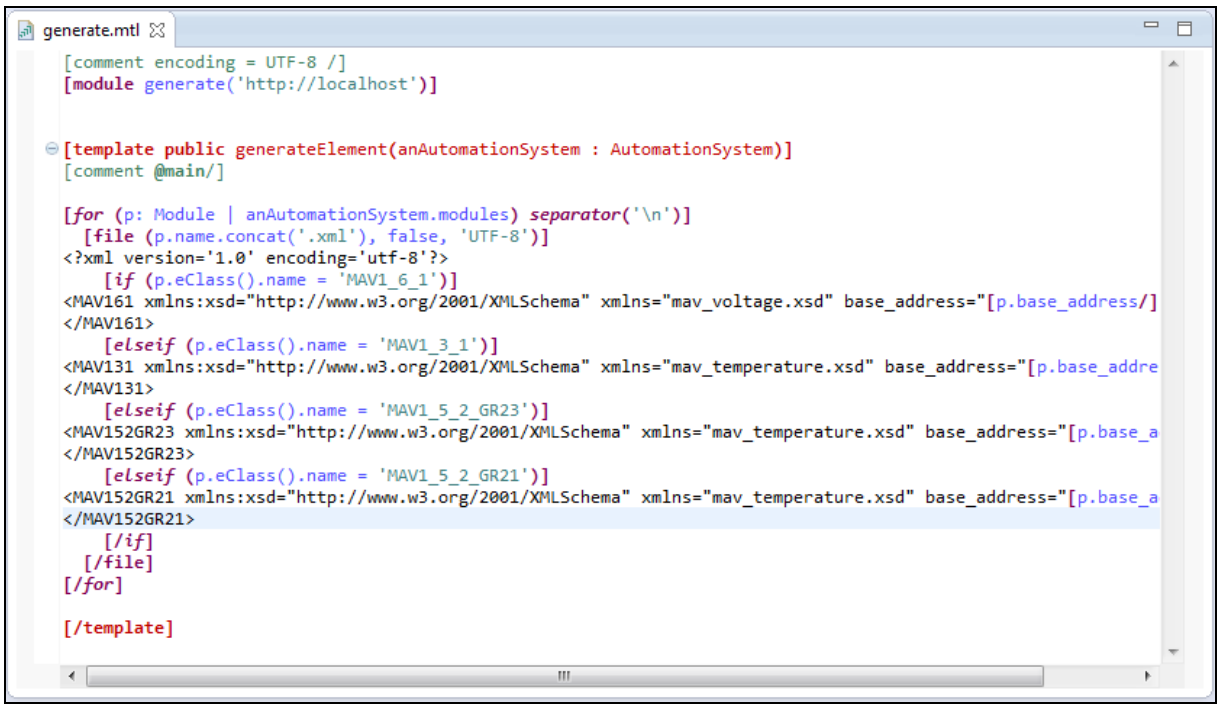

Рис. 4. **Пример описания шаблона для Acceleo**

Синтаксис языка, встроенного в Acceleo для формирования шаблонов генерации кода, позволяет использовать различные управляющие конструкции высокоуровневых языков программирования: циклы, условия и т.д. После формирования всех необходимых шаблонов процесс создания предметно-ориентированного языка, в рамках рассмотрения его в данной статье, можно считать законченным. Далее представлено решение конкретной практической задачи с использованием полученного инструмента.

Пример реализации задачи

Программно-аппаратные комплексы СМ1820М применяются в системе контроля технологических параметров экспериментального реактора на быстрых нейтронах. Модернизация данной системы управления и мониторинга затрагивает и программную составляющую, в т.ч. подходы в разработке. Для создания программных компонентов используется предлагаемый модельно-ориентированный подход. С помощью созданного ранее редактора и генераторов кода строится модель данной системы и генерируется соответствующий код.

На нижнем уровне располагаются КП (Контролирующие пункты), представленные в виде ПЛК СМ1820М, на верхнем уровне – OPC-сервера, которые необходимы для связи со SCADA-системой. Взаимодействие между ними осуществляется с помощью TCP протокола СМ1820М. Требуемая система управления реализуется с помощью полученного редактора и кодогенератора (рис. 5).

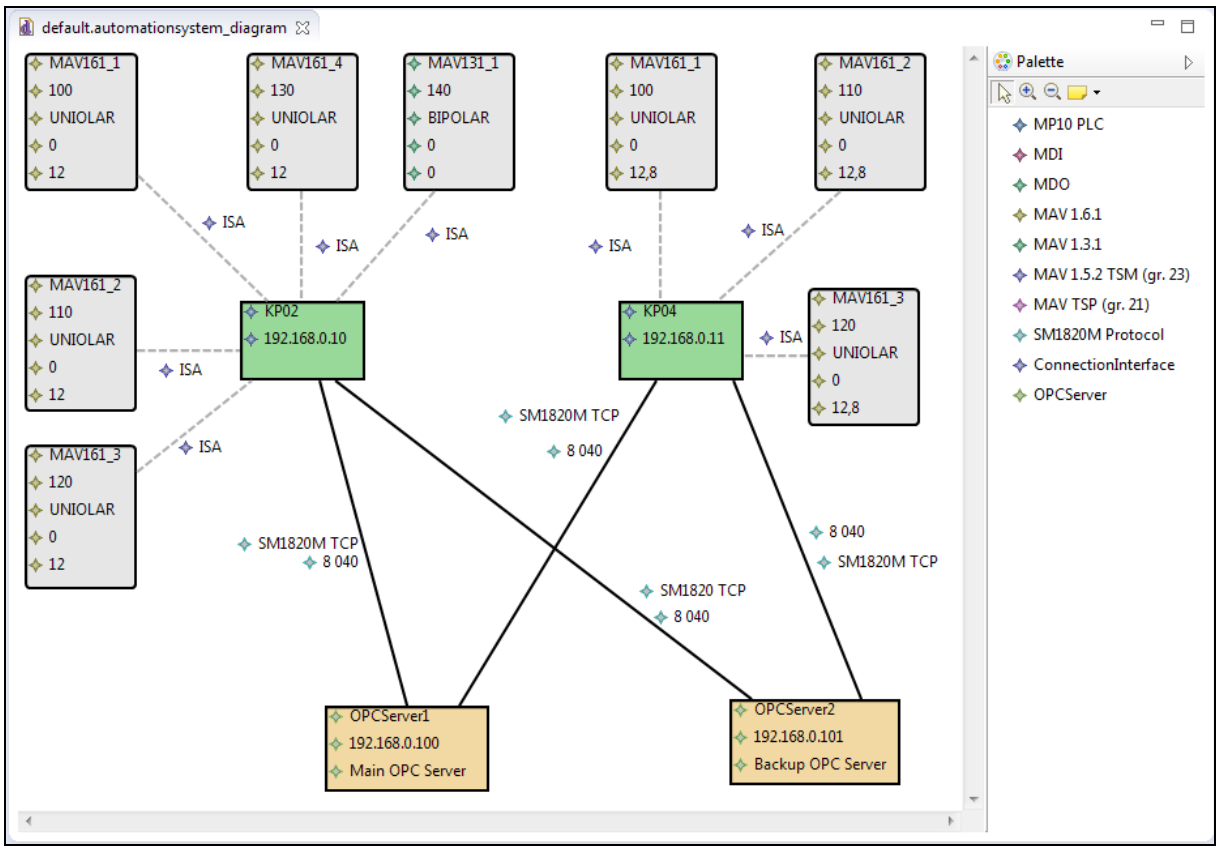

Рис. 5. **Процесс создания диаграммы программных компонент с помощью редактора**

На диаграмме показаны комплексы СМ1820М, представленные в виде процессорных модулей с именами KP02 и KP04, модули ввода/вывода для каждого из них (MAV161_1, MAV131_1 и т.д.), OPC-сервера и протокол взаимодействия SM1820MProtocol между СМ1820М и OPC-серверами. После того как система реализована, т.е. добавлены все необходимые компоненты и заданы соответствующие свойства инженером технологом, можно приступить к генерации всех программных компонентов.

Сгенерированный код для среды разработки Beremiz (рис. 6) соответствует формату PLCopen TC6 – XML. Приведено описание модуля аналогового ввода MAB 1.6.1, содержащее его общие настройки и настройки каждого отдельного канала. Данные сгенерированные проекты могут быть открыты в среде разработки Beremiz и отредактированы в соответствии с заданными требованиями. Сгенерированные конфигурационные файлы для OPC-серверов используются на соответствующих вычислительных комплексах, обеспечивающих исполнение верхнего уровня системы управления.

```
<?xml version='1.0' encoding='utf-8'?>
<MAV161 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="mav1_6_1.xsd" valuepresentationtype=
  <module_condition indexnumber="0" name="module_condition" type="DINT" class="output"/>
  <signals>
    <signal number="0">
      <value indexnumber="1" name="mav1_6_1_value_1" type="UINT"/>
      <condition indexnumber="2" name="mav1_6_1_value_1_condition" type="DINT" class="output"/>
    </signal>
    <signal number="1">
      <value indexnumber="3" name="mav1_6_1_value_2" type="UINT"/>
      <condition indexnumber="4" name="mav1_6_1_value_2_condition" type="DINT" class="output"/>
    </signal>
    <signal number="2">
      <value indexnumber="5" name="mav1_6_1_value_3" type="UINT"/>
      <condition indexnumber="6" name="mav1_6_1_value_3_condition" type="DINT" class="output"/>
    </signal>
    <signal number="3">
      <value indexnumber="7" name="mav1_6_1_value_4" type="UINT"/>
      <condition indexnumber="8" name="mav1_6_1_value_4_condition" type="DINT" class="output"/>
    </signal>
```

Рис. 6. **Сгенерированный XML код для модуля MAB 1.6.1**

Важно отметить, что все программные компоненты генерируются из одной модели. Тем самым сокращаются риски несогласованности конфигурационных данных между серверной (настройки протокола SM1820MProtocol) и клиентской стороной (настройки тегов OPC-сервера).

Заключение

Результатом работы является выработанная методика формализации основных программных компонентов, из которых создаётся система управления, реализация на их основе предметно-ориентированного языка и его специализированного редактора. В итоге можно сделать вывод, что процесс проектирования заметно ускорился благодаря использованию кодогенератора для создания проекта. Полученное решение отображает всю систему на более высоком и понятном уровне абстракции для инженера-технолога.

Список литературы

1. Управляющие вычислительные комплексы для промышленной автоматизации : учеб. пособие / [Н. Л. Прохоров и др.]; под ред. Н. Л. Прохорова, В. В. Сюзева. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2012. – 372 с.

2. 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, Irsee, Germany, 2012.

3. M. Fowler Domain-Specific Languages, 1 edition, Addison-Wesley Professional, 2010. – 640 pp.

4. Список паттернов предметно-ориентированных языков [Электронный ресурс] – URL: http://martinfowler.com/dslCatalog/ (дата обращения 26.04.2014).

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

6. Разработка, управляемая моделями – Википедия [Электронный ресурс] – URL: http://ru.wikipedia.org/wiki/Разработка_управляемая_моделями (дата обращения 28.04.2014).

7. M. Brambilla, J. Cabot, M. Wimmer – Model-Driven Software Engineering in Practice, 2012. – 182 pp.

8. 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.

9. Eclipse Modeling Framework – Википедия [Электронный ресурс] – URL: http://ru.wikipedia.org/wiki/Eclipse_Modeling_Framework (дата обращения 26.04.2014).