Автоматное моделирование систем автоматизации с реализацией на ПЛК

О.А. Большаков (Институт конструкторско-технологической информатики РАН),
А.В. Рыбаков (МГТУ «СТАНКИН»)

В статье на примере конкретной технической задачи рассмотрена методика автоматного моделирования систем автоматизации, реализуемых на базе ПЛК. Для проектирования и разработки ПО предложено использовать концепцию модельно–ориентированного подхода с применением автоматного программирования. Для автоматического преобразования автоматных моделей в программный код предлагается использование генератора MetaAuto, доработанного для возможности получения исходного кода на языке структурного текста.

Ключевые слова: модельно–ориентированный подход, алгоритм, автоматное программирование, программируемый логический контроллер, технологический процесс.

Традиционный подход к разработке программного обеспечения для ПЛК

В настоящее время ведущие мировые вендоры в области автоматизации, такие как Siemens (контроллеры серии S-300, S-400), Schneider Electric (контроллеры серии Modicon M340, Premium, Quantum), Rockwell Automation (контроллеры серии MicroLogix, SLC500 и др.) используют для программирования ПЛК специализированные среды разработки. У компании Siemens – это программный пакет SIMATIC STEP 7, для контроллеров компании Schneider Electric используются программные пакеты Unity Pro XL и Concept XL, для программирования ПЛК концерна Rockwell Automation используется среда программирования RSLogix. Все эти программные пакеты объединяет поддержка стандарта 61131-3 международной электротехнической комиссии [1].

Данный стандарт описывает пять языков программирования и фактически определяет возможности построения программ для ПЛК. Автоматизация достигается за счет локализации задач путем создания пользовательских функциональных блоков с их многократным использованием по принципу «тип – экземпляр». Это традиционный подход, при котором программист на основе вышеперечисленных программных сред разрабатывает ПО согласно техническому заданию на пяти доступных языках программирования. При работе над небольшими проектами такого подхода к организации процесса разработки программного продукта вполне достаточно. Однако при больших и сложных проектах, длящихся не один год и построенных на ПЛК разных производителей, этих возможностей уже недостаточно для эффективной разработки ПО (под термином «эффективность разработки ПО» понимается отношение показателей качества программного продукта к затраченному времени и стоимости его разработки). С увеличением топологии аппаратной части проекта (число используемых контроллеров, устройств управления, устройств сбора данных ТП) и возрастанием сложности логики управления эффективность традиционного подхода значительно снижается. Затрачивается неоправданно много времени на проектирование, реализацию и интеграцию ПО. Основные причины, по которым это происходит.

1. Сложность предметной области. Заложенные в техническое задание требования к системе программист понимает и читает по-своему. Причина в том, что текст имеет тенденцию быть нечетким, неполным или вводящим в заблуждение. Поэтому, при переходе проекта с этапа на этап в цикле разработки ПО, возникают барьеры понимания. Из-за неточной передачи информации в программном коде появляются ошибки.

2. Из рис. 1 видно, что чем больше времени тратится на проект, тем выше цена исправления ошибок. При проверке на соответствие программного продукта предопределенным требованиям возникает необходимость в исправлениях и внесении изменений. Этот процесс очень трудоемок и занимает много времени. Вероятность допустить ошибку очень высока, так как программный код исправляется вручную и часто не автором программного продукта.

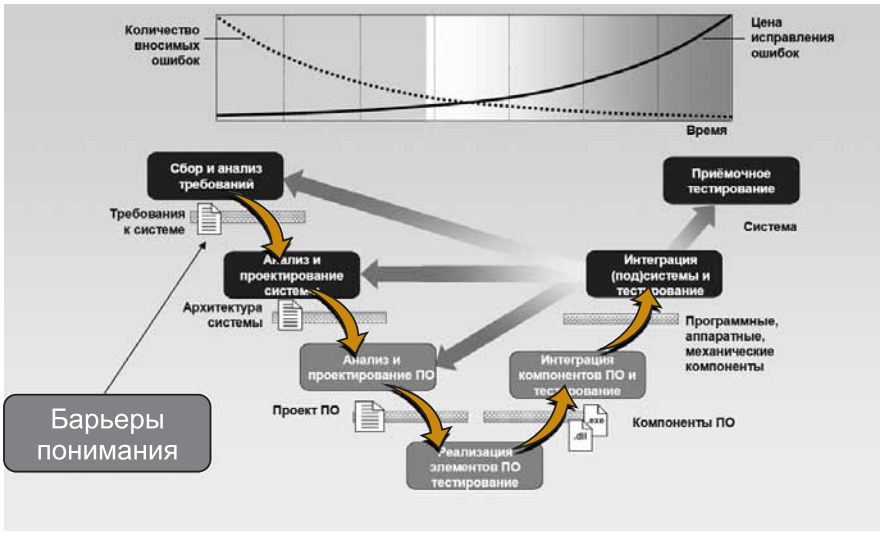


Рис. 1. Как работает традиционный метод разработки ПО на основе документации

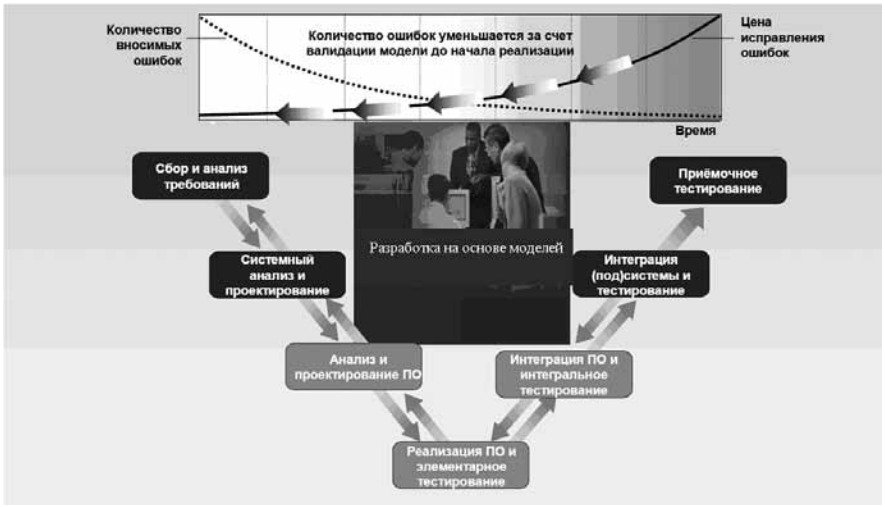


Рис. 2. Разработка ПО на основе моделей

3. При составлении документации на программный продукт часто используются словесное описание поведения системы, блок-схемы алгоритмов управления, не привязанные к программному коду модели. Информация дублируется в различных документах и нотациях, является противоречивой и нецелостной.

Также наиболее сложными и трудоемкими процессами при создании программ для ПЛК являются декомпозиция задачи на программные блоки, корректное преобразование алгоритмов программных блоков и их межагрегатных связей на языки ПЛК. Любой инструмент, позволяющий автоматизировать (в лучшем случае) или хотя бы оптимизировать эту работу, будет полезен для практического проектирования АСУТП.

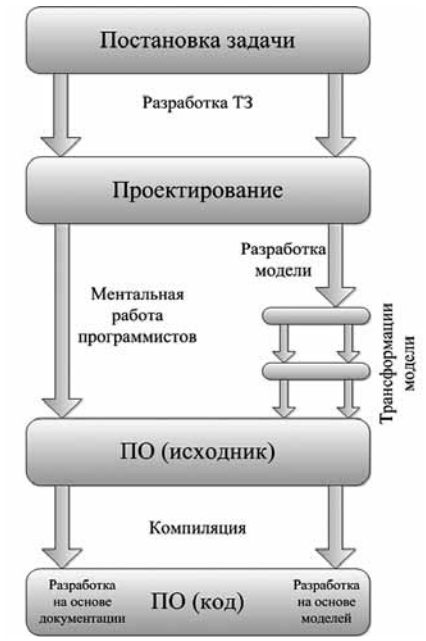


Рис. 3. Сравнительный вид методов разработки ПО

Необходимость повышения качества программного продукта, сокращения сроков и стоимости разработки ПО привело авторов данной статьи к созданию новой методики, охватывающей весь жизненный цикл программного продукта: от проектирования до документирования. Также было разработано программное решение, позволяющее автоматически создавать программный код с заранее определенной структурой. Это значительно облегчило разработку ПО и упростило процесс внесения изменений в программный продукт.

Формулировка предлагаемого метода

Предлагается проектировать основанные на ПЛК системы автоматизации, используя идеологию разработки ПО на основе моделей. Идея модельно-ориентированной разработки ПО принадлежит группе Object Management Group (OMG), запустившей проект Model Driven Architecture (MDA, http://www.omg.org/mda/). Этот проект стал новым направлением в программной инженерии. Основной идеей этого проекта является независимое рассмотрение моделей, создаваемых при проектировании системы, от деталей их реализации на конкретной программно- аппаратной платформе. Эта идея позволяет создать связь между этапами жизненного цикла ПО. Модель не является чем-то специфическим, что может понять только программист. Модель в данном случае – это абстрактное отражение ТП. Модель проста в понимании и редактировании для всех участников проекта жизненного цикла ПО. На рис. 2 видно, что применение моделей неразрывно связывает все этапы разработки ПО: от сбора и анализа требований до приемочного тестирования.

Однако описанный выше проект не ориентирован на разработку ПО для промышленных контроллеров. Это связано с тем, что в подходе MDA нет формального и однозначного описания правил интерпретации (операционной семантики) поведенческих диаграмм для моделирования сложной логики управления ТП. Поэтому в качестве моделей предлагается использовать диаграммы из нотации Switch-технологии (технология автоматного программирования) [2]. Switch-технология ориентирована на разработку ПО для микропроцессоров, микроконтроллеров и ПЛК; применяется для разных областей использования ПО: клиент- серверных приложений, Web-приложений, визуализаторов, мобильных систем, встроенных систем, систем высокой надежности (военные приложения, аэрокосмическая индустрия).

Автоматные модели строятся на основе унифицированного языка моделирования UML [3] и реализуются в разных парадигмах разработки ПО: процедурная, объектно-ориентированная, языки контроллеров (лестничные схемы, функциональные схемы). Для автоматического преобразования диаграмм в программный код существует математический аппарат и инструменты [4]. Но для автоматической генерации программного кода по автоматным моделям на язык структурного текста в настоящий момент не разра-

ботано специального инструментального средства. Для решения этой задачи авторами доработано инструментальное средство MetaAuto [5], так как на сегодняшний день это единственный генератор автоматных моделей с открытым исходным кодом. Язык структурного текста был выбран как единственный высокоуровневый язык в стандарте IEC 61131-3.

Идея использовать модельно-ориентированную идеологию разработки ПО с применением технологии автоматного программирования вызвана тем, что ни один из графических языков стандарта 61131-3 не подходит для описания сложной логики управления и математических функции в рамках одного функционального блока. А зачастую при написании сложных программ управления интеллектуальными системами задача состоит именно в этом. Для решения этой задачи был выбран язык структурного текста, как наиболее удобный для подобного рода задач, а в качестве его графического представления – автоматные модели. Очевидно, что только графический язык может выступать в роли модели, наглядно и целостно отражающей информацию об управлении технологическим процессом.

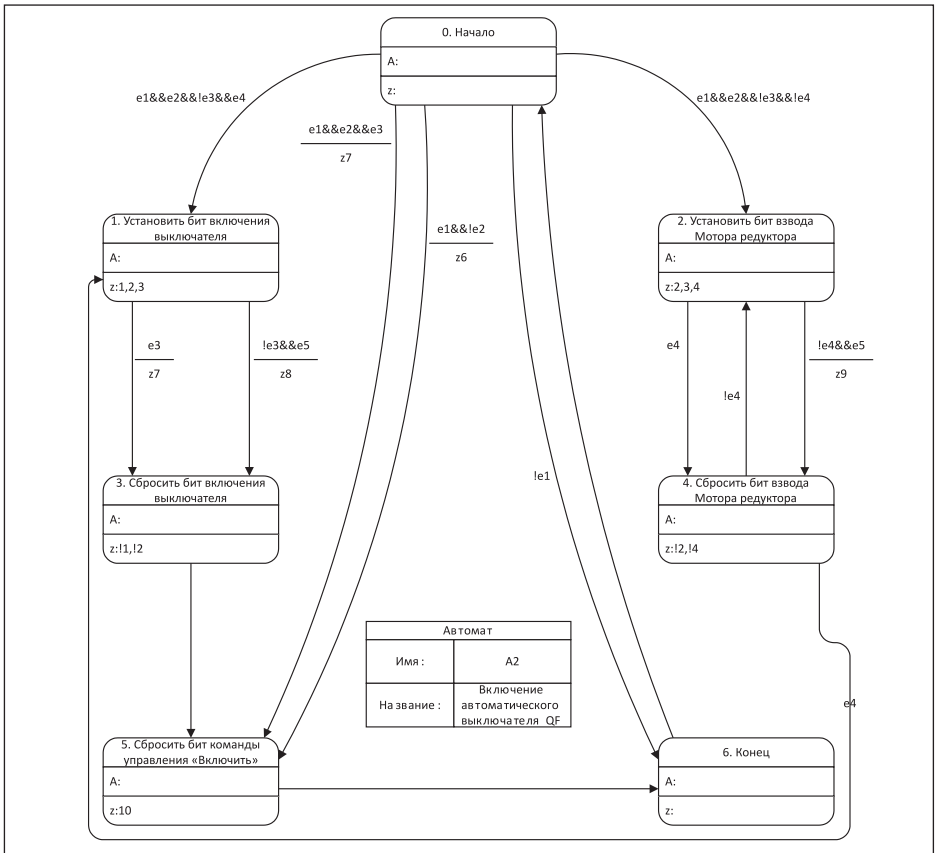
Еще одна причина разработки новой методики – отсутствие возможности динамического структурирования больших проектов, то есть графическое отображение логики управления ТП не привязано к программному коду. При внесении изменений в проект приходится редактировать вручную различные нотации и программный код, затем проверять всю документацию на предмет отсутствия рассогласования и дублирования информации. В связи с этим возникает необходимость автоматически редактировать программный код при изменении модели.

На рис. 3 представлены основные отличия методов разработки ПО, рассматриваемых в данной статье. При разработке ПО на основе моделей основная часть программирования автоматизированных систем – это визуальная разработка логики управления и математических функции.

Автоматные модели и ПЛК

Рассмотрим приложение автоматной модели к программированию ПЛК. В качестве примера на рис. 4 представлена автоматная модель алгоритма включения автоматического выключателя QF подсистемы управления комплектной трансформаторной подстанцией собственных нужд (КТП СН) и распределительным устройством (РУ 0,4 кВ). Данный пример взят из реального проекта компании Schneider Electric. Проект требовал быстрой подготовки и разработки технического решения с реализацией на контроллерах Modicon M340. Наличие сложной логики управления вызвало необходимость в структурировании программного кода и моделировании логики управления подсистемы, так как проект реализовывался несколькими программистами, у каждого из которых свой стиль конструирования программ и свое понимание предметной области. Чтобы проект имел единую структуру ПО и точно соответствующие программному коду модели предложено использовать разработанную методику автоматного моделирования. Ниже на примере автоматического выключателя QF описан процесс создания автоматной модели.

После автоматной декомпозиции алгоритм включения



e1	QF1_CMD_ON
e2	QF1_EN_ON
e3	QF1_ON
e4	QF1_ST
e5	T_imp_fin

z1	QF1_S_ON
z2	T_imp_pusk
z3	T_ust
z4	QF1_S_ST

z6	Формирование сообщения оператору –«включение блокировано»
z7	Формирование сообщения оператору –«команда выполнена»
z8	Формирование сообщения оператору –«команда не выполнена»
z9	Формирование сообщения оператору –«автомат не взводится»
z10	Сбросить бит команды включения QF

Рис. 4. Автоматная модель алгоритма включения автоматического выключателя QF

автоматического выключателя QF выделяется в отдельный автомат. Затем система разбивается на компоненты (автоматы, выполняющие строго определенные программистом функции), им назначаются входные/выходные переменные, и моделируется логика работы каждого из них (рис. 4).

Далее по словесному описанию строится набор управляющих состояний. Например, «Установить бит включения выключателя», «Сбросить бит взвода мотора редуктора» и т.д. Управляющие состояния связываются между собой переходами, включающими условия и действия при переходе из одного состояния в другое.

Вся логика работы автомата строится на основе двоичной математики (операции с переменными типа boolean). Переменные разделяются на «события (E)» – сигналы от объектов управления (например, сигналы от выключателя «Включен/Выключен»); «Входные воздействия (X)» – сигналы от устройства управления (например, кнопки «Включить/выключить выключатель» на пульте оператора); «выходные воздействия (Z)» – управляющие сигналы для объектов управления на выходе автомата.

Автоматные модели проектируются в системе Microsoft Visio с использованием встраиваемых шаблонов, разработанных авторами данной статьи. Когда автоматы спроектированы, их необходимо сохранить в отдельный файл стандартного формата Microsoft Visio «vsd» для обработки конвертером. Сгенерированный программный код после обработки автомата конвертером сохраняется в обычном текстовом файле, доступном для редактирования практически в любом текстовом редакторе (например, Notepad). Перед генерацией программного кода предлагается выбрать среду программирования из имеющегося списка, так как компиляторы языка структурного текста в различных средах несколько отличаются. В настоящий момент список состоит из трех программных сред: SIMATIC STEP 7, Unity Pro XL и CoDeSys. При генерации программного кода имена и названия автоматов, описания переменных и состояний используются как комментарии для улучшения читаемости программного кода. Также для информационной привязки к процессу переменных, имеющих вид «X1», «E1», «Z1», есть возможность их автоматической замены на соответствующие описания (рис. 4). Все автоматы наглядно разделены в программном коде и имеют одинаковую структуру, согласно моделям, что значительно упрощает процесс внесения изменений и формализует структуру программ.

Генератор преобразует каждый файл.vsd в функциональный блок на языке структурного текста. Это может быть любая задача или подзадача, решаемая описанными взаимодействующими автоматами.

Сам процесс преобразования автоматных моделей в программный код основан на общепринятых принципах, описанных в [2] в разделе «Реализация», и выполнен в строгом соответствии с идеологией, семантикой, дизайном программного кода автоматной парадигмы. Проверка модели на соответствие требованиям автоматной парадигмы осуществляется автоматически перед генерацией программного кода по методу Model Checking [6].

При построении автоматной модели программируется только логика управления в чистом виде (операции с переменными типа boolean). При этом математические операции программируются вручную. Используя доработанный конвертер MetaAuto, программист ПЛК получает 80% программного кода, сгенерированного автоматически, и 20% дорабатывает вручную (рис. 5).

Однако и эти 20% можно разбить на визуальное и «ручное» программирование. Для программирования сложных математических функций компания MathWorks разработала программный пакет Simulink PLC Coder (http://www.mathworks.com/products/sl-plc-coder/), который генерирует аппаратно-независимый программный код на языке структурного текста по встроенным функциям MATLAB. Таким образом, на долю привычного «ручного» программирования остается 5..10%.

Важно отметить, что разработанный генератор автоматных моделей не является интегрированным в современные программные среды (SIMATIC STEP 7, Unity Pro XL, CoDeSys, MATLAB) продуктом. Полученный с использованием генератора программный код добавляется в компилятор из текстового файла вручную с помощью операций «Копирование» и «Вставка» текста. Это единственный недостаток разработанного программного продукта, который, однако, делает его универсальным, так как отсутствует привязка к определенной программной среде. Данный подход позволяет генерировать программный код, соответствующий требованиям компиляторов, для упомянутых выше программных пакетов. В настоящий момент этот список расширяется.

Заключение

При больших и сложных проектах автоматизации связь между этапами разработки программного продукта теряется. Неоправданно много времени тратится на разработку, верификацию и документирование программного продукта. В качестве ме-

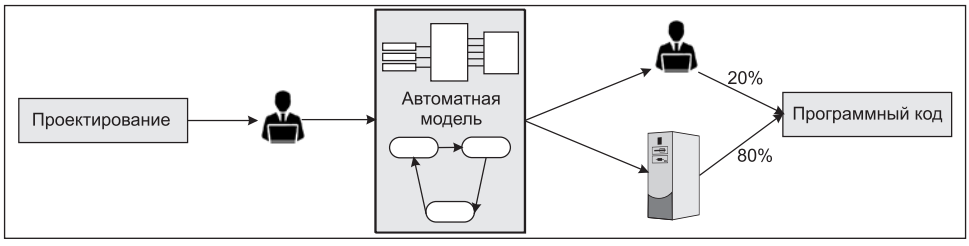


Рис. 4. Автоматная модель алгоритма включения автоматического выключателя QF

тодики, позволяющей решить данную проблему, авторами предложено использовать технологию разработки на основе моделей с применением автоматного программирования и доработанного генератора MetaAuto.

Совместное использование идеологии модельно-ориентированной разработки ПО и технологии автоматного программирования решили проблему сложности предметной области, а также верификации и документирования. Наличие доработанного генератора MetaAuto позволяет разработчикам ПО динамически структурировать большие и сложные проекты. Модель, служащая единым языком для участников проекта, обеспечивает его точное понимание и соответствие программного продукта требованиям на этапе верификации. Таким образом, модель неразрывно связывает все этапы разработки ПО – от проектирования до документирования.

Данная методика не является интегрированным в существующие программные пакеты решением. Но она используется как расширение существующих возможностей для программирования ПЛК и адаптирована к основным программным пакетам, поддерживающим международный стандарт IEC 61131-3.

За счет использования данной методики повышаются показатели качества программного продукта, так как тестируется не рукописный код, а модели. Значительно меньше требуется времени на верификацию и документирование. Эффективность применения автоматных моделей возрастает с увеличением сложности логики управления, так как структура проекта строго определена. Как следствие, показатели стоимости разработки ПО значительно уменьшаются.

На основе представленной методики построены автоматные модели, по которым был сгенерирован программный код для реального ТП. Данная методика была успешно апробирована в металлургической и энергетической отраслях на оборудовании компаний Schneider Electric, Siemens и программных пакетах SIMATIC STEP 7 и Unity Pro XL. В результате доработки инструментального средства MetaAuto получен инструмент, позволяющий реализовать автоматные модели любой сложности на любом оборудовании, поддерживающем международный стандарт IEC 61131-3.

Список литературы

1. *Karl-Heinz John, Michael Tiegelkamp.* Programming Industrial Automation Systems. Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools. 2001.
2. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. Второе издание. 2011.
3. *Буч Г.М., Рабо Г., Якобсон И. UML.* Руководство пользователя. М.: ДМК. 2000.
4. *Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.* Инструментальное средство для поддержки автоматного программирования // Программирование. №6. 2007. http://is.ifmo.ru/works/_2008_01_27_gurov.pdf.
5. *Канжелев С.Ю., Шалыто А.А.* Преобразование графов переходов, представленных в формате MS Visio, в исходные коды программ для различных языков программирования (инструментальное средство MetaAuto), http://is.ifmo.ru/projects/metaauto.
6. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО. 2002.

Большаков Олег Андреевич *– аспирант Института конструкторско-технологической информатики РАН,*
Рыбаков Анатолий Викторович *– канд. техн. наук, доцент кафедры «Автоматизированные системы обработки информации и управления» МГТУ «СТАНКИН».*
Контактный телефон (917)582-63-04.
E-mail:oleg.bolshakov@schneider-electric.com
E-mail: avr48@rambler.ru