Программно-алгоритмическое обеспечение конфигурируемых контроллеров в системах автоматизации и управления зданиями

Н.Г. Махмутов, Ю.П. Страшун (МГГУ)

Рассматриваются программно-алгоритмическое обеспечение (ПАО) цифровых контроллеров в системах автоматизации и управления зданиями и нестандартный подход к его разработке, нацеленный на использование графов переходов (автоматного подхода). Рассматриваемый подход в полной мере может быть использован при проектировании промышленных систем автоматизации и управления.

Цифровые контроллеры в архитектуре систем автоматизации и управления зданиями (САиУЗ) [1] занимают место между нижним уровнем датчиков и исполнительных механизмов (полевой уровень) и системами верхнего уровня управления (уровнем менеджмента). Основная функция контроллеров в системе – это сбор, обработка и передача на верхний уровень первичной информации, а также генерация и передача управляющих воздействий на исполнительные механизмы. В работе [1] подробно рассматривается концепция многоуровневой сетевой инфраструктуры САиУЗ. Для управления САиУЗ применяется ряд моделей контроллеров.

Контроллеры, применяющиеся в автоматизации зданий, в отличие от контроллеров, используемых в промышленной автоматизации, включают в себя специфические функции обеспечения комфорта, эргономики, энергоэффективности и безопасности.

Цифровые контроллеры могут быть следующих типов [1]:

- специализированные прикладные контроллеры (СПК) – узкоспециализированные миниатюрные контроллеры с фиксированными функциями;
- конфигурируемые контроллеры – с перестраиваемыми стратегиями управлениями;
- программируемые контроллеры – с программируемыми стратегиями управлениями.

Цифровые контроллеры могут нести основную нагрузку по управлению процессом, выдавая на верхний уровень запрашиваемую информацию, а могут быть лишь передаточным звеном между компьютером и конкретными элементами управления ТП.

В мире автоматизации программируемые цифровые контроллеры находят применение в подсистемах распределенных систем управления (РСУ) [2] и в, частности, в DCS (Distributed Control System), RCS (Remote Control System) и т.п.

Цифровыми контроллерами, например, являются [3-6]:

- ПЛК согласно стандарту IEC 61131-1-92;
- промышленные PC-совместимые контроллеры и т.п.

Программируемые PC-совместимые контроллеры предоставляют выбор между программированием на языке высокого уровня или использованием стандартных средств разработки ПО (стандарт IEC 61131-3. ftp://ftp.cle.ab.com/stds/iec/sc65bwg7tf3/html/welcome.htm). Программирование контроллера подразумевает создание в его памяти алгоритма функционирования контролируемой и управляемой технологической системы или ее части, и часто осуществляется посредством языков программирования IEC 61131-3. Поэтому контроллер должен обладать достаточными аппаратными средствами, позволяющими инсталлировать ядро ОС, обеспечивающей поддержку средств разработки и выполнения пользовательского приложения на базе языков IEC 61131-3. Стоимость проекта, реализуемого на базе такого контроллера и средства разработки пользовательского приложения, зачастую бывает высока. Это объясняется в основном тем, что к производительности аппаратных средств предъявляются высокие требования, а именно: процессор, способный поддерживать работу ядра ОС (как правило, 32 бит), большой объем памяти для хранения ядра средства разработки и пользовательского приложения (≥1 Мбайт). Благодаря выполнению таких требований пользователь контроллера обладает свободой задания поведенческой и функциональной моделей системы управления. Средства разработки согласно стандарту IEC 61131-3 позволяют задавать статические и динамические характеристики системы управления (функциональную и поведенческую модель) [7].

Однако как первичная настройка САиУЗ, включающей такой контроллер, так и ее дальнейшее обслуживание могут производиться лишь специалистом, обладающим высокой квалификацией в области программирования и отладки, а следовательно, сопряжены с весьма высокими затратами. Это объясняется сложностью и многогранностью задач, решаемых контроллером.

СПК – это, как правило, контроллеры с жестко заданным (производителем) алгоритмом работы или библиотекой алгоритмов для применения в ряде типовых ТП. Это значит, что алгоритм работы таких контроллеров неизменен – строго определен набор функциональных блоков, заданы связи между ними и режимы работы устройства управления. Пользователь может лишь подстраивать работу такого контроллера под имеющийся ТП путем задания некоторых параметров, например, уставок, условий перехода с режима на режим, временных задержек, порогов срабатывания сигнализации и др. В отличие от программируемых контроллеров к производительности СПК не предъявляются высокие требования, поскольку они не должны поддерживать функции ОС. Это позволяет успешно решать задачи управления ТП в РВ при невысоких затратах.

Таким образом, можно сделать вывод о том, что в отличие от пользователей программируемых кон-

троллеров, пользователи СПК сильно ограничены в возможности применения таких устройств для решения задач, непредусмотренных производителем. Кроме того, при изменении требований к процессу управления во время эксплуатации технологического контроллера с жестко заданным алгоритмом, возникнет несоответствие между алгоритмом управления и алгоритмом, заданным производителем контроллера, и пользователь окажется перед неразрешимой задачей изменения алгоритма работы контроллера.

Компромиссным направлением, позволяющим пользователю задавать алгоритм работы системы управления в рамках имеющейся аппаратной части и не вдаваться при этом в тонкости программирования, являются конфигурируемые контроллеры. Задача создания таких контроллеров и средств их разработки представляется чрезвычайно актуальной.

Задачи автоматизации и управления зданиями требуют от контроллеров обеспечения работы в различных режимах, то есть для каждого режима работы системы контроллеру задаются выполняемые функции.

Формально контроллер, служащий для решения задач управления ТП, характеризующимися различными состояниями, по существу является контроллером конечных состояний.

Средства разработки пользовательского приложения и программно-алгоритмическое обеспечение (ПАО) конфигурируемых контроллеров должны решать следующие задачи:

• задание функциональной модели системы (параметров и связей между имеющимися функциональными блоками);

• задание поведенческой модели системы, отражающей функционирование системы во времени.

Функциональная модель системы описывает связи и параметры функциональных модулей (регуляторов, коммутаторов, фильтров и др.). Данная модель может быть достаточно просто описана в текстовой форме или наглядно представлена в виде диаграммы функциональных блоков (Function Block Diagram). Соответственно гибкость функциональной модели контроллера определяется возможностью задавать характеристики функциональных блоков и связи между ними. Для устройств, не требующих высокой квалификации от наладчика, важно, что функциональные блоки могут быть как элементарными (например, логические функции), так и комплексными для конкретных технических процессов (например, канал управления контуром отопления).

При использовании конфигурируемых контроллеров функциональная модель системы управления может быть задана параметрически, то есть при помощи параметров, описывающих характеристики функциональных блоков и связи между ними. Средство разработки лишь передаст в контроллер эти параметры на основе функциональной модели, описанной пользователем в удобной ему форме. Такие средства разработки существуют – это редакторы FBD-диаграмм.

Совсем не так просто решается задача задания поведенческой модели системы управления, отражающей ее функционирование во времени в зависимости от условий.

Режимы работы системы управления характеризуются параметрами таких функциональных модулей, как ключи, коммутаторы и уставки. Таким образом, можно сказать, что для каждого режима работы контроллера задаются параметры динамических функциональных блоков. За выбор режима работы отвечает так называемая "машина состояний". Этот программный блок непрерывно анализирует все внутренние и внешние величины и проверяет, выполняются ли условия перехода в другой режим. В то же время "машина состояний" задает значение уставок и состояния ключей в зависимости от того, в каком режиме работы находится в данный момент система управления. В общем случае для каждого режима работы "машина состояний" определяет множество внутренних параметров системы управления, а также отвечает за выбор режима работы.

"Машина состояний" может быть представлена либо как конечный автомат, либо как сеть Петри. Эти формальные методы описания дискретных систем находятся в основе алгоитмического обеспечения современных контроллеров.

Имеется прецедент задания поведенческой модели контроллеров при помощи языка последовательных функциональных схем (SFC) или языка лестничных диаграмм (LD).

SFC используется для представления последовательности операций управляющих систем в графической структурированной форме, в основе данного языка лежат сети Петри. Язык SFC позволяет описать логику программы на уровне чередующихся функциональных блоков и условных переходов. Функциональные блоки представляют собой действия, которые должны быть исполнены, в том числе и параллельно, а условные переходы задают условия, которые должны выполняться для перехода к следующему функциональному блоку. Инструкции для функциональных блоков и условных переходов могут быть написаны на одном из четырех других языков IEC 61131-3. Следует отметить, что SFC подразумевает не только последовательное выполнение задач, но и возможность нахождения системы одновременно в нескольких состояниях, то есть описывает последовательно-параллельные задачи. Альтернативой применению SFC при программировании контроллеров является язык лестничной логики. Программы, написанные на этом языке, состоят из последовательности ступеней, которые выполняются последовательно. Каждая ступень состоит из набора графических элементов, ограниченных слева и справа шинами питания. Графические элементы соединяются вертикальными и горизонтальными связями. Кроме того, каждая ступень разделена на две зоны: зону условий, которая содержит условия, необходимые для выполне-

ния действий, и зону действий, содержащую операторы, которые должны быть исполнены по результатам, полученным в зоне условий. Существенным недостатком языка лестничной логики является необходимость последовательного перемещения между состояниями системы, а возможность свободного перехода от одного состояния к другому реализуема лишь с большим трудом, поскольку не характерна для этого языка.

Также имеет место задание поведенческой модели системы управления программным способом на языке программирования высокого уровня. Программная реализация на языке высокого уровня создается на основании автоматного описания алгоритма работы, такая реализация называется "switch- технологией" [8]. То есть функции "машины состояний" выполняет программно заданный конечный автомат (Finite State Machine – FSM) [9], характеризующийся множеством состояний и переходов. Такой подход дает большую свободу в задании алгоритма функционирования устройства, но сводится к написанию программы на языке программирования высокого уровня. Значительные наработки в этой области принадлежат А.А. Шалыто [10] и Н.И. Туккелю [11]. Хотя одним из первых, кто взял понятие "автомат" в качестве математической абстракции структуры ЭВМ, был В. М. Глушков. В 1961 г. в работах по теории автоматов В.М. Глушков говорил так: "Вычислительные машины проектировались на основе инженерной интуиции. Я решил превратить проектирование машин из искусства в науку".

Между тем, в 1993 г. фирма Modicon (США), входящая в настоящее время в группу Shneider Electric [12], а также фирма Siemens (Германия) в 1996 г. предложили нестандартный подход: использовать графы переходов (наглядное графическое представление конечных автоматов, т.е. автоматный подход) в качестве одного из языков программирования для своих контроллеров. Фирма Siemens утверждает, что описание на таком языке не только подходит для программиста контроллера, но также понятно инженеру-механику, инженеру по запуску оборудования и инженеру по обслуживанию. При этом отметим, что новый язык был введен указанными фирмами в практику проектирования, несмотря на то, что он не входит в состав языков, рекомендуемых Международным стандартом IEC 61131-3.

Работы упомянутых фирм позволяют сделать вывод, что подход к заданию поведенческой модели на основе теории конечных автоматов обладает оптимальным сочетанием наглядности и функциональности.

Автоматный подход к программированию "машины состояний" предполагает использование одного или нескольких конечных автоматов (FSM), который определяет реакцию системы управления на получение событий извне. Такая реакция описывается с помощью переходов, каждый из которых указывает на событие, происходящее после наступления условия перехода из одного состояния в другое. Конечные автоматы позволяют описать поведенческую модель системы в упрощенной форме, что выгодно применять при разработке конфигурируемых контроллеров.

В настоящее время схожий подход применяется при разработке технологических контроллеров с жестко заданным алгоритмом работы. Разработчик задает поведенческую модель в виде диаграммы, которая затем транслируется в машинный код и записывается в микроконтроллер. Этот подход обеспечивает такие средства, как графический редактор конечных автоматов IAR Visual State. Таким образом, применение автоматного подхода будет естественным развитием разработки конфигурируемых контроллеров.

К основным понятиям автоматного подхода относятся понятия "состояние" и "переход". Состояние, как правило, соответствует какому-либо режиму работы системы управления, а переход – это перевод системы из одного режима работы в другой.

Главное различие между этими двумя понятиями заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что предельное время перехода из одного состояния в другое равно нулю (если дополнительно ничего не сказано). Другими словами, переход объекта из состояния в состояние происходит мгновенно. Это понятия из так называемой "классической" теории конечных автоматов. Применение такого подхода к описанию достаточно сложной системы управления бывает неудобно из-за большого числа состояний, в которых может находиться данная система, и всех возможных переходов между ними.

Для минимизации формального представления поведенческой модели сложной системы управления, выраженной автоматом, существует ряд понятий, позволяющих группировать состояния и переходы (композитное состояние, подсостояние, историческое состояние и ряд др.). Таким образом, если в общем случае автомат представляет динамические аспекты моделируемой системы в виде ориентированного графа, вершины которого соответствуют состояниям, а дуги – переходам, то при использовании обозначенных выше понятий применяются так называемые диаграммы состояний и переходов (State Transition Diagram). STD – это диаграмма, на которой изображается конечный автомат, к которому относятся простые состояния, переходы и вложенные композитные состояния. Изначальная концепция STD была разработана Дэвидом Хэрелом [13].

Текстовое описание STD-диаграмм также достаточно компактно и удобно для описания систем. Применение такого инструмента, как STD позволяет перейти от формального и наглядного описания поведенческой модели к синтезу системы управления.

Чтобы контроллер мог отработать такую поведенческую модель, заданную пользователем, необходимо отразить ее в "машине состояний" контроллера. Это

значит, что программно "машина состояний" должна работать с паттернами понятий (автомат; композитное, историческое, начальное и конечное состояния; переход). На их основе при помощи средства разработки пользователь может сконфигурировать программный блок, отражающий необходимую поведенческую модель системы.

Такой подход к заданию алгоритма работы конфигурируемого контроллера, как и в случае задания алгоритма работы средствами IEC 61131-3 реализуется при помощи специализированного средства разработки. При задании алгоритма работы конфигурируемого контроллера средство разработки должно обеспечивать возможность задания функционала контроллера (например, при помощи диаграмм функциональных блоков FBD), а также обеспечивать возможность задания всех необходимых режимов работы контроллера, то есть поведенческую модель (например, при помощи диаграмм состояний и переходов STD). Другими словами, специалист, настраивая контроллер на выполнение определенной задачи, пользуется терминами самой задачи (процесса), а не конкретного языка программирования.

На основе созданной пользователем функциональной схемы системы управления средство разработки передаст контроллеру необходимые параметры функциональных блоков и параметры, описывающие их связи. На основе изображенной пользователем в специальном редакторе STD-диаграммы, описывающей поведенческую модель системы, средство разработки должно передать в контроллер информацию о состояниях и переходах, необходимую для отработки этой модели "машиной состояний". В этом случае средство разработки преобразует графическое отображение поведенческой модели в структуру паттерна case или switch. Заданные пользователем условия переходов должны преобразовываться средством разработки в соответствии со стековым алгоритмом вычислений в форму, которую контроллер сможет обработать без сложных преобразований (КНФ или ДНФ элементарных условий, польская запись, IL и др.). Таким образом, конфигурируемому контроллеру задается та или иная стратегия управления САиУЗ, а благодаря наглядности и формальности средства разработки, она может быть легко изменена или перестроена.

Заключение

Применение рассмотренного подхода к разработке ПАО конфигурируемых контроллеров позволит в будущем не только существенно расширить область их применения за счет увеличения гибкости алгоритмического обеспечения, но и значительно облегчить нелегкий труд разработчика. Это обусловлено тем, что рассмотренный подход позволяет задать технологу или наладчику требуемый алгоритм работы САиУЗ, содержащий конфигурируемый контроллер. Такой подход сократит сроки разработки устройства и сделает его применение более широким. Кроме того, существует возможность наработки библиотеки наиболее распространенных алгоритмов управления, которые можно использовать как шаблоны при создании оригинальных конфигураций, как это делается с приложениями для программируемых контроллеров. Таким образом, рассмотренный подход к разработке конфигурируемых контроллеров позволит совместить в одном устройстве функциональную гибкость и невысокую стоимость аппаратного, программного и алгоритмического обеспечения. Наглядность и детерминизм САиУЗ, использующей такой контроллер, позволит минимизировать ошибки при разработке и обеспечить ее высокую надежность. А высокая надежность и простота в обслуживании – это важные критерии оценки систем автоматизации и управления зданиями и промышленными инженерными системами.

Список литературы

1. Стандарт АВОК-5-2004. Системы автоматизации и управления зданиями. Ч. 2. Основные положения. Аппаратные средства. М.: АВОК-Пресс, 2004.
2. *Анашкин А.С.* Техническое и программное обеспечение распределенных систем управления. СПб.: Медный всадник, 2005.
3. ГОСТ Р 51840-2001 (МЭК 61131-1-92) Программируемые контроллеры. Общие положения и функциональные характеристики. ИПК Издательство стандартов. 2002.
4. *Анзимиров Л.В.* Тенденции мирового рынка промышленной автоматики // Автоматизация в промышленности. 2003. №4.
5. *Захаров Н.А.* ПЛК и PC-совместимые контроллеры: два подхода к построению систем // Там же
6. *Егоров Е.В., Малиновский Д.И.* PC против PLC или вперед к победе оппортунизма // Там же
7. *Кознов Д.В.* Конечный автомат – основа для визуальных представлений поведения объектов /Объектно-ориентированное визуальное моделирование. СПб.: Изд-во СПбГУ, 1999.
8. *Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.* UML. Switch-технология. Eclipse // Информационно-управляющие системы. 2004. № 6.
9. *Кудрявцев В.Б., Подколзин А.С., Ушчумлич Ш.* Введение в теорию абстрактных автоматов. МГУ, М.: 1985.
10. *Шалыто А.А.* Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления// Известия Академии Наук. Теория и системы управления. 2000. № 6.
11. *Шалыто А.А., Туккель Н.И.* SWITCH-технология: автоматный подход к созданию программного обеспечения "реактивных" систем// Программирование. 2001. №5.
12. Электронный каталог средств автоматизации компании Schnider Electric v 5.62. 2005.
13. *D. Harel and M. Politi.* Modeling Reactive Systems with Statecharts: The STATEMATE Approach. McGraw-Hill, 1998.

Махмутов Наиль Гаптрахманович *– аспирант,*
Страшун Юрий Павлович *– канд. техн. наук, доцент МГТУ.*
Контактный телефон 8-910-458-07-71. E-mail: Nail_Makh@rambler.ru