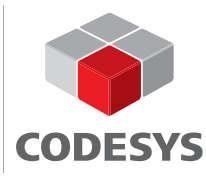
Программирование систем автоматизации завтрашнего дня



В области программирования систем автоматизации на производстве сегодня наблюдаются две тенденции: во-первых, в этой сфере применяются компьютерные информационные технологии (ИТ), во-вторых, при написании прикладных программ с успехом используется модульный принцип. Программа Application Composer, разработанная компанией 3S-Smart Software Solutions, позволяет пользователям идти в ногу со временем и соответствовать обеим тенденциям.

Роланд Вагнер, 3S-Smart Software Solutions GmbH

Оптимизация затрат на проектирование систем автоматизации в производстве, в том числе в машиностроении, а также в мобильных приложениях, автоматизации зданий сегодня является одним из высокоприоритетных вопросов. Программирование прикладных задач составляет значительную часть этих затрат. Неудивительно, что многие ведущие компании, работающие в данной области, всерьез задумываются над тем, какие концепции или парадигмы программирования систем управления им стоит использовать в будущем.

Конкурентная ценность систем все больше достигается именно за счет программного обеспечения. Пользователям постоянно требуется развитие сервисных и коммуникационных функций при одновременном упрощении применения, что вызывает непрерывное наращивание внутренней сложности систем. Это явно показано в исследовании «Требования изготовителей оборудования» немецкого союза машиностроителей (VDMA), крупнейшего объединения промышленных компаний в Европе [1].

Компания 3S-Smart Software Solution GmbH является разработчиком CODESYS, всемирно известного комплекса программирования на языках стандарта МЭК 61131-3. Многолетние собственные исследования и результаты обсуждений с большим числом пользователей явно выявили две тенденции:

- внедрение технологий и инструментов из мира компьютерных информационных технологий (ИТ) в автоматизированные системы управления;
- концентрация сложных программных частей в виде специализированных прикладных модулей, из которых в дальнейшем строятся программы управления отдельными машинами или производством.

Эти тенденции, их мотивация и способы практического применения в комплексе CODESYS описаны ниже.

Чему мы можем научиться у мира ИТ?

Современные методы и парадигмы программирования ИТ-приложений, в первую очередь объектно ориентированное программирование (ООП), значительно повышают эффективность разработки программ. Естественно, есть смысл применять их в системах автоматизации. Раньше это не удавалось сделать в основном из-за радикализма их введения. «Ветераны» программирования контроллеров категорически не принимали попыток безальтернативного ввода ООП. Кроме того, пользователи были вынуждены в будущих проектах переходить на компьютерные среды разработки, такие как Visual Studio.

Исследование, проведенное в 2011 году, показало: «В разработке программ для систем управления большинство пользователей используют языки стандарта МЭК 61131-3 (98 из 157 опрошенных). Другие языки, такие как C+, UML, C #, Matlab/Simulink, играют очень незначительную роль» [2].

Инновации из мира ИТ должны внедряться поэтапно, не перечеркивая наработанный опыт и не разрушая привычный инструментарий.

Зачем в МЭК 61131-3 нужно ООП?

Возможность использования ООП в контексте языков стандарта МЭК 61131-3 впервые была реализована в среде разработки CODESYS V3. Это позволило преодолеть пропасть между ИТ и контроллерным программированием.

Благодаря этому квалифицированные пользователи могут запрограммировать, например, наиболее сложные, «фундаментальные» блоки с помощью объектно ориентированного программирования. Они могут использовать интерфейсы, методы и наследование, динамическое связывание, перегружать методы и определять свойства точно таким же образом, как делают это в языках высокого уровня. Им не придется ради этого покидать среду МЭК 61131-3 – методы и их вызовы могут быть запрограммированы на языках IL, FBD, LD или ST. Созданные же ранее блоки могут вызываться из ООП-кода обычным образом. Пользователи, знакомые только с функциональным программированием, могут пользоваться новыми «фундаментальны-

ми» блоками точно так же, как они это привыкли делать без ООП.

По инициативе 3S-Smart Software Solutions концепт ООП был выдвинут на рассмотрение комитета по стандартизации, получил одобрение и был включен в третью редакцию стандарта МЭК 61131-3, выпущенную в начале 2013 года.

UML: дополнительный или «единый язык»?

В большом ООП-проекте разработчику бывает непросто сориентироваться в зависимостях тех или иных объектов друг от друга. UML-диаграмма классов позволяет наглядно отображать зависимости между классами, объектами, методами и интерфейсами, специально планировать и создавать эти зависимости. Таким образом, ООП становится более удобным и легче поддается контролю. Введение UML-диаграмм классов – это следующий логичный шаг после добавления ООП в инструмент МЭК 61131-3. В результате диаграммы классов были добавлены в CODESYS как дополнительный инструмент.

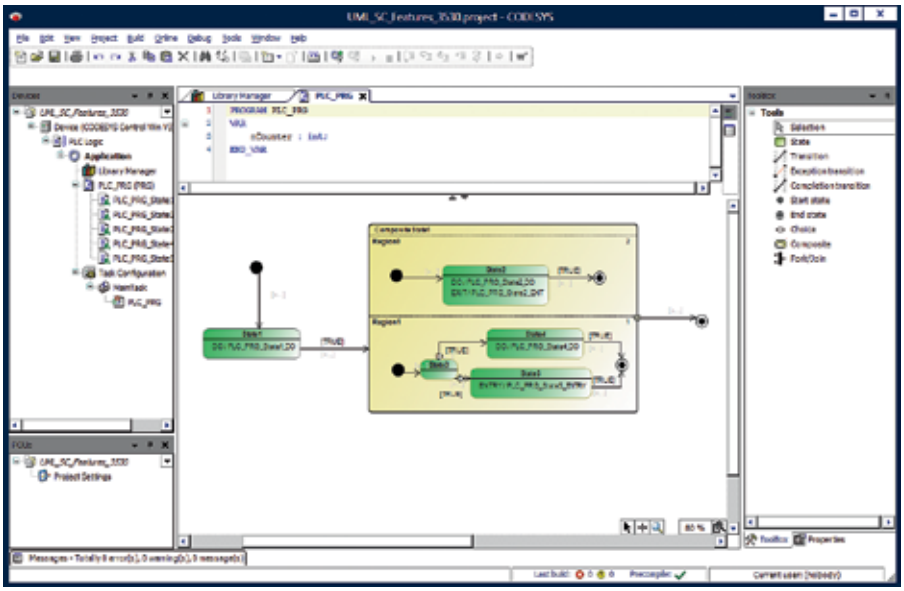
Но это еще не всё. Из всех 14 различных типов диаграмм особо выделяются диаграммы состояний, как наиболее полезные в автоматизации. Они также были интегрированы со средой программирования CODESYS. Теперь пользователи могут описывать программы и даже генерировать их код автоматически по стандарту МЭК. Специалисты, занятые в АСУ ТП, давно знакомы с такого рода диаграммами, но не имели возможности непосредственно использовать их в прикладном программировании. В среде разработки МЭК 61131-3 UML-диаграммы становятся средством коммуникации или «единым языком», который поможет «преодолеть вавилонское смешение языков между машиностроителями и программистами», как образно выразился доктор Хуттерер из компании Trumpf Maschinen Austria на форуме Benchmark Forum Intelligent Engineering, состоявшемся в Мюнхене в марте 2012 года. Используя UML-диаграммы, специалисты описывают технологические процессы, функции и порядок их вызовов для программистов. Благодаря интеграции UML с CODESYS, отпадает необходимость ручной конвертации проектов, что, естественно, уменьшает количество ошибок.

Вспомогательные инструменты для увеличения производительности программирования

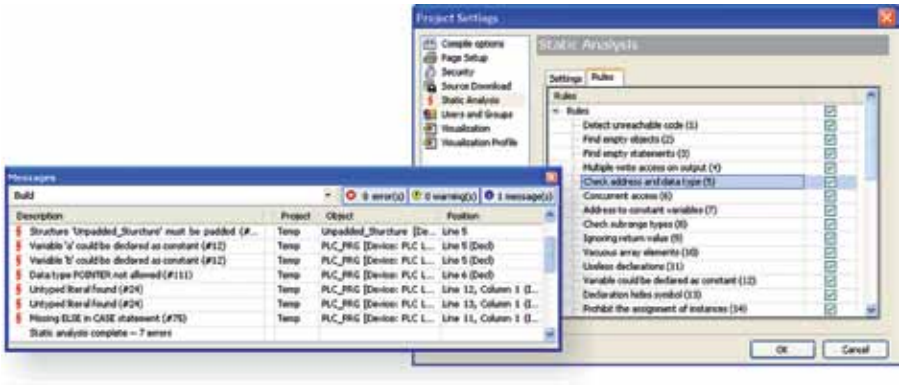
Помимо ООП и UML, есть еще несколько дополнительных инструментов из мира ИТ, которые могут весьма пригодиться в автоматизации.

Очевидно, что разработка масштабного проекта или набора однотипных проектов одним человеком может занять непозволительно много времени. В этих обстоятельствах необходимо работать группой. Как правило, у команды разработчиков возникает потребность синхронизировать свой труд и совместно использовать готовые фрагменты кода. В результате образуется набор проектов и библиотек с множеством вариантов улучшений и доработок. У программиста нередко возникают сложности при поиске исправлений, вызвавших неочевидную проблему. В таких случаях имеет смысл использовать систему контроля версий. В мире ИТ одной из самых распространенных систем такого рода является Apache Subversion (SVN). Разработчики могли бы хранить программные блоки и библиотеки в SVN, используя механизмы импорта и экспорта. Однако без интеграции системы контроля версий с МЭК 61131-3 ее было бы очень неудобно применять, особенно с графическими языками. Поэтому разработчики CODESYS интегрировали SVN в среду. Управление версиями программных модулей выполняется привычными средствами в среде программирования. Пользователь может сравнить различные версии программных компонентов, проанализировать отличия и восстановить нужную версию разрабатываемого приложения в любое время, с любой глубиной отката, что очень часто бывает необходимо на практике.

Еще одним полезным вспомогательным инструментом является статический анализатор кода CODESYS Static Analysis. Благодаря ему сокращается время разработки путем распознавания и исправления логических ошибок в коде приложения до того, как он будет загружен в контроллер. В анализаторе предусмотрены десятки тестов с настраиваемым набором правил. В числе прочего он проверяет одновременный доступ к переменным, множественную запись выходных переменных, ищет в проекте неиспользуемые переменные и пустые программные блоки. Пользователь может настраивать правила тестов для всего проекта. На случай, если ему необходимо намеренно обойти какое-то правило в конкретном фрагменте, предусмотрены специальные исключающие директивы, записываемые в коде приложения.



▲ UML-диаграмма состояний интегрирована со средой разработки МЭК 61131-3



▲ Статистический анализ кода для приложений МЭК 61131-3

Четвертым вспомогательным инструментом CODESYS является менеджер тестирования. Без тщательного тестирования не может обойтись ни один значительный проект. Иногда даже в самых тривиальных фрагментах кода может скрываться затаенная ошибка. Блок сравнений способен прекрасно работать на тысяче значений переменной и дать сбой на тысяча первом из-за банальной невнимательности в порядке вычислений и сравнений. Проверить это вручную нереально. Но на работающем объекте будет происходить сбой с непонятной симптоматикой и периодичностью. Менеджер тестирования позволяет написать сценарии тестирования. Работая по сценарию, CODESYS не поленится прогнать выполнение прикладной программы с каждым возможным значением переменной, оценить результаты и сформировать детальный отчет. Также нередко бывает, что «безобидная» доработка программы вносит неожиданную ошибку в уже отлаженный код. Менеджер тестирования позволяет пользователю выполнять полную проверку всего проекта при каждом его исправлении, сколько угодно раз. Ошибки в прикладной программе устраняются до выхода на объект, экономя время и повышая авторитет разработчика.

Представленные выше вспомогательные инструменты не входят в стандартный бесплатный пакет поставки CODESYS. Они выполнены в виде отдельных плагинов. Их можно приобрести и установить через интернет-портал CODESYS Store. Необходимые для этого средства встроены в среду программирования.

Преодоление сложности программирования

Сегодня многие люди, так или иначе связанные с работами по автоматизации, задаются вопросами:

- «Помогут ли нам новые концепты и стандарты программирования отвечать все возрастающим требованиям рынка?»;
- «Можем ли мы оценить риски, связанные с заменой нашей философии программирования? Стоят ли перемены этих усилий?»;
- «Можно ли добиться значительного скачка производительности в программирования ПЛК, не выходя за рамки МЭК 61131-3?».

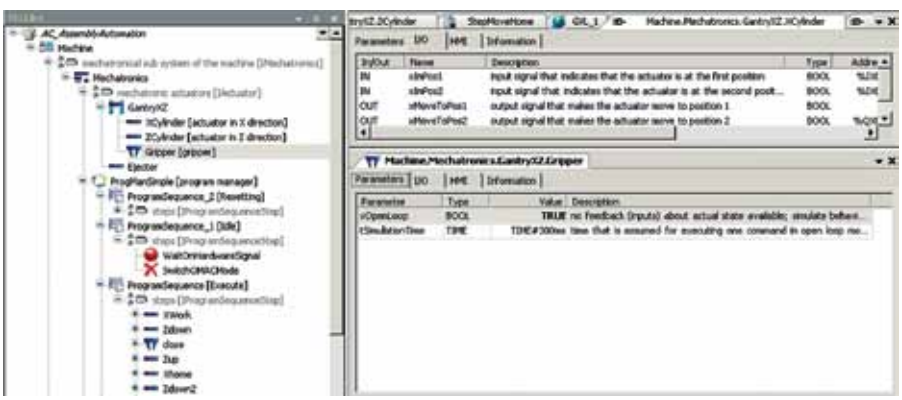
Все это представляется вполне возможным, особенно в случае тиражирования похожих либо однотипных проектов. Существуют сотни машин и систем, собираемых из достаточно стандартных узлов, механических и электрических устройств. Например, системы освещения и безопасности, системы приточно-вытяжной вентиляции, котельные, системы водоподготовки, деревообрабатывающие, полиграфические, пищевые и многие другие машины. Возникает мысль, что лучше всего иметь один раз подготовленный специалистами высшей квалификации набор гибко настраиваемых программных модулей для всех стандартных узлов подобных систем. Из них можно было бы, как из кубиков, собирать проект системы управления, настраивать необходимые параметры модулей, их связи и автоматически генерировать готовый, заведомо работоспособный код. Если делать это в рамках МЭК 61131-3, то сохраняется возможность легко внести в такой почти готовый код специфические для конкретной установки изменения или дополнения, не предусмотренные разработчиком модулей.

Данная идея вылилась в разработку принципиально нового инструмента программирования, получившего название CODESYS Application Composer.

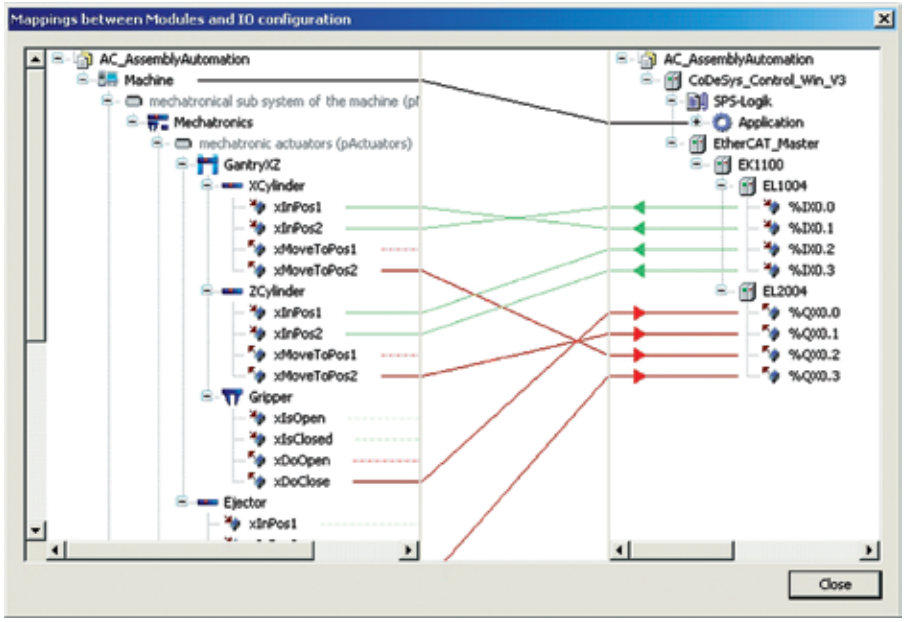
CODESYS Application Composer

В отличие от других обсуждаемых выше концептов, при использовании Application Composer разработчику не нужно приобретать дополнительных знаний или выходить за рамки привычной среды разработки МЭК 61131-3. Хорошо знакомый инструмент расширяется по принципу модулей, из которых будет «собрано» конечное приложение.

Понятие «сборки» говорит о том, что для создания приложений уже необязательно быть программистом: построением прикладного проекта способны заниматься технологи, инженеры и даже монтажники. Для этого достаточно иметь общие представления о работе машины.



▲ Собранное дерево модулей с параметрами входов



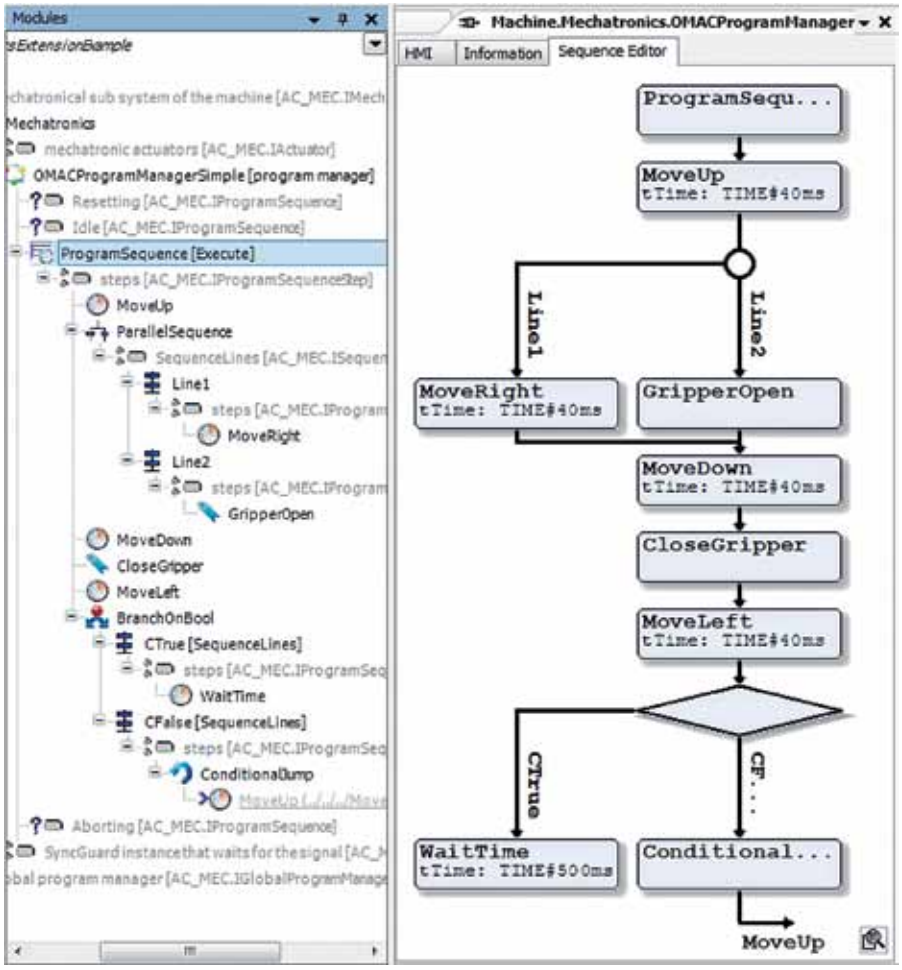
▲ Маппинг входов/выходов

Типовой модуль состоит из функциональных блоков и реализует функционал различных частей машины/установки. Модулями могут быть, с одной стороны, различные механические компоненты, такие как пневматические цилиндры, теплообменники, клапаны и т.п. С другой стороны, модуль может выполнять и чисто программные функции, такие как архивация событий, формирование аварийного сообщения и т.д. Важный момент – каждый модуль содержит базовые инжиниринговые черты CODESYS:

- МЭК 61131-3 код программы;
- входы и выходы для связи блоков;
- возможность параметризации значений;
- графическое представление для операторского управления или обслуживания.

Пользователь создает (собирает) древовидную структуру системы на основе этих модулей, которые полностью описывают работу машины. Модули имеют конкретные интерфейсы с описанием «родителей» и «потомков», и их можно поместить на дерево только в соответствующие места. Например, интерфейс модуля «циркуляционный насос» может требовать модуль плавного пуска либо частотного привода. Неподходящий модуль просто не установится. Модуль за модулем конструктор проекта собирает и настраивает установку целиком. Зная связи в дереве проекта, система подстроит модули друг к другу. Благодаря этому настройка параметров сокращена до действительно необходимого минимума. С помощью графического маппинга пользователь соединяет физические входы/выходы контроллера с соответствующими модулями.

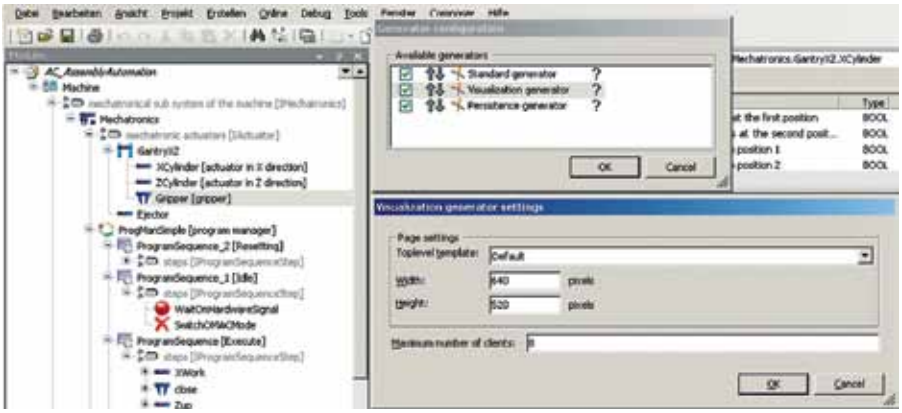
Если модулям необходим контроль хронологической последовательности их работы, то используется так называемый «модуль программируемых последовательностей». Для более удобного представления информации он оснащен собственным графическим редактором блок-схем по стандарту DIN66001/ISO5807.



▲ Задание хронологической последовательности

МЭК-программа в 0 строк

На основе подготовленного дерева модулей встроенный генератор по команде выдает полноценное МЭК 61131-3-приложение, включая визуализацию. Оно может быть скомпилировано с помощью CODESYS в исполняемый код и загружено в контроллер. Это значит, что пользователю даже не придется написать ни единой строчки МЭК 61131-3-кода самому – вся информация будет взята из дерева



▲ Задание параметров для генератора МЭК-кода и визуализации

модулей и библиотек и корректно встроена в структуру проекта. Исходный код будет доступен для редактирования, если такая необходимость вдруг возникнет. Приложение автоматически запустится при старте контроллера. Никаких дополнительных действий не требуется.

Если понадобится изменить приложение по требованию заказчика или из-за изменения конфигурации контроллера, то разработчику всего лишь нужно будет внести изменения в параметры требуемых модулей и переназначить входы/выходы.

Доступные и специальные модули

Эффективность использования Application Composer подразумевает доступность разнообразных модулей. 3S-Smart Software Solutions, как разработчик CODESYS, считает необходимым участвовать в развитии универсальных аппаратно-независимых модулей для, например, машин состояний, архивирования данных, коммуникаций, распределенных систем, обработки тревог и уведомлений.

Предполагается, что каждый изготовитель машин или систем должен сам разработать набор специализированных модулей для CODESYS Application Composer под свои конкретные продукты и области применения. Для этого необходим инструмент CODESYS Application ComposerToolkit.

Компания, имеющая достаточный опыт в программировании определенных систем, также может разработать собственный набор модулей и предложить заказчикам во всем мире через CODESYS Store [4].

Заключение

Каким же видится будущее программирования в автоматизации? Напрашивается несколько логичных выводов.

МЭК 61131-3 останется стандартом, которому будет соответствовать подавляющее большинство приложений для ПЛК, но инструменты программирования станут более мощными и сложными, чем раньше.

С одной стороны, программисты, имеющие соответствующее образование и опыт, смогут более эффективно работать в области прикладного программирования контроллеров. Они будут снабжены необходимыми технологиями и инструментарием, хорошо зарекомендовавшим себя в создании программ для персональных компьютеров в области ИТ. В самом деле, многие профессиональные программисты давно ждут таких расширений.

С другой стороны, есть технологи и инженеры, которым хорошо известна функциональность и специфические характеристики их машин и производств. В будущем они смогут сосредоточиться на своих основных задачах и создавать собственные машины, включая полноценные управляющие программы, с помощью модулей, которые разработают для них программисты. И всё это не написав ни единой строчки кода!

В конечном счете все выиграют от разработок, описанных выше. Сосредоточив внимание на своих задачах, каждая группа пользователей будет выполнять их быстрее и с меньшими ошибками. Рассмотренные инновации уже интегрированы в лидирующий продукт на рынке МЭК 61131-3-систем разработки – CODESYS. Будущее программирования систем автоматизации начинается уже сегодня!

Литература

1. The requirements of machine manufacturers // Computer&Automation, WEKA. 2012. № 1.

2. Jörn Linke. The PLC benchmark: the result // Computer&Automation, WEKA. 2011. № 9.

3. Dr. Hutterer. Trumpf Maschinen Austria demanded // Benchmark Forum Intelligent Engineering. Мюнхен, 2012.

4. Сайт Codesys Store. URL: http://store.codesys.com/

Р. Вагнер,
3S-Smart Software Solution GmbH,
ООО «ПК Пролог», г. Смоленск,
тел.: (4812) 382-931,
e-mail: info@prolog-plc.ru,
www.prolog-plc.ru