2. В тех случаях, когда у объекта инерционность канала передачи действия возмущения на управляемую величину меньше, чем инерционность канала передачи управляющих воздействий на эту величину, передаточную функцию корректирующего устройства целесообразно выбирать так, чтобы сократить нули числителя и доминирующие (ближайшие к мнимой оси) полюсы знаменателя передаточной функции системы управления, отвечающей каналу действия возмущения на управляемую величину.

3. Разработанный метод синтеза корректирующих устройств позволяет значительно повысить качество переходных процессов, возникающих на выходе системы при скачкообразном изменении возмущающих воздействий с неизвестными статистическими характеристиками.

Работа выполнена на кафедре "Системы управления" Московского государственного университета технологий и управления (МГУТУ).

Контактный телефон: (495) 670-91-90.
E-mail: syscontr@mail.ru

Список литературы

1. *Ротач В.Я.* Теория автоматического управления: Учебник для вузов. М.: МЭИ, 2004.
2. *Солдатов В.В., Агабекян Н.Г.* Робастное управление линейными стационарными системами на основе оптимального соотношения между составляющими хаоса и порядка // Приборы и системы. Управление, контроль, диагностика. 2005. № 5.
3. *Солдатов В.В., Жиров М.В., Шаховской А.В.* Многопараметрические цифровые регуляторы и методы их настройки // Приборы и системы. Управление, контроль, диагностика 2002. № 6.

Г.М. МАРТИНОВ, д-р техн. наук, профессор,
Н.В. КОЗАК, аспирант

Декомпозиция и синтез программных компонентов электроавтоматики

В статье представлена декомпозиция программных компонентов электроавтоматики и последующая их систематизация с учетом уровней абстракции виртуальной машины и функционального назначения компонентов. Предложен оригинальный исследовательский аппарат – матрица компонентов, позволяющая анализировать существующие системы, классифицировать прикладные программные компоненты электроавтоматики и на этапе проектирования системы определять оптимальный набор компонентов в соответствии с функциональными требованиями.

This article deals with decomposition of the software components and their systematization by levels of virtual machine and by their functional application is presented. The original approach is suggested. The essence of the approach is the component matrix, which analyzing existing systems, classifying software components of logic systems and determines the optimal set of components in accordance with required functional demands.

Введение

Стремительное развитие ПК в сторону увеличения вычислительной мощности и относительное снижение цены способствовало широкому применению ПЛК на базе ПК. Программируемые логические контроллеры на базе ПК характеризуются высокой универсальностью в отношении управляемого объекта и круга решаемых задач, возможностью быстрой перенастройки при перепланировке производства. Разработка и отладка прикладного ПО для таких контроллеров осуществляется с помощью стандартных инструментальных средств ПК, что существенно снижает затраты и сокращает время на разработку [1].

Традиционные производители ПЛК (например, *Siemens*, *Bosch*, *Beckhoff*), предоставляют собственные инструментальные среды для разработки и исполнения прикладного ПО [2]. Эти среды достаточно сложны, несовместимы между собой и не предусматривают возможности работать с ПЛК сторонних производителей.

Попытка унифицировать объекты электроавтоматики и создать единую среду разработки, сохраняя при этом стандарт языков программирования *МЭК 61131-3*, обозначила свое направление развития. Лидером здесь является компания *Smart Software Solutions – 3S*, а ее продукт *CoDeSys* уже становится стандартом [3].

Параллельно с рассмотренными направлениями, со стороны технологии разработки систем управления установились тенденции применения компонентного подхода и программной реализации контроллера электроавтоматики (*SoftPLC*) [4, 5]. Это предполагает использование стандартных технологий разработки, таких как *COM*, *DCOM*, *ActiveX*, применение платформы *.NET* и превращение *Windows XP* в ОС РВ.

Независимо от существующих тенденций и способа реализации, к прикладной компоненте электроавтоматики предъявляют вполне определенный набор требований:

- открытость в отношении используемых ПЛК-систем, в т. ч. и аппаратно реализованных;
- программирование ПЛК в соответствии с открытым стандартом *МЭК 61131-3*;
- подключение ПЛК к стандартным промышленным шинам (*ProfiBus*, *CanBus*);
- распределенное функционирование системы.

Многообразие предлагаемых на рынке решений в области систем управления электроавтоматикой с ограниченной открытостью вызывает затруднения при выборе и недопонимание возможностей, которые предлагает то или иное решение.

Целью статьи является декомпозиция прикладной составляющей электроавтоматики для выявления оптимального набора компонентов и синтеза механизма интеграции этих компонентов в единую систему.

Для достижения поставленной цели решены следующие задачи:

- систематизирован набор компонентов электроавтоматики и выявлены критерии определения оптимального набора компонентов на стадии проектирования системы;

• специфицированы программные интерфейсы взаимодействия между компонентами;

• разработаны механизмы интеграции внешних компонентов сторонних производителей.

Такой подход позволил определить базовый набор компонентов (минимально необходимый) для сокращения времени выпуска новых систем управления на рынок.

Классификация программных компонентов

Прикладная программная составляющая электроавтоматики характеризуется ОС, пользовательским интерфейсом, библиотеками прикладных программных компонентов, интегрированной средой разработки и исполнения.

Сегодня наиболее популярны ОС, обладающие графическим интерфейсом (*Windows* и *Linux*). Пользовательский интерфейс приложений разрабатывается на основе предоставляемых возможностей библиотек ОС и стандартных средств разработки ПО. С помощью библиотек прикладных программных компонентов создаются и отлаживаются управляющие программы электроавтоматики, реализуются задачи процессов управления, осуществляется доступ к ПЛК. Интегрированные среды разработки и исполнения управляющих программ электроавтоматики предлагают унифицированный способ работы в приложениях. Эти среды предоставляют операции перетаскивания (*drag-and-drop*) рабочих элементов в приложения, операции настройки панели управления как кнопок панелей инструментов и меню, механизмы вывода на печать и смены языка.

На базе рассмотренной функциональности прикладной составляющей электроавтоматики выделим две группы. Первая группа – это компоненты окружения, вторая – прикладные компоненты.

Компоненты окружения (*frame works* компоненты) обеспечивают механизм интеграции прикладных компонентов в приложение и их управление. С помощью этих компонентов исполняются прикладные программы, осуществляется настройка на конкретное оборудование и конфигурация прикладных задач. Они предоставляют главное окно приложения и размещаемые в нём области меню, панелей инструментов, документов, строки состояния, плавающие окна и механизмы управления этих окон.

Прикладные компоненты реализуют управление электроавтоматикой. Они осуществляют функциональное наполнение приложения, используя предоставляемые сервисы и интерфейсы, реализуют содержимое плавающих окон, команды в меню и панелях инструментов, отображают статус в строке состояния.

По своему функциональному назначению компоненты окружения делятся на три подгруппы:

• *компоненты каркаса системы* обеспечивают общий механизм интеграции приложения в единую среду исполнения. Этот механизм определяет правила работы с документами, способ передачи фокуса для активизации окна, интерфейсы для добавления пунктов меню;

• *компоненты конфигурирования системы* позволяют производить конфигурирование состава компонентов системы управления без перекомпиляции исходных кодов. Таким образом, конечный пользователь оперирует только тем набором компонентов, который ему необходим;

• *компоненты интерфейса пользователя* предоставляют свои интерфейсы для реализации экранов оператора. Это предполагает использование дерева проекта, строки состояния, окна пользовательских настроек, библиотек устройств, галереи кнопок панели оператора.

Компоненты окружения могут быть собственной разработкой, а также в их качестве используются доступные на рынке решения. Интерес представляет продукт *ddpFRAMEWORK* фирмы *M&M Software GmbH* [6], специализирующейся в области приложения промышленной автоматизации. Компании *Divelements Limited* и *LidorSystems*, специализирующиеся в области офисных решений, предлагают компоненты, реализующие расширенный пользовательский интерфейс *MS Office*, *Visual Studio* и *Outlook*. Подобные компоненты могут быть эффективно использованы.

Выделение пользовательских задач предполагает группирование прикладных компонентов электроавтоматики в подгруппы, в соответствии с выполняемыми задачами:

• *компоненты разработки управляющих программ*, предназначенные для создания управляющих программ электроавтоматики с помощью различных текстовых и графических редакторов и для последующей компиляции этих программ;

• *компоненты эмуляции исполнения и отладки управляющих программ*, обеспечивающие эмуляцию выполнения управляющих программ ПЛК, отслеживание изменений и редактирование значений переменных, вывод графической информации, графическую эмуляцию процессов производства;

• *компоненты выполнения управляющих программ*, реализующие запуск управляющих программ, управление последовательностью вызовов и остановку выполнения набора управляющих программ;

• *компоненты моделирования объекта управления*, обеспечивающие эмуляцию аппаратных панелей управления и процессов производства с использованием виртуальных приборов (тахометров, термометров, панелей настроек) и технических средств (двигателей, насосов, приводов осей станка);

• *компоненты диагностики оборудования*, производящие диагностику аппаратных средств, состояний соединений с ними, состояния аппаратуры при исполнении процессов управления, а также информирующие об аппаратных ошибках и аварийных состояниях процесса, неисправностях оборудования;

• *компоненты конфигурирования ПЛК*, обеспечивающие построение топологии контроллерных сетей, настраивающие режимы работы и координирующие совместную работу аппаратных средств, формирующие логические связи между исполняемыми управляющими программами и контроллерами.

Расширение пользовательских задач для систем электроавтоматики не ограничено. Конечным пользователям и сторонним производителям предоставляется набор системных средств разработки (*Developer's Kit*) для наращивания функциональных возможности системы.

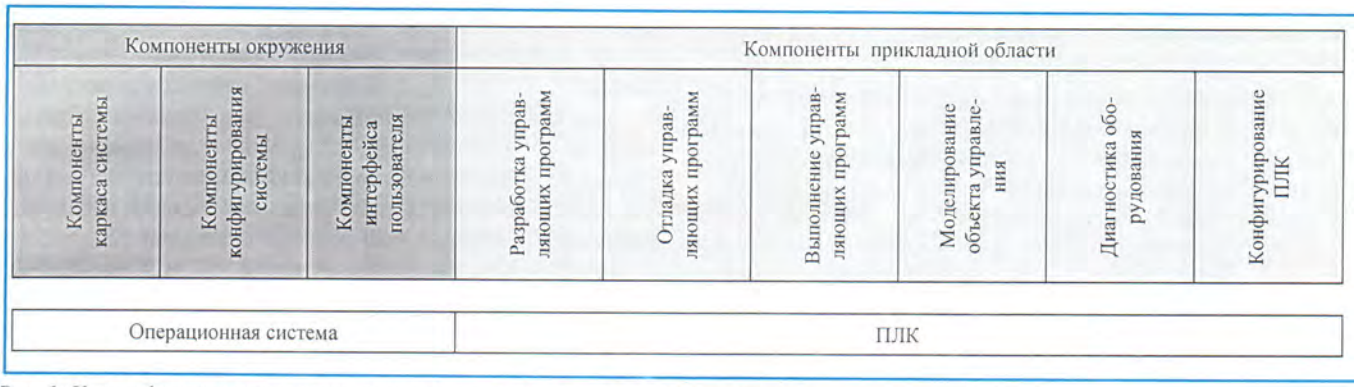

Рис. 1. Классификация компонентов систем управления электроавтоматикой

Но основное расширение пользовательских задач осуществляют разработчики ПО с выпуском очередной версии продукта или пакетом обновления.

Предложенная классификация (рис. 1) позволяет систематизировать компоненты электроавтоматики и определять функциональную нагрузку, которую несет каждая из групп и подгрупп.

Компоненты окружения определяют ОС и платформу исполнения прикладных приложений электроавтоматики. Смена ОС неминуемо влечет за собой смену компонентов окружения.

Компоненты прикладной области абстрагированы от особенностей платформы исполнения, но они привязаны к пользовательским задачам. Многообразие подгрупп определяется изобилием функциональных возможностей ПЛК-систем и разнородностью управляемых объектов. Такие традиционные задачи, как создание, отладка и выполнение управляющих программ, входят в любую поставку прикладного ПО электроавтоматики. Особенно активное развитие получили задачи моделирования объекта управления и удаленная диагностика оборудования. Это объясняется стремлением снизить расходы на проектирование производства и сократить внеплановые простои оборудования.

С точки зрения объекта управления, более интересной представляется группа компонентов прикладной области, поэтому далее акцент сделан на нее.

Виртуальная структура прикладной области

В своем вертикальном сечении ПО прикладной области электроавтоматики имеет многоуровневую структуру и полностью соответствует виртуальной машине [9].

На рис. 2 проведена аналогия между виртуальной структурой и архитектурой концепции *Microsoft Windows DNA* (*Distributed interNet Applications*), которая вводит промежуточные слои бизнес-логики для отделения представлений от данных.

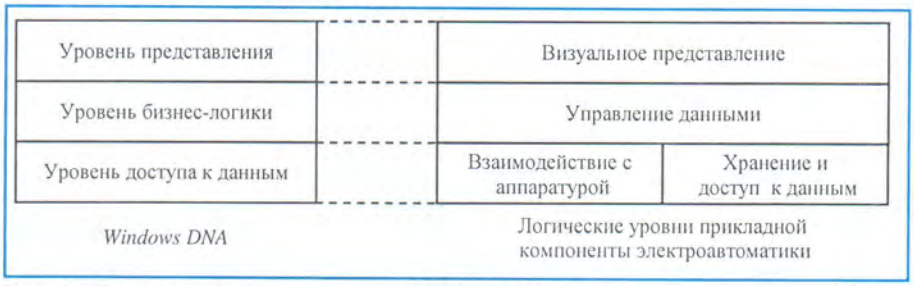

Рис. 2. Трехуровневая архитектура Windows DNA и уровни прикладной области электроавтоматики

Уровень взаимодействия с аппаратурой маскирует особенности работы аппаратных решений. На этом уровне инкапсулируют доступ к ПЛК-устройствам, реализуют работу с протоколами промышленных сетей (*ProfiBus*, *CanBus*), встраивают *OPC*-серверы для интеграции в *SCADA* систему.

Уровень хранения и доступа к данным определяет внутреннее представление информации, работу с БД, файловой системой и сетевыми протоколами обмена данных. Здесь реализуют менеджеры для предоставления данных, их хранения во внутреннем формате, а также для записи и чтения данных в различных файловых форматах.

Уровень управления данными реализует функциональность двунаправленного управления потоками данных. На этом уровне располагают трансляторы, компиляторы, генераторы кодов для различных процессоров, механизмы конвертирования и фильтрации БД. Уровень, отвечающий за преобразование данных, полностью соответствует функциональности уровня бизнес-логики *Windows DNA*-архитектуры.

Уровень визуального представления реализует компоненты пользовательского интерфейса по типу виртуальных приборов [9]. На этом уровне располагают редакторы управляющих программ, инструменты моделирования объекта управления, инструменты для построения графиков диагностики и отслеживания состояния процесса, панели управляющих элементов, а также реализуют диалог взаимодействия с оператором [10].

Матрица компонентов

Для анализа прикладных компонентов электроавтоматики предложен оригинальный исследовательский аппарат – матрица компонентов. Матрицу компонентов строят в виде пересечения пользовательских задач по горизонтали с логическими уровнями виртуальной структуры по вертикали (рис. 3). Таким образом, столбцы полученной матрицы определяют многообразие пользовательских задач системы управления электроавтоматикой, а строки – способ компонентой реализации ПО. Здесь проиллюстрировано, в каких ячейках следует искать традиционные модули электроавтоматики, такие как редакторы языков управляющих программ в стандарте *МЭК 61131-3*, *OPC*-сервер, драйвера промышленных сетевых протоколов *ProfiBus*, *CanBus*, сервис сетевых подключений, модуль шифрования данных.

Логические уровни	Пользовательские задачи системы управления						
	Разработка управляющих программ		Отладка управляющих программ	Выполнение управляющих программ	Моделирование объекта управления	Диагностика оборудования	Конфигурирование ПЛК
Визуальное представление	Редакторы *ST, LD, FBD* и других языков			Панели управления оператора	Редакторы графических объектов		
Управление данными	Компиляторы	Генераторы кода для процессоров					Сервис сетевых подключений
Хранение и предоставление данных						Диагностика во время управления объек-	Драйвер для сетевых протоколов *OPC, Profibus, CAN*
Взаимодействие с аппаратурой							

Рис. 3. Принцип построения матрицы компонентов

Применение матрицы компонентов позволяет:

- упорядочить необозримое множество компонентов современных систем, разделив их на группы по задачам пользователя и на логические уровни в соответствии с реализацией;
- выявить полноту охвата пользовательских задач и выделить минимальный (базовый) набор компонентов, достаточный для скорейшего выпуска на рынок изделий с ограниченным набором функциональностей;
- исследовать модульность системы с целью ее последующего развития и расширения;
- определить набор компонентов на этапе проектирования и разработки системы для устранения дублирования функциональностей.

Модульность системы характеризуется количеством логических уровней, которые занимает компонент. Если компонент занимает все четыре уровня матрицы – это означает отсутствие модульности в реализации прикладной задачи.

Исследование систем электроавтоматики с помощью матрицы компонентов

Современные системы электроавтоматики обладают несколько десятком базовых и множеством дополнительных компонентов, в т. ч. компонентами сторонних производителей. Вследствие этого прикладные приложения, как правило, частично перекрывают, а иногда и дублируют свои функциональные возможности. Конечному пользователю не легко разобраться в такой ситуации, учитывая, что дополнительные компоненты – это и дополнительные расходы на их приобретение.

WinCC (Windows Control Center)

Программный комплекс *WinCC* разрабатывался как универсальная система контроля и управления объектами, обладающая свойствами масштабируемости в соответствии с потребностями задач автоматизации и распределенного функционирования на базе ОС *Windows NT*. Наблюдение за процессом управления предполагает моделирование объектов, отслеживание аварийных сообщений, построение графиков и генерацию отчетов.

WinCC поддерживает все языки программирования контроллеров стандарта *МЭК* и в дополнение реализует работу с глобальными сценариями на языке программирования *ANSI-C*. Глобальные сценарии состоят из *C*-функций и действий.

WinCC хранит списки данных в стандартной БД *Sybase SQL Anywhere*, встроенной в систему. Доступ к БД осуществляется с помощью языка структурированных запросов *SQL* или через драйвер *ODBC*. В *WinCC* реализованы одновременно *OPC*-сервер и *OPC*-клиент, что позволяет осуществлять доступ к любому внешнему *OPC*-клиенту и предоставлять данные о процессе управления.

Матрица компонентов *WinCC* (рис. 4) имеет слабо выделенные логические уровни. Компоненты системы в основном реализованы по типу модуля, ориентированного на решение одной и более пользовательских задач.

Компоненты создания и компиляции управляющих программ занимают три логических уровня матрицы, это объясняется их реализацией в виде отдельных приложений, использующих файловую систему для обмена данными. Список адресов переменных контроллеров, используемый в управляющей программе, поставляется отдельным компонентом, который функционирует на уровне взаимодействия с аппаратурой этой же пользовательской задачи.

Взаимодействие системы с ПЛК в задаче конфигурирования обеспечивают драйвера каналов связи. Эти драйвера располагаются в матрице компонентов на

Рис. 4. Матрица компонентов системы WinCC

уровнях взаимодействия с аппаратурой, хранения и предоставления данных.

Задачу моделирования объекта управления реализует набор приложений для разработки графической модели объектов управления (*Graphics Designer*), для моделирования процессов управления (*Graphics Runtime*) и для построения диаграмм переменных процесса управления (*Function Trend Control*).

Матрица компонентов иллюстрирует полноту реализации пользовательских задач в системе. В матрице отсутствуют пустые столбцы; более того, каждой задаче соответствуют несколько компонентов на каждом уровне, что говорит о полноте охвата пользовательских функций системой, иногда переходящей в избыточность.

CoDeSys

Программный комплекс электроавтоматики *CoDeSys* фирмы *3S* поддерживает контроллеры более ста производителей (в т. ч. *Motorola*, *Hitachi*, *Intel* и *ARM*) [2]. В дополнение к языкам программирования стандарта *МЭК*, которые предоставляет *CoDeSys*, реализован язык непрерывных структурных схем *CFC* (*Continuous Function Chart*). Этот язык программирования основан на стандартном языке функциональных блоков (*FBD*), но в отличие от него использует свободно позиционируемые графические элементы, что позволяет, например, создавать петлю обратной связи.

Концепция создания управляющих программ в *CoDeSys* основывается на объектно-ориентированном подходе, что повышает эффективность процесса разработки. Пользователю предоставляются средств автоматического форматирования данных, объявления переменных и ввода шаблонов программ и функциональных блоков. Встроенный компилятор *CoDeSys* генерирует быстрый машинный код для обеспечения максимальной производительности. Интеллектуальные технологии на базе "инкрементального компилятора" (осуществляющего компиляцию только изменившегося модуля проекта) позволяют очень быстро обрабатывать проекты, содержащие тысячи переменных и сотни программных модулей. Комплекс *CoDeSys* предоставляет разработчику набор инструментальных средств для эмуляции ПЛК, пошаговой отладки управляющей программы, с использованием точек останова, визуализации объекта управления, возможности трассировки значений переменных и корректировки кода во время выполнения.

Матрица компонентов *CoDeSys* (рис. 5) имеет выраженный модульный характер логических уровней. Компоненты обладают более узкой специализацией, что повышает гибкость настройки пользовательских конфигураций системы.

CoDeSys-OPC Server предоставляет компонентам данные для построения визуализации и хранения информации о ходе процесса управления. Соединение с ПЛК происходит по сетевому протоколу *TCP/IP* с помощью сервера *CoDeSys Gateway Server*. Компоненты визуализации процессов управления *CoDeSys* позволяют пользователю создавать собственное ПО с графическим интерфейсом.

Согласно матрице компонентов в системе *CoDeSys* лучше всего соблюдено разграничение компонентной реализаций пользовательских задач по логическим уровням. Уровень хранения и предоставления данных

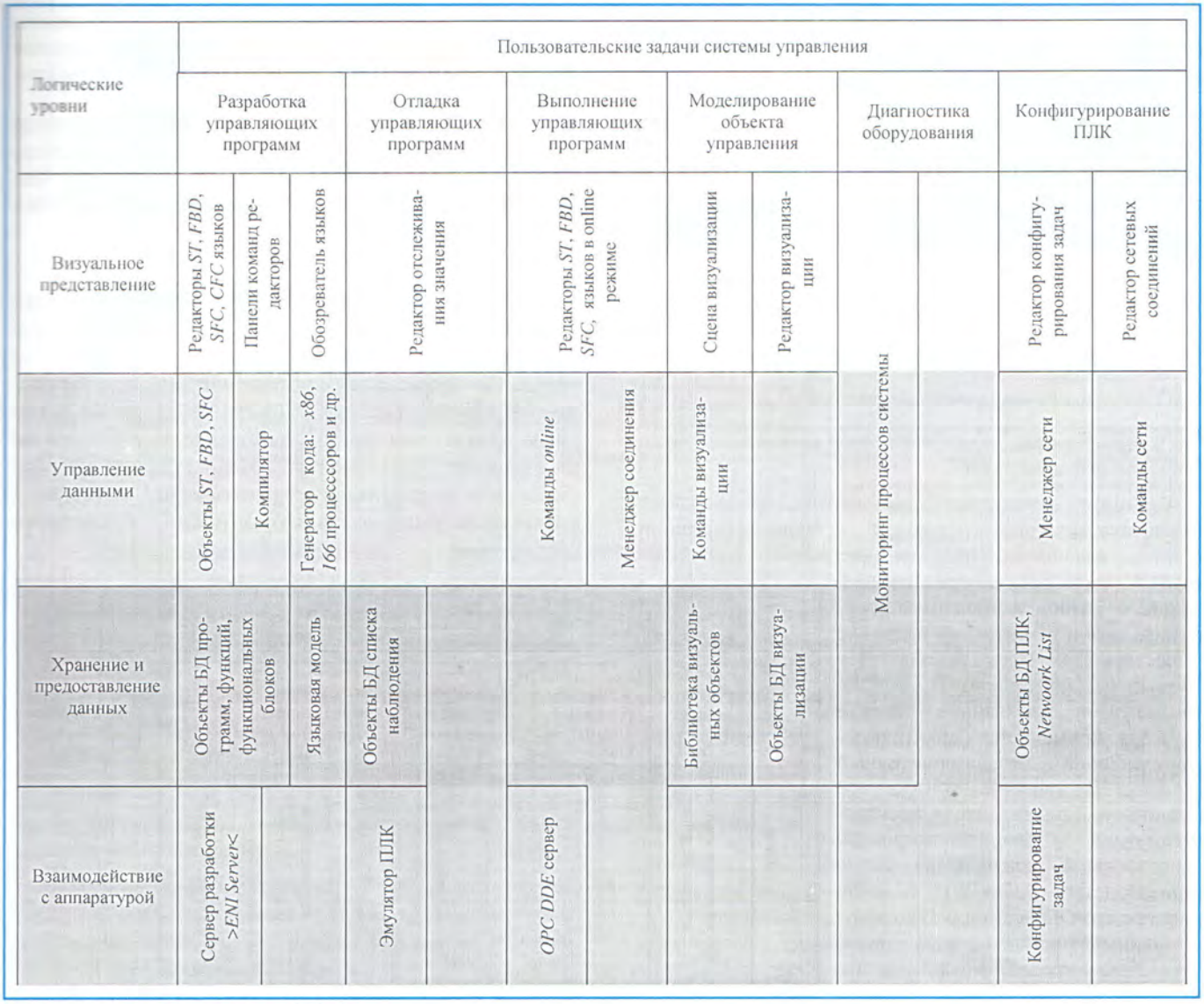

Рис. 5. Матрица компонентов CoDeSys

очень часто реализуется объектами БД. Доступ к БД со стороны уровня управления осуществляют компоненты, реализованные по типу менеджеров, например, менеджер объектов или менеджер языковых моделей. Другой пример – реализация взаимодействия с контроллерами через цепочки менеджеров соединения (*OnlineManager*) и *CoDeSys Gateway* сервером.

Пустые места в матрице компонентов иллюстрируют недостаточно полную реализацию пользовательских задач в *CoDeSys*, например, в задаче диагностики.

Принцип интеграции программных компонентов

Задача интеграции сводится к связыванию в единую систему модулей программированного контроллера, прикладной области и окружения. Связывающим звеном здесь выступает специальный модуль, разработанный для передачи и трансформации данных.

Интеграция компонентов предоставляет единые механизмы для печати, смены языка, работы с диалогом ожидания. Эти механизмы обеспечиваются за счет использования и реализации общих интерфейсов интеграции. Обобщенная спецификация интерфейсов интеграции приведена в таблице.

Интерфейсы интеграции

Интерфейс	Описание
IPrint	Предоставляет функции форматирования листа и работы с контекстом печати документа
ILanguage	Обеспечивает динамическую смену языка пользовательского интерфейса
IThemes	Определяет механизм унификации оформления компонентов внешнего представления в соответствии с выбранной темой, которая включает в себя: наборы шрифтов, цветов, фоновых рисунков и других графических объектов, которые определяют внешний вид пользовательского интерфейса
IWaitDialog	Предоставляет единый механизм работы с диалогом ожидания и блокировки действия пользователя на время выполнения системы длинных операций
IControlPanel	Обеспечивает работу с панелями управления (меню, панелями инструментов, деревом навигации), создает и размещает элементы управления и конфигурирует панели управления
IConfig	Определяет общий механизм конфигурирования компонентов с использованием вкладок настройки

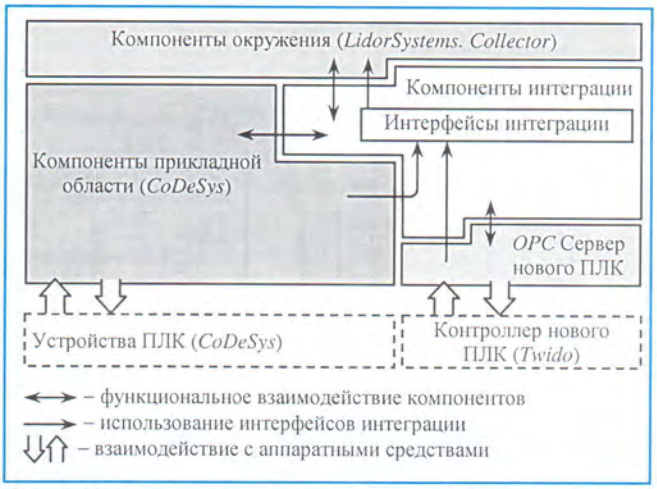

Рис. 6. Принцип интеграции компонентов электроавтоматики сторонних производителей

Принцип интеграции компонентов предполагает разработку связывающего модуля, а также реализацию и предоставление интерфейсов интеграции. Рассмотрим вариант интеграции, приведенный на рис. 6. Выбор компонентов окружения, хотя и не велик, но на рынке существует, и здесь используется продукт от *LidorSystems*. В качестве компонентов прикладной области *CoDeSys* применяется практически единственный безальтернативный вариант. В качестве контроллера стороннего производителя электроавтоматики проиллюстрировано использование контроллера *Twido* (производство *Schneider Electric*) вместе с его *OPC*-сервером.

Одним из основных результатов применения общего механизма интеграции компонентов является возможность конфигурирования системы.

Формально конфигурирование прикладной составляющей электроавтоматики можно разделить на три последовательных этапа.

1. Этап конфигурирования режимов системы закрепляет набор прикладных задач электроавтоматики за конкретный режим системы управления. Режимы формируются на базе пользовательских задач, при этом одна и та же задача может использоваться в нескольких режимах одновременно (рис. 7).

2. Этап конфигурирования задач определяет компоненты, которые используются в каждой пользовательской задаче. Задачи, одновременно используемые в разных режимах, могут иметь разную конфигурацию. Например, режимы программирования и отладки использует задачу моделирования объекта управления. Разница в том, что режим отладки не использует компоненты для редактирования модели объекта управления и для работы с библиотеками визуальных объектов (рис. 8).

3. Этап конфигурирования интерфейса пользователя предполагает настройку прикладных компонентов уровня визуального представления и размещения элементов управления компонентов задач в интерфейсе пользователя.

Заключение

Удельный вес прикладного ПО электроавтоматики неуклонно растет, что затрудняет ориентацию в многообразных предложениях на рынке. Решение проблемы предлагает формальное применение матрицы компонентов. Матрица систематизирует ПО электроавтоматики по прикладным задачам (разбиение по горизонтали) и оценивает модульность архитектурных решений (разбиение по вертикали) с точки зрения адаптации к конкретным решениям и возможности последующего расширения.

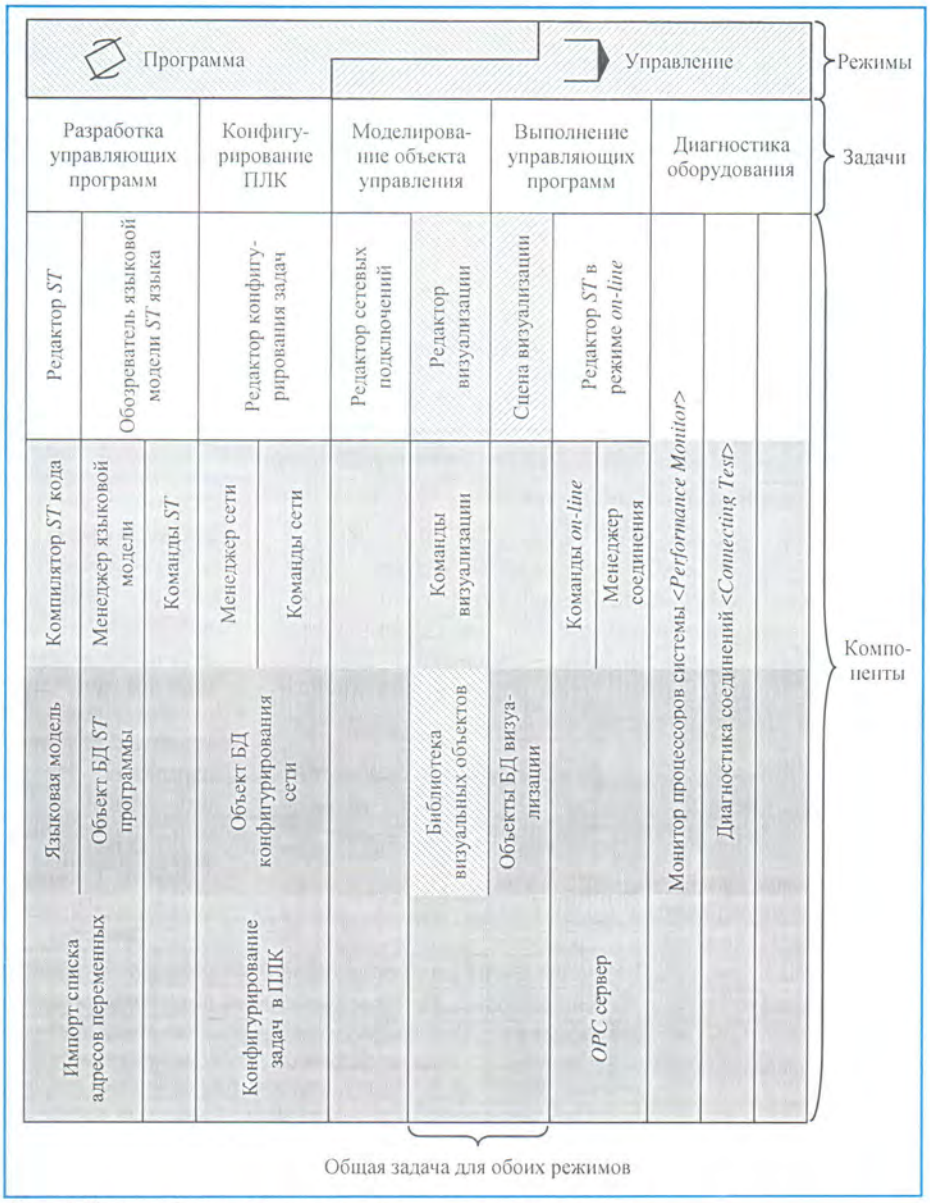

Рис. 7. Пример формирования режимов системы управления

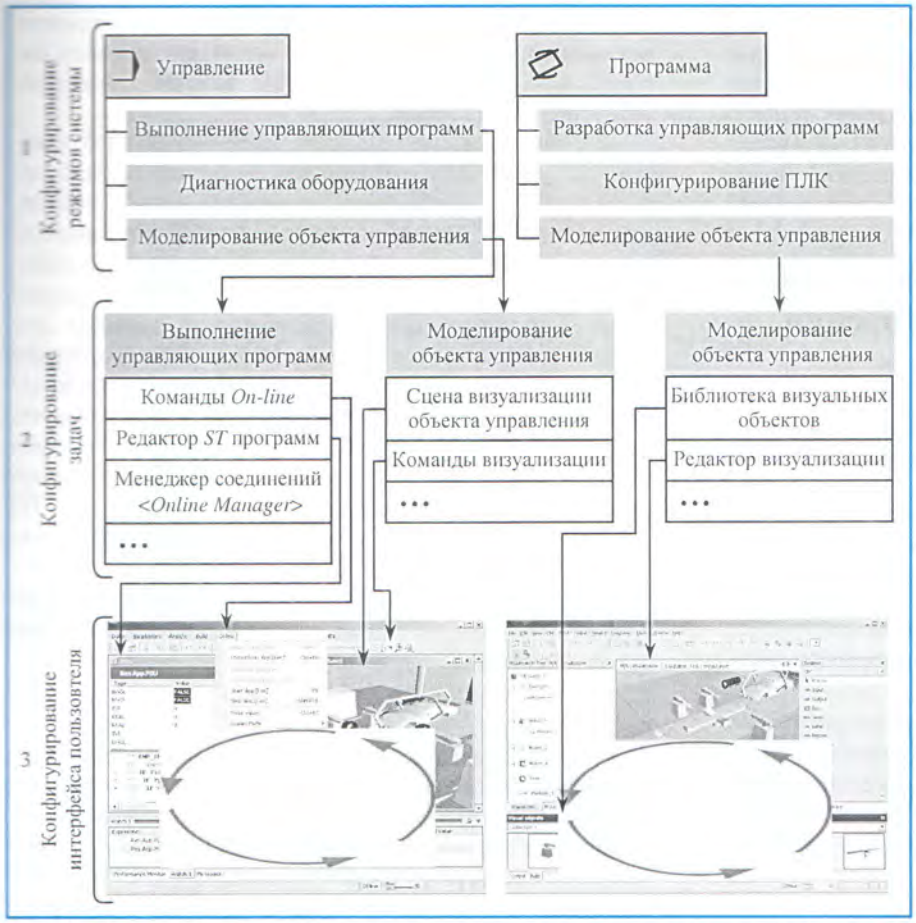

Рис. 8. Этапы конфигурирования компонентов системы управления для режимов "Управление" и "Программа"

Из рассмотренных систем матрица компонентов выявила в *WinCC* избыточность функциональностей для прикладных задач, что характерно для систем управления фирмы *Siemens*, и оптимальную модульную архитектуру для системы *CoDeSys*.

С помощью матрицы компонентов определяется минимальный набор прикладных задач, достаточный для выпуска на рынке облегченной версии ПО, без ожидания конца разработки продукта.

Разбиение на логические уровни и предложенный метод интеграции компонентов определяют, какие компоненты должны быть разработаны и как они будут интегрированы в системы для того, чтобы подключить ПЛК стороннего производителя.

Интеграция модулей программирования контроллера, прикладной области и окружения в единую систему предполагает разработку связывающего модуля и реализацию интерфейсов интеграции.

Работа выполнена в Московском государственном техническом университете "СТАНКИН".

Контактный телефон: (495) 972-94-40.

E-mail: book@ncsystems.ru

Список литературы

1. *Сосонкин В.Л., Мартинов Г.М.* Концепция числового программного управления мехатронными системами: проблемы управления электроавтоматикой // Автотракторное электрооборудование. 2002. № 4.
2. *Катцель Д.* Инвестиции в HMI отражают расширяющийся рынок // Control Engineering Россия. 2006. № 1.
3. *Петров И.В.* Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В. П. Дьяконова. М.: СОЛОН-Пресс, 2004. Серия "Библиотека инженера". ISBN 5-98003-079-4.
4. *Сосонкин В.Л., Мартинов Г.М.* Системы числового программного управления: Учеб. пособие. М. Логос, 2005. ISBN 5-98704-012-4.
5. *Сосонкин В.Л., Мартинов Г.М., Перепелкина М.М.* Концепция числового программного управления мехатронными системами: управление электроавтоматикой станков с ЧПУ по типу виртуальных контроллеров SoftPLC // Приборы и системы. Управление, контроль, диагностика. 2003. № 7.
6. *http://www.nnn-software.com*
7. *http://www.lidorsystems.com*
8. *http://divelements.co.uk*
9. *Мартинов Г.М.* Виртуальные приборы диагностики в системе ЧПУ // Информатика-машиностроение. 1998. № 4.
10. *Сосонкин В.Л., Мартинов Г.М., Любимов А.Б.* Интерпретация диалога в windows-интерфейсе систем управления // Приборы и системы управления. 1998. № 12.

Высокопроизводительные промышленные компьютеры по низким ценам

Компания *ИКОС* объявила о снижении цен на ряд популярных промышленных компьютеров *ROBO-2000*.

Начиная с сегодняшнего дня, компьютеры *ROBO-2000-4085TL*, *ROBO-2000-4175TL*, *ROBO-2000-4175TL-1*, *ROBO-2000-4175TLRHN*, *ROBO-2000-4385TL*, являющиеся одними из последних разработок компании *ИКОС*, предлагаются по специальным ценам. Все перечисленные выше компьютеры представляют собой современные высокотехнологичные системы, отличающиеся своей производительностью, надежностью и способностью работать в условиях сложных промышленных и специализированных применений. Компьютеры разработаны на чипсетах *Intel 915GV* и *Intel 945GV*, выпускаются с установленными процессорами *Intel Pentium 4* или *Pentium D* с частотой 3 ГГц. Такая производительность в совокупности с надежностью и низкой стоимостью делает эти компьютеры лучшим выбором среди систем для промышленных и специализированных применений. Компьютеры выполнены в конструктиве *4U* для 19" стойки. В качестве платформы в них используются процессорные платы *PICMG* или промышленная материнская плата *ATX* (модель *ROBO-2000-4085TL*). Все они представляют собой готовые к работе системы, способные бесперебойно работать в сложных условиях. Возможности расширения компьютеров представлены большим количеством слотов – 6 или 12 (в зависимости от модели).

Подробнее: *http://www.icn.ru/i/a.nsf/uv/robo-discount*