УДК 681.3

КОНЦЕПЦИЯ ПОСТРОЕНИЯ СРЕДСТВ ДИАГНОСТИКИ И УПРАВЛЕНИЯ УСТРОЙСТВАМИ ЭЛЕКТРОАВТОМАТИКИ НА БАЗЕ ОРС ТЕХНОЛОГИИ

Н.В.Козак, Р.А.Абдуллаев

В статье предложена структура компонентов интеграции устройств электроавтоматики и интерфейсы для их внедрения в прикладное программное обеспечение, описан механизм реализации OPC компонентов диагностики для контроллеров на базе CoDeSys SP.

Введение. Современные автоматизированные системы управления (АСУ) технологическими процессами (ТП) предусматривают объединение всех подсистем управления отдельными участками производства и технологическими процессами в единую интегрированную систему, которая обеспечивает выполнение всех требуемых функций управления предприятием. В организации управления предприятием выделены уровни (рис. 1), которым соответствуют применяемые на них информационные системы (SCADA, MES, ERP, BPMS). Ключевыми функциями в работе этих информационных систем являются диагностика и управление производственными процессами. Удобными инструментом в реализации вертикали этих функций является OPC технология, предоставляющая единый интерфейс для управления объектами автоматизации и технологическими процессами [0].

Рис. 1. Уровни управления предприятием

Целью публикации является представление концепции организации коммуникационной среды диагностики и управления устройствами электроавтоматики на базе OPC технологии.

Исходя из области применения OPC-серверов, в АСУ предприятия различают несколько уровней управления:

нижний уровень обеспечивает доступ к данным и управление полевыми шинами (fieldbus) и отдельными устройствами электроавтоматики (рис. 2);

средний уровень предоставляет данные и управление производственными процессами в рамках выделенного участка производства;

уровень АСУ ТП предоставляет обобщенные данные и более общие функции управления технологическим процессом (уровень работы информационных систем типа SCADA);

уровень АСУП - приложения управления ресурсами предприятия предоставляют обобщенные данные с нижних уровней управления ТП и других производственных процессов.

Каждый из уровней может обслуживаться OPC-сервером, поставляя данные OPC-клиенту на том же или на более высоком уровне.

Далее рассмотрим детально представленные на рис. 2: стандартные интерфейсы OPC, предоставляемые сервером; интерфейсы интеграции, их реализацию и использование в компонентах интеграции контроллеров электроавтоматики (PLC); интерфейсы программных компонентов взаимодействия с PLC на примере PLCHandler устройств электроавтоматики на базе платформы CoDeSys SP.

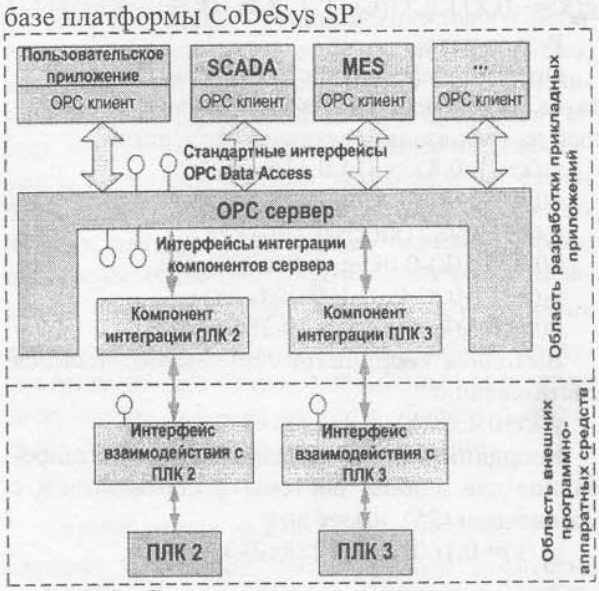

Рис. 2. Структура компонентов управления стройствами электроавтоматики

В заключение приведем пример реализации компонентов интеграции для предоставления доступа к диагностическим данным контроллеров CoDeSys SP клиентам OPC сервера.

Реализация интерфейсов OPC в сервере

OPC - (OLE for Process Control) стандартное описание интерфейса для локального и сетевого обмена данными. Технология OPC основана на механизме DCOM (Distributed Component Object Model) Microsoft Windows. Интерфейс IUnknown является базовым интерфейсом в COM/DCOM. Объект OPC сервера является прямым наследником интерфейса IUnknown (*Рис. 3*). Идиома точек стыковки выражает общую концепцию регистрации экспортируемых интерфейсов как небольшого числа интерфейсов стандартной инфраструктуры. Наиболее фундаментальным из этих интерфейсов является IConnectionPoint.

Интерфейс IOPCServer предоставляет возможность управлять OPC группами. Этот интерфейс определяет методы для добавления и удаления OPC групп. С его помощью можно преобразовать стандартный код ошибки в описание, понятное конечному пользователю. Интерфейс IOPCServer позволяет следить за общим состоянием сервера. IOPCBrowse является необязательным, позволяет приложению просматривать имена тегов и атрибутов, которые доступны внутри сервера. Интерфейс IOPCItemIO используется для выполнения операций чтения и записи элементов данных OPC.

OPC сервер вне зависимости от стандарта OPC (Data Access, Alarm & Event, Historical Data Access и т.д.) использует интерфейс IOPCCommon. Этот интерфейс предоставляет возможность устанавливать и запрашивать значение идентификатора LocaleID, который будет действовать для данной клиент-серверной сессии. Благодаря этому, действия одного OPC клиента не затрагивают других клиентов. Также как и для других интерфейсов, например IUnknown, объект, реализующий интерфейс IOPCCommon, для каждого сервера уникален. И объект сервера доступа к данным, и объект сервера уведомлений должны обеспечить реализацию интерфейса IOPCCommon. Клиент, который поддерживает соединение с этими серверами, будет использовать интерфейсы этих двух объектов независимо.

Интерфейсы интеграции компонентов OPCсервера

Верхний уровень взаимодействия OPC сервера и компонентов интеграции представлен интерфейсом IDAIDevice (DAI сокр. от data access interface). Основным назначением данного интерфейса является предоставление OPC серверу данных с физического устройства электроавтоматики. Интерфейс IDAIDevice является связующим звеном между коммуникационным уровнем и уровнем устройств. С помощью данного интерфейса OPC сервер имеет возможность создавать объекты данных (CDAIItemBase). Эта операция выполняется методом CreateItem интерфейса IDAIDevice.

Базовая реализация интерфейса IDAIDevice представлена в классе CDAIComponentBase (рис. 4), находящемся на стороне OPC сервера. При разработке компонентов интеграции для кокретного устройства электроавтоматики необходимо реализовывать производные от CDAIComponentBase класс. В этом классе переопределяются требуемые методы в соответствии со специфическими особенностями устройства электроавтоматики.

Рис. 3. Стандартные интерфейсы OPC сервера

Рис. 4. Интерфейсы интеграции компонентов сервера

Для того, чтобы скрыть подробности создания конкретного экземпляра, производного от базового класса CDAIComponentBase, используется интерфейс фабрики устройства IDAIDeviceFactory. Производный объект создается с помощью метода CreateDevice интерфейса фабрики устройства. Следует отметить, что каждое устройство имеет свой производный класс от CDAIComponentBase, а следовательно и свою особенную реализацию интерфейса IDAIDevice. OPC сервер имеет возможность создавать единообразно различные устройства IDAIDevice на логическом уровне.

Интерфейс IDAIItem предоставляет доступ к определенным элементам данных, которые находятся в устройстве электроавтоматики. Объект, реализующий интерфейс IDAIItem, на логическом уровне представляет собой переменную. Такие объекты имеют ряд свойств, кото-

рые характерны для переменных. Основной характеристикой таких объектов является имя, содержащиеся в поле Name. Тип переменной определяется интерфейсом IDAIItemType, имеющий единственное поле ItemType. Приоритет работы с элементом данных определяется интерфейсом IDAIPriority. Чем выше приоритет элемента данных, тем раньше он будет обработан сервером.

Текущее значение переменной можно получить из свойств Value и VariantValue интерфейса IDAIItem. Свойство VariantValue возвращает объект типа VARIANT, в то время как Value предоставляет экземпляр класса, реализующий интерфейс IDAIData. Этот интерфейс по своей сути является оберткой для работы с типом Variant. Информация внутри него структурирована в виде таблицы. Это означает, что доступ к данным осуществляется по номерам строк и столбцов. Единственным полем интерфейса данных является количество строк NumberOfRows. Для определения количества столбцов используется метод NumberOfColumns. Такой подход в определении количества строк и столбцов связан с тем, что каждая строка может иметь различное количество столбцов.

Еще одним важным свойством переменной, реализующей интерфейс IDAIItem, является качество чтения, задающееся в свойстве Quality. Оно может принимать одно из трех значений: Хорошее (good); Плохое (bad); Неопределенное (uncertain).

Для циклического доступа к набору переменных используются группа (свойство ParentGroup интерфейса IDAIItem), представляемая интерфейсом IDAIGroup. Группа необходима для того, чтобы предоставить пользователю возможность организовать доступ к данных, которые логически связаны между собой. Переменные, объединенные в группу, имеют общую скорость обновления данных. Среди основных свойств группы следует выделить ее имя Name, набор переменных Items и скорость обновления данных UpdateRate.

Также как и рассмотренное ранее логическое устройство (IDAIDevice), интерфейс IDAIItem имеет свою базовую реализацию на стороне OPC сервера, представленную классом CDAIItemBase. Фабрика CDAIItemFactory (рис. 4), реализующая интерфейс IDAIItemFactory, используется для создания определенных типов переменных методом CreateItem. Потомки класса CDAIItemBase реализуются в компонентах интеграции определенным образом в соответствии с особенностями физического устройства электроавтоматики.

Большинство классов компонентов интеграции реализуют интерфейс IDAIErrInfo с целью своевременного получения информации о возникающих ошибках. Этот интерфейс имеет ряд свойств, обеспечивающих полную информацию об обнаруженной ошибке. К таким свойствам относятся номер ошибки DetailedNumber и номер класса Number, в котором она имела место быть. Свойство Source указывает на источник ошибки, т.е. имя исходного файла и номер строки, в которой она возникла. Информация об ошибке представлена в виде короткого (ShortDescription) и длинного (LongDescription) описаний возникшей проблемы на естественном языке.

Интерфейсы взаимодействия с ПЛК на примере PLCHandler

PLCHandler - это класс C++, который позволяет создавать клиентские приложения (например, системы визуализации и диспетчерского управления), обменивающиеся данными с контроллерами с системой CoDeSys SP. Интерфейс PLCHandler предоставляет функции и службы установки и разрыва соединения с ПЛК (в том числе одновременно с несколькими ПЛК), чтения и записи значений переменных ПЛК (синхронно, асинхронно или циклически). PLCHandler автоматически восстанавливает соединение в случае разрыва. Таким образом, PLCHandler можно использовать в качестве базового компонента коммуникации для OPC-серверов или систем визуализации [2, 3].

Соединение с ПЛК может быть установлено через различные коммуникационные каналы. В CoDeSys реализован специальный интерфейс ARTI (Asynchronous Runtime Interface), который открывает символьный доступ к значениям переменных контроллера. Также соединение с ПЛК может быть установлено с помощью Gateway сервера. Он представляет собой автономное приложение, работа которого состоит в обеспечении связи системы программирования CoDeSys и программируемого устройства. Сервер позволяет поддерживать удаленные соединения с программами верхнего уровня по протоколу TCP/IP и с контроллерами через последовательный канал, CAN или другую сеть. Изготовители контроллеров могут добавлять к стандартным драйверам (RS232, CAN и Ethernet) собственные драйверы для поддержки фирменных протоколов.

Чтобы запрограммировать некое устройство PLC в среде программирования CoDeSys, в нем должна быть установлена система исполнения CoDeSys SP. С ее помощью любое микропроцессорное устройство, включая PC совместимые, приобретает функциональность ПЛК с поддержкой программирования на языках МЭК

61131-3. Установка системы исполнения выполняется изготовителем устройства или специалистами 3S. Конечный пользователь в этом не участвует. CoDeSys SP – масштабируемая среда исполнения УП, ее функциональность можно изменять в широких пределах, адаптируя к разным аппаратным платформам и техническим требованиям[4].

Система исполнения включает все аппаратно-зависимые и системные модули. Это позволяет МЭК программисту сосредоточится исключительно на прикладной задаче, не теряя времени на вспомогательную работу. Прикладные проекты в CoDeSys легко переносимы, поскольку программная поддержка специфичных для конкретного устройства узлов реализована изготовителем и скрыта в системе исполнения.

Реализация OPC компонентов диагностики для контроллеров CoDeSys SP

В качестве базовой реализации элементов данных используется класс CDAIItemBase, определенный на стороне OPC сервера (Рис. 4). Наследование от этого класса позволяет не реализовывать базовую функциональность, которая является общей для всех элементов данных. Методы, которые требуют специфической реализации, можно переопределить в производных классах.

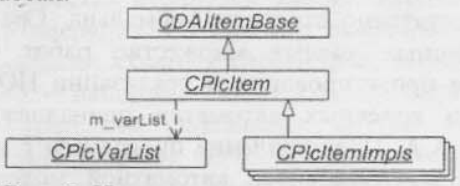

Рис. 5. Иерархия классов элементов данных

Внутри компонента интеграции с помощью механизма наследования определяется базовый класс CPlcItem (рис. 5) для всех разновидностей элементов данных. Характерной особенностью объектов этого класса является то, что они содержат в себе объект CPlcVarList, который представляет собой связующее звено между запросом к элементу данных компонента интеграции и интерфейсом взаимодействия с устройством электроавтоматики (рис. 2). Конечной реализацией элементов данных являются классы CPlcItemImpls, унаследованные от CPlcItem, которые реализуют получение диагностических данных от контроллеров.

Обобщенная схема получения диагностических данных OPC клиентами представлена на рис. 6. При обращении OPC клиента к OPC серверу (вызов 1 на рис. 6) происходит запрос к элементу данных CPlcItemImpl (вызов 2) компонента интеграции в основном потоке. После этого элемент данных добавляется в очередь запросов (вызов 3) и обрабатывается в отдельном потоке функцией обработки запросов (вызов 4). Дополнительный поток обработки запросов необходим для того, чтобы организовать одновременный доступ к синхронным сервисам OPC сервера для его клиентов.

Рис. 6. Обработка синхронного запроса OPC клиента

Функция обработки запросов (вызов 5), выполняемая в отдельном потоке, осуществляет взаимодействие с устройством электроавтоматики через интерфейс PLCHandler (вызовы 6.1. и 6.2). Уведомление элемента данных CPlcItemImpl (вызов 9) о завершении обработки запроса (вызов 7) происходит через скрытое окно (вызов 8). После этого происходит возврат фокуса управления обратно OPC серверу (вызов 10) и OPC клиенту (вызов 11).

Представленный механизм обработки синхронных запросов к устройству электроавтоматики является универсальным для одновременного использования несколькими OPC клиентами. Наличие очереди обусловлено необходимостью обработки нескольких синхронных запросов от OPC клиентов. Уведомление о завершении работы с интерфейсом PLCHandler, отправляемое через скрытое окно, означает готовность поля данных CPlcVarList.

Результаты

Применение технологии OPC на нижнем уровне управления предприятием обеспечивает стандартный интерфейс доступа к данным и управления устройствами эллектроавтоматики.

Описанный механизм реализации OPC компонентов диагностики для контроллеров CoDeSys SP предоставляет возможность одновре-

менной работы нескольким OPC клиентам с одним OPC сервером.

Предложенная концепция компонентов интеграции позволяет существенно сократить время и затраты при внедрении новых типов устройств электроавтоматики в производственный цикл современных предприятий.

Список использованных источников

1. http://www.opcfoundation.org/

2. PLCHandler Programming Guide Document Version: 1.2. - http://www.3s-software.com/se_data/_filebank/ReleasedDocuments/PLCHandler%20Programmming%20Guide.pdf

3. Мартинов Г.М., Козак Н.В. Декомпозиция и синтез программных компонентов электроавтоматики// Приборы и системы. Управление, контроль, диагностика. 2006. №12. С. 4-11.

4. Козак Н.В. Основы интеграции компонентов электроавтоматики на примере графического редактора 3D-сцены визуализации// Тр. Междунар. НТК «Информационные средства и технологии». в 3-х т. Т.3. – М.: МЭИ, 2007, с. 132-135.

МГТУ «Станкин», г.Москва

УДК 681.518.54

Тестирование объектно-ориентированного программного обеспечения на основе моделирования конечными автоматами

Л.С.Ломакина, С.Н.Малинин

Разрабатывается диагностическая модель объектно-ориентированного программного обеспечения (ОО ПО) на основе теории конечных автоматов. Исследуются ее свойства. Разрабатываются алгоритмы определения состояния ОО ПО на основании результатов тестирования или других экспериментальных данных.

Введение

Несмотря на сравнительно недавнее появление объектно-ориентированных (ОО) языков, таких как Smalltalk, Object Pascal, C++, Ada, объектно-ориентированное программирование (ООП) стало наиболее широко используемым стилем разработки программного обеспечения (ПО) в мире. ООП – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. ООП подразумевает событийную управляемость. Б.Мейер, один из основателей ООП, определял объекты как "нечто имеющее идентичность, состояние, поведение". С целью управления качеством ОО ПО необходимо решать задачи автоматизации тестирования, что требует разработки новых моделей, методов и алгоритмов, приспособленных к ОО ПО.

Фундаментальные теоретические и прикладные вопросы тестирования ПО изложены в работах В.В. Липаева, П.П. Пархоменко, В.И. Сагунова, G. Myers, B. Beizer, D. MacGregor, B. Boehm, C.V. Ramamoorthy и ряда других отечественных и зарубежных ученых. В качестве модели тестирования ПО в этих работах предлагается использовать управляющий граф программы. Однако, методы, разработанные для данной модели, хорошо зарекомендовали себя при тестировании традиционного (процедурного) ПО, но при непосредственном применении к ОО ПО имеют трудно решаемые проблемы.

Удобным средством описания ОО ПО является модель, построенная на основе теории конечных автоматов: она обладает ясностью семантики (трактовки элементов модели), наглядностью и выразительностью, и в то же время достаточно строга и формальна. Среди отечественных ученых множество работ по проблеме проектирования и реализации ПО с помощью конечных автоматов принадлежит Шалыто А.А. Представление программы с использованием элементов автоматной модели: состояний, событий, действий, переходов позволяет повысить эффективность тестирования ПО за счет проверки не только возвращаемых кодов, но и содержащихся в объекте данных.

Несмотря на то, что конечные автоматы используются уже давно для формального описания поведения систем, моделей, приспособленных к тестированию ОО ПО, нет. Классическая автоматная модель обладает рядом недостатков: отсутствие понятия параметров, в том числе и с бесконечной областью определения, отсутствие понятия времени, не определена семантика взаимодействия нескольких автоматов. Итак, разработка новых моделей и методов тестирования ОО ПО на основе теории конечных автоматов является актуальной задачей.

1. Теоретический анализ

Для описания функционирования программы целесообразно рассматривать автоматную модель программы, в которой выходы ассоциируются с переходами по событиям, а состояния соответствуют стабильным состояниям программной системы. Кроме того, в ОО ПО эле-