Стандарт OPC — путь к интеграции разнородных систем[1]

Игорь Куцевич, Антон Григорьев

OPC — это аббревиатура от OLE for Process Control, или OLE для управления процессами. Если принять это во внимание и попробовать прокомментировать название статьи, то придется сказать, что ключевыми словами для понимания изложения являются технология Microsoft OLE и интеграция. Точнее, “спрятанное” под термином OLE понятие COM.

Но прежде всего нужно сделать несколько замечаний по терминологии. Дело в том, что по ходу изложения материала встречаются термины близкие или даже одинаковые по написанию, но имеющие совершенно разный смысл.

1) *Автоматизация.* Это, конечно, собственно автоматизация (например, производства). Но это также и OLE-автоматизация (даже без приставки OLE). И еще это интерфейс автоматизации, который объясняется далее.

2) *Интерфейс.* Это, разумеется, интерфейс в общепринятых смыслах (их несколько). Но это также интерфейсы объектов COM и еще интерфейсы серверов (например, интерфейс автоматизации).

А теперь к делу.

Историческая справка

Сравнительно недавно, в 1994 г., под эгидой Microsoft была создана организация OPC Foundation (http://www.opcfoundation.org). Как определяет сама OPC Foundation, ее цель — разработка и поддержка открытых промышленных стандартов, регламентирующих методы обмена данными в режиме реального времени между клиентами на базе ПК и ОС Microsoft. Сейчас организация насчитывает более 270 членов, включая почти всех ведущих поставщиков контрольно-измерительного и управляющего оборудования для АСУ ТП. Достаточно назвать такие фирмы, как Siemens, Schneider Automation, Rockwell Software, Wonderware, Intellution, Ci Technologies.

Интеграция

Что же мы понимаем под интеграцией? Предположим, в результате огромных усилий программистов создана сложная комплексная система, охватывающая автоматизацию на всех уровнях предприятия, начиная от самого нижнего — управления датчиками и исполнительными механизмами — и заканчивая уровнем управления предприятием, вплоть до представления обобщенных данных завода у директора. Реализована ли в данном случае концепция интеграции? В общем случае нет. Здесь интеграция подразумевает не единую глобальную систему как таковую, а прежде всего взаимодействие всех уровней ПО между собой (разумеется, это не исключает объединение их в цельную систему). Фактически, речь идет о том, что различные программные системы, созданные с помощью различных средств, установленных на различных платформах, работающих на разных компьютерах, умеют “договариваться”. То есть они знают, как запросить друг у друга данные и как послать друг другу “указания”. По большому счету, интеграция сводится к конфигурированию “высоких договаривающихся сторон”.

Для того чтобы договориться, необходимо установить общий язык. И этот язык должны принимать хотя бы де-факто, а лучше де-юре все программные участники интеграции. Можно привести множество примеров всего, что употребляется с ключевыми словами интерфейс, протокол, API, язык и пр. Создаются комитеты, комиссии, организации, форумы, выпускаются спецификации, стандарты, и во многом последнее десятилетие проходит под знаком обеспечения возможностей интегрирования в той или иной области. Все крупнейшие компании увлечены этим потенциально очень прибыльным процессом, вдохновленные в том числе и поражающими воображения успехами Internet-интеграции.

Экскурс в COM/DCOM

Не осталась в стороне от этого процесса и компания Microsoft. Мы не будем здесь касаться борьбы Microsoft за Internet. О ее результатах хорошо известно. Но у Microsoft есть еще одна глобальная интеграционная тенденция, и о ней необходимо сказать несколько слов.

Итак, COM — Component Object Model (модель составных объектов) — и ее сетевое расширение DCOM — Distributed COM (распределенная COM) — это технология, введенная первоначально Microsoft для интеграции различных офисных приложений в Windows. Интеграция подразумевала использование объектов одного приложения, например, таблицы Excel, в другом приложении, например, в редакторе Word. Все это известно под аббревиатурой OLE. Начиная с версии OLE 2.0 в ее основу - была положена модель COM.

Первоначально название COM не “выставлялось на показ”, было скрыто от широких масс. Но постепенно COM пронизала все варианты Windows 9.x/NT/CE. Достаточно упомянуть такие ее производные, как ActiveX (OLE-

[1] Статья опубликована в PCWeek/Re №32 4 сентября, 2001

автоматизация) или OLE DB, не говоря уже о самой офисной OLE. Эта технология все больше и больше увлекала Microsoft. И вот в Windows 2000 к COM добавляются некоторые компоненты (транзакции, безопасность, очереди и др.), она преобразовывается в COM+ и объявляется главной, “склеивающей” технологией программирования в архитектуре DNA (Distributed interNet Application — распределенные приложения Internet), а связанные с этим технологии объединяются под общим названием Component Services (сервисы компонентов).

Обо всем этом сейчас уже написано достаточно много. Нам же в первую очередь нужно понять основные положения, связанные с нынешним состоянием технологии COM.

Модель COM оперирует *объектами*, очень похожими на объекты в объектно-ориентированных языках программирования типа Си++. Но сама технология COM не является языком программирования. Она только регламентирует поведение своих объектов. Нам нужно знать, что объект после создания предоставляет свою функциональность вызвавшему процессу, а после использования — уничтожается.

Объекты COM передают свою функциональность через *интерфейсы.* Интерфейс в COM (не путать с тем, что обычно понимается под словом интерфейс) объединяет группу взаимосвязанных функций, предоставляемых объектом. Главная особенность интерфейсов COM заключается в их “публичности”. Интерфейсы используются после того, как они “опубликованы”, и после этого их нельзя изменять *никогда* (имеются в виду прототипы и набор функций интерфейса, но не их реализация, которая вполне может изменяться). Если необходима новая версия интерфейса, издается новый интерфейс при сохранении старого. Этим обеспечивается совместимость при обновлении и модернизации объектов. И это *первый шаг на пути к интеграции.*

Именно интерфейс, вернее, указатель на него является тем, с чем работает вызывающий процесс (читай программист). Объект может предоставлять несколько интерфейсов. Чтобы получить указатель на любой интерфейс, нужно воспользоваться функцией QueryInterface обязательного для всех COM-объектов интерфейса IUnknown. Указатель на этот интерфейс передается инициирующему процессу при создании объекта.

Объект COM — сторона пассивная. Он лишь передает через интерфейсы свои функции. В этом смысле употребляется термин COM-сервер. Запрашивающая программа, соответственно, называется COM-клиент. Но это не исключает того, что обе программы одновременно могут быть и COM-серверами, и COM-клиентами. Забегая вперед, скажем, что именно здесь ключ к пониманию того, что OPC-сервер может поставлять данные “по подписке”, т. е. сам инициализировать обмен с OPC-клиентом при их обновлении.

Чтобы создать объект, нужно знать, где он находится. В Windows для этого используется регистрация объектов в системном реестре. И не только их. В реестре регистрируются также интерфейсы и кое-что другое. При этом каждый COM-предмет регистрации имеет уникальный в полном смысле этого слова идентификатор, называемый GUID (Globally Unique Identifier — глобально уникальный идентификатор, иногда используется название UUID — Universally Unique Identifier). Присваивает идентификаторы своим COM-детищам их создатель, используя, например, программу GUIDGEN.EXE. Заметим также, что многие COM-объекты могут (а ActiveX просто обязаны) саморегистрироваться.

Регистрация делает доступной информацию о расположении объектов всем приложениям. И это *второй шаг на пути к интеграции*.

Вопросы, затрагиваемые здесь, очень важны для понимания всего излагаемого. Объекты COM должны быть достаточно независимыми. Они зачастую, если не сказать в большинстве случаев, находятся вне программы COM-клиента, а могут быть запущены также и на другом компьютере. Это имеет далеко идущие последствия.

Даже на одном компьютере разные приложения Windows функционируют в своих собственных адресных пространствах. Это означает, что требуется кто-то для передачи вызовов из одного процесса в другой, так как даже простое создание или уничтожение объекта в другом адресном пространстве вовсе не тривиальное дело.

В COM эти и другие проблемы решаются с помощью специальных библиотек, таких, как OLE32.DLL. С одной стороны, эти библиотеки предоставляют функции для работы с объектами. Например, вызов CoCreateInstance создает объект. С другой стороны, активизируемые специальные компоненты выполняют диспетчерские функции, в частности, упаковку и передачу параметров вызываемым методам объектов (так называемый marshalling). В связи с этим упомянем два важных модуля: заместитель (proxy) и заглушка (stub). Они функционируют в адресном пространстве COM-клиента и COM-сервера соответственно и обеспечивают прозрачность вызовов. Механизм таков: COM-клиент вызывает функцию COM-интерфейса, которую ему подсовывает заместитель. Тот передает вызов заглушке через RPC. А заглушка вызывает функцию COM-сервера.

Для нас важно следующее. Первое — поддерживающие компоненты автоматизируют работу с COM-объектами и делают ее прозрачной для COM-клиента (с его точки зрения объект находится в его собственном адресном пространстве). И это *третий шаг на пути к интеграции.*

И второе — необходимость некоего программного обеспечения напоминает о том, что это в первую очередь технология Microsoft и добиться полной платформенной независимости не совсем просто.

Удаленные объекты

Без сетевых решений разговора об интеграции в настоящее время можно даже и не начинать. В COM по этому поводу существует DCOM — расширение COM, позволяющее добираться до объектов на других компьютерах. Важно то, что с точки зрения программирования ничего не меняется: DCOM — это системный сервис, делающий COM прозрачным в локальных сетях. И это *четвертый шаг к интеграции.* Но с тем же очевидным недостатком: DCOM *должен присутствовать* в операционной системе.

Еще одно существенное замечание. Сервис DCOM базируется на RPC. А это очень сильно затрудняет его использование в глобальных сетях. Требуются специальные программные компоненты и изощрённое сетевое администрирование, чтобы добиться взаимодействия через DCOM в глобальных сетях. Увы! Шаг на пути к интеграции несколько меньше желаемого.

Предоставление объектов

Чтобы использовать объект, необходимо знать, как он устроен, вернее, как устроены его интерфейсы. Для этого они должны быть опубликованы, например, в виде официальной документации, или стандарта. Таким образом, вырисовываются две возможности:

1) вы разрабатываете некий COM-объект, “украшаете” его и его интерфейсы GUID, снабжаете документацией (если это объект ActiveX, то официальная документация может и не понадобиться: на программном уровне общаться с распространяемым вами через Internet объектом будете только вы сами) и передаете в виде бинарного кода;

2) вы намечаете какую-либо проблему, изучаете ее, возможно, даже собираете “тусовку” под названием Foundation или Committee и издаете стандарт, подробно описывающий объекты, призванные решать данную проблему. Реализацию вы оставляете другим. Если дело стоящее, желающие найдутся. *Именно это можно сказать об OPC!*

Использовать COM-объекты должны COM-клиенты. Но они могут быть разными, если мы говорим об интеграции. И они могут применять разные языки программирования, не исключая скриптовых типа Visual Basic. Технология COM предусматривает две возможности. Либо вы программируете на Си++ и тогда описываете интерфейсы с помощью предоставляемых с документацией h- и c-файлов. В этом случае говорят об интерфейсе (не путать с COM-интерфейсами!). Либо вы используете для скриптовых запросов так называемую автоматизацию (OLE Automation). В этом случае для доступа к функциям объекта имеется специальный COM-интерфейс IDispatch, который COM-объект в этом случае обязан поддерживать, предоставляя интерфейс автоматизации (опять не путать с COM-интерфейсами!). Не вдаваясь в подробности, скажем, что при этом никакие компилируемые файлы не нужны, но нужна так называемая библиотека типов.

Программирование COM было бы нелегким занятием, если бы не предоставляемые средства. Процесс программирования упрощенно выглядит так. С помощью Си-подобного языка MIDL (Microsoft Interface Definition Language — язык определения интерфейсов) вы описываете интерфейсы. С помощью компилятора MIDL.EXE они преобразовываются в описанные выше файлы, в том числе и в библиотеку типов. А далее используется библиотека ATL (Active Template Library – библиотека активных шаблонов), “умеющая” интерпретировать эти файлы и многое другое, связанное с COM и ActiveX.

Все выше изложенное преследовало единственную цель – подвести понятийную базу под то, что будет рассказано далее. Думается, этого очень схематичного, если не сказать больше, описания хватит для более-менее непринужденного разговора собственно об OPC.

Как уже отмечалось выше, технология OPC реализована и продолжает реализовываться по второй схеме предоставления объектов — путем разработки стандартов. OPC Foundation определяет направления исследований и организует комитеты, которые делают следующее:

- создают спецификации COM-интерфейсов и COM-объектов;
- присваивают объектам GUID;
- оформляют все в виде стандартов и опубликовывают;
- генерируют или создают вспомогательные файлы: idl-, h- и c-файлы для Custom-интерфейса; библиотеки типов для интерфейса автоматизации; заместители (proxy) и заглушки (stub) для поддержки межпроцессного взаимодействия;
- разрабатывают вспомогательные компоненты, например утилиту opcenum, позволяющую OPC-клиенту “увидеть” список всех OPC-серверов локальной сети;
- занимаются рекламой и популяризацией технологии, включая выпуск демонстрационных программ и оценку производительности.

Практически все является общедоступным: зайдя на сайт, вы можете скачать то, что вас интересует, или заполнить небольшую анкету и бесплатно получить компакт-диск со всеми имеющимися материалами. Но кое-что (например, демо-программы), все-таки, предоставляется только членам организации. Кстати, стоимость членства зависит от вашего бизнеса и колеблется от 100 долл. в год для университетов и непрофильных организаций до 10 000 долл. в год для крупных фирм с оборотом более 100 млн. долл.

Существует достаточно большой перечень стандартов OPC (см. врезка 2). Консорциум OPC Foundation пытается охватить все аспекты взаимодействия с технологическим оборудованием. В разработке самих спецификаций принимают участие ведущие производители оборудования и систем автоматизации, которые стараются максимально учесть свой опыт и предоставить абсолютно все необходимое тому, кто будет использовать OPC. Далее мы проиллюстрируем это на примере спецификации Data Access (DA). Нет никакой возможности хоть сколько-нибудь подробно рассмотреть остальные.

OPC-стандарты

OPC Common Definitions and Interfaces — общие для всех OPC-спецификаций интерфейсы.

Data Access Custom Interface Standard — спецификация COM-интерфейсов для обмена оперативными данными, программирование на Си++.

Data Access Automation Interface Standard — спецификация COM-интерфейсов для обмена оперативными данными, программирование на языках типа Visual Basic.

OPC Batch Custom Interface Specification — спецификация COM-интерфейсов конфигурирования оборудования, программирование на Си++.

OPC Batch Automation Interface Specification — спецификация COM-интерфейсов для конфигурирования оборудования, программирование на языках типа Visual Basic.

OPC Alarms and Events Custom Interface Specification — спецификация COM-интерфейсов для обслуживания событий (event) и нештатных ситуаций (alarm), программирование на Си++.

OPC Alarm and Events Automation Interface Specification – спецификация COM-интерфейсов для обслуживания событий и нештатных ситуаций, программирование на языках типа Visual Basic.

Historical Data Access Custom Interface Standard — спецификация COM-интерфейсов для работы с хранилищами данными, программирование на Си++.

Historical Data Access Automation Interface Standard — спецификация COM-интерфейсов для работы с хранилищами данными, программирование на языках типа Visual Basic.

OPC Security Custom Interface — спецификация COM-интерфейсов для обработки прав доступа к данным, программирование на Си++.

OPC-сервер

Кто же использует OPC? Первая категория — производители оборудования автоматизации, или OEM. Предполагается, что тот, кто создает, например, плату сбора данных, снабжает ее не только драйвером, но и реализует OPC-сервер, работающий с этой с платой через драйвер или даже напрямую. Тем самым OEM-производитель предоставляет стандартный доступ к своей плате.

Список потенциальных изготовителей OPC-серверов неограничен. OPC-сервером можно снабдить контроллер, плату ввода/вывода, адаптер полевой шины, программу пересчета, генератор случайных чисел, что угодно, лишь бы это устройство поставляло или принимало данные. Но все-таки здесь речь идет в первую очередь о программном обеспечении для более низкого уровня в системах автоматизации.

Что необходимо сделать производителю, если он решил обеспечить свой продукт стандартным интерфейсом? Сначала он должен получить нужную спецификацию и прилагаемые программные компоненты, затем изучить *COM-интерфейсы тех COM-объектов* этой спецификации, которые относятся в ней к *модели OPC-сервера,* и, наконец, посадить самого опытного программиста за Visual Studio, который с помощью ATL-библиотеки реализует требуемые интерфейсы, а значит, и OPC-сервер. Это все. Можно только добавить, что если он не хочет использовать самого опытного программиста, ему придется купить какой-нибудь комплект инструментальных средств (Toolkit). Но об этом ниже.

OPC-клиент

Правила игры заданы: OPC-сервер поставляет данные, OPC-клиент их потребляет. Этим определяется вторая категория пользователей спецификаций OPC. К ней относятся, в первую очередь те, кто реализует программное обеспечение более высокого уровня, например, поставщики SCADA-пакетов или чего-то близкого по назначению.

Что же требуется от производителя “верхнего” ПО, если он задумал обеспечить свой продукт стандартным интерфейсом? Как и в предыдущем случае, ему надо получить нужную спецификацию и прилагаемые программные компоненты. Затем он должен изучить *COM-интерфейсы тех COM-объектов* этой спецификации, которые относятся в ней к *модели OPC-клиента,* и только после этого посадить достаточно опытного программиста за Visual Studio, чтобы тот с помощью ATL-библиотеки реализовал требуемые интерфейсы, а значит, и OPC-клиент для Custom-

интерфейса. Можно использовать Visual Basic или, скажем, Delphi, и тогда будет создан OPC-клиент для интерфейса автоматизации (если она предусмотрена для данной спецификации). По-прежнему, сэкономить на квалификации программиста помогает Toolkit.

Остальные потребители собирают системы из OPC-серверного оборудования и соединяют его с OPC-клиентным ПО. Главный фокус здесь — каждому OPC-серверу найти OPC-клиента и наоборот. Очень часто чего-то из этого не хватает, и тогда не исключена вероятность перехода потребителя в категорию изготовителей, но чаще заказчиков OPC-продукции.

Чтобы лучше почувствовать, что такое OPC, рассмотрим подробнее главный, по большому счету, стандарт Data Access (DA), предназначенный для поставки оперативных данных от оборудования и/или к оборудованию (термин “OPC” часто используют как синоним OPC DA). Для DA реализованы спецификации как пользовательского интерфейса, так интерфейса автоматизации. Как функциональный интерфейс, последний ничем не отличается от пользовательского, за исключением того, что не позволяет одновременно работать с несколькими OPC-серверами, и к нему добавлен упомянутый выше COM-интерфейс IDispatch, обязательный в OLE Automation. Это позволило OPC Foundation издать "обертку" (wrapper) в виде dll, преобразующую один интерфейс в другой. Таким образом, разработчик OPC-сервера заботится только о Custom-интерфейсе, а если клиент предпочитает автоматный интерфейс, он использует эту библиотеку в качестве переводчика.

Стандарт DA имеет две версии интерфейсов: 1.0 и 2.0. С точки зрения COM - это самостоятельные спецификации. OPC-клиент предварительно запрашивает, может ли он работать с нужным ему COM-интерфейсом в используемом OPC-сервере. В версии 2.0 механизм уведомления клиента приведен к стандартному механизму COM/DCOM, что упрощает программирование.

Более интересно рассмотреть, с чем работает DA. Основной единицей данных в OPC является переменная (Item). Переменная может быть любого типа, допустимого в OLE: различные целые и вещественные типы, логический тип, строковый, дата, валюта, вариантный тип и т. д. Кроме того, переменная может быть массивом.

Каждая переменная обладает свойствами. Различаются обязательные свойства, рекомендуемые и пользовательские. *Обязательными свойствами,* понятно**,** обязана обладать каждая переменная. Это, во-первых, текущее *значение* переменной, ее *тип* и *права доступа* (чтение и/или запись). Во-вторых, очень важные свойства — *качество* переменной и *метка времени.* Технология OPC ориентирована на работу с оборудованием, а оборудование может давать сбои, так что корректное значение переменной не всегда известно OPC-серверу, о чем и уведомляется клиент через качество (хорошее/плохое/неопределенное и дополнительная информация). Метка времени сообщает о том, когда переменная получила данное значение и/или качество. Еще одним обязательным свойством является *частота опроса* переменной OPC-сервером, хотя его можно было бы и не относить к обязательным, так как не все OPC-серверы работают в режиме опроса оборудования. Последнее из обязательных свойств — *описание* переменной. Это строковое значение, содержащее информацию для пользователя о том, что представляет собой эта переменная.

Дополнительные свойства необязательны для OPC-сервера. Это, например, *диапазон изменения* (выход за границы диапазона должен специальным образом обрабатываться клиентом) и *единица измерения*. Есть перечень рекомендуемых свойств, но можно добавить и свои собственные, то есть *пользовательские*.

Существует три основных способа получения OPC-клиентом данных от OPC-сервера: синхронное чтение, асинхронное чтение и подписка. При *синхронном чтении* клиент посылает серверу запрос со списком интересующих его переменных и ждет, когда сервер его выполнит. При *асинхронном чтении* клиент посылает серверу запрос, а сам продолжает работать. Когда сервер выполнил запрос, клиент получает уведомление (через интерфейс соответствующего COM-объекта, реализованного в клиенте!). И, наконец, в случае *подписки* клиент передает серверу список интересующих его переменных, а сервер затем регулярно присылает клиенту информацию об изменившихся переменных из этого списка (опять же, через интерфейс соответствующего COM-объекта клиента!). Эти списки в терминологии OPC называются группами. Каждый клиент может поддерживать одновременно много групп с разной скоростью обновления.

Запись данных ничем не отличается от чтения, за исключением того, что нет записи по подписке.

Технология OPC регламентирует только интерфейс между OPC-клиентами и OPC-серверами (как и положено в технологии клиент-сервер, допускаются множественные подсоединения). И она *абсолютно не регламентирует способ получения этих данных от оборудования!* Разработчик сам определяет, где и как их брать. Но, тем не менее, есть некоторые разумные, с точки зрения разработчиков OPC, модели взаимодействия с оборудованием. Для их рационального обслуживания предлагаются соответствующие механизмы. Например, можно попросить OPC-сервер получать данные не напрямую, а извлекать их из своего внутреннего буфера (кэша). Разумеется, если сервер не делает кэширования, он вправе эту просьбу "игнорировать".

Переменные в OPC-сервере могут быть представлены либо в виде простого списка, либо в виде дерева, напоминающего дерево файлов на диске (только вместо термина “папка” в OPC говорят “ветвь”). И есть соответствующие интерфейсы для навигации по этому дереву, позволяющие, в частности, в любой момент запросить дерево переменных, поддерживаемых OPC-сервером. Если оборудование допускает, дерево может изменяться динамически. Впрочем, если быть до конца точными, интерфейс для просмотра дерева объявлен в OPC-спецификации как необязательный. Тем не менее, он настолько удобен, что практически все OPC-серверы его реализуют.

Кроме того, есть механизм оповещения о завершении работы OPC-сервера, запроса информации о самом сервере и списка зарегистрированных групп. В общем, разработчики OPC-спецификаций предусмотрели многое для облегчения организации взаимодействия поставщика данных (OPC-сервера) и потребителя данных (OPC-клиента). Однако цель этого раздела — описание не DA-интерфейсов, а того, как OPC ориентируется именно на работу с оборудованием, в частности на обмен данными.

Инструментарий

Как уже было сказано, чтобы создать OPC-сервер или OPC-клиент, нужно только взаимодействие с OPC Foundation (получить OPC-спецификации) и Microsoft (купить Visual Си++ и пр.). Но имеется очень много сложных вопросов, которые придется решить при программировании OPC-интерфейсов.

Во-первых, само программирование COM не такое уж незатейливое, даже с применением ATL. Есть над чем подумать. Во-вторых, сами OPC-объекты и их OPC-интерфейсы достаточно сложны и громоздки. Так что придется потрудиться. И, в-третьих, надо решить вопросы системного уровня, затрагивающие фабрики класса (новый COM-термин!), заглушки и заместители, апартаменты (новый COM-термин!), асинхронный обмен, многозадачность, синхронизация, память.… Кстати, проблема памяти весьма актуальна, так как в COM допускается (и сплошь и рядом в OPC используется) выделение памяти в сервере, а удаление ее возлагается на клиент. Малейшая неточность — и пойдут трудно устранимые утечки памяти. А учитывая, что OPC-сервер обычно должен работать стационарно, рано или поздно крах системы неизбежен.

Всего этого избежать можно, если воспользоваться специальным инструментарием разработчика. Есть достаточно много фирм, которые реализацию OPC-спецификаций избрали своим бизнесом. Они в той или иной степени уже "наступили на все грабли" и предлагают средства, позволяющие более-менее безопасно и легко создавать OPC-продукцию.

Типичный инструментарий представляет собой библиотеку, состоящую из OPC-объектов выбранной спецификации. Разработчику же, например, OPC-сервера, предлагается некий набор вызовов, достаточно простых (read, write, …), которые необходимо "подцепить" к своему оборудованию для доступа к его данным. Для тех, кто знает объектное программирование, заметим, что эти функции могут быть реализованы как виртуальные функции некоторого класса, их нужно перегрузить в своем приложении. Так устроен, например, инструментарий фирмы FactorySoft (www.factorysoft.com).

OPC и интеграция

Теперь настало время взглянуть на OPC с точки зрения главной темы статьи. На рисунке представлена схема, иллюстрирующая возможные области применения OPC-серверов в АСУ предприятия. Мы различаем несколько уровней управления:

- нижний уровень — полевые шины (fieldbus) и отдельные контроллеры;
- средний уровень — цеховые сети;
- уровень АСУТП — уровень работы систем типа SCADA;
- уровень АСУП — уровень приложений управления ресурсами предприятия.

Каждый из этих уровней может обслуживаться OPC-сервером, поставляя данные OPC-клиенту на более высоком уровне или даже "соседу".

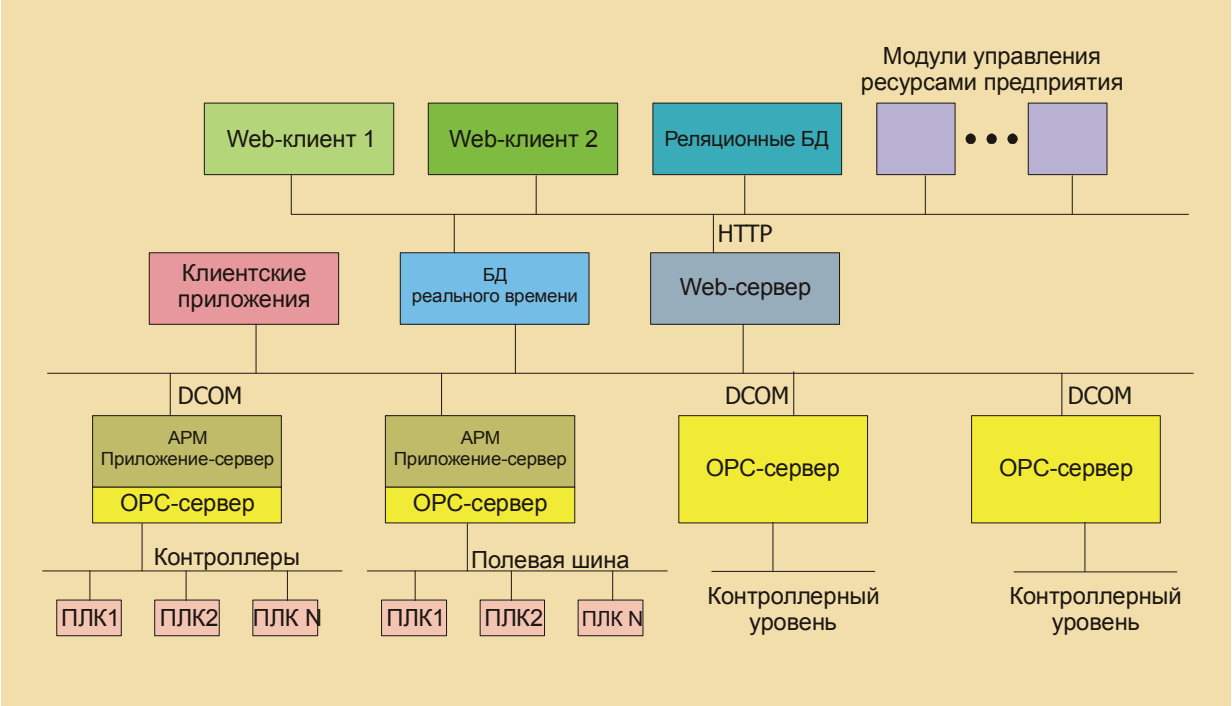

Рис.1 Возможные области применения OPC-серверов в АСУ предприятия

Опишем подробнее “места работы” OPC-сервера.

Если имеется оборудование, например, плата АЦП, управляемая через драйвер на компьютере с Windows или другой ОС, поддерживающей COM/DCOM, то это самый главный кандидат на то, чтобы непосредственно поверх драйвера реализовать OPC-сервер. Замена устройства не потребует изменения остальных приложений: драйвер изменяется, но OPC-интерфейс поверх него остается прежним.

При наличии устройства, управляемого через какой-нибудь сетевой протокол, вполне возможна реализация OPC-сервера, получающего данные по этому протоколу. Единственная особенность — следует предусмотреть механизмы восстановления связи в случае сбоев.

Несколько более сложной будет схема при работе управляющих приложений на компьютере, не поддерживающем COM/DCOM. В этом случае применим двухкомпонентный OPC-сервер. На стороне ОС, не поддерживающей COM, устанавливается сетевой модуль, который, с одной стороны, связан с приложением(ями), а с другой — через сеть с OPC-сервером. Заметим, что сетевой модуль может быть стандартным (как, например, ISaNet в системе ISaGRAF). В этом случае необходимо разработать только OPC-сервер. По такому принципу функционирует OPC-сервер ISaGRAF фирмы “РТСофт” (www.rtsoft.ru). Иногда сетевой модуль создается специально для OPC-сервера. Возможна даже реализация, при которой этот модуль не ориентирован на конкретное приложение, а предоставляет некоторый API-интерфейс для любых приложений, желающих обслуживаться с помощью OPC. Так действует OPC-сервер для операционной системы OS-9, также разработанный в “РТСофт”.

Еще одна разновидность OPC-сервера — шлюз к сети полевой шины, такой как Profibus или Lonworks. Реализация этой схемы очень похожа на предыдущие случаи. Скорее всего, на компьютере с ОС Windows будет установлен адаптер fieldbus-сети, а OPC-сервер будет взаимодействовать с этой сетью через драйвер адаптера. В Internet можно найти немало таких примеров. Идея подобной схемы достаточно очевидна. Сеть полевой шины работает в жестком реальном времени, а OPC предоставляет менее требовательный шлюз к этой сети из приложений более высокого уровня.

Можно назвать много других мест применения OPC: для работы с базами данных в качестве вспомогательных или промежуточных OPC-серверов и т. д. Возможности для фантазии безграничны. Одну такую фантазию хотелось бы привести. Технология DCOM, как уже говорилось, не очень пригодна для глобальных сетей. Поэтому для привлечения к OPC-технологии Internet-технологий можно набросать такой путь. Расширение Web-сервера является OPC-клиентом, собирающим данные от OPC-серверов. А на стороне клиентов запускается динамическая html- или xml-страница, получающая данные от этого Web-сервера. Ее можно сделать даже OPC-сервером для других приложений.

Полезность применения OPC с точки зрения интеграции достаточно прозрачна и вытекает из самой сути OPC. Это стандарт на интерфейс обмена данными с оборудованием. Первое преимущество — если вы заменяете какой-нибудь компонент, то нет нужды корректировать другое ПО, так как даже при замене драйвера поверх него работает OPC. Второе — если вы хотите добавить в систему новые программы, нет необходимости предусматривать в них драйверы устройств, кроме OPC-клиента, разумеется. Ну и так далее.

Состояние дел

До сих пор мы описывали технологию и ее возможности. А каково же состояние дел с ее внедрением?

Идеальной была бы следующая картина. Все в мире признают OPC своим стандартом. При этом поставщики оборудования, в том числе полевых шин, снабжают выпускаемые продукты OPC-серверами, а поставщики программ для систем управления делают собственные продукты OPC-клиентами и во многих случаях еще и OPC-серверами. При этом и те, и другие реализуют все спецификации и поддерживают все интерфейсы OPC. А все производители операционных систем встраивают в свои ОС технологии COM/DCOM, а также предоставляют сервисный инструментарий как для него, так и для OPC. И при этом все делается на высоком профессиональном уровне и очень грамотно рассказывается сборщикам систем, как это все собирать и конфигурировать. Вот тогда вопрос обмена данными в системах автоматизации можно было бы считать закрытым.

Но пока положение дел несколько иное.

В настоящее время общепризнанным стандартом является только OPC DA, а остальные спецификации только начинают завоёвывать себе место под Солнцем. Не все спецификации завершены, по крайней мере, с точки зрения интерфейса автоматизации (например, для OPC Batch уже существует версия 2.0 custom-интерфейса, а интерфейса автоматизации – только версия 1.0, для некоторых других спецификаций тоже существует отставание интерфейсов автоматизации от custom-интерфейсов; для OPC Security спецификации на интерфейс автоматизации вообще не существует).

Соответственно, широкое распространение получил лишь стандарт OPC DA. Можно сказать, что сейчас действительно очень многие производители снабжают свои продукты OPC DA-серверами. Чего нельзя сказать о других спецификациях.

Среди программ высокого уровня аналогичная картина. Спросом пользуется лишь OPC DA. Все известные нам SCADA-продукты являются OPC-клиентами, например Wonderware InTouch, CiTect (Ci Technologies), а многие из них и OPC-серверами (в частности, CiTect). Другое ПО подвержено влиянию OPC в гораздо меньшей степени.

Из операционных систем технологию COM/DCOM поддерживают следующие:

- все Windows, начиная с Windows 95. Это обеспечивается самой компанией Microsoft;
- большинство Unix-подобных ОС, включая Linux; поддерживается фирмой GE Software;
- ОС реального времени VxWorks; обеспечивается фирмой-разработчиком WindRiver; имеется поддержка OPC, встроенная в систему разработки Tornado.

В других операционных системах, насколько нам известно, поддержки COM/DCOM нет. Это не очень отрадный факт, поскольку разработчиков систем автоматизации в первую очередь интересуют ОС реального времени.

Перспективы

Итак, в настоящее время картина далеко не идеальна. Еще довольно много оборудования и ПО не охвачено OPC-технологиями. Даже технологией DA. Но нам представляется, что сейчас в мире налицо некий бум OPC, по крайней мере, в отношении опять же DA. Думается также, что Microsoft рано или поздно доведет все до желаемого уровня по всем направлениям[1]. Тем более что альтернативных вариантов пока нет. Мы имеем в виду не COM/DCOM, а именно спецификации на обмен технологическими данными. Поскольку для COM/DCOM замена как раз имеется — CORBA. Это действительно изначально платформенно-независимая технология взаимодействия приложений. Но это не обмен технологическими данными, реализующий более высокий уровень абстракции. Кстати, заметим, что на рынке имеются OPC-шлюзы к CORBA (это возможно, как и к любому другому протоколу).

Заключение

Технология OPC предлагает стандарты для обмена технологическими данными, в которые заложены самые широкие возможности. Учитывая большой авторитет вовлеченных в данную деятельность фирм можно ожидать, что технология OPC будет набирать силу. Это перспективная технология для интеграции разнородных систем. Хотя процесс становления еще далеко не завершен и есть много проблем, которые предстоит решить.

С авторами статьи можно связаться по телефону: (095) 742-6828 или по e-mail: ikutsev@rtsoft.msk.ru (И. Куцевич), grigoriev@rtsoft.msk.ru (А. Григорьев), http://www.rtsoft.ru, http://www.rtsoft.ru/products/OPC

[1] Хотелось бы верить. Но опыт показывает, что скорее Microsoft придумает новую технологию, чем доведет до блеска старую, так как главная цель MS, как, впрочем, и других компаний, — развитие бизнеса, а не технологий. — *Прим. редактора.*