ПЛАТФОРМА ARCHESTRA: ПРЕИМУЩЕСТВА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА

Компания Klinkmann

Сформулированы основные различия между системами автоматизации, построенными на базе объектно-ориентированного подхода, и системами, базирующимися на тегах. Показано, что объектно-ориентированная модель обеспечивает существенную экономию времени и средств на протяжении всего жизненного цикла систем автоматизации. Рассматривается архитектура первой реализованной в России системы автоматизации, функционирующей на Котласском целлюлозно-бумажном комбинате, выполненной на компонентной объектно-ориентрованной платформе ArchestrA.

Архитектура ПО, основанная на компонентах, долгие годы используется в мире вычислительной техники, а в последнее время широко применяются объектно-ориентированные разработки. И только с недавних пор обе эти модели начали использовать в АСУТП и в SCADA-системах. Промышленный сервер приложения (Industrial Application Server – IAS) на данный момент является единственным доступным продуктом из множества ПО для автоматизации, который поддерживает обе эти модели. IAS является частью FactorySuite A2 компании Wonderware®, входящей в Invensys PLC.

Сравнение систем, основанных на тегах, и компонентных объектно-ориентированных систем

Длительное время построение ПО систем диспетчеризации (HMI и продукты диспетчерского контроля, доступ к данным, создание скриптов, алармы и анализ данных) основывалось на использовании концепции тегов. Среда, базирующаяся на тегах, очень проста и легко переносима из проекта в проект, но ей присущи недостатки плоского пространства имен, что не позволяет связывать элементы в сложные связанные структуры. Глобальные изменения в БД тегов чаще всего выполняются вне среды разработки с использованием текстовых или Excel файлов с последующим импортированием в приложения. Повторное использование наработок в системах, основанных на тегах, обычно организовывается посредством динамического или клиент-серверного обращения, которое позволяет создавать, например, общую графическую среду и во время исполнения скрипта переключать теги для просмотра их в режиме реального времени (РВ). Более того, из-за плоской структуры приложения, необходимо внимательно отслеживать все изменения и анализировать их влияние на приложение в целом.

Объектно-ориентированные и основанные на компонентах разработки в мире информационных технологий (ИТ) первоначально относились к инструментарию, который освобождал разработчиков от рутинных операций программирования и при этом позволял повторно использовать общие компоненты при разработке.

Однако этот инструментарий не совсем подходит для промышленной среды. Во-первых, системные интеграторы и инженеры-технологи обычно не являются программистами. Более того, существуют некоторые ключевые отличия в архитектуре ИТ-приложений и приложений автоматизации. Например, ИТ-приложения общего назначения обычно включают доступ к БД посредством недетерминированных, основанных на формах интерфейсов, которые чаще относятся к банковскому обслуживанию в оперативном режиме, составлению отчетов, финансовому учету или статическому просмотру информации. И наоборот, приложения управления производством и диспетчерского контроля организуют доступ к данным ПЛК в режиме РВ, производят сложные расчеты для определения количества продукции и отображают эти данные в режиме РВ в графической клиентской среде, а также обеспечивают чтение и запись в БД информации, связанной с производством и эксплуата-

Объектно-ориентированное программирование (ООП) – модель языка программирования, основой которого является "объект" и данные, а не "действие" и не логика. Ранее программа рассматривалась как логическая процедура, которая принимает входные данные, обрабатывает их и выводит обработанные данные. Программирование считалось построением логики, а не описанием данных. Для объектно-ориентированного программирования ключевыми элементами являются объекты, которыми необходимо управлять.

Компонент. В объектно-ориентированном программировании и распределенной объектной технологии компонентом называется многократно используемый программный компоновочный блок, который может взаимодействовать с другими компонентами на том же или на другом ПК в распределенной сети для формирования приложений. Примерами компонентов могут быть: кнопка в графическом пользовательском интерфейсе, калькулятор с простейшим набором функций, интерфейс для администратора БД. Компоненты могут быть развернуты на разных серверах в сети и связываться друг с другом для обмена необходимыми услугами. Компоненты запускаются в оболочке, называемой контейнером. Примерами контейнеров могут быть страницы на Web-сайтах, Web-браузеры и текстовые процессоры.

Следовательно, термин ***"объектно-ориентированный компонент"*** относится к системе, которая использует объектно-ориентированный подход для создания приложений, основанных на компонентах, которые могут быть распределены среди любого числа компьютерных ресурсов. Поскольку поддерживается связь от компонента к компоненту и от среды разработки к среде функционирования, то нет необходимости останавливать работу всей системы при внесении изменений.

Таблица 1

Параметры	Объектно-ориентированная компонентная архитектура		Архитектура, основанная на тегах	
	Разработка	Исполнение	Разработка	Исполнение
Структура приложения	Иерархическая – создание компонентов, используя объектно-ориентированную методологию	Иерархическая – компоненты представляют собой физические устройства, которые по мере необходимости запускаются согласованно из различных мест	Иерархическая – создание графического содержания, иногда с использованием объектной ориентации	Плоская – единые экземпляры ПО, которые запускаются на одной или многих машинах как отдельные "приложения"
Разработка графики	Выполняется в последнюю очередь	–	Выполняется в первую очередь	–
Скрипты	Разрабатываются в шаблонах объектов как часть компонента		Разрабатываются отдельно, далее связываются с графическим представлением	
Проверка на соответствие стандартам	Строго		Нестрого	
Глобальные изменения в проекте	Наследуются из шаблонов объектов	Возможно распределять и заменять/усовершенствовать компоненты	Базируются на графических средствах или инструментарии, например Excel	Требуют перекомпиляции проектов
Представление данных	Логические компоненты как физические устройства (например, клапаны, насосы) или логические устройства (например, ПИД-регулятор и расчеты) как объекты и компоненты	–	Графические устройства как объекты и теги	–

цией. В табл. 1 приведена сравнительная характеристика объектно-ориентированной компонентной архитектуры и архитектуры, основанной на тегах.

Таким образом, задачи ИТ-приложений и производства существенно отличаются, и одни и те же объектно-ориентированные компоненты не могут удовлетворить требованиям обеих систем одновременно. Объектно-ориентированная компонентная архитектура IAS специально разработана для упрощения разработки, управления и эксплуатации больших и сложных систем диспетчеризации.

Преимущества для промышленных приложений

В объектно-ориентированной SCADA-системе объекты приложения содержат параметры, связанные с устройствами, которые они представляют. Например, объект "клапан" может содержать все события, алармы, безопасность, связи и скрипты, связанные с устройством. Однако объекты не просто представляют оборудование. Они могут также моделировать общие расчеты, методы доступа к БД, ключевые показатели производительности (KPIs), диагностику состояния, планирование ресурсов предприятия (ERP), операции по передаче данных и многие другие задачи, которые необходимо решать посредством информационной системы предприятия. Так

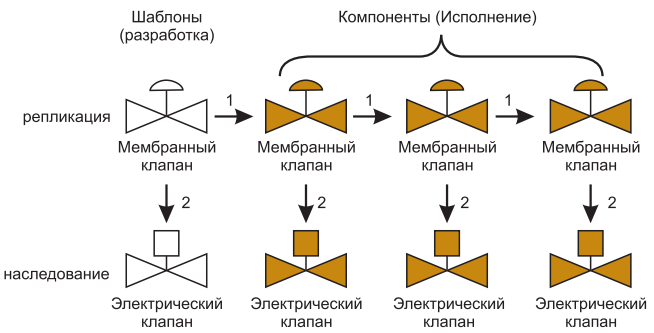

Рис. 1.
1. Репликация – создание компонента из шаблонов объекта.
2. Наследование – это внесение изменений через шаблоны объектов. Изменения могут распространяться на выбранные или на все связанные компоненты

как эти операции являются модульными, то их можно легко добавить в любую часть приложения. Например, если существует стандартный метод организации расчетов и рабочее задание на техническое обслуживание насоса, то можно инкапсулировать эти функции в объект и применять их к любому насосу в системе.

Промышленные приложения обычно имеют некоторые общие компоненты, которые делятся на следующие типы: устройства и оборудование; способ эксплуатации; средства измерений; расчеты; графические экраны.

Это дает возможность использовать шаблонный подход, который позволяет разрабатывать маленькие программы (шаблоны) как объектные модули, из которых впоследствии создаются экземпляры объектов и собираются вместе, таким образом формируя приложения. Этот подход на данный момент поддерживает большинство производителей. Объектно-ориентированная компонентная SCADA-система отличается тем, что после того, как из шаблонов созданы экземпляры, можно вносить изменения в шаблон, и эти изменения автоматически распространяются на все дочерние экземпляры. То есть разрабатывается шаблон объекта "родитель" и компоненты являются либо копиями, либо наследуют родительский объект. Все объекты-потомки связаны с "родителями" поэтому изменения в родителе отображаются на всех потомках (рис. 1).

Большинство функций в интегрированной среде разработки ArchestrA основаны на таких действиях, как перенос объекта с фиксацией на новом месте (drag-and-drop), выбор объекта для выделения (click-to-select) или заполнение текстовых окон (fill-in-the-text box). Целью является создание шаблона один раз и дальнейшее его многократное использование в приложениях. В большинстве случаев разработка объектно-ориентированных приложений намного проще, чем модификация скриптов по строчке. К тому же, число синтаксических ошибок и ошибок периода исполнения минимизируется за счет использования инструментария, который определяет специфические правила.

Пути минимизации затрат на протяжении жизненного цикла системы

Изначально пользователи SCADA-систем создавали графику и связывали ее с тегами, которые были представлены в виде адресов в ПЛК или в системах управления. Все внимание сосредотачивалось на локальном компьютере и на прикладном ПО. Этапы разработки традиционного SCADA-приложения, основанного на тегах, включают: разработку нового HMI приложения на локальном ПК; создание графического представления для окон; импорт описания тегов из ПЛК или конфигурацию их вручную; определение для каждого тега алармов и скриптов для выявления событий (Event Detection Scripts); связывание тегов с графическими элементами; создание анимационных скриптов или связей; определение тегов ввода/вывода и связывание их с приложением. Если приложение будет развертываться в клиент-серверной среде, то оно перестраивается для централизации алармов, определения событий, архивирования исторических данных, графического представления и серверов ввода/вывода. Внесение изменений в систему требует останова приложения; изменения вносятся в многочисленные скрипты, теги, ссылки на БД для внедрения новых функциональных возможностей. Также необходимо перезагружать новые HMI приложения на каждой рабочей станции.

IAS и ArchestrA Integrated Development Environment (интегрированная среда разработки – IDE) открыли новую эру в разработке ПО, позволяя создавать законченную объектную модель предприятия. В данном случае разработчики не обязаны вникать в сложности компьютерного окружения, что позволяет сосредоточить все усилия на "моделировании" производственного оборудования, создании гибких производственных модулей и процессов, которые участвуют в диспетчерском контроле производства.

Модель предприятия строится один раз, после чего функции диспетчерского контроля легко выполняются. Небольшие затраты на этапе построения шаблона объекта ведут к значительному повышению эффективности разработок. Рассмотрим десять простых шагов создания приложения диспетчеризации с использованием IAS.

1. Проводится обзор участков для составления структурной схемы технологических операций или процессов.

2. Составляется список одинаковых (повторяющихся) частей оборудования. Описываются отдельные производственные участки.

3. Шаблоны объектов конфигурируются для каждого общего устройства или компонент в установке. Это процесс определения стандартов для диспетчерских приложений и для любых других приложений, которые будут разработаны в дальнейшем.

4. Шаблоны объектов могут содержать друг друга, что позволяет строить более сложные устройства.

5. Шаблоны объектов имеют свойства, которые предоставляют возможность ввода/вывода данных в ПЛК или в АСУТП. Эти свойства в дальнейшем связываются с устройствами ввода/вывода посредством Device Integration Objects (объекты интеграции с устройствами).

6. Приложения могут быть собраны, путем простой операции "drag and drop" внутри IDE.

7. Компоненты в дальнейшем привязываются к группам безопасности.

8. Модель, созданная в среде IDE, может быть развернута на главном компьютере.

9. Графическое представление конфигурируется с помощью SCADA-системы InTouch®.

10. Приложение разрабатывается один раз и в дальнейшем систему легко сопровождать и обслуживать. Изменения, внесенные в шаблоны объектов (Object Templates), могут легко распространяться на дочерние компоненты (Child Components).

Причины минимизации затрат на протяжении всего жизненного цикла системы приведены в табл. 2.

Возьмем простой пример, чтобы показать минимизацию затрат при использовании разработчиками SCADA-систем объектно-ориентированной компонентной технологии. Рассмотрим приложение диспетчеризации участка, состоящего из 27 клапанов, каждый из которых имеет 6 точек ввода/вывода, за которыми необходимо непрерывно наблюдать. В традиционной SCADA-системе необходимо создать 162 тега (27 клапанов х 6 точек ввода/вывода на клапан). В компонентной объектно-ориентированной SCADA-системе создается шаблон для одинаковых объектов типа "клапан" и компоненты, которые представляют каждый клапан в отдельности, т.е. дублируют шаблон. Используя традиционную SCADA-систему, основанную на тегах, для создания приложения необходимо потратить 0,4 часа на каждый тег. При этом не учитывается время на построение графического представления или на разработку логики управления ПЛК. Чтобы разработать шаблон объекта "клапан" необходимо затратить 2 часа и еще 20% (или 0,4 часа) на каждый компонент для создания конкретного клапана в приложении. При этом шаблон

Мы должны стремиться не к тому, чтобы нас всякий понимал, а к тому, чтобы нас нельзя было не понять.

П. Вергилий

Таблица 2

Этапы жизненного цикла системы	Причины минимизации затрат
Создание приложения	За счет однократного определения шаблонов объектов и дальнейшего их многократного использования в проекте; использования шаблонов и приложений, разработанных для одного проекта, в последующих проектах
Внесение изменений в приложение	За счет распространения изменений, произведенных в шаблоне объекта, на все дочерние компоненты. Когда во время разработки необходимо внести изменение в сложное приложение, эффективность значительно повышается
Техническое обслуживание и сопровождение распределенной системы	Путем удаленного мониторинга, дистанционного внесения изменений и развертывания на всех узлах сети. Пользователям нет необходимости посещать каждый производственный участок для сопровождения и внесения изменений, что экономит время и деньги

Таблица 3

Традиционный HMI, основанный на тегах, (X)	Компонентная объектно-ориентированная SCADA, (Y)	Уменьшение затрат (X-Y)
Начальные затраты на разработку		
162 тега × 0,4 ч на тег = 64,8 ч	(2 ч на создание шаблона объекта × 1 шаблон) + (27 экземпляров клапанов × 0,4 ч на один экземпляр) = 12,8 ч	52 ч или 80%
Затраты на изменение приложения		
64,8ч × 10% усилий на внесение изменений = 6,48 ч	2 ч на построение шаблона объекта × 10% усилий на изменение шаблона = 0,2 ч	6,28 ч или 96%

объекта инкапсулирует скрипты, алармы, безопасность, события, конфигурирование исторических данных и устройства связи. В системе, основанной на тегах, все это должно быть запрограммировано с помощью тегов дополнительной памяти (табл. 3)

Теперь предположим, что пользователь просит изменить 10% приложения. В SCADA продуктах, основанных на тегах, можно сказать, что 10% усилий от разработки пойдет на внесение изменений. В компонентной объектно-ориентированной SCADA-системе благодаря связи "родитель-потомок" между объектами и компонентами 10 % усилий необходимы только для изменения шаблона (табл. 3).

Обычно эффективность проектирования снижается при разработке диспетчерских приложений и SCADA приложений по мере увеличения числа разработчиков. IAS помогает решить эти проблемы, используя компонентный объектно-ориентированный подход к разработке приложений, который позволяет создать один раз шаблон объекта для устройства и в дальнейшем использовать его в приложении. Также создается централизованное хранилище для разработки приложений и для интегрированной среды разработки, что позволяет нескольким пользователям модифицировать и создавать приложение одновременно.

Первым реализованным в России проектом на базе ArchestrA Technology компании Wonderware стала автоматизированная система оперативно-диспетчерского управления (АСОДУ) Котласского целлюлозно-бумажного комбината. Рассмотрим особенности реализации системы и ее архитектуру.

АСОДУ Котласского целлюлозно-бумажного комбината

Разработанная в 2004 г. НПФ "Ракурс" (С.-Петербург) АСОДУ производства сульфатной беленой целлюлозы Котласского ЦБК предназначена для представления оперативных данных о состоянии производства диспетчерской службе, руководству производства целлюлозы, а также руководству комбината.

АСОДУ решает следующие задачи:

- автоматический сбор технологических параметров производства и формирование БД;
- представление мнемосхем, оперативных и отчетных документов, графиков, характеризующих состояние производства, основного оборудования, расчет простоев;
- сигнализацию технологических и системных нарушений;
- контроль производительности и выработки полуфабрикатов основными отделами производства;
- контроль запасов химикатов и полуфабрикатов в емкостях;
- оперативное регулирование выработки и потребления волокна;
- подготовку данных для АСУ предприятия.

Концептуально АСОДУ содержит три уровня иерархии (рис. 2). На нижнем уровне производится сбор и первичная обработка информации, которая выполнена на базе контроллеров Siemens S7-300 и станций распределенного ввода ET200M, объединенных сетью Profibus.

На среднем уровне системы находится сервер производства. Он отвечает за хранение данных, решение расчетных задач, а также за организацию взаимодействия с клиентами. Ядром системы является новый программный продукт IAS фирмы Wonderware. Использование этого продукта дает следующие преимущества:

- концентрацию всей бизнес-логики в рамках единой иерархической производственной модели, что существенно сокращает время на разработку и сопровождение системы;
- отсутствие ограничений на размер приложения, возможность масштабирования;
- возможность внесения изменений в отдельные узлы и элементы без остановки всей системы;
- мощный скриптовый язык с возможностью использования библиотек Microsoft.NET;
- встроенную систему безопасности;
- тесную интеграцию с другими продуктами Wonderware.

В реализованном проекте программные продукты IAS, InSQL, InTouch фи-

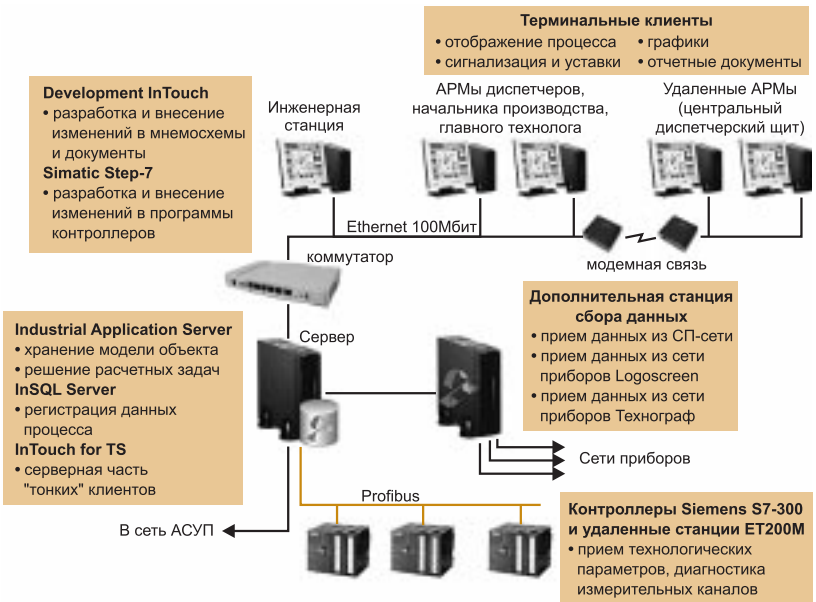

Рис. 2. Функциональная схема АСОДУ

зически установлены на одном сервере на базе Intel Xeon, мощности которого с запасом хватает для решения этих задач. Однако при дальнейшем расширении системы возможна установка каждого продукта на выделенный сервер для увеличения общей производительности.

Архивирование принятых и рассчитанных данных производится в БД PB Wonderware Industrial SQL. Открытая архитектура InSQL позволяет построить оптимальную структуру для Internet/Intranet сетевых приложений, легко осуществить интеграцию с уже имеющейся системой АСУП, а также использовать стандартные средства отчетности.

Для возможности организации так называемых "тонких" клиентов на сервере установлена специальная версия InTouch для терминальных сессий - InTouch Terminal Services.

В качестве клиентских мест используются существующие на производстве компьютеры под управлением разных ОС Windows 98, 2000, XP с установленным терминальным клиентом Microsoft. Использование терминальных клиентов позволяет установить InTouch один раз на сервере, а затем для каждого пользователя запускать свою терминальную сессию, которая имеет ту же функциональность, что и обычный InTouch с возможностью просмотра мнемосхем, графиков процесса, отчетных и оперативных документов без необходимости инсталляции InTouch на каждой клиентской машине.

Основные преимущества использования терминальных клиентов заключаются в централизованной эксплуатации и управлении приложениями. Администратор системы, не вставая со своего места и ни на минуту не прерывая работы клиентов, может внести необходимые изменения в систему, которые тут же отобразятся на экране пользователя. Это позволяет проводить быстрое и эффективное внедрение новых клиентских приложений, упрощает процедуры резервного копирования и восстановления информации, а также снижает общую стоимость системы. Сетевой трафик, создаваемый одним терминальным клиентом составляет около 5 Кбит/с при обновлении информации на экране один раз в секунду, что позволяет использовать даже медленные модемные соединения при сохранении комфортных условий работы.

НПФ "Ракурс" имеет богатый опыт в области создания систем диспетчерского управления, которые успешно эксплуатируются на различных производствах Светогорского и Архангельского ЦБК. С применением нового продукта IAS компании Wonderware система АСОДУ производства целлюлозы Котласского ЦБК была разработана и внедрена в сжатые сроки. Кроме того, базируясь на стандартных решениях от Wonderware, разработанная система имеет в своей основе открытую архитектуру, поддерживает большое число контроллеров и измерительных устройств, предоставляет богатые сетевые возможности для Intranet/Internet решений и связи с ERP-системами. Благодаря этому, внедренная система АСОДУ имеет большой потенциал для дальнейшего расширения и развития, а использование новых возможностей, предоставляемых IAS, делает эту задачу простой и прозрачной для разработки и дальнейшего сопровождения.

Преимущества от использования IAS:

- возможность создания модели, наглядно отображающей структуру всего производства;
- создание единого информационного пространства в рамках всего производства;
- возможность интеграции уже существующих автоматизированных систем производства;
- способность масштабирования 250...1000000 точек ввода/вывода с переразмещением объектов IAS на другие ПК с целью выравнивания загрузки;
- возможность бесшовной стыковки IAS с IndustrialSQL Server;
- наличие централизованного источника производственных данных, которым является IAS, для унифицированных клиентских мест: InTouch View ("легкий" клиент), Internet Explorer (WEB клиент) на базе SuiteVoyager, Terminal Client ("тонкий" клиент) на базе Terminal Server (Microsoft или Citrix);
- возможность создания резервированной системы;
- возможность централизованно поддерживать, расширять и обслуживать все созданные производственные приложения;
- наличие средств связи с ERP-системами.

Заключение

В наше время предприятиям тяжело найти свободные средства на внесение изменений в существующую систему управления. В большинстве случаев производственное оборудование и аппаратное обеспечение для управления процессом настолько эффективно, насколько это возможно. Необходимые изменения лучше всего вносить в ПО, расположенное во главе иерархии системы управления (управление активами, ERP интеграция, оптимизация производства и приложения для построения отчетности). Это области, в которых IAS эффективно работает как платформа для интеграции приложений и обеспечивает интеграцию данных и безопасность коммуникации, что необходимо при связывании отдельных островков автоматизации. Имея в своем распоряжении более 800 интерфейсов, ПО Wonderware может связать практически все устройства на предприятии.

Отметим, что компания Wonderware разработала калькулятор FactorySuite A2 Life-Cycle Savings Estimator, чтобы дать понять пользователям, каким образом их приложения диспетчерского контроля или SCADA проекты могут выиграть от применения Industrial Application Server и ArchestrA Technology. Рассчитать возможное снижение затрат для конкретного проекта можно на сайте: http://fsestimator.wonderware.com

Контактный телефон (812) 327- 37-52. Http://www.klinkmann.spb.ru