invensys™

Invensys is now Schneider Electric

White Paper

The Benefits of Object-Based Architectures for SCADA and Supervisory Systems

Author: Steven D. Garbrecht, Vice President of Software and Advanced Applications Marketing, Invensys Operations Management

What's Inside:

The Benefits of Object-Based Architectures for SCADA and Supervisory Systems

1. Executive Summary

Most industrial facilities today use SCADA and supervisory HMI (human-machine interface) applications based on traditional tag-based architectures. However, readily available and technologically mature object-based architectures can deliver cost savings of up to 80% over tag-based systems. Cost calculations are explained and demonstrated. Other benefits of object architectures include easier enforcement of best practices across multi-site industrial enterprises, more consistent plant performance information and greatly enhanced system scalability and maintainability.

2. Introduction

Object-based software architectures have been around for many years in the commercial computing world. Now these architectures are being applied in process control and SCADA applications to deliver significant cost and operational benefits. In this paper we will discuss what object-based architectures are, how they improve SCADA and HMI application development and how you can calculate potential cost savings over traditional tag-based systems.

3. Tag-Based vs. Object-Based Systems

i. Tag-Based Systems

From the inception of PC-based HMI and supervisory systems, process data access, scripting, alarming and data analysis has been based on the concept of tags. These systems use a "flat" list of tags with built-in hierarchy, relationships or interdependencies.

Large, global changes to a tag system's database are typically done externally to the application, often via a text file or through tools like Microsoft Excel®. Once made the changes are imported into the application's database. Reuse of engineering in a tag-based system is commonly instituted through dynamic or client-server referencing. The system creates common graphic containing scripts that switch the tags in runtime. Because the application structure is flat, the user must then change each tag in the system and analyze how the change affects the rest of the application.

Maintenance of tag-based applications typically involves tag by tag analysis and update that can consume significant amounts of labor. Since system changes are time consuming and often involve the use of outside labor improvements to tag-based systems are limited.

ii. Object-Based Systems

The concept of object-oriented development originated in the information technology (IT) world. The goal was to provide tools that would release the developer from mundane, repetitive program tasks, while at the same time maximizing reuse of code through the development of common software objects.

As you would expect, these tools are not an exact fit for the industrial environment. For one thing, systems integrators and production engineers are typically not computer programmers. Furthermore, there are some key architectural differences between IT and production automation applications.

For example, IT applications typically involve accessing databases from non-deterministic, forms-based interfaces that accomplish things like online banking, business reporting, HR management, financial accounting or static information look-ups.

Conversely, supervisory control, manufacturing execution systems (MES) and plant intelligence applications involve acquiring real-time process data; performing sophisticated calculations to determine flows and production values; displaying real-time data on operator displays or process reporting or analysis tools; and storing this data to process historians or production-related databases.

The two environments are different enough to dictate that object-based tools be purposely built for the industrial environment. The ArchestrA® System Platform uses an object-based architecture which is called ArchestrA. It is specifically designed for industrial customers who develop, manage and maintain supervisory systems.

iii. Comparing the Two Systems

The following table contrasts object-based and tag-based architectures.

	Object-Based Architecture		Tag-Based Architecture	
	Development	**Runtime**	**Development**	**Runtime**
Application Structure	Hierarchical – Objects created using object-oriented workflow methodology	Hierarchical – Components represent physical devices and can coordinate with components in different computers	Hierarchical – Graphical content sometimes created using object orientation	Flat – Monolithic instances of software run on one/multiple machines as separate "applications"
Graphics Development	Performed last	N/A	Performed first	N/A
Scripting	Developed in object templates then deployed to specific runtime application	N/A	Developed separately, linked to a graphical interface	N/A
Standards	Strictly enforced	N/A	Not strictly enforced	N/A
Application Changes	Propagated from object templates	Objects can be distributed, exchanged or enhanced	Based on graphics or changed using tools like Excel	Require recompilation of the application
How Data is Represented	Logical constructs such as physical devices (e.g., valves or pumps) or logical devices (e.g., PID loops or calculations) are represented as objects	N/A	Graphical devices are represented as objects or tags	N/A

4. Objects Help Drive Consistency and Enforce Best Practices

In object-based SCADA applications, application objects contain aspects or parameters associated with the asset they represent. For example, a valve object can contain all the events, alarms, security, calculations, data collection, integrations, communications and scripting associated with the asset.

Objects don't just represent plant equipment. They can include calculations, database access methods, key performance indicators (KPIs), condition-monitoring events, ERP data-transfer operations, mobile operator procedures, workflow activities and MES tasks. All of these objects can be standardized and used across all supervisory applications to drive consistency of system design and operation.

For example, a standardized work request object can be created then added to any plant asset, such as a pump, within a supervisory application ensuring a consistent and standardized approach to initiating work requests.

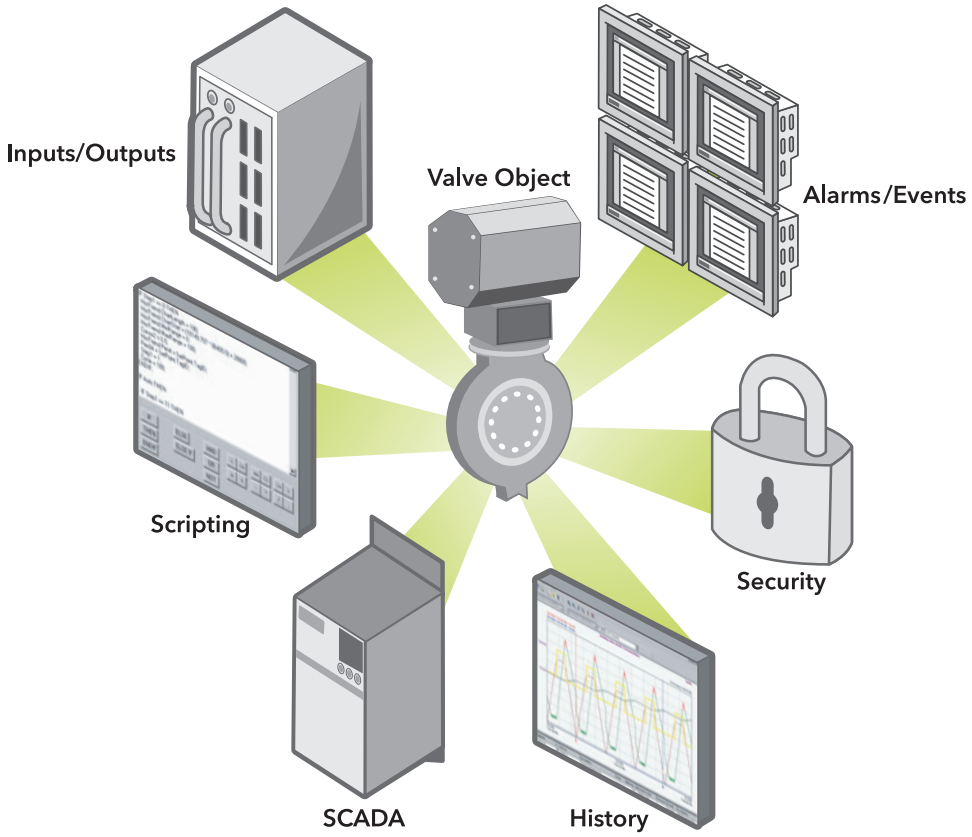

Figure 1: An object template contains valuable information about Alarms/Events, Security, History, SCADA, Scripting and Inputs/Outputs.

Industrial facilities controlled by a modern supervisory system share a set of common characteristics:

- Plant devices and equipment
- Operating procedures
- Process measurements
- Calculations
- Graphical operator displays

Object-based architectures facilitate a cookie-cutter type approach to supervisory system design, in which system functionality such as the characteristics mentioned above can be encapsulated into object templates, duplicated and joined together to form a complete supervisory system.

A key advantage of the object-based approach is this concept of object templates. Below is a graphical representation of how object templates allow rapid system design and change propagation.

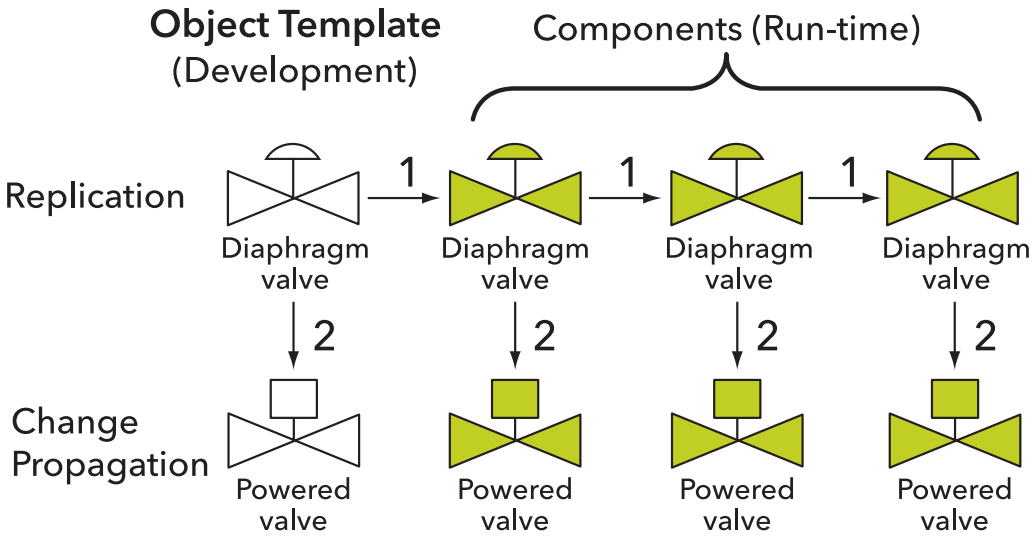

Figure 2: Replication of Objects and Change Propagation

The first row of Figure 2 shows the replication of an object template representing a diaphragm valve and all of its inherent characteristics. Replication is the process where run-time instances or components are created from object templates. The next row illustrates how a change to a characteristic of the valve (manual activation to powered activation) is propagated throughout all the run-time instances of the valve object.

This parent-child relationship is a key benefit of the object-based approach. Changes are automatically propagated to all run-time instances of the object template including any number of supervisory applications running in different physical locations. No one has to visit every site to make the required changes which could run into the hundreds or even thousands of instances of common assets like a valve.

- Application creation is optimized by using object templates to automatically generate run-time components
- Project changes are easily accommodated by making changes in the object template and having the components inherit the changes via change propagation
- Ongoing system changes and expansions are easier and more cost effective because of automated replication and change propagation

5. Develop Once, Reuse Many Times

The ArchestrA System Platform's object-based approach greatly simplifies supervisory application development and maintenance. The software's Integrated Development Environment (IDE) enables the use of simple Windows drag-and-drop, click-to-select or fill-in-the-text box techniques to create and manipulate objects.

In most cases, this approach is far easier than modifying scripts line-by-line. In addition, the number of syntax and run-time errors is reduced because the IDE enforces system-specific rules. What's more, users can develop object templates once and then reuse them many times in many different applications- maximizing return on engineering.

6. Object-Oriented HMI Graphics

The term "object-oriented graphics" has been used in the SCADA/HMI environment since the early 1990s. Object-oriented graphics enable users to build a symbol and replicate it across HMI application. They can then edit the symbol and easily distribute the changes to all the similar symbols at the same time.

While this is useful functionality, SCADA/HMI applications require more than graphics. Much of the development work that goes into designing supervisory applications is spent on creating functionality such as:

- Alarm monitoring
- Animation scripts
- Security scripts
- Supervisory scripts
- Historical data storage
- Integrations to other applications or databases
- Event detection
- Flow and movement calculations
- Device integration
- Workflow

To fully realize the benefits of a object-based architecture, a SCADA/HMI system needs to include all of these functions or capabilities in object templates including graphics.

7. Development Advantages of Object-Based Architectures

i. Tag-Based Architectures

From the inception of PC-based HMI and SCADA software, users have built operator graphics and linked them to tags that represented addresses in a PLC or a control system. The steps below represent the typical development process for a traditional tag-based SCADA application:

1. Single development computer used.
2. Operator graphics and screens are created for the application.
3. Tag definitions are imported from the PLC or manually configured.
4. Alarm and event-detection scripts are defined for each tag.
5. Tags and associated I/O are linked to graphic elements.
6. Graphical animation scripts or links are created.
7. Changes to the system require shutting down the application, making changes to the many scripts and tag database references to enable the new functionality. The application is then re-installed on each operator workstation.

i. Object-Based Architectures

Object-based architectures associated with supervisory and SCADA/HMI applications have been pioneered by Wonderware®. The ArchestrA System Platform and its core development tool, the Integrated Development Environment (IDE) has fundamentally changed the way supervisory and SCADA/HMI applications are developed.

Using the Integrated Development Environment, the application developer creates a single plant model using re-usable object templates. The developer is thus removed from the complexities of the computing environment and empowered to concentrate on modeling the production facility. The developer can focus on the different assets and production processes that comprise plant-wide supervisory control.

After the plant model is captured, it is easy to implement supervisory control functions. A small investment in creating object templates yields big results in engineering productivity. Creating a supervisory application using the ArchestrA System Platform involves:

1. A site survey conducted to understand the layout of the production operation or process.
2. Creation of a list of similar pieces of equipment/assets. Distinct areas of operation are also identified.
3. Object templates are configured for each common asset in the facility including HMI graphics. This key step enables best practices and standards to be created for use in all future application projects.
4. Device or component object templates can be contained within each other to create complicated pieces of equipment.
5. Device object templates have attributes which represent real I/O available in the PLC or control system. These attributes are then linked to the I/O through device integration objects (DI Objects).
6. The application can then be assembled in the IDE by using simple drag-and-drop operations.
7. Application objects are then assigned to security groups.
8. The plant model created in the IDE can now be deployed to the computers that will host the application.
9. Once the application is developed, system maintenance is easy. Changes made to object templates can be propagated to the child components found in deployed applications.

8. Lifecycle Savings

Object-based architectures can provide significant saving during their entire lifecycle. These saving can be categorized into four basic areas, as illustrated in the following table.

Savings Area	Explanation
Initial Development Savings Related to Application Generation	This represents the savings that result from the time saved when users develop applications by defining object templates once and then using these templates multiple times.
Initial Development Savings Related to Application Changes	This represents the development savings gained through the ability to propagate changes from object templates to all the run-time instances derived from those templates. When multiple application changes are requested during development, the savings can really add up.
Maintenance Savings Throughout the System's Lifecycle	Using a distributed system significantly reduces maintenance costs through the ability to remotely monitor, change and deploy software to all HMI computers on the network. This is especially important for geographically distributed networks because users can save both time and money by eliminating the need to travel to each site for maintenance or upgrades.
Savings Across All Sites	These savings result from re-using the templates and applications created for this project on other projects. Companies use this to drive standards in their projects. This is particularly beneficial for system integrators, value- added resellers (VARs), original equipment manufacturers (OEMs), machine builders and facility operators.

Here's a simple example to illustrate how object-based development can lower costs.

Let's assume we are developing a plant supervisory application that has, among other things, 27 double seat valves, each having six process parameters (I/O) that will be monitored. These are I/O points in the PLC that measure the performance of this valve.

In a traditional tag-based system, 162 tags (27 valves * 6 parameters (I/O) values per valve) would be created. In an object-based SCADA system, a common valve object template is created and objects that represent each individual valve are instantiated or replicated from that object template. Now assume that it takes 0.4 hours per tag to develop the application using a traditional, tag-based SCADA system. This does not include process graphics or PLC control logic development. Let's estimate that it takes two hours to develop a valve object template and an additional 20% more (or 0.4 hours) per object instance to customize each individual valve in the application.

Device Example:

Device Type	Number of Instances	I/O per Instance
Double seat valve	27	6

Individual Estimations:

Device Type	Number of Instances	I/O per Instance
Double seat valve	27	6

Remember that an Object Template encapsulates scripting, security, alarming, events, history configuration and device communications. In a tag-based system, all of this needs to be programmed using additional memory tags. Now let's compare the total time to develop the application using each type of development approach.

Initial Development Effort:

Traditional, Tag-Based HMI	Component Object-Based SCADA	Savings
162 tags * 0.4 hrs. per tag = **64.8 hours**	(2 hrs. * 1Object Template) + (27 Valve Instances * 0.4 hrs per instance) = **12.8 hours**	**52 hours** or 80%

This is impressive savings -- even if you estimate half of this number, you save 40% in development costs!

Now what happens if a changes is required that affects 10% of the application? Using tag-based development, it's reasonable to assume that 10% of the effort spent on the original development would be required to make the changes. However, using object-based development such as the Wonderware ArchestrA System Platform, the 10% change effort only needs to be applied to the object template -- because of the parent-child relationship between objects and components. In this situation, the additional savings can be calculated like this:

Application Change Effort:

Traditional, Tag-Based HMI	Component Object-Based SCADA	Savings
64.8 hrs. * 10% change = **6.48 hrs.**	2 hrs. per Object Template * 10% change = **0.2 hours**	**6.28 hrs** or 96%

9. Object-Based Development is the Future

Object-based architectures provide compelling advantages in the development and maintenance of SCADA and supervisory systems. When evaluating architectures is it important that the following technical aspects be considered, including:

- Does the development tool provide a realistic model of plant equipment and manufacturing areas, processes and production lines?
- Can network security be easily integrated into the application including centralized security configuration?
- Does it offer flexible device connectivity and cost effective tools to interface with all field devices in the plant?
- Does it provide centralized diagnostic utilities?
- Can the development environment allow application scaling from a single node to many nodes without re-architecting the entire application?
- Can HMI applications be remotely deployed to computers across the network?
- Does the development tool provide a unified namespace that facilitates tag browsing across the entire PLC network, both in run-time and in off-line development?
- Can computing loads be distributed across multiple computers?
- Does the system provide cost-effective redundancy using commercial off-the-shelf virtualization technology?
- Is the alarm sub-system distributed?
- Is historical archiving defined during HMI development or is a separate tool required?

A modern SCADA system should be able to offer all of the above.

10. Summary

Economic conditions facing industrial facilities dictate that engineering productivity be maximized along with production agility. Today's object-based architectures for the development of supervisory and SCADA applications offer up to 80% savings in engineering costs over tag-based architectures.

Being able to build once and use many times is critical for managing project costs and object based development allows best practices to be encapsulated into applications and standardizes across the enterprise. Object based systems can be quickly enhanced or changed to respond to a market condition – improving agility. Additionally, object-based process graphics not only allow engineering reuse but provide a uniform look and feel which reduces operator training requirements and allows easier acceptance of system changes by operators.

A single "namespace" allows existing control systems to be kept and enhanced through an object based supervisory "layer." No need to "rip and replace" existing systems while allowing new system capabilities to be added for a fraction of the cost of an entirely new supervisory system.

So, object-based architectures are a very compelling alternative to traditional tag-based systems and allow compelling cost savings and operational flexibility.

Invensys Operations Management • 5601 Granite Parkway III, #1000, Plano, TX 75024 • Tel: (469) 365-6400 • Fax: (469) 365-6401 • iom.invensys.com

Rev. 04/15 PN IN-0222