

## Evolution of Object Based Control

The problems facing industrial automation have not changed much in the last hundred years - only the applied solutions have changed. Consider the following, seemingly disparate concepts:

- Open Systems: the ultimate end-user nirvana, a long-standing goal of all standards committees and the antithesis of classical control system vendor behaviour.
- Object-oriented technology (OOT): the promise of reusable software components and virtually extensible product life cycles, a promise as yet unrealized for industrial automation users.
- The "industrial" PC: a computing environment of various physical varieties, enabled through a common, industry endorsed operating system with a consistent networking paradigm.

Today we stand in the twilight of the third industrial revolution, eagerly awaiting the dawning of the fourth. The first began when industrial machinery was invented to reduce repetitive physical labor content for manufactured goods.

It ran its course of over 75 years until a clever engineer decided to replace classical relay logic or servo control with more sophisticated special computer-based solutions, which are known today as Programmable Logic Controllers and Distributed Control Systems.

This second revolution was only 20 years young when the PC began to press upon the industrial automation threshold, comprising a third revolution that is now only 10 years along. Each of these revolutions has had a profound impact in the behavior, efficiency and overall economics of industrial automation. Each has taken us closer to a utopian concept of open, universal solutions. At this threshold we stand today.

For the past few years it has been generally accepted that this fourth industrial revolution would be driven by a software initiative and supported by a wide range of interoperable, commodity hardware.

It is also reasonable to expect that the viability of this revolution pivots around the concept of openness and distributed solutions; that it must solve automation problems better, easier, cheaper and faster than heretofore available solutions; and that it must reduce the wide disparity of solution sets, yield huge economic benefits and do all of this in the face of a growing deficit of competent software developers.

The fourth industrial revolution has begun and it is embodied in the concept of Object-Based (Control) Solutions. What are the prevailing conditions that have enabled the emergence of this revolution? What needs does it satisfy? What benefits will be realized as Object-Based Control begins to provide solutions for the future of industrial automation?

## Economics - the driving factor

The basis of any revolution is invariably derived from economics with competitive underpinnings, and a growing, "I need it now" demand from the ultimate consumers. Technology almost never wins unless it satisfies these economic needs, and yet technology is invariably the response. Classical responses to some of the more significant issues has been underwhelming.

Ideally, a response to an issue should be timely, effective and produce immediate and long ranging benefit.

How does Object-Based Control purport to address these issues?

### Open Strategies

In order for a system to be truly "open," it must be comprised of easily replaceable, standard, hardware and software modules. You cannot obtain an open system from a "closed" company that practices account control by forcing you to purchase every component from the same vendor, whether that coercion is based upon contractual or technological constraints.

From the ground up, an open solution must be based on a hardware/software platform that follows a model, loosely referred to as Moore's Law: "Computing power will double every eighteen months."

While a strict quotation of Gordon Moore, co-founder of Intel, properly refers to the number of transistors deployed in future versions of Intel processors, we have all enjoyed the benefits of this reality

This benefit accrues to the wide range of PCs available for today's industrial automation requirements. These range from standard desk-top devices, to industrially hardened systems, to plug-in CPU cards to more traditional control systems, such as PLCs and DCS.

This hardware platform is powered by Microsoft® Windows XP/2003 , which has become the de-facto standard operating system for industrial automation (IA) computing. It provides the software and network infrastructure to anchor the new generation of IA solutions.

Concurrently, widespread familiarity with the Internet is driving user expectations on the factory floor toward a similar paradigm: Graphical tools, large monolithic software code being reduced to applets and objects in the form of OCX's, ActiveX components, and standard COM modules that simply inter-operate within single workstations or among several PCs connected with Internet or Intranet technology.

## Universal Control

There is a wide disparity of control solutions, which characterize the evolving IA market. Motion and Computer Numerical Control (CNC), robotics and PLCs populate the discrete manufacturing environs, while batch, simulation and DCS systems serve the process market.

Every facility requires some building automation system for security, lighting and environmental. And there are markets that have traditionally employed general-purpose computers for control rather than PLC or DCS systems, i.e. semiconductor fabrication and electronics manufacturing.

Imagine the benefit of a singular distributed system, with a common user interface, a common event and fault management system serving all of these seemingly disparate requirements equally well. Every programmer, every operator, everyone tasked with systems maintenance may be trained on any/all aspects of the system. Such a system promises, at a minimum, greatly reduced personnel costs, fewer people and far better mobility among human resources.

Such a system shall hereafter be referred to as a Universal Control System. If this were the only benefit of the Object-Based Control model, it would provide a huge return on investment (ROI), but the benefits extend way beyond this desirable goal.

### Promise of Reusable Software Components - Realized!

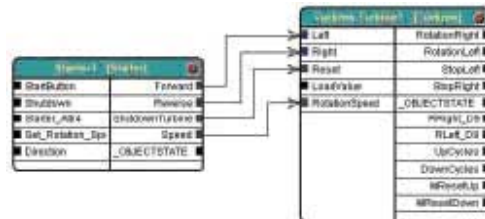
OOT has long promised reusable software components, a promise office automation users have begun to realize on the Internet. Yet, the industrial automation sector has some unique requirements that transcend merely dragging an object down from the Web and plunking it into the middle of a mission critical control environment - security and robust, long term behavior, to name two of the most important.

It has only been recently that the computing environment has had the power to support the huge computer bandwidth demands that object technology requires. Finally, few in the IA sector has been willing to embrace the requisite jargon and immature software tools, which have historically been provided in support of OOT solutions.

Object-Based Control and Moore's Law, which we've already discussed, solve these problems. OBC takes the user way beyond the need to understand "cloud diagrams" and Rumbaugh charts. It provides an easy to understand, yet extremely powerful, graphical object model; one in which objects are simply dragged onto a control screen, connected with lines ... and they work.

These are functional objects, which, in the case of control, are easily identifiable by the work that they do:

switches, solenoids, motors, conveyors epitomize discrete functions; pumps, valves and tanks for process. The different behavior of these objects is embodied within their methods. They exchange data through their respective attributes.



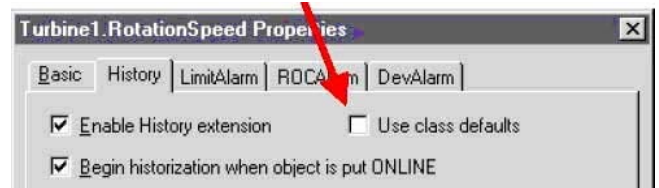
## Write Once - Run Anywhere

Each object is derived from a predefined object-class. Thus, the behavior of an object is defined once, then used over and over again.

This is referred to as "write-once-run-any-where" or WORA. Object-Based Control objects enjoy the full benefit of object inheritance ... each object derived from the class has all of the behavior and connections of the parent class.

If the class object is defined to historize one or more of its attributes, all objects will do so according to its inherited definition.

Yet, in some cases, it may be desirable to inhibit any instance of an object from such behavior. This is as simple as marking a check box in an object configuration box.



It may also be desirable to create a redundant copy of an object, one that obviously should run on a different node in the system. This, too, is as simple as unchecking a box in the object configuration dialog.

The behavior of an object is embodied in its methods. These are written in any number of languages familiar to the user: IEC-1131-3 Ladder Diagram (LD), Function Block (FB) or Sequential Function Chart (SFC), Flow Diagram, or they may be written in Java or C++.

Each method is a complete function, or sub-function, and may be assigned to any control engine, which is part of the distributed control domain. Methods may be scheduled for execution when any of their associated attributes change state, thus creating an event driven model.

Methods may be executed in a predefined sequence, prescribing a classical scan model. And methods may be scheduled to execute periodically according to a predefined priority (up to 64,000 levels).

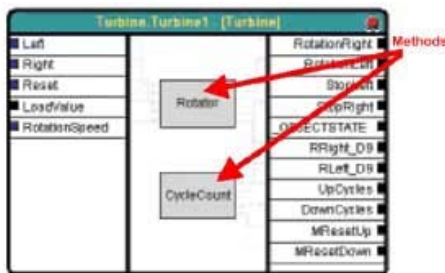
## Aggregation - Combining Objects

Another concept derived from the OOT legacy, is the concept of combining two or more objects together to create a new object with all of the characteristics and interconnects of the original objects.

This process is easily accomplished with Object-Based Control systems by simply enclosing the target objects in with the mouse pointer and pressing the aggregation button on the toolbar.

Quickly, the Object-Based Control system creates a new class, combining the methods and interconnected attributes of the original objects, and giving the user the ability to select which of the resulting attributes to expose to the outside world of this new, aggregated, class.

This new class has the same behaviour characteristics of all other classes.



## ZeroProgramming

When there are enough classes to prescribe each of the major functions of a control environment, the configuration of a complex control strategy is rather trivial. In fact it can be done without having to write a line of control code.

This is the Object-Based Control, ZeroProgramming initiative.

It is one of the most significant benefits of OBC, which does not attempt to make the work easier ... it eliminates the majority of the work entirely. Configuring IA solutions is done in a fraction of the time of traditional control solutions.

Moreover, because OBC diagrams are easy to follow and because objects are prescribed in industry standard languages, eg. IEC-1331-3, maintenance and support costs for the life of the solution are proportionally lower.

## Object Framework for Control

A single node PC or a network of "open" PCs running WindowsXP provides the platform for the OBC system, but it does not provide for the management of the entire lifecycle of the objects which make up such a system. This is the domain of an object framework, an abstracted software layer that sits between the operating system and the objects themselves.

This object framework, or object brokering system, provides equivalent services and facilities that a CORBA system provides for a transaction-processing environment, including:

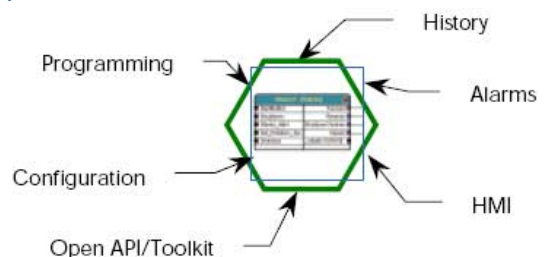
- Naming Service
- Event Service
- Security
- Licensing
- Messaging Services
- Redundancy
- Object persistence

A significant departure from the CORBA transaction model, an OBC object framework also provides some standard IA facilities which enable classical control functionality, such as:

- History
- Alarms
- Configuration
- Human Machine Interface
- Programming
- Others - an open API/toolkit interface

These facilities are perhaps easier to envision if we think of the OBC object model as being represented by a polygon. Each surface of the polygon represents a facility defined by the object model and exposed through the object's attributes.

The number of attribute extensions (represented by the number of sides of the polygon) may be extended through an OBC framework toolkit. Thus, the object model grows to meet the demands of the control system environment.



Third party software developers may define and add additional capability to the object model, or they may replace an existing capability through the application of the appropriate toolkit. This OBC object model extension defines a new standard for open software tools for the industrial automation market.

The Object-Based Control system provides plug-ins for execution of control code (run-time engines), recording events (an event History system), resolution and management of anomalies (an Alarm management system), an integrated Human Machine Interface where the graphical elements may be assigned to the object class, a graphical workbench for managing the user-oriented behaviour, an integrated HTML/URL

documentation system, for both user based Help and for object level documentation.

Each of these "plug-ins" is designed to manage the appropriate attribute access to the object model as depicted in the polygon surfaces above.

## Progression of Control

In order for OBC to provide write-once-run-anywhere functionality, several conditions must be prescribed. It begins with a commitment that is one of the foundations of Object-Based Control design: control run-time engine must run on a variety of hardware platforms.

This is one of the principle distinctions between PC-based and Object-Based Control. OBC code may run in a soft-real-time or hard-real-time engine, an embedded controller (Windows CE), an open PLC or DCS processor. Moreover, the decision as to which run-time engine the object methods will be assigned may be made long after the control strategy has been defined.

The object methods may be moved from one run-time environment to another through a simple dialog-box reassignment. The entire object may be moved from one node to another - while it is running.

Thus, an entire control strategy may be defined and tested on a soft control engine. Critical objects may then be moved to a hard real time or embedded controller to assure timely, deterministic responses. It is extraordinarily easy to develop optimization algorithms which will move objects to assure load balancing or I/O throughput.

## Legacy/Proprietary Control Applications Tamed

When there is an existing legacy of predefined control solutions, e.g. a running program in a PLC, the OBC model provides for a "class server" object to "wrap" the entire PLC program, or a portion of that program accessible through PLC I/O into an OBC object. Thus, legacy solutions are easily encapsulated into Object-Based Control models and may be extended in the same manner as any other OBC object.

Sometimes there is a requirement for a very tight control loop, a high-speed servo controller or some proprietary control algorithm, written in a language such as C++, to be wrapped in much the same manner as a legacy PLC application. This is easily accomplished with the OBC class server tools.

## The Big Payoff

If an Object-Based solution set provided no more than has been described so far in these pages, it would provide a return on investment measured in weeks or months.

An OBC system doesn't stop there, for the model is easily extended to provide tightly integrated MES

objects for real-time planning and scheduling activities, integration into ERP systems through an object model interface, predictive maintenance, statistical process and quality control, troubleshooting, job costing, inventory and WIP tracking, recipe management and work order control.

This is all done using exactly the same user interface paradigm, the same programming methodology, the same object model, the same object framework the same OBC system processors.

For the first time, the controls engineer and the MIS department use the same tools! The means of interconnecting the control objects and the MES resources and work order objects is through another simple, highly intuitive graphical user interface.

The reporting is done through the same reporting tools. The UI is based on the same workbench/graphical package. Thus the user experience remains consistent while the system remains open.

The classical Computer Integrated Manufacturing (CIM) pyramid, originally presented by Advanced Manufacturing Research (AMR), depicting four independent and disassociated layers has long represented a dysfunctional factory.

For years, IA vendors have been able to solve large problems in the IA sector by providing stand-alone, "point-solutions," which have absolutely no means of interoperating with other solutions in the same space. The burden of interfacing these disparate solutions has, for far too long, been placed upon the shoulders of the users and their systems integrators.



By tightly integrating the lower three levels and linking them to that of the Enterprise Resources, one can finally reach for the ultimate industrial automation solution.

Finally, through the power and flexibility of Object-Based Control tools, these seemingly disparate solutions can be tightly integrated into a singular environment, with a common user interface, a common look-and-feel, a singular data base structure with access to real time data anywhere in the control process by any data client all the way up to the ERP level.

Through the concept of Object-Based Control and it's common object model, the set of solutions that comprise the Real-time Planning and Control space easily interoperate through the OBC framework, although they may be designed and provided by independent software suppliers.

## **Object Based Control - The Benefits**

The benefits of Object-Based Control are enormous:

- Fully integrated solutions arise from multiple vendors, from the IA/factory floor up to an interface with the industry standard ERP systems.
- User access to full object functionality ... the promise of reusable software components ... realized.
- Rapid development environment with write-once-run-anywhere and ZeroProgramming.
- A Universal Control environment, a singular environment for building automation, motion, CNC, robotics, discrete control, batch, simulation, process control, real-time planning and scheduling, and maintenance solutions, with common user interface, common object model, common interoperating paradigm.
- Leverages commodity hardware.
- Leverages industry standards: IEC-1131-3, Fieldbus, Profibus, DeviceNet, InterbusS, S88, IPC, ODVA, OPC, OACG, OMAC, IEEE, etc.
- Provides a rich, easily extended environment for users and third party developers.
- Encourages technological partnerships, leveraging the scarce software developer resources available to build IA solutions.
- Frees the end user community from its dependency on single-vendor solutions.
- Finally, fulfills on the promise of "open" solutions, each module is easily replaceable with another, "best of breed" solution.