

OAenterprise and DCOM in Manufacturing

The OAenterprise product, offered by ObjectAutomation® (OA), is comprised of many manufacturing-oriented components, for both the visualization and back-end support of such services as control, HMI, history storage and alarm management. All of the components within OAenterprise have evolved from the incorporation of both Microsoft®'s DCOM and OA's own distributed object service layer, the OAframework.

OA has precisely and effectively integrated both DCOM and OAframework object models, resulting in an extremely robust platform for manufacturing systems.

One of OA's early decisions was to use DCOM rather than CORBA (Common Object Request Brokering Architecture) to develop OAenterprise, after an extensive evaluation of both technologies. Microsoft's DCOM is a technology that provides the basic, distributed object functions required for a robust distributed system on the Microsoft Windows® family of platforms. Additionally, DCOM/ActiveX provided OA with the appropriate mechanisms to extend and enhance the technology, to build services that supplement the base DCOM brokering system, with respect to industrial automation requirements.

With OA's development efforts focused to meet industrial automation requirements, OAenterprise is enhanced with concepts and specifications from CORBA, resulting in OA's own object framework, the OAframework, building upon DCOM with additional key services essential in supporting the object life cycle and rich functionality demands of industrial automation.

These demands are not entirely met by DCOM. The life cycle model of DCOM is such that when the DCOM services determine an object is not being used and therefore, the reference count drops to zero, the object usually destroys itself.

In a manufacturing environment this is not acceptable, for the object may represent or contain the control code for a crucial element in a control solution. OA has addressed this issue through the OAframework Life Cycle Service and OAframework Name Service, that guarantees the object always has at least one reference and maintains its location information, respectively.

With the OAframework, life cycle operations are performed explicitly (e.g., shut down, start up) rather than implicitly (reference count going to zero) and client connections are maintained during the object's life cycle.

This ensures that objects, whether they represent pumps, reactors or work orders, remain within the

system until someone with appropriate permissions executes a command to destroy the object.

While those objects responsible for the real-time monitoring and control of the plant floor have their own life cycles, independent of the HMI or other applications, the objects responsible for retrieving and maintaining production information follow the normal COM/MTS (Microsoft Transaction Server) model.

The MTS objects are primarily responsible for supporting transacted order fulfillment and provide a front-end to most of the common production information systems.

When a production order is created, the MES (Manufacturing Execution System) system is consulted by using a transacted production request object (hereby referred to as 'the request') provided by the MES system. The request then creates an order fulfillment object, which will verify the feasibility of the request and then allocate the resources for the request.

The fulfillment object is responsible for using the transactable objects provided by the ERP (Enterprise Resource Planning), Recipe and Financial systems to guarantee the immutability of the operation. Whether it be at the ERP, MES or plant floor level, security plays a large part with software in a manufacturing environment.

First, objects must be securable in the sense that different end-users, or groups of end-users, must have security characteristics associated with their system login that prevent dangerous executions or allow critical operations upon objects. The security mechanism must be integrated with the underlying operating system to provide truly robust security. For example, the OAenterprise uses standard Windows NT C2 level security for securing object access.

If OA had selected its own methodology of user names and passwords, it would have only caused security loop-holes whenever the system was extended to interact directly upon the operating system. Another important characteristic of objects within manufacturing is that they must be able to tolerate faults and assume a peer object's task when that peer object fails.

This is one of the most important features that the OAenterprise provides above and beyond normal DCOM and also requires a change to the normal COM Reference Counting Life Cycle Model. According to DCOM rules, if an HMI creates an object on another node performing control and a fork lift runs over the Ethernet cable connecting the two, the object performing the control shuts down because its one reference has "gone away."

Furthermore, a second object (or an 'x' amount of other objects) may wish to attempt to take over for the original object if that original object is no longer available (eg., hot backup).

The AOenterprise provides the mechanisms required for fault-tolerant, redundant, distributed object control systems. A mechanism for efficiently managing the unique, highly granular type of asynchronous event messaging that occurs within industrial automation systems is also a necessity for such an object-based system in manufacturing.

The event distribution throughout the network of objects must be prioritized or otherwise managed to prevent event "storms" (e.g. cascading alarm events) from either occurring or seriously affecting proper control handling of those events. When a user attempts to manage these events in a predictable manner, over commodity, non-deterministic networks such as Ethernet, the problem is only compounded.

OA provides "real-fast" event delivery via our Event Services and is now in the process of implementing "real-time" event delivery in the predictable, non-

operations and efficient object management. Regarding the manufacturing plant floor software systems, a user can generalize a similar three-tier model and highlight the differences among the tiers. The execution and control objects on the plant floor generally have different life cycle and maintenance requirements than the business objects as specified by Microsoft. Whereas the business objects in the Microsoft model primarily exist due to a client request, the execution and control objects exist despite nonexistent HMI interaction.

Essentially, the life cycle focus is within the control objects and outside of the business objects. Also, the data layer of the plant floor is more often real-time I/O (an OPC server, for example) or a specific device, as opposed to a relational database in the Microsoft data tier. In fact, the communication among the tiers of the plant floor is inherently bi-directional and functions in

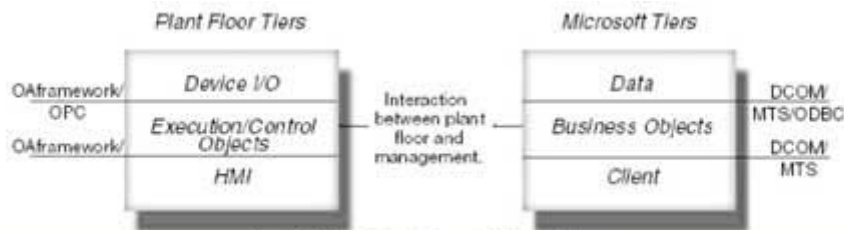


Figure 1: Plant Floor Tiers and Microsoft Tiers.

deterministic sense.

Generally speaking, multi-user development and overall distributed systems management goes hand in hand with object-based software. Since the seeds of software development have migrated from the world of monolithic applications to the world of object granularity, multiple developers may work concurrently within the object-based system. In a deployed system, multiple end-users or objects may be executing in a truly asynchronous manner (as opposed to multi-threading on a single chip for example).

All of these multi-user, asynchronous situations must be managed and the integrator should have control over specifying behaviour as a result of these dynamics. This is yet another justification for the necessity of the OAFramework services for manufacturing-oriented objects. Microsoft's DCOM and DNA architecture are designed with the "Three-Tier" model in mind. The three tiers (see Figure below) are separated into data, business objects implementing business rules and client or user interface layers.

In general, the data tier is implemented using databases, the business objects may be implemented using any COM-compatible language and may be organized hierarchically within the tier, and some form of graphical user interface application implements the client tier.

Additionally, the MTS provides a great environment for hosting business objects, allowing the transaction of

real-time, requiring more advanced peer-to-peer communication than the basic client-server type of information flow that exists in the Microsoft model.

When analysing the Microsoft Three Tiers Model and the plant floor tiers model (see Figure 1), the key concept to recognize is the interaction point between the two models, located at the middle tier, or in other words, among the objects. Since the OAFramework is a set of services that are built upon and supplement DCOM (the specification for Microsoft's model), the OAFramework embodies a refined model of the interacting three-tier models with a seamless interaction between them.

ObjectAutomation has effectively used the native DCOM architecture in conjunction with the OAFramework services to provide an overall solution for manufacturing enterprise systems, contained within the AOenterprise offering.

Extended life cycle management, real-time information flow and real instance-based naming are the essential services required by the critical plant floor environment; hence, the reason for the necessity of the OAFramework.

Microsoft's DCOM and ObjectAutomation's OAFramework are not competing, distributed object platforms, but complementary solutions for the world of industrial automation.