

Object Databases Improve Process Control Performance and Reliability

**By Leon Guzenda
Director, Special Projects
Objectivity, Inc.
Mountain View, California**

Object oriented database technology is delivering significant improvements in performance and reliability to a considerable number of leading-edge process control applications. Delays or downtime can be disastrous in process control because these applications are often required to choreograph the activities of equipment in situations where the cost of a shutdown can run well into seven figures. The great majority of process control software developers use object technology because its network-based model provides a much better fit with industrial processes than traditional procedural programming methods. But a big programming effort and performance hit are involved in moving back and forth between the objects used to represent the process and a traditional relational database. In recent years, object oriented databases have overcome these problems by storing objects directly without the need for mapping into tables and rows. Major process control applications that taken this path, such as Fischer-Rosemount's DeltaV, Siemens' SIGMASYS and NEC's Object Oriented and Open (O3), have demonstrated major improvements in response time and availability.

Process control is by its very nature a network-oriented task. A manufacturing plant can best be depicted as a hierarchical structure with the plant divided into departments, departments divided into cells, cells divided into machines, machines divided into subsystems, etc. Both the process equipment and the material being processed are connected by an extremely complicated network of interrelationships. For example, a machine used to package and assemble semiconductors contains different types of equipment such as die bonders, curing machines, wire bonders, etc. A number of different models will often be included of some equipment types in order to increase the ability of the system to handle different products. In turn, multiple copies may be included of certain machine models to reduce cycle times. At the same time, individual machines may have multiple processing units. For example, a curing oven with eight boxes might have two of the boxes reserved for special applications.

Inherent hierarchical structure

Object oriented technology offers dramatic advantages in modeling a system of this type. Object oriented languages such as C++ and Java have an inherent hierarchical structure that allows lower-level objects to inherit properties from higher-level objects. Another advantage of the object paradigm is that objects combine data with methods used to operate on the data. So if, for example, you create a new wire bonding machine it will from its creation be capable of performing all of the basic operations that you have defined for this class or type of object. Data, such as temperature information coming from a sensor, can be wrapped in self-describing objects that contain sample information such as temperature, pressure, and liquid level, as well as identification information such as date, time, sensor location, alarm status, etc. This makes it easy to communicate the data to heterogeneous systems throughout the plant.

Most process control system developers have migrated to object oriented technology and the majority of new systems are being developed in Java. A key advantage of Java is that it runs the same on any hardware platform that supports it. This frees Java developers from concerns over

operating system and hardware specifics and eliminates the need for porting efforts. Java can also distribute applications across networks and distribute full or partial applications to thin clients as needed. For example, imagine that a design engineer's UNIX-based workstation needs data from a factory system run by a programmable logic controller. Information from the factory system can be wrapped in an object written in the Java language and seamlessly and efficiently sent to the workstation, ready to be published or used in other programs such as a spreadsheet or quality assurance analysis package.

Problems with the relational model

The developers that created the first process control systems using object technology faced a major data storage challenge. Relational databases are not very good at traversing complex relationships. They generally need an artifact called a JOIN table to represent many-to-many relationships between objects. Traversing and maintaining these JOIN tables can involve extra B-tree index access and additional I/O's. If N is the cardinality of the relationship, a relational database will need N^2 B-tree lookups and N^2 logical reads. The result is an enormous performance hit. In addition, often as much as 70% of the code in process control relational database applications is involved in translating many-to-many relationships into the table and row structure of the relational database.

The problems involved in using a relational database in an object oriented process control application are critical because these applications demand extremely high performance. Typically, the application is required to receive status information from a few to many thousands of different interdependent processes that are operating simultaneously. The application receives and stores the status information, then generates instructions for each of these processes. Clearly the ability to perform its functions without any delays is critical because leaving factory equipment without instructions even for a fraction of a second can delay the process and may even cause the equipment to fail in some situations. Downtime on the part of the control system is just as bad. It can result in shutting down a factory that may have cost hundreds of millions of dollars to build and could easily contain a million dollars worth of work in progress. The cost of halting and restarting such an operation is enormous.

Use of file systems

To avoid this problem, some process control system vendors originally developed their own file systems in which the tree structure is built in RAM and stored in a file. When the equipment is turned on, the tree structure is read into memory. The main problem with this approach is the difficulty of maintaining and improving a system that was built in machine level code. Programmers typically need to keep track of when persistent objects are changed and issue a save operation. This usually results in frequent errors that require a considerable portion of the programming effort to fix and diagnose. This approach also presents scalability difficulties because larger tables won't fit into RAM. Another problem is that there is no standard SQL way to query the network structure, for example, to produce reports required for quality tracking purposes.

For these reasons, process control system vendors have long been considering the object oriented database approach. The most appealing aspect of object databases to most vendors is that they are capable of modeling process equipment with all of its complex interrelationships directly, without forcing it into a row and table format. The object oriented data model allows for an unlimited number of associations and interconnections among objects as well as the use of variable length data without a performance penalty. While this approach is ideal in theory, it has taken some time to grow firm roots in the process control industry.

Trends towards Java

But, in the last several years, the move to object databases has picked up a good deal of momentum and the majority of the major new systems have been developed from the ground up for object databases. An important factor aiding this trend is the advent of powerful Java-based Integrated Development Environments (IDEs) that were created originally to build commercial web sites. The significance of these tools is that they provide both a powerful modeling language for creating objects and methods as well as an easy-to-use environment for graphically creating transaction screens. Best of all, they are so intuitive that any good programmer can learn one of them in a matter of days. Estimates are that 90% of the programmer community in the process control industry is either using or considering Java.

It is extremely easy for a Java programmer to add object oriented database functionality to a Java application. The development process typically consists of: 1) Creating a federated database to manage metadata; 2) Using an integrated development environment (IDE) to design the data model and application classes and; 3) Running the application to store objects that are created and modified within transactions. The entire development process can proceed using standard Java development tools and utilities and the developer can control as much or as little of the database interaction as he or she desires. In order for objects to be stored in the database, they must be instances of persistence-capable classes. This means that persistent objects are initially created as standard Java objects. The application must perform an operation called clustering that causes a transient object to be assigned a storage location in a database. Classes can be made persistence-capable either by implementing a persistent interface or by deriving from a persistent class. Any object can be made persistent by establishing an association between the transient object and a persistent object or by making an explicit call to cluster an object in a particular container or database.

Directionality and cardinality

In order to access the object, the developer first needs to create a local representation. The next step is to gain control of the object by opening it for read or update usage. If the object has application-specific persistent data, then the object data must be copied into the local representation of the object. The database allows the developer to specify the directionality and cardinality of relationships, how relationships should be handled, when objects are copied or versioned, whether operations on objects propagate along relationships and how relationships should be stored. The relationship's cardinality indicates the number of objects on one side of a relationship that may be related with objects on the other side, such as one-to-one, one-to-many, many-to-one and many-to-many. Relationship directionality is defined by the declaration of traversal paths that enable applications to locate related objects. An object that maintains a unidirectional relationship can locate its related object while bi-directional relationships allow related objects to locate each other.

Multiple Java threads can be used to execute concurrent persistent operations. The interaction between a session and a thread is governed by the session's thread policy. With an unrestricted thread policy, a thread can operate on the session, while restricted thread policies require that a thread join the session. Transactions perform operations that create, read, modify or delete objects while guaranteeing consistency between the local representations of the objects and application and the objects in a federated database. The basic steps for creating a transaction block are: create a session object; start a transaction; create/delete objects or open and modify objects; manipulate the objects and commit the transaction.

Retrieving objects from a database

A variety of methods can be used to retrieve objects from a database. You can retrieve individual objects based on their application-defined names and links with other objects. In addition, you can search storage objects for objects of a given class, possibly restricting the search based on values of certain persistent fields. Finally you can traverse the entire storage hierarchy, retrieving all objects within each storage object. An object graph is a group of related persistent objects that are linked together in a directed graph structure. Each link in the graph can be a persistent field that references another object or a relationship. Once you have retrieved one object in the graph you can follow links to retrieve its other objects. In addition to finding objects via traversal, object oriented databases often provide flexible query capabilities. A typical example is a predicate query language that includes standard arithmetic, relational and logical operators as well as string matching operators that test whether a string matches regular expressions.

Object oriented databases offer greater flexibility than relational databases in their use of indexes. With relational databases, you must typically define all indexes when the database is created or take the database off-line for a massive change. With an object-oriented database, indexes can be created, updated and even dropped on the fly. For example, if an application routinely scans a particular container (a cluster of logically related objects) to find the persistent objects that satisfy a particular predicate, the application can define an index to speed the search. No other containers are affected. Indexes are created dynamically by application programs and continue to exist until they are explicitly removed.

DeltaV process automation system

One of the most exciting new object-database-based process control applications is Fisher Rosemount Systems' DeltaV scalable process automation system. DeltaV is the first of a new generation of systems that harness the power of field device intelligence to improve plant performance. The DeltaV system's scalability and ease-of-use have earned international recognition as well as numerous industry awards. Even more important, many process industry giants have adopted the new system. Fisher Rosemount recently announced a \$4 million contract to supply AstraZeneca PLC with a DeltaV scalable process system at a new pharmaceutical intermediates facility. Global drug giant Pfizer installed a system valued at over \$6.5 million for a plant in Ireland. Coca-Cola uses the new process automation solution to serve syrup blending and quality monitoring facilities at a plant in Japan.

"Our first step in developing DeltaV was to undertake a research project, comparing the capabilities of relational vs. object-oriented database technology," said Mark Nixon, Fisher-Rosemount Lead DeltaV System Architect. "Upon completion of the research project, we concluded that object oriented technology was the way for us to go for DeltaV. Evaluating several object-oriented database projects, we chose Objectivity/DB because of the tremendous amount of flexibility and extendibility it gave us in our product design. We needed to be able to manage tremendous amounts of complex data -- but still think of ease-of-use first and foremost."

Using the object oriented database helped bring the DeltaV system to market faster. The product was written entirely from scratch in only two years -- an amazingly short period of time for a complex piece of software that contains one million lines of code, 1200 classes and 25,000 methods. In fact, the software is one of the largest applications ever managed by Microsoft's Visual SourceSafe code management system.

"Objectivity/DB holds 100% of the CONFIG engine, the part of the software that provides most of the user interface and time-saving features of the product," Nixon explains. "DeltaV's user interface 'Explorer' shows the plant layout on the computer screen. If an operator wants to see what's going on in more detail, he or she simply clicks the area with a mouse to see the next level of detail, clicks again to see even greater detail, and so on. In programming terms, this means we're using nested inheritance and abstractions to hold the configuration that the user needs."

Because of Objectivity/DB, DeltaV uses a much more intuitive, visual and flexible methodology. It's a lot easier to conceptualize what you already see, rather than trying to mold data into a different type representation."

ESEC controls semiconductor manufacturing

The use of object oriented technology is one of the keys to the first comprehensive solution to the integration of semiconductor manufacturing's back end. As semiconductor functionality increases at the same time prices are dropping, manufacturers are paying increased attention to reducing test, assembly and packaging costs. Partitioning a manufacturing lot into smaller units that advance independently through the various process steps can play a major role by reducing work in process (WIP) and cycle time. But this greatly increases the complexity of the scheduling and planning function. It requires the development of software capable of managing production logistics while coordinating WIP, planning, scheduling and schedule execution, logistics control, process control, data collection and equipment monitoring. Object oriented technology provides an ideal platform to manage this complex network problem without bogging programmers down in the onerous task of normalizing these complicated processes as relational data and mapping them back and forth between the real world and rows and tables.

The first, and to date only, effort to integrate back end manufacturing operations is the Autoline from ESEC, Cham, Switzerland, an autonomous, flexible production cell. The system covers process steps where material is moved in magazines, typically from die attach down the line to transfer molding. Material transport between process equipment is automated by means of a robot moving on a linear track. To enable optimum process sequences within the cell, the cell integrates third party dispensing, optical inspection, plasma cleaning and molding equipment.

When it began developing this system, ESEC used the Smalltalk programming language and stored persistent objects in a file system in which objects were serialized in binary files. The program needed to keep track of when persistent objects were changed and issue a save operation. This resulted in frequent errors that required approximately one-third of the total programming effort to fix and diagnose. Switching to an object oriented database immediately increased productivity by about 50% by eliminating the need for programmers to be concerned about persistent object storage. The total amount of time required to convert the application was about four person-months. In addition, the implementation of the object oriented database eliminated earlier problems involving the recovery of the system after a crash. This is significant because if the system crashes in the middle of a production run it is likely to be full of high value WIP. The elimination of persistent data problems makes it possible to easily recover the state and resume processing, avoiding loss of any material.

Of course, there are many, many other successful object database implementations in the process control field. Siemens AG uses object database technology in its SIGMASYS security alarm system, a collection of system building blocks that processes fire, burglary, and other emergency messages. NEC, one of the world's largest semiconductor manufacturers, employs object database technology within its O3 system that manages distributed manufacturing for 16 factories around the world. Neles Automation uses object database technology for its nelesDNA engineering environment that networks all automation and information activities for process industries. Technology Answer's CIMPLEX Generative Numerical Control system automatically generates multi-axis numerical control programs that drive machines cutting extremely complex aerospace and automotive parts.

As these and scores of other successful applications prove, object oriented databases are clearly a technology whose time has come in the process control industry. The primary driving force is the fact that nearly every process control application demands 24 X 7 availability -- millions of dollars worth of production machinery and customer orders are depending upon it. By eliminating

the bottleneck of translating between the object and relational worlds, object databases provide a dramatic performance boost. That's why the majority of new high-end process control applications are being written for object databases.

For more information contact Objectivity, Inc, 301B East Evelyn Ave, Mountain View, CA 94041.
Phone: 650-254-7100 Fax: 650-254-7171 <http://www.objectivity.com>