Control Solutions
Formerly I&CS

CONTROL SOFTWARE FORUM

Improving the OI in process control using Java-enhanced integration and access

Tom Lubinski
CEO
SL Corporation
Corte Madera, California

Praful Bhayani
Process Control Manager
SL Corporation
Corte Madera, California

In a typical process information system, large numbers of continually changing data variables need to be viewed by multiple clients in a large network. The user of the operator interface (OI) expects to perform complex interactions with the graphics displays and to see, with minimal delay, dynamic information in near real-time. A control system that can compete successfully under these conditions must have an architecture that is compact and scalable.

Displays should be updated by transferring only the data necessary for a particular display, minimizing the overhead. When a new display is requested by the user, the wait for a new screen should be, at most, a few seconds. The challenge becomes more difficult if the goal is to make the system available over a low- or medium-bandwidth Internet connection. Three alternative, Web-based technologies can be used to update data displays: server pages, ActiveX, and Java™.

Server pages and ActiveX solutions

One common solution is to have a server-based process respond to requests from Internet clients for a specific view of some information. The server process produces an HTML page and one or more images, which are sent back to the client. This approach is very demanding on the server when there are multiple users. It is also bandwidth intensive, requiring that complete images be transmitted on a regular basis in order to keep the screens updated.

A better solution is for the server to only transfer specifically-requested data across the Web, and make use of a thin client application to update a display on the client side. The process industry embraced the Microsoft ActiveX solution for this purpose. It was available and worked well enough, but was limited to the Microsoft Windows environment. Installation required attention to details, and there was always the possibility of resulting errors.

Java solutions

In a number of ways, the Java™ solution is very similar to ActiveX technology. A Java program on the client machine renders the graphics displays, but only data updates need to be transferred across the network.

Initially, there was reluctance from many developers to use Java in process applications, and for valid reasons. The Netscape and IE Version 3.x browsers provided only limited, buggy Java 1.0 support. This was a serious limitation to the use of these early versions.

Now, with version 4+ browsers, support for Java 1.1 is very good, and Sun Microsystems continues to release newer versions of its JDK (1.3+). Real-world applications can be provided in this environment. As a result, a Java thin client becomes a real alternative to an ActiveX solution as a means of accessing and viewing large amounts of dynamic process data. Java has the distinct advantage that an application written in Java can be run on almost any platform, but most importantly, Windows, UNIX, and Mac.

Moreover, the Java language supports an object-oriented programming style that is much easier for many programmers to use than C++. The syntax is easier to follow, and the interface construct provides a nice way to achieve a sort of multiple inheritance, something that is too complex in C++ for many casual users. It is just this suitability of the Java language to supporting object-oriented architecture that makes it so valuable. The nature of an operator interface is very much object oriented. Good tools and language support are necessary to develop, maintain and enhance large-scale OI applications.

Browser-based Java applets

Any software platform hosting the Java Virtual Machine can execute a Java application. For local intranet support, a Java application is much like any other program running on your system.

However, for Internet access the Java support that is provided within the current crop of browsers, Netscape and IE, is very good. In fact, deploying an application within the framework of a browser as a Java *applet* has significant advantages. A browser anywhere on any machine can access an HTML page at a specific Web address and automatically execute a Java applet, using the Java VM built into the browser. There is nothing to download to a user's system.

Applets can be designed to make use of newer Java features, such as Swing controls, and Java 2D graphics. However, the user may need

Reprinted from the February 2001 edition of **CONTROL SOLUTIONS**

to download a special Java plug-in for that particular browser, making it seem more like an ActiveX solution, although more portable.

Performance considerations

Any Java application is going to be slower than an equivalent native C++ application, simply because of the interpreted nature of Java implementations. The performance issue has slowed the acceptance of Java for large-scale systems, but this is changing rapidly, as Java improves and hardware performance accelerates.

Java will eventually be compiled. There is no reason why Java cannot be processed just like C++ to produce compiled code. It is interpreted now, for convenience—you don't need to make a compiler for every machine. However, as more applications are done in Java, demand will grow for optimizations that can only be achieved through compilation.

Pick third-party support built on Java

There are many issues to address when building an OI for a process application. It makes sense to use a third-party product that is designed specifically for that purpose, and uses the latest technologies that include Web functionality and Java-based features. For example, a number of well-designed third party development systems provide the following basic features:

1. A run-time system that manages the display of graphics screens and associated data variables.
2. A graphics editor that allows the construction of graphics and associated dynamic behaviors.
3. Compatibility with common Web file formats, such as SVG, XML, etc., along with proprietary file formats that may be used for optimization.

A number of other features are not so obvious, but are of significant importance in the current Java deployment environments:

4. Scalability achieved using special techniques such as Java Code Generation. As the power of a system increases, so do the demands of the application. If the system is capable of it, why not put even more detail into your application. For example, your competitors will be "pushing the envelope," along with you. To succeed, you must provide a range of optimal solutions. There is no room for slop.
5. A state-oriented architecture with routines for handling user interaction. This often involves the use of a finite state machine architecture and API support for an active state hierarchy.

Developing for the future

With the maturing of Java, development tools and libraries are becoming readily available, making a Java thin client a serious option for the deployment of a process graphics OI. When trying to determine what tools or products to use to help you build your product or service, or if you are considering developing a system internally, recognize that building an OI for a large-scale control system is a complex undertaking. It is not just the graphics part of the OI. There are many integration, performance, and deployment issues. The Internet has dramatically changed the way the information flows from suppliers to customers. Supply chains are becoming global and increasingly complex. The flow of raw materials is being accelerated through globalized supply chains, requiring more efficient and customizable process control.

This requires real-time information flow, enterprise-wide integration and visibility. MES has become the connector between the plant floor and ERP/SCM (Enterprise Resource Planning/Supply Chain Management) to provide a mix of plant graphics and trend analysis that reports how well production units are performing and the status of customer order fulfillment.

In the Java environment, you can develop with the future in mind. You can provide for multiple deployment scenarios by building a system that functions as both an applet or an application, and works with AWT (Abstract Window Toolkit) or Swing controls. You can also make use of object-oriented and state-oriented approaches to application architecture to protect your investment in the logic behind the application by designing modules as classes and state classes, which can be implemented equivalently in C++ or in Java. Only then can you future-proof your OI for Java and beyond.

About the authors

Tom Lubinski, who founded SL Corp. in 1984, is currently the company's president and CEO, and has been instrumental in developing SL's Graphical Modeling Solutions (SL-GMS) software. Lubinski attended the California Institute of Technology (Pasadena).

Praful Bhayani, SL process control manager, manages the technical direction of SL-GMS. Bhayani, who has 15 years of experience in process automation, graduated from Bangalore University in 1984 with a bachelor's in engineering.