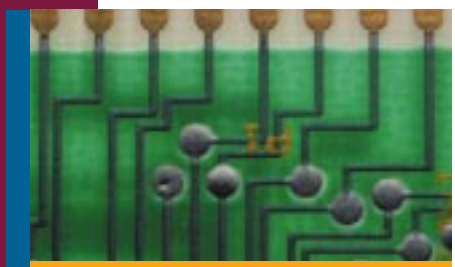


BY RICHARD B. VIPOND

Legacy Systems Enablement: Part III – Development Environment Essentials

The next step in our goal of Internet-enabling our legacy systems is to examine how the numerous tools, technologies and standards selected will fit and work together.

THE goal of this four-part series is to demonstrate how my client's site is creating an open systems development environment model from which they can more readily migrate into a fully functional object-oriented environment such as CORBA.

Part I (July 1998) set the stage for this four-part series and examined how companies are rushing to take advantage of the Internet and all the potential it holds for the future of the enterprise. However, as I pointed out, these companies also need to embrace the legacy systems that have supported their enterprise for all these years.

In Part II (August 1998), I examined the criteria necessary for selecting a development tool but did not specifically list the products we have reviewed at my client's site.

This article further examines our open development environment model, which was presented in Part II, as well as software configuration management (SCM) software and the object repository. I will also further discuss the CORBA standard as well as the other standards that are being adopted by select companies and vendors. Additionally, I will explain how GUI and WYSIWIG development tools will become vital to the future success of application development and examine their relationship to current "bleeding edge" development concepts.

APPLICATION DEVELOPMENT WITHIN THE ENTERPRISE

Our open development environment model will consist of numerous tools, technologies and standards that must conform to the fundamental principles outlined in the previous articles. Now that the application development tools have been selected, we will see how they should all fit and work together. The most important principle is using proven open technology standards in the design of all systems. While referring to Figure 1, realize that all the tools being used will create code that will fit into some type of object repository. This repository will become an integral part of our open development environment model and will be controlled by an SCM program that manages the enterprise. In the future, a product like IBM's Component Broker, which manages the CORBA environment, will ultimately manage the global development environment. This will be discussed in the concluding article.

Application development within an enterprise will consist of existing legacy system programs, existing standards being enforced and new development efforts, all of which must transition into our open development environment model.

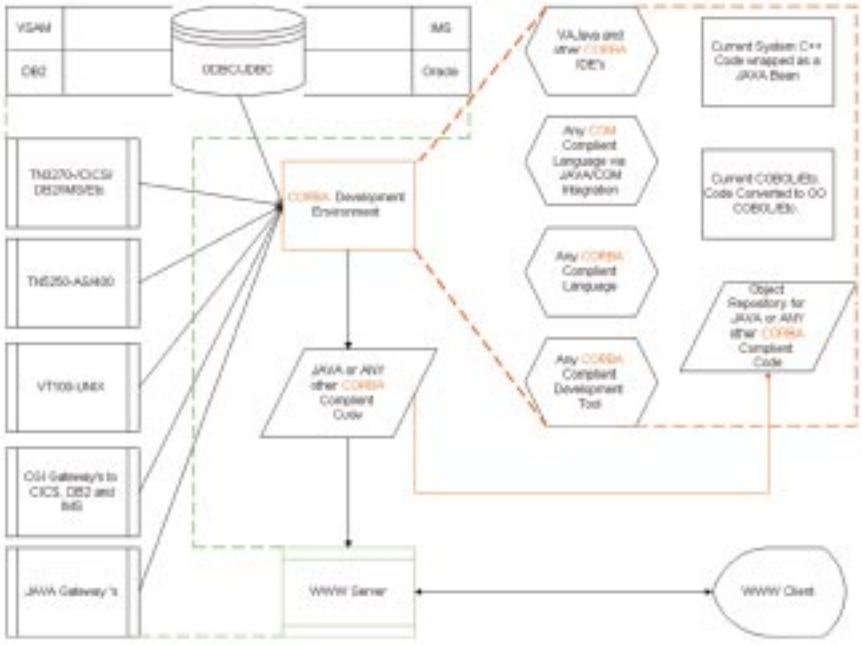
Existing Legacy System Programs

Our existing legacy system programs are fundamental in our quest to preserve and utilize their functions. There are several reasons we may want to use these programs directly, rather than indirectly through the use of pseudo screen scraping development tools or other development tools that merely mask the original legacy-system-based interface. The most obvious reason to utilize legacy programs directly is because of the performance overhead associated with any bridge, gateway or development tool.

The object-oriented (OO) capabilities of programming languages such as OO-COBOL allow us to utilize the programs within our legacy system environment. Current C++ code can also be wrapped as a Java Bean and introduced into the OO environment. I

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Figure 1: Example of Concept Requirements



predict that all programming languages will allow for OO capabilities in the near future.

The biggest obstacle to using compiler OO capabilities will be in how the legacy applications have been designed and structured. Since most legacy programs have been poorly designed and structured or are the result of years of maintenance, thus creating a poor design and structure, I think most companies will find that rewriting these applications using our open development environment model principles will produce the best results.

Design principles for OO applications rely on properties such as simplicity, reusability, security and good performance. For these properties to be effective, an object should implement a solution to only one problem and delegate responsibility for implementation of solutions for all problems other than its own to other objects.

Existing Standards Being Enforced

Part I mentioned a few current standards that when used with gateways or bridges can be incorporated into our open development environment model. It's important to remember that gateways and bridges mean extra overhead, so make sure that the tool or technology that implements the open standard justifies the additional overhead required to use it.

Some examples of such object-oriented application development models include Microsoft's Distributed Component Object Model (DCOM), which like CORBA, separates the object interface from its implementation and requires that all interfaces be declared using an Interface Definition Language (IDL). DCOM is not based on the classical object model of CORBA. DCOM does not support IDL-specified multiple inheritance and thus, achieves object reuse via containment and aggregation instead of inheritance. In other words, DCOM objects do not maintain state between connections and thus, cannot be reused to create a unique object. These differences between CORBA and DCOM are circumvented by a method known as the Java/COM Integration Bridge, which allows CORBA compliance and subsequent use in our open development environment model.

Application development within an enterprise will consist of existing legacy system programs, existing standards being enforced and new development efforts, all of which must transition into our open development environment model.

Remote Method Invocation (RMI) is a technology that uses remote Java objects, whose methods can be invoked from another Java Virtual Machine (JVM), even across a network. However, RMI can not communicate with other Object Request Brokers (ORBs) or languages such as CORBA's Internet Inter-ORB Protocol (IIOP). Since RMI needs the services of IIOP to be a suitable backbone for Internet use, RMI requires a bridge that allows us to write distributed objects in pure Java. The bridge currently used for this is called Caffeine from Netscape and Visigenic, and it provides all the benefits of CORBA transparently from Java and allows for subsequent use in our open development environment model.

Hypertext Transfer Protocol/Common Gateway Interface (HTTP/CGI) was the predominant model for creating three-tier client/server applications over the Internet. However, the protocol is clumsy, stateless, slow and resource-intensive on the server. Netscape is currently addressing this problem by bundling a CORBA ORB with every browser. This will provide CORBA/IIOP capabilities. Java Bean wrappers encapsulate CGI Languages, such as C++, which provides CORBA compliance. Subsequently, these methods will allow for use in our open development environment model.

Peer-to-peer communication that hides the details of the network (commonly referred to as simply sockets) was the only way for a Java program to communicate to the world before CORBA and RMI. However, sockets introduce a major maintenance and programming nightmare and are the reason you need an ORB. ORBs hide these nightmares and let you program at a higher level of abstraction using CORBA or RMI methods. In the future, these methods will allow for use in our open development environment model.

Extensible Markup Language (XML) is a language that defines other languages. It also has the potential to give structure and meaning to the information contained in HTML documents or any other data form, making such information as searchable and structured as the information locked in a database. The potential of such a meta-language is huge if it is implemented as an open standard. Right now, XML is an open standard, and if it continues to evolve along open lines, it could drastically improve Web-based development and fit nicely into our open development environment model.

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

New Development Efforts

New application development efforts should use a combination of tools, processes and languages that conform to our open development environment model. New development efforts will also include those legacy programs that do not lend themselves well to the OO capabilities of the various compilers being introduced. I currently prefer the Java programming language for application development; however, we also need to be aware of existing staff expertise and resources currently being utilized. The introduction of Java to the computing environment has finally given CORBA the vehicle it needs to be implemented in the real world.

The existing expertise and resources will comprise most of the standards previously mentioned and will be found in all the various Integrated Development Environments (IDE), languages and tools that must all conform to our open development environment model principles. All code that is produced from these IDEs, languages and development tools will have to be maintained in some type of object repository. Code reuse is a fundamental requirement to on open technology standard and an object repository will be mandatory. At the time of this writing, we have not yet determined what object repository we will be using. We realize that major advances will undoubtably be made in object repository technology in the future.

THE KEY IS STANDARDS

We must only concentrate on standards that follow and incorporate what CORBA establishes. Remember, in the event we do want to adopt another standard in the future at least we will have a standard to convert from if we insist on using CORBA as the foundation. If we use a tool, technology or standard that is not CORBA compliant, that tool, technology or standard should be used with the understanding that it is a short-term fix to be remedied by a CORBA-compliant solution in the near future.

APPLICATION DEVELOPMENT IN THE 21ST CENTURY

I have come to realize that GUI development tools are a mandatory requirement. I feel GUI and WYSIWIG development tools are vital to the future success of application development in the 21st century. The complexity of application development in relation to application objects that will have to be manipulated and the eventual global location of these objects will require SCM tools that rival the current capabilities of the bleeding-edge technology, called VRML (Virtual Reality Modeling), which I've been studying for the past three years. GUI and WYSIWIG development tools are a precursor to VRML type application modeling techniques just as CAD was the precursor to current VRML modeling technology.

For future SCM tools to work, we must use products that contain all of the elements necessary for the creation of an open development environment. Communication with programs outside of the development tools environment must be supported, as must communication between application programs and objects. The concept of communicating with a program that isn't actually executing refers to dynamic code reuse and code manipulation of objects. The code produced by the development tool must be able to accept input parameters from other programs or processes outside its own execution environment. This capability is currently available in a new technology called VRML. Because of its necessity to maintain constant and direct contact with every object of its nodes, VRML is a highly complex language. It can only be programmed using a visual modeling tool, and, in fact, the only reason to learn the language is to perform minor tweaking and performance modifications to enhance the final result. All of the development and modeling concepts inherent in VRML technology can be directly applied to future application development and SCM tools.

The concluding article will complete the analysis of our open development environment model by examining its integration with the WWW servers and further relate how important current bleeding-edge developments are in relation to the concept of a global development environment. ts

NaSPA member Richard B. ViPond is a senior consultant for Ciber Network Services, Inc.

Special thanks to NaSPA members and technical editors Dwight S. Miller and Stephen J. Pryor for their help with this article.

©1998 Technical Enterprises, Inc. For reprints of this document contact sales@naspa.net.

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.