Building Distributed Systems

Everyone is building a distributed systems these days, but the ''how'' has become a matter for religious debate and the ''why'' is often neglected completely. This panel brings together a variety of practitioners to explain the joys and woes of building a distributed system including:

- why a distributed solution was chosen
- what middleware was selected and why
- what issues and gotchas reared their ugly heads and what was done about it

Moderator:

Doug Lea, University of Oswego

Panelists:

- David Forslund, Los Alamos National Laboratory
- Tom Barry, Lucent
- Don Vines, Genesis
- Rajendra Raj, Morgan Stanley
- Ashutosh Tiwary, Boeing Information and support services

Position Statements:

David Forslund, Los Alamos National Laboratory

Providing secure electronic medical records over the internet is a continuing challenge to the healthcare enterprise. We discuss our experiences with the deployment of a secure, Java/CORBA-based multimedia electronic record system in a rural healthcare setting. In particular, issues of scalability, portability, extensibility, interoperability, useability, and quality of service will be discussed in detail.

Tom Barry, Lucent

Lucent began building its custom Assistance Request system about two and a half years ago. Smalltalk was chosen for the client and a relational database was chosen as the backend. Not long after the project was began, GemStone/S replaced the relation database in order to provide smoother integration between the client and the server.

The intent has always been to make this system global, however having clients attach to a single server from across the world is not workable. So, instead of bringing the clients to the server, it was decided to distribute the repository to bring the information closer to the clients. Using GemStone's GemEnterprise, a repository is placed in each region (North America, Europe, etc.) and the information available to the client is synchronized amongst the distributed repositories.

In addition, Lucent provides web access to some of the system's information to its customers. Since customers cannot access Lucent's internal network, it is necessary to move the data to the firewall server. Here again GemEnterprise is used to federate selected (non-restricted) information to a repository on the firewall server. A special toolkit is then used to connect this repository to the web server on the firewall serving Lucent customers.

GemEnterprise provides multiple policies for replicating information, (copies, read-only replicates or modifiable replicates) as well as the usual mechanisms for limiting access to data, such as stubbing and class mapping. GemEnterprise is transparent to the client so the usual GemStone security mechanisms also work such as groups and segments. Refreshing of replicate information is scheduled as often (continually for the globally distributed replicates) or as infrequently (daily for the web accessible repository) as necessary.

The main problems encountered were system response and conflict management. Federated repositories communicate via the same mechanism a client communicates with the repository. By distributing the repositories, difficulties clients might have had connecting to a single server must now be dealt with between servers. In addition, the potential exists for clients on separate servers to update the same object. Once a refresh happens, someone's data could potentially be overwritten without warning.

The application has been deployed for some time. The web piece is scheduled to come on line in June. The globally distributed solution is currently on hold. Perhaps the most important lesson is that federation should be planned into a product deployment early. Since federation is affected by such things as the complexity of the object model and the nature of updates that will take place, it is much easier to address these issues up front, rather than in an ad hoc way later on.

Don Vines, Genesis

A number of recent publications have proposed that the combination of Java and CORBA is a good match for supporting leading-edge, three-tier client/server architectures over the Internet. A major assumption is that Java Applets make wonderful cross-platform clients that can be safely downloaded and run in a browser, with commercial CORBA products providing the transparent access to the middle-tier servers over the Internet. In our experience, much of this can in fact be accomplished today, but nobody should underestimate the engineering required to effectively integrate these two technologies. This position statement describes our experience with one such WWW-based system, including a discussion of key accomplishments, major issues, and architectural tradeoffs.

Genesis has been working with one of the world's leading biotechnology companies to introduce a new series of applications for managing complex, chronic diseases. These applications provide healthcare workers with valuable information and decision support over the Internet in a form that is not currently available from traditional information sources such as medical journals or research briefings. These applications depend on a distributed component infrastructure which manages clinical data over the Internet with required security/confidentiality, while encapsulating that data with business logic in the form of distributed objects and associatied rules.

Clinics are able to access these objects and rules through a Java applet over low bandwidth lines, e.g. COTS 56Kb modems, where the applet uses standard CORBA IIOP to talk to the middle-tier object servers. Since the applet must communicate with servers on the other side of the firewall, we opened a single IIOP port through the firewall and used an IIOP proxy to forward requests to the CORBA servers. This entire distributed object infrastructure for the middle-tier was built/integrated primarily from COTS components. This includes a WWW server, WWW-enabled CORBA/IIOP ORB, a commercial firewall, and an IIOP proxy forwarding service.

To get over the concern that clinics were storing their data at a remote site across the Internet, a commercial encryption algorithm is being used in a way that allows clinical data to be stored in its encrypted form such that only the clinics themselves could actually see the data in the clear. Commercial products have also been used to construct an application server which provided an interface to the business objects and a mapping of the business objects onto a relational database. The system also integrates a commercial reporting engine to generate the reports on the server and to download them over the web for display on the client.

A platform-independent business architecture was specified using standard UML artifacts. This formal business object archtiecture was implemented largely as modeled in the application server, which insulated the business object logic from object storage specifics. JavaBeans was used to provide the business object

developers with a Java-centric (versus a CORBA-centric) programming model, further insulating the GUI from the changes to the middle-tier, and totally encapsulating all the distributed computing mechanisms and policies.

Despite the usual dose of annoying problems, the project was very successful. The engineering team kept pace with numerous changes to the requirements, GUI updates, etc. There was no serious schedule slippage due to product or design problems. The hardware, software, and personnel resources remained as originally planned. The system met its look and feel requirements as well as its performance requirements. Despite these key accomplishments, however, there were some serious issues that resulted in significant architectural tradeoffs.

First, the GUI design found AWT deficient in terms of the ability of its widgets to display the sophisticated graphs and charts of the required clinical data. Appropriate commercial widgets are available, but downloading these libraries of 500 Kbytes is too big for 56 Kbit modems to handle in an acceptable startup time. The tradeoff was to make a less sophisticated GUI with AWT and avoid the download, or to use the sophisticated widgets and pre-install them at the clinics. The decision was to keep the sophisticated GUI, and accept the fact that client software installs and upgrades would not be painless or transparent for some time.

Another issue arose when the ORB that comes with Netscape (VisiBroker) was found to be incompatible with the ORB that we were using (Orbix). In fact, it was necessary to delete the native Netscape ORB so that Netscape would download our ORB. Other alternatives might fix this problem, e.g. a patch from one of the vendors or compiling the client stubs with the Netscape ORB.

Other incompatibility issues occur at the browser as well. For example, the browsers differ on how to preinstall applications, and how to grant privileges to local resources. Signing of applets also differs among the browsers. There are a number of other setup issues with regard to the firewall, the IIOP proxy forwarding engine, and so on.

Finally, Internet applications must be designed to get all the data that is required to populate a screen in a single call or small number of calls. For acceptable performance over low bandwidth lines, the applet can't make dozens of CORBA calls back to the middle-tier -- the connection overhead is simply too great over the Internet. We reduced network calls by returning lists of structures instead of object references, and by passing an entire object's state to 'create' and 'update' operations, rather than setting individual attributes one at a time. Follow this pattern and an applet will usually perform better than downloading an HTML page.

In summary, Java and CORBA do indeed provide the benefits that have recently been described in the trade journals and commercial component are available that can be integrated into an infrastructure for such applications. Beans provides an excellent way to provide a Java-centric programming model and still use CORBA to access remote servers transparently. To be successful, establish a distributed object infrastructure, design a platform-independent business architecture, and reduce the number of roundtrips over the Internet.

Rajendra Raj, Morgan Stanley

As enterprises become more global and distributed, so does the need for enterprise-wide software applications. For the past several years, I have functioned in various capacities---architect, developer and manager--of successive generations of globally distributed persistent object infrastructures that currently are "successfully" being used for a variety of high-volume real-time mission-critical financial applications. In late 1993, since no appropriate solution was found in the vendor marketplace, we decided to build Our own solution using vendor components and leveraging previous experiences within the firm. The project was successfully implemented in 1994, with successive functional releases continuing to this day; the current generation also incorporates CORBA and related technologies. It is likely that the next generation of this infrastructure will be relegated to a supporting role, primarily on the server side, in an increasingly Web-centric world of distributed computing.

This talk will summarize the lessons during the development and deployment of the above infrastructures, and raise issues that must be addressed when constructing large distributed systems. In short, there are no canned solutions to building large scale distributed object systems, and common distributed issues (e.g., naming, reliability, availability, replication, fault-tolerance, transactional integrity, persistence, security, object mobility, heterogeneity, scalability, and performance) need to be re-addressed in an integrated manner for each application due to different practical tradeoffs.

Ashutosh Tiwary, Boeing Information and support services

Building a large scale distributed system is intrinsically hard. Current distributed object technologies provide good ways of managing the complexity of such systems. However, the builders still have to deal with the other hard problems in building such systems. For example, when performance and scalability are critical requirements, the builders frequently need to look below the covers of such technologies. This either results in significant customization of existing technologies or the construction of home grown distributed object infrastructures. In this talk, I will describe my experience in building two commercial distributed object applications with home grown infrastructures. In both cases, performance and scalability were critical requirements. For such applications, I will argue that a single language environment with customizable support for distribution and persistence is more important than "buzzword compliance".

Participant Biographies

Doug Lea is a professor of Computer Science at the State University of New York at Oswego. He is author of the book ''Concurrent Programming in Java'', and co-author of the text ''Object-Oriented System Development''. He is the author of several widely used software packages, as well as articles and reports on object oriented software development including those on specification, design and implementation techniques, distributed object systems, and software reusability.

David W. Forslund, a fellow of the American Physical Society, and a Fellow of Los Alamos National Laboratory, is a theoretical and computational plasma physicist who has made major contributions in a broad range of plasma physics from space plasma physics to magnetic fusion to laser fusion and, more recently, in computer science. As deputy director of the Advanced Computing Laboratory and Los Alamos National Laboratory Fellow, he has helped guide the installation and operation of one of the largest massively parallel supercomputer systems in the world and led research projects in the practical applications of parallel and distributed computing. He has over 50 publications in refereed scientific journals and has given numerous invited talks in plasma physics and computer science for the past 25 years.

Tom Barry is an Object-Technologist (a.k.a. consultant) with LinkSoft Corp. in Chicago. He has worked with GemStone's GemEnterprise technology for the past 18 months and has worked with object-oriented technology since 1989, including C++ and Smalltalk. His latest project is a gateway to connect GemStone/S to a commercial web server such as Netscape Enterprise or Microsoft IIS to serve dynamic content. Tom has an MSEE from U.C. Berkeley and an MBA from the University of Illinois.

Don Vines is Director of the Distributed Object Computing (DOC) practice at Genesis Development Corporation, a world leader in object technology consulting. At Genesis, Don plays a pivotal role in managing major customer
engagements, technical partnerships with the major ORB vendors, and technical strategies relative to the OMG and other standards bodies. He is also co-authored numerous OMG specifications including: COM-CORBA
inter-working, Extended IDL data types, ORB-to-ORB interoperability, and IDL-to-C++ mapping. Prior to Genesis, Don was technical architect for both NEC's and IBM's CORBA product lines. He has been a

proponent and developer of the object paradigm since 1983, and is a well-known speaker and author, with articles in numerous technical journals such as C++ Report and Object Magazine.

Rajendra K. Raj is Vice President in Morgan Stanley's Information Technology Division, Where he has led the development of several generations of buzzword-compliant object infrastructures used for financial applications. Most recently, he has been focusing on the problem of semantic integration of diverse business data for access by distributed applications. Before joining Morgan Stanley, Dr. Raj was an Assistant Professor at the State University of New York at Oswego. He has authored several scholarly publications and Has given talks at universities, corporate centers, workshops and conferences including ECOOP and OOPSLA. He holds a Ph.D. from the University of Washington, Seattle.

Ashutosh Tiwary is a researcher with the Research and Technology Group of Boeing Information and Support Services and a PhD candidate in the Computer Science Department at the University of Washington. He has had about 8 years experience building OO systems and has worked on several commercial and research Large Scale Distributed OO system. He was also the chair of the OOPSLA'95 Workshop on Building Large Software Systems using Objects and OOPSLA'96 Workshop on Objects in Large Distributed and Persistent Software Systems. His current work is focused on persistence of object-oriented systems and performance analysis and tuning of large scale systems.