Real-World Java Development Experiences
A Background Data Collection System

Jerome B. Soller, James Clingenpeel, Patrick W. Hayes Jr.,
Mark Muday, Brian Larsen, and Tamara Jones
CogniTech Corporation

An Internet-based infrastructure can enable easier data entry, validation, and report generation capabilities when users are at different locations or use diverse hardware platforms and operating systems. However, there are many technical obstacles to tackle before such a solution can be successfully implemented, especially when users with diverse requirements need access to extensive capabilities and dynamic information. This article discusses the development and run-time implementation of a background data collection system, based on a three-tier architecture using Java™, the eXtensible Markup Language (XML), and relational database technologies. This solution is representative of a broader class of solutions, applicable to problems requiring the collection, verification, auditing, and reporting of data for users at different locations or who use different hardware and software platforms.

One of CogniTech's customers collects and verifies background data on various applicants to determine their suitability to perform specified tasks as a contractor or an employee for other organizations (clients). The background data include education, professional certification, employment history, criminal allegations, and citizenship. The business processes of collecting and verifying these data are time- and resource-intensive and require the centralized storage and maintenance of the resulting records.

To increase the efficiency, responsiveness, and quality of its verification services, the customer wanted to reengineer its business processes through the use of software packages for each of the three distinct but interdependent services:

- Collection of background data.
- Verification and auditing to authenticate the data.
- Generation of reports and other forms of data access based upon the verified data.

The profile of each applicant determines the data collected, verified, and reported during the process. Therefore, each applicant requires a customized user interface, which is determined dynamically, based on the clients requesting the collection and verification services. CogniTech developed an Internet solution to support the collection of the background data.

The Development of Internet-Based Data Collection System

User Perspective and Involvement

Typical users of the Internet-based data collection are applicants. Under the traditional form system, an applicant might be required to complete as many as 15 forms for different clients, many of which request the same data. From the point that the paper form is sent to the applicant, there is typically a two-week delay until the completed information is returned by mail. This translates into a two-week delay before the customer's verification stage can begin. The transcription of this information from the paper form to a computerized system can lead to incorrect copying of handwriting, a lack of consistent terminology, and inefficiencies within the verification process. To help prepare a precise set of functional requirements, two categories of domain experts—applicants and verification process domain experts—assisted in the development process.

Data Model

A common data model served as the foundation for all of the software solutions. The data model was based on input from domain experts and existing paper data collection forms associated with clients. This model divided the data into two categories: generic to all clients and client-specific. When the valid values for a field could be enumerated, these values were coded with a standard internal representation and vocabulary for presentation to the user.

Technology Selection and Challenges

Decision to Support Internet Data Entry. The first technology decision for the software development was the use of an Internet interface for data entry. This allowed centralized storage and maintenance of the software, databases, and data. It required a minimal information systems investment (a browser and an Internet connection) from its users.

Selection of Programming Language and Target Platforms. Common Gateway Interface (CGI) was the first technology considered for this project. CGI scripts running on a server can dynamically generate HyperText Markup Language (HTML) pages. The advantage of this approach is that it does not require special browser support. Even with the addition of JavaScript (a scripting language for HTML pages unrelated to the Java programming language), static HTML could not capture the dynamic behavior required by the functional specifications.

In addition to dynamic behavior, the user interface specifications required a minimal security risk, cross-platform support, software reuse, and rapid development. The Java programming language was the programming language that best met these criteria. We first looked at Java Development Kit (JDK) 1.02, an older version of Java supported by most Web browsers in use, but found that it did not support the necessary infrastructure for the rapid development of business solutions. Instead, we chose JDK 1.1 (a later version of the JDK released in spring 1997) and its Java Beans component model [2], which provided the necessary foundation for the solution. The trade-off was that JDK 1.1-compliant software requires that the user have a recent version of Netscape Navigator, Microsoft Internet Explorer, or another Java Virtual Machine (run-time Java environment).

Selection of the Software Development Environments. Microsoft Windows NT was chosen as the software development platform. After developing and testing the JDK 1.1-compliant Java software on NT, it could later be ported to a more stable, scalable, and secure enterprise-wide production environment, such as a UNIX-based server, an AS/400 server, or a mainframe. For a number of reasons, the CogniTech development team selected IBM VisualAge for Java Enterprise Edition as the primary Java development environment for this project. Among its advantages, VisualAge for Java

- Offered strong support for the platforms and technologies within the target enterprise, including JDK 1.1, Windows NT, AS/400 servers, the IBM DB2 Database, and the Lotus Domino E-mail/groupware server.
- Offered strong functionality for team development, including a development repository and strong revision control.
- Supported the model-view-controller paradigm, which encourages developers to fully utilize the object-oriented paradigm by dividing objects by their functionality.

NetObjects Fusion was chosen as the Web site authoring environment for the complex site, which anchored the Java applets. RoboHelp Office Edition supported the creation of the HTML help pages and their associated index, implemented as a Java applet.

Use of XML. We first looked at the Standard Generalized Markup Language (SGML) [1], an international standard to provide a simple, platform-independent and flexible representation. It also is widely used to encode documents. SGML does not define a set of element types (such as tags or attributes) but provides a mechanism to define markup languages to solve a class of problems. HTML is an application of SGML. We chose to implement the eXtensible Markup Language (XML) [9], a subset of SGML that provides many of the benefits of SGML, such as generalized markups to create customized tags, but removes several complex features.

The software uses XML to represent three distinct forms of business domain content:

- The visual layout of components within the user interface.
- The relationships of the screens within the user interface.
- The mapping of the visual components to external data object references, e.g., relational database data.

An intuitive visual XML editor accelerates the development and maintenance processes.

Run-Time Environment

Three-Tier Architecture

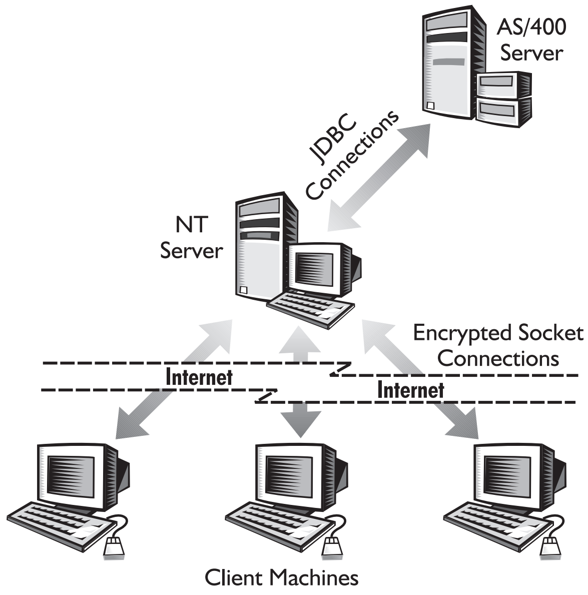
Figure 1 shows the architecture for the solution. The IBM AS/400 server contains the DB2 databases used by the software. A Windows NT server provides a firewall and second tier, which runs the Java implementations of business logic and data access. The XML represents the dynamic mapping to these server-side objects. The Java applets communicate with the server-side Java through sockets, which are low-level communications application programming interfaces (APIs). The client-side Java applets contain the user interface.

Client-Side Java

Since each applicant in the process has a different user profile, the solution generates and dynamically merges content at run-time based on each profile. This results in a user interface that supports a wide variety of navigation paradigms. However, the typical user would only use the system once every two years. Therefore, it is difficult to standardize the choice of client-side Java Virtual Machine for all users.

In theory, one of the advantages of client-side Java is "write once, run anywhere." As the technology is currently supported by many Web browsers and Java Virtual Machines, the reality of cross-platform Java support is "write once, test everywhere." Support for JDK 1.1.2 by Microsoft Internet Explorer and Netscape Navigator has only recently been implemented, and both browsers still contain minor incompatibilities with the standard. Because of the inability to fully control the client platforms used for data entry, the Java applets will run under

Figure 1. *Three-tier architecture.*



Netscape Navigator 4.04 with the Java 1.1 patch, Microsoft Internet Explorer with service pack 1, and fully JDK1.1.2-compliant Java Virtual Machines.

To address the concerns over support for future releases of the JDK, JavaSoft (Sun Microsystem's Java division) recently released a standard plug-in Java Virtual Machine for Web browsers called the Activator. In spite of these "growing pains," Java's cross-platform capabilities have significant advantages over other technologies. In environments where the enterprise controls the client platform, it is significantly easier to upgrade and standardize upon a browser, Lotus Notes client, or Java Virtual Machine than to repeatedly upgrade all computers to new operating systems.

Given customer requirements to support multiple browsers, the solution could not make strong assumptions about locally resident Java class libraries, e.g., Java Foundation Classes. Any classes not resident in the local browser or Java environment would require downloading. To minimize download time, we chose to create a lightweight set of graphical components. Due to incompatibilities in browsers, sockets were used in place of a more robust distributed object framework, such as the Object Management Group's (OMG's) [7] Common Object Request Broker Architecture (CORBA) [8]. This eliminated the need to download the class libraries for the Object Request Broker (ORB) or the Java stubs for the Interface Definition Language (IDL) of a given interface. However, the design supported the possibility of a migration to a CORBA architecture in the future.

Server-Side Java

The server-side Java application on the middle tier of the three-tier architecture handled all database transactions with the third tier AS/400 DB2 database, business logic, and dynamic user profiling. Communication with the database used the Java Database Connectivity (JDBC) [6] standard, which encapsulates Structured Query Language (SQL) statements in Java classes. For Java components only running on the server, run-time cross-platform support is not as important as run-time performance. Java has advantages over C++ because of the ease of developing and porting the software to another server platform.

Lessons Learned and New Technologies

During and after CogniTech's efforts to develop the solution, technologies and developer tools have evolved. Given its experience, the development team formulated a technical road map for future projects.

Most current and future CogniTech projects, including the development of decision support systems and occupational medicine and worker's compensation records, use an object modeling tool, e.g., Rational Rose, and a data modeling tool, e.g., Platinum ERwin, for the design phases. Rose can generate Java class definitions and CORBA IDL specifications, which serve as the starting point for the software implementation. The data modeling tools can generate an SQL Data Definition Language (DDL) specification for the database tables.

In the three-tier architecture previously discussed, the JVM, the Web server, and the database server ran as separate processes. The increasingly popular Java application servers offer an alternative for the run-time implementation of Java middle tiers. Java application servers, such as IBM WebSphere, increase the performance of server-side Java software by supporting a tight coupling of Web server HTTP services, CORBA Internet Inter-ORB Protocol, database connectivity, and JVMs.

The server-side Java issues focus on

- Speed performance.
- The ability to port development code.
- The ability to integrate enterprise data and applications.

Unlike client-side Java, a large percentage of server-side Java software requires high performance and does not require the run-time cross-platform support of Java bytecode (the intermediate Java code sent to the client machine). In the past, components with these requirements could be written in C++ and linked to the Java code through a native interface. With the creation of native code Java compilers (such as IBM's high performance Java compiler), high performance code previously written in C or C++ can now be written in Java. This combines the development advantages of Java, such as its lack of pointers, with the performance advantages of C++.

However, whether the native methods are written in Java or C++, they must interact with cross-platform Java bytecode. Many CogniTech projects will require a high-performance component written in the Java language (such as database queries, user profiling, statistical computations, manipulation of an expert systems knowledge base) to impact the dynamic generation of Java objects, which are then downloaded to many platforms.

A large business issue today is the selection of the appropriate Internet technology to solve a problem. If the user accesses a Web site infrequently, there may be limited incentive to upgrade or standardize their browser or JVM. When the user interface requirements are not complex, CogniTech recommends a dynamically generated HTML and JavaScript user interface.

Servlets [5] and Java Server Pages [4] offer Java programmers a high-performance alternative to CGI for dynamic server-generated HTML. Servlet engines provide a high-performance plug-in that runs a Java Virtual Machine and maintains data values associated with a session. The IBM WebSphere Java Application Server includes a servlet engine and manages database connections. Servlet design and maintenance benefit from the separation of business logic and presentation logic.

The Java Server Pages technology supports HTML produced from a variety of sources, including Web authoring tools. Java Server Pages allow the substitution of HTML within print statements and the combination of Java expressions with HTML tags and content.

If the user interface requires the richness of Java, the organization associated with the Web site must create a strong incentive for users to upgrade their browser or JVM. It is easier to require this standardization if the user accesses the same Internet or enterprise Intranet site repeatedly. Given this usage pattern, there is a strong business case for supporting the most recent JDK and providing the richest possible Java user interface. In these circumstances, CogniTech recommends that its clients standardize either on the Lotus Notes 5.0 client or a Web browser with the Sun Java Activator plug-in.

For many current projects, CogniTech is developing two user interfaces:

- An HTML and JavaScript user interface generated by Java Server Pages.
- A rich Java user interface, using the Java Foundation Classes [3].

In the solution described above, the developers chose not to use CORBA. However, several technology changes will impact the future use of Java and CORBA. The upcoming JDK 1.2.x will include a lightweight CORBA ORB and components of the CORBA security service. This eliminates the need to download the ORB for browsers using JDK 1.2. For browsers not supporting 1.2, the OMG is releasing a specification for minimum CORBA, which defines a minimum set of CORBA services. This specification could standardize the minimum download of a CORBA ORB.

The CORBA standards offer many technical advantages over Microsoft's competing Component Object Model (COM). However, one significant advantage of COM is that Microsoft bundles it free within the operating system. We believe the greatest challenges of CORBA technology vendors are

- Decreasing the cost of CORBA development tools, ORBs, CORBA services, and application servers.
- Improving the usability of software development environment support of CORBA, e.g., hiding the complexity of CORBA from developers.
- Continuing the development of vertical industry interface standards.
- Developing a business object facility.

Security of confidential information passed across the Internet is another rapidly evolving area. Technology and licensing issues impact the use of security in Java applications and applets. Today, implementations of Secure Sockets Layer (SSL) use RSA encryption. Although public domain implementations of RSA encryption exist, RSA's patent requires a licensing fee in the United States. From an independent developer perspective, licenses for applet downloads of the RSA encryption technology are expensive. This has hampered developer use of these technologies with Java applets. Today's options for Internet developers who do not wish to purchase the RSA licenses include

- The use of other encryption algorithms.
- Creating only HTML user interfaces, using HTTP as the protocol, and leveraging the SSL support of the browsers and Web servers.
- Using CORBA IIOP as the protocol and SSL services provided by several ORB vendors.

Several future occurrences will impact the use of encryption technologies on the Internet. In the year 2000, RSA's patent expires, and RSA encryption becomes effectively free. The Internet Engineering Task Force (IETF) developed a public key infrastructure (PKIX) standard draft, and IBM has released the source code for a PKIX reference implementation to the public domain.

Conclusion

The data collection solution described in this article represents a large class of problems that benefit from a three-tier architecture leveraging Java, XML, and relational database technologies. It is important to understand the current limitations and projected capabilities of these technologies to meet project deadlines and create software components, which can be reused and maintained over time. ◆

About the Authors

Jerome B. Soller is president of CogniTech Corporation in Salt Lake City, Utah, which provides information systems research, design, development, and consulting services. He has a bachelor's degree in electrical engineering from the Johns Hopkins University and holds a doctorate in computer science from the University of Utah. He previously served as a research instructor at the University of Utah and held several research positions for the Department of Veterans Affairs and other organizations.

CogniTech Corporation
1060 East 100 South #202
Salt Lake City, UT 84102
Voice: 801-322-0101
Fax: 801-322-0975
E-mail: Info@Cognitech-UT.com
Internet: http://www.cognitech-ut.com

James Clingenpeel, Patrick W. Hayes Jr., Mark Muday, Brian Larsen, and Tamara Jones are software engineers at CogniTech Corporation.

References

1. International Organization for Standardization (ISO), "The Standard Generalized Markup Language (SGML)," *ISO 8879:1986*, 1986.
2. JavaSoft, "JavaBeans FAQ: General Questions," http://www.javasoft.com/beans/faq/faq.general.html, 1998.
3. JavaSoft, "Java Foundation Classes: Now and the Future," http://www.javasoft.com/marketing/collateral/foundation_classes.html, 1998.
4. JavaSoft, "Java Server Pages: The Front Door to Enterprise Applications," http://www.javasoft.com/products/jsp/index.html, 1998.
5. JavaSoft, "Servlet," http://www.javasoft.com/products/jdk/1.2/cast-out/servlet/index.html,1998.
6. JavaSoft, "The JDBC Database Access API," http://www.javasoft.com/products/jdbc/index.html, 1998.
7. Object Management Group, "OMG Home Page," http://www.omg.org, 1998.
8. Object Management Group, "OMG Technical Library," http://www.omg.org/library/downinst.html, 1998.
9. World Wide Web Consortium, "Extensible Markup Language (XML)," http://www.w3.org/XML/,1998.