# Enabling Technologies for Web-Based Legacy System Integration

Ying Zou Kostas Kontogiannis
University of Waterloo
Dept. of Electrical & Computer Engineering
Waterloo, ON, N2L 3G1
Canada

## Abstract

*With the exponential growth of the Internet and the multi-tier distributed system architectures, there is an urgent demand to develop Web-based and component-based applications to reduce the time to the market and to leverage existing software. This position paper presents an approach to integrate existing legacy applications to a Web-enabled Network-Centric environment. The integration process focuses on the identification of the legacy components, their consequent wrapping using CORBA Objects, and finally the deployment of the application in the Enterprise JavaBeans platform. A scripting language that is encoded in an XML format can be used for allowing thin clients to communicate with legacy components.*

## 1. Introduction

With the growth of the Internet, there is an urgent need to develop Web-based applications. Initially the Web was presented as a giant URL-based file server for publishing static information in a hypertext electronic form. With the incorporation of client/server architectures, CGI scripts and, Java applets the Web evolved into an interactive medium that provides dynamic information services. However, these tools are slow and mostly support stateless transactions. Morover, downloading “fat” Java applets imposes many limitations due to low bandwidth Internet access. Today, new requirements for Web-enabled applications are emerging.

Firstly, organizations would like to take advantage of the Web in its various forms (Internet, Intranet, and, Extranets), for their enterprise computing. This will allow them to leverage functionality from their existing legacy systems without having to rebuild these systems [1]. Furthermore, most organizations need to port their legacy software assets to a distributed Web-enabled environment.

Secondly, the integration of heterogeneous applications and systems is forced through market requirements due to the plethora of operating systems, languages, networking protocols and, data representations [2]. With its universal access across heterogeneous hardware and software, the Web is the best choice to address this challenge [9].

Thirdly, there is a constant shift towards thin-clients in multi-tier architectures away from the “fat-client” paradigm that is predominant in the traditional client/server topologies. The latter requires more memory and longer download time. Thin client architectures shift the focus from the client to the server, by leaving as little code as possible on the client side. This results in faster applet downloading and less client RAM requirements. Thin-clients require the servers to process the data, and to provide complete services such as transactional service, security service and, naming service.

Finally, portability of code and applications become a critical issue, as more types of devices and embedded systems (handheld, wireless) are participating in distributed applications.

It is apparent that the current HTTP/CGI paradigm cannot meet these requirements. The next generation of the Web, the so-called Object-Web, applies distributed object technologies to multi-tier architectures and, is gradually adopted as the development and deployment platform for distributed applications [5].

To integrate a standalone legacy application with other systems the Web offers unique opportunities. Technologies, such as CORBA [6], XML and, Enterprise JavaBeans [10], offer powerful integration mechanisms. This position paper discusses these technologies and sketches an architecture for legacy system integration using the Web as the medium.

## 2. Enabling Technologies

### 2.1 CORBA

The predominant architecture for applications requiring access to remote objects is OMG-CORBA. In particular, it offers several advantages over DCOM and COM+, including platform, language, and vendor independence. CORBA ORBs are available for almost all operating systems, such as Windows/NT, Unix and, AIX. The CORBA IDL is used for separating the interfaces from the implementations of the remote objects. IDL as a uniform interface, bridges different programming languages including C, C++, Smalltalk, COBOL and Java. It hides the implementation details from the client code and integrates heterogeneous languages. Moreover, the CORBA IIOP over TCP/IP is an open industry standard, which allows ORBs from different ORB vendors to seamlessly inter-operate and, provides an effective backbone for system integration.

### 2.2 Component-Based Enterprise JavaBeans

Due to its platform neutrality, multithreading, dynamic loading into JVM, security mechanism and, portability, Java is a widely used environment for object-oriented programming and Web-based software development.

In the software industry, the current trend is towards developing software components with specialized functionality instead of large, special purpose software applications. Enterprise JavaBeans is a Java-based, server-side component architecture for developing, deploying and, managing enterprise-level applications. It deals with the issues that allow components of a distributed application to communicate with each other. Moreover, it allows for the developers to focus on the application business logic and provides a standard component infrastructure to quickly assemble distributed components.

### 2.3 CORBA and Enterprise JavaBeans

CORBA and Enterprise JavaBeans have been developed independently, they are complementary to each other and, can be seamlessly integrated. JavaBeans uses CORBA/IIOP as the transport mechanism for the pure CORBA client and server. An EJB server can be built on top of an ORB by using the CORBA Naming Service and transactional services.

### 2.4 XML

The Extensible Markup Language (XML), is one of the key technologies for the future Web-enabled applications. It provides a machine understandable and human readable syntax and, allows for the developers to define their own tags in different domains. XML is widely used as the standard format for data representation, data integration, data storage, message exchanging, and for defining scripting languages.

## 3. Proposed Architecture

Businesses, agencies and, software houses have invested a lot of money on their legacy applications. At this point, no one can afford to spend the same money, effort and, risk again by re-implementing the same systems for another platform or programming language. One way to minimize risk and cost is to translate the legacy code from one language to another by using translator programs [7]. Even though this approach solves the problems of porting a legacy application form one platform to another, does not solve the integration issue. Another approach is to re-architect and to translate the legacy system in an Object Oriented way and language (i.e. C++ or Java) [2]. Each identified object along with its associated methods is supposed to implement a specific function of the original system. Wrappers can be applied to encapsulate each C++ or Java object obtained from the original system and make it available to a distributed Network-Centric environment. Even though this approach is promising, it causes a lot of network traffic due to the large number of objects involved.

The proposed approach focuses on the use of Web as a infrastructure medium, the utilization of Reverse Engineering to identify coarse granularity components and, the use of Distributed Object Technologies to address issues that stem from distribution (message passing, concurrency control, interaction semantics). In brief the proposed approach consists of the following steps:

1. Use reverse engineering techniques and restructuring to identify and generate a

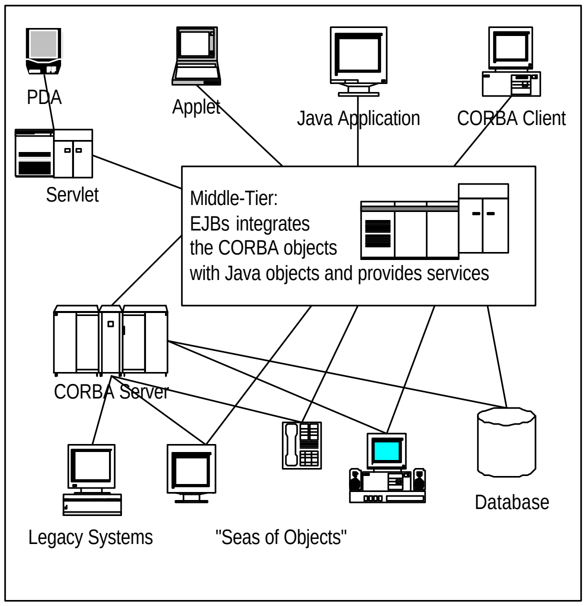

**Figure 1:** Overall Architecture

decomposition of the legacy system into modules

2. Analyze the interfaces of the selected legacy components and store their signatures in a component repository using XML format
3. Generate the CORBA/IDL and CORBA wrappers from the component repository
4. Use EJBs (Enterprise JavaBeans) to develop the application server, in order to integrate the CORBA wrappers and to provide the services to the Web-based applications
5. Define a scripting language using XML, to allow for the utilization of Web technologies for legacy components to be invoked and computational tasks to be specified.

### 3.1 Identification of Components

A component can be considered as a reusable unit, that offers specific functionality and can be easily integrated with other COTS or custom made software components. So far, most software analysis efforts have been focused on clustering and decomposition techniques based on cohesion, coupling and, other source code features. In most cases, legacy system decomposition has become a clustering problem. Nevertheless, no matter how successful clustering techniques are on decomposing a legacy system into subsystems, they do not take into account constraints related with the target architecture sought. We believe that decomposition should take into account constraints of the target architecture and, allow for these constraints to be incorporated in the clustering/decomposition process [4].
Moreover, in order to obtain the required target decomposition that allows for integration and distribution, domain knowledge has to be incorporated in the decomposition process.

For the integrity issue, a clear component interface should be specified to decoupage the internal dependencies from the clients, thereby promoting component independence and portability. The interface separates the detailed implementation from the abstract description. The specification hides the collaborations amongst the group of classes and publishes the services (functions or operations) to the consumers.

Finally, it is important to choose a standard format to describe interfaces so that, the identified legacy components can be easily shared with other applications. XML provides a light-weight data representation platform as compared to the heavy weight object database. XML markup tags can also be used for denoting the components' interfaces.

### 3.2 Wrapping Identified Components

In order to integrate the identified components to a heavily heterogeneous distributed environment, we must define an appropriate middle-ware. CORBA is the appropriate infrastructure mechanism due to its platform, language, and vendor independence specifications it supports
The wrapping process can proceed in three steps.

First of all, a single IDL interface is generated, by denoting the component interfaces in an XML format. Second, externally visible operation identifiers are extracted from the component specification and registered in the CORBA IDL. Finally, the IDL compiler generates the client-side stub and the server-side skeleton object.

The CORBA wrapper inherits from the server-side skeleton classes and encapsulates the legacy components. The wrapper acts as a façade: it offers clients a single, simple interface to the underlying remote objects. It glues together the CORBA distributed capabilities and the standalone components. The wrapper re-directs its public operations to the appropriate external-visible operations in the components.

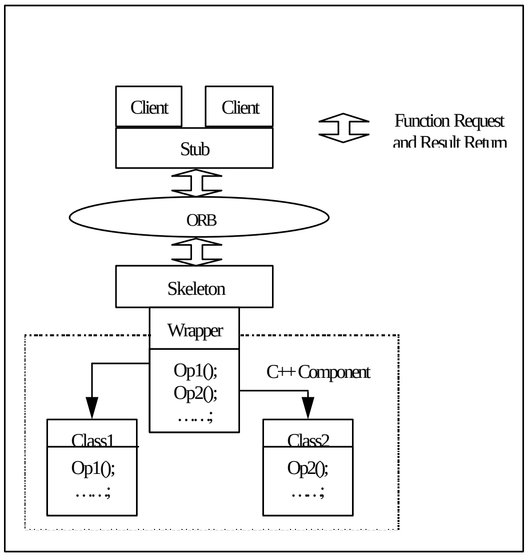

**Figure 2:** Wrapping a component

It is notable that the object wrappers are housed within the CORBA server infrastructure, providing the back-end services.

### 3.3 Integration of CORBA Objects

Application servers occupy the middle tier in multi-tier distributed applications. They have been widely adopted as the runtime environment of choice for integrating heterogeneous applications. Different application servers are implemented on different technologies. Enterprise Java Beans (EJBs) provide means to facilitate the invocation of distributed objects in a more standardized way than CORBA. CORBA is suitable for the lower level aspects of distributed applications, while EJBs handle in a more standardized way issues related to communication, security and, concurrency control. EJB is considered as a workbench for "plug and play" components and, for developing application servers. The application server assembles the individual CORBA wrappers, integrates them other remote objects and, makes services available to a front-end Web-enabled application (i.e. a Web browser).

EJBs utilize the CORBA naming services to locate the CORBA objects, and to dispatch the request to the CORBA objects over the CORBA/IIOP. They provide a link between Web-based applications and the back-end diversity software applications, databases, transactional systems.

### 3.4 Scripting Language for the Thin Clients

In a thin-client environment, applications are downloaded to the clients on-request. Clients can issue individual requests to specific servers using CORBA or any other message pasing mechanism. However, given the infrastructure that the IIOP, http, CORBA and mark-up languages now offer, it is much more efficient to allow thin-clients issue multiple requests to many servers. These requests can be building blocks of aggregate tasks that are requested by a thin-client. Moreover, the requests can implement a more sophisticated paradigm such as the Event-Condition-Action paradigm where, specific requests and actions are carried out only after specific events have been intercepted and specific conditions have been fulfilled. In order to allow for thin-clients to posses such functionality, we must define a scripting language that allows for the composition of services available in the application servers and, the formation of aggregate on-demand tasks. The scripting language can be implemented using XML and be sent to servers using the standard http protocol. EJBs can be used to intercept and interpret the transmitted by the thin-client scripts. In such a way, the client can take advantage of a Web browser to compose scripts (i.e. Web forms), and does not need a special compiler or, script interpreter. The script can be sent to a servlet. The servlet can resolve the script, call the corresponding components in the application server, and return the result to the front-end client. From the client's point of view, the scripting language is directly executable.
In such a scenario, XML becomes the primary candidate as the implementation vehicle for the scripting language. To interpret the scripts, the servlet should maintain the repository for the mapping between the script keywords and the available services in order to be directly runnable by the servlet.

## 4. On-going Work

This position paper is exploring Web technologies as an integration medium for legacy systems. Currently, we are implementing a prototype for the proposed integration architecture at the IBM Toronto Lab, Center for Advanced Studies.

We are using Refine, Rigi and PBS [3], for analysis and legacy system decomposition. We have selected the IBM Visualage Enterprise Edition to develop the Enterprise JavaBeans application server [8], the IBM Websphere as the EJBs deloyment environment and, the Visibroker as a CORBA implementation.

## References

[1] Brown, A., Barn, B., "Enterprise-Scale CBD: Building Complex Computer Systems From Components", In Proceedings of STEP'99, Pittsburgh, PA. September, 1999.

[2] G. Canfora, G., A. Cimitile, A. DeLuccia, ``Decomposing Legacy Programs:A First Step Towards Migrating to Client-Server Platforms", In Proceedings of IEEE IWPC'98, June 1998.

[3] P. Finnigan et.al. "The Software Bookshelf", IBM Systems Journal, Vol. 36, No. 4, November 1997.

[4] Gamma, E., Helm, R., Johnson, R., Vlissides, J., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994.

[5] Held, J., Susch, C., Golshan, A., "What Does the Future Hold for Distributed Object Computing", StandardView Vol. 6, No.1, March/1998.

[6] Hoque, R., "CORBA 3", IDG Books Worldwide, Inc., 1998

[7] Kontogiannis, K., Martin, J., Wong, K., Gregory, R., Muller, H. and Mylopoulos, J., "Code Migration Through Transformations: An Experience Report", In Proceedings of CASCON'98, Toronto ON., November 1998.

[8] Picon, J., Edwards, C., Scenini, G., "Using VisualAge for Java Enterprise Version 2 to Develop CORBA and EJB Applications", International Technical Support Organization, http://www.redbooks.ibm.com.

[9] S. Tilley, et.al. "On Using the Web as Infrastructure for Reengineering.", In Proceedings of IWPC '97: Dearborn, MI; May 1997.

[10] Vogel, A., Rangarao, M., "Programming with hEnterprise JavaBeans, JTS and OTS: Building Distributed Transactions with Java and C++", John Wiley & Sons, Inc, 1999