

Distributed Systems: An Effective Information Sharing Approach for Legacy Systems

S. Kami Makki

Department of Electrical Engineering and Computer Science, University of Toledo, Toledo, Ohio, USA

kmakki@eng.utoledo.edu

Abstract

An important issue of distributed systems is interoperability. Lack of interoperability between distributed systems is a common problem with current and legacy applications. Since the legacy applications represent a major investment and for the most part can not be abandoned. Enterprises are looking for solutions that enable the integration of the legacy environments with the new breed of distributed applications, as well as the integration between their existing applications caused either by mergers and acquisitions, or a company's desire to move into a new and uncharted business activity. This paper discusses the affect of legacy systems migration and integration utilizing the existing of-the-shelf middleware. It presents an analysis of distributed systems concepts and design as seen from the perspective of migration and integration of a monolithic legacy system into a distributed system, and it concludes that distributed systems protocols and architecture must be tailored to various levels within the legacy systems in order for an integration project to succeed.

1. Introduction

Traditionally there have been large differences between different vendor's development languages and operating environments, such as different primitive types, proprietary "standards", inconsistent object models, different networking protocols and different operating systems with proprietary code bases. What this has meant for businesses (large and small), is that the ability to transact between systems (which were implemented in separate vendor's products), has usually required the hand-developed codes to handle the system differences or to process the messages between systems to undertake the requests. This has resulted in an architectural maze. Also, since the software and hardware within a company were either

developed gradually or acquired and patched together over a period of time. Each element has been obtained according to the prevailing ideas relating to systems and reflects technologies that was available at the time. Investigation into these systems seems like delving through the archaeological layers in the system's bedrock. One feature of layering development is that, most layers are close together, and the migration between consecutive architectures tends to be reasonably gradual. Therefore, the businesses were given the time to find new solutions or catalyzers for integration of their legacy environments with the new breed of applications [1].

However, the gap between monolithic legacy systems and distributed systems is widening [1, 2]. Therefore, the core questions are: How do the companies move forward? How do they close the gap? Certainly, this develops a mindset that the replacement of legacy systems is as inevitable as changing a company's car fleet for the new models. This is often done as companies migrate to Commercial-Off-The-Shelf (COTS) software, enterprise management systems, and distributed systems. Many legacy systems however, have remained intact for many years through the effects of the following:

- The lack of a workable alternative (that is, an alternative that may or may not be better but guarantees a future migration).
- The lack of finance to implement a workable alternative or lack of willingness for further expenditures.
- A perfect or inadequate assessment of the requirements for a specific business.

Typically in businesses today any or all of the above factors can and may do occur. Therefore, the aim of a business which, rises from the rationale for the change, provides the focus of the business strategy. Also, the following factors such as identification of the aim, analysis of conditions (timings, standards, deliverables), and measures of success or failure (which, identifies the business or project is on or off its

track), effect the implementation of an integration. For the purpose of this paper, an integration project aim is only referred without detailing how it came to be agreed upon. Therefore, we will accept a simple generic integration project aim, such as, integration of the legacy systems into a new distributed system in accordance with the requirements of the new distributed systems, in which they need to have the majority of functionalities of the legacy systems. However, the rationale (why/whether to) for integration of legacy systems into distributed systems will not be focused on in this paper as well. It is only referred in the light of its effect on how the project is approached. To frame the analysis, this paper focuses on a distributed architecture such as Web services (since they provide a standard means of communication readily available among different applications) and the following two questions:

1. In integration of a monolithic legacy system into a distributed system, why would one apply new distributed software and architecture if the legacy system can already read and write files across the network?
2. If one would want to apply a distributed architecture, how can one re-frame and refocus understanding of the existing legacy system's architecture for integration and future distributed design?

Nevertheless, the following detailed justification for the purpose of change needs to be available at all times and be relevant to all involved:

- The tasks that we currently can not do, and are seriously important for the company and can be done using distributed systems.
- The tasks that only legacy systems can do which are important to the company.
- The system integration is required.

There may well be some other subtle assumptions or philosophies within the business aim. For simplicity we will not expand on them in this paper. Nevertheless they may need to be mentioned prior to proceeding from 'why' to 'how'. It is true that distributed systems advocate advanced technology and have to be considered as the master architecture(s) within the company. However, legacy systems are able to service the requirements of the companies. This concept may be argued in detail but here it will be accepted as a useful structure within which to proceed.

This paper is organized as follows: In the next section, we review the existing issues in the legacy

systems and distributed systems counterparts, and answer questions such as, how a legacy system integration process can be achieved. Then in section 3, we discuss the mapping of legacy systems issues. In section 4, the usage of distributed technologies in combination with legacy systems will be discussed. Section 5 provides Integration Architecture and related issues, and finally section 6 provides a summary of the issues related to the integration process.

2. Existing Issues in Legacy Systems and Distributed Systems

The "enterprise application integration" is, what the company does require. Therefore, an IT manager needs to address how and to what degree re-factor the legacy systems according to new distributed architecture requirements [4].

The existing legacy systems generally are able to read and write files across the network using tools available in the operating systems (e.g. nfs, ftp, email, etc). In other words, the legacy systems already communicate with other systems in various ways. These systems are sometimes distributed around the world, or occasionally within the boundaries of the companies however, with limited EDI/B2B activity and customer proprietary ordering and information systems. The prevailing theory can sound something like; 'give me a common interface file format standard and the power of network operating systems and I will give you a distributed system'. There is considerable temptation, which is, to 'do distributed system integration' in this way.

There is an associated philosophy that at least a 'more of the same' project to integrate/optimize legacy systems will provide or stop the gap until one or more of the above limitations to transition are overcome and the legacy systems are no longer needed. Therefore, designers which contemplating the complexity of distributed systems protocols tend to 'go with what they know' when designing architectures involving the integration of monolithic legacy systems within distributed systems.

For example, if the architecture in Figure 1 is currently implemented, then the question is; why can not we just do something similar as shown in Figure 2.

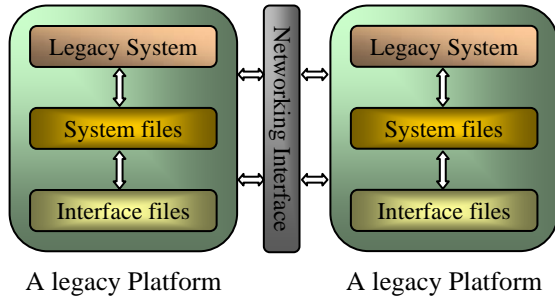


Figure 1. Basic Legacy Communication Architecture

The greatest accomplishment a legacy system analyst/ programmer can achieve is to move distributed systems in the direction of ‘front-ending’ legacy systems in an integration process. This can be made by using legacy proprietary software commonly available for the legacy systems, such as GUI, Web enabled, ODBC enabled. Therefore, this success combined with the simple architecture of Figure 2 can provide a one-tier monolithic system that has been recreated as a distributed system. At this point in time, seems that the requirements have been met; and we have a robust (legacy) back office that can talk to the Web and talk to other parts of distributed systems. The gap seems to have been bridged and the project is complete.

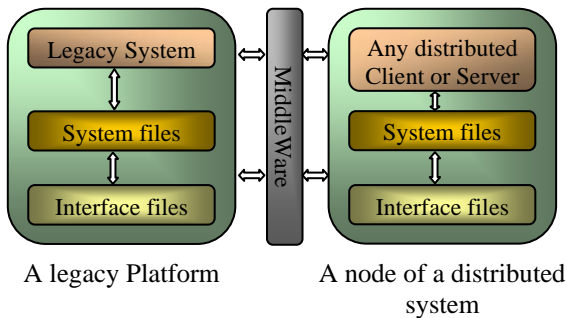


Figure 2. Basic Legacy Communication Architecture links to a section of distributed systems

As a result, many of these systems can be built, however a post project discussion with the design team reveals that, we do not have a distributed system. We have created something else. It seems that much could have been made from the degrees of openness and requirements of a distributed system compared to the simplified structure above. However, we will allow a simple expression of some distributed system requirements to guide us forward, which is:

- The applications possibly need to be decoupled. That is, different components of an application

should not be strongly dependent on each other. Instead, they should communicate via interfaces which, can result in a significantly reduced impact on the overall system, if one application is modified or replaced [5].

- Also, an important point are the interfaces to application components which are maintained at a higher level of abstraction and therefore the extensions to these interfaces can be handled more efficiently [6].

Therefore in a three-tier model the client tier becomes thinner. It is also focused on providing an easy to use graphical user interface, whereas some of the business logic moves from the client tier to application server. At the same time, the business logic that used to be buried in proprietary database extensions (i.e. stored procedures) has moved to the application server. In general, this model provides more choices in how business rules of distributed application can be partitioned across different tiers. This in turn can enable better exploitation of resources and supply an infrastructure that meets scalability requirements [5, 6]. But, how is a legacy system integration process can be achieved? This question will be dealt with in two parts; mapping of the legacy systems, and use of distributed technologies.

3. Mapping of the Legacy Systems

We must begin with the admission that preserved legacy systems are generally extremely complex systems in which distributed systems cannot replace them easily. This heavyweight nature of legacy systems tends to lean the entire system in the direction described above. Therefore, a simple syntax for describing legacy systems in the framework of distributed systems is required. It may be helpful to understand the legacy systems in terms of what they produce and consume. Whether it is a file update or, a file download or, an invoice, since a legacy system create some products.

Previously, many legacy systems were mapped to the new systems using Dublo approach [12] or the OZ approach [13]. In Dublo approach, the migration of applications takes place from legacy system to a multi-tier architecture system. The basic idea of this approach is formulating old business logic in a new business logic tier and managing the new business tier by the new presentation tier. In this approach mapping is done in step-by-step process replacing old business logic tier with new business logic tier using a legacy adapter. The new business logic tier simply pass the

data through the new presentation tier without implementing the business logic. This makes the old legacy code to bypass the new business logic tier and decouples the development of new presentation tier by porting the legacy code to the business logic tier. The advantages of using this approach are smooth migration, database consistency, database independence, reuse of legacy business logic. In the Oz approach, the migration of applications of legacy system to new system take place using a mediator object such as HTTP proxy (for gaining access to web applications). In this approach, legacy system is mapped into the Web with the help of Mediator (Http proxy). This mediator is designed to intercept the requests for data and transform them into requests for the legacy system. This mediator based approach can be used to map the legacy systems to distributed environments. The challenges to the Oz approach are, as the HTTP protocol is completely stateless, the mediator has the responsibility of maintaining the state information on behalf of the every web-based client and must serve the functionality of some of the web browser client tasks. The migration of legacy client/server applications is easy as it requires no change to legacy code. In the past CGI programs are used as mediator. Some of the problems that encountered using CGI programs are session's problem, moving large legacy client/server applications to web was very difficult as large number of CGI scripts to be written to accomplish this task. The legacy systems can also be mapped to distributed environments using CORBA [14]. In this approach, legacy systems are migrated to the distributed environments through the wrapping technique. In this technique the components of the legacy system are wrapped and are invoked from object-oriented distributed environment.

The general issues of migration that are presented in the approaches which have been discussed in the above are the degree of reusability of the legacy code; reduction of performance; difficulty in implementating code; selection of language; and the difference in data models. In the reusability of the code one cannot simply use the old code for the new system as the new system has its own requirements. One solution to this issue is the use of SOA (Service-Oriented Architecture). SOA is an architecture for building software applications that uses available network services such as Web. It allows the reuse of existing elements from where new services can be created from an existing legacy system and thus promises interoperability between heterogenous applications. The reduction of performance increases with the addition of more layers to the new system

architecture. Also the existence of various interfaces to legacy systems makes it difficult for the server-side application of the new system to implement code for the legacy systems. The selection of programming language is difficult when the new system has different functionality than the existing legacy system which cannot be fulfilled by a particular language. The difference in data models of the legacy system and the new system makes it difficult for the programmer to convert both the data models into one data model.

Also to be able to classify the various entities that the businesses amassed within their legacy systems which will be referred here as legacy 'artifacts'. These artifacts are very much essential for their day-to-day business and they can be divided into two groups: public artifacts such as statements, invoices, interface files, dispatch advices, reports, EDI uploads; and private artifacts such as internal data files, internal locks, semaphores, registers, reports. For legacy systems, these artifacts lend themselves to two basic activities: exposure of public artifacts to the distributed systems through the means discussed below, and internal management of private artifacts according to rule/protocol based integrity and state management through the application of distributed systems methodology or architecture. In this paper, we only focus on the public artifacts. Although out of scope here, the implications of the management and migration of private legacy artifacts within the architecture of Web services may also be helpful.

4. Usage of Distributed Technologies

As in [3], "before its move to Web services, Texas A&M used a series of batch file updates delivered over FTP to various applications to integrate the systems". Also as explained in [4], "Web services are building blocks for creating open distributed systems". Web services are a set of interoperable or connecting protocols which standardize distributed systems, however far it extends. In terms of our syntax above, they can be used to facilitate the introspection of data services, to allow services to embed the nature of public artifact that legacy systems have to offer. It allows legacy and the remainder of the system to understand each other.

To support an integration process with the methodology reflected in Figure 2, many tools would have to be created within the legacy systems. These include tools for rerunning, canceling, recovery, rollback, transaction, general housekeeping communication and metrics/performance measurement. These however can be provided by the

distributed architecture within a Web services framework. The Web services technology can work hand in glove with legacy systems through the understanding of the legacy system's public artifacts, and also how they are integrated using Web services or middleware. Also by changing our way of thinking about the legacy systems, that is, from a stand-alone system which is only capable of sending and receiving proprietary formatted files to an element of a Web service. The purely data passing approach to integration can be developed into a data passing/processing, transferring approach. As stated in [10], "It can make proven production level applications available as Web services. This allows for a rapid deployment of stable Web services to customers of a Web service provider". Therefore the Web services architecture (WSA) is an ideal technology to incorporate enterprise legacy applications into this new area [10]. The architecture in simple terms will look something like in Figure 3.

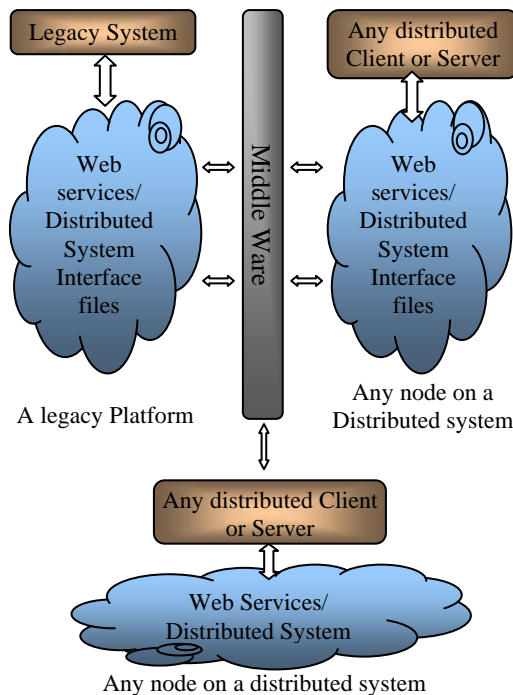


Figure 3. Basic Distributed System Architecture

The eXtensible Markup Language (XML) is a central part of the architecture [11]. As stated in [7], "An XML-RPC message is an HTTP-POST request. The body of the request is in XML, a procedure executes on the server and the values that it returns are also formatted in XML". This style and format of interface can also be used for communication between legacy and distributed systems.

Although the application of Web services to legacy systems integration has the methods and protocols to provide a robust and common framework. There is still much to achieve. Caution and a lot of specific work must still be applied. Caution relates to the maintenance of global standards as stated in [9], "companies have tendency to implement detailed standards differently, causing compatibility problems". The work required mainly refers to the actual act of setting up legacy systems 'hand in glove' with Web services systems. The challenge lies in "the options (and constraints) that exist for the activation of mainframe transactions. By supporting features such as triggering, which allows existing IBM Customer Information Control System (CICS) or IBM Information Management System (IMS) transactions to be started upon arrival of a message on a specified queue, and/or Microsoft Operations Manager (MOM) enables legacy applications to become part of an event driven application architecture" [5].

Many old and proven links methods such as simulating input/output (IO) via screen scrapers and the like will still be needed. The great benefit of Web services architecture is that, it provides the legacy system with a close associate which does have a good relationship with the rest of the distributed systems. Conducting an exercise with legacy systems to re-understand them in terms of public and private artifacts will assist in allowing Web services and legacy systems to communicate in the same language. Thus legacy systems are supported by the technologies of Web services, and distributed systems. Also distributed systems via the Web services have the power of the legacy systems within their architecture.

5. Integration Architectures and Issues

Some of the examples of architectures that are used for the integration of legacy systems are CORBA, Java J2EE connector architecture, XML, SOAP, DCOM, Java RMI, EAI, and OMG MDA [15]. CORBA allows legacy applications to communicate with one another efficiently. Using CORBA, users can gain access to information of the system without having any prior knowledge of the hardware and software platform on which it resides. Thus making CORBA an excellent technology to integrate legacy systems. Java J2EE connector architecture defines a set of standard services that allows the developers to quickly connect and integrate legacy applications with the enterprise information systems. The Extensible Mark up Language (XML) is broadly adopted for structured documents and data on Web. This can be used for integrating legacy applications into Web applications.

The Simple Object Access Protocol (SOAP) can be used as integration medium between legacy application and new system. The Distributed Component object Model (DCOM) is a protocol that enables components of the architecture to communicate directly over a network. This protocol can be used for integrating legacy applications into the new network architecture in a reliable and efficient manner. Also Java RMI can be used in the application level of integration. It enables the programmer to create distributed Java technology-based applications. Using Java RMI, methods of remote Java objects can be invoked from different Java enabled machines present on different hosts. Enterprise Application Integration (EAI) is the process of creating an integrated infrastructure for linking different systems such as legacy system, financial system, Customer Relationship Management system (CRM), Enterprise Resource Planning system (ERP), Supply Chain Management system (SCM). Object Management Group Model Driven Architecture (OMG MDA) provides tools that are necessary to integrate the middleware technologies such as SOAP, EJB, XML, and CORBA. This architecture can be used to design, implement and integrate applications using standard services. It assures interoperability, platform independence and productivity.

The important integration issues to be taken into consideration when integrating a legacy system into a new system are, how data of legacy system is to be integrated with the data of the new system, data transformation and validation into and out of each legacy application, interconnectivity of each component in the architecture, message routing between components, legacy system security issues that must be addressed in the new system.

6. Summary

In summary, the philosophy and rationale of migration and integration of legacy systems and distributed elements are valid based on rigorous analysis in which the project will be approached. The aim comes from a profound understanding of what it is that must be provided to the clients for their use. Accordingly, standards for interfacing legacy systems to distributed systems can range from simple interface file passing using proprietary standards to Web services integration. Though there are justifications for limiting integration to file passing, the requirements of a distributed system may be better served through a more thorough and closer integration across the systems, including the legacy systems. In this manner, the migration towards open distributed systems may

bring legacy systems along with it. This activity is an extreme challenge and Web services architecture may be of great value in this integration. Understanding legacy systems in terms of some 'distributed systems'-like syntax can allow legacy systems to present public artifacts to distributed systems via Web services. The XML standard provides a structure for data presentation and process management. The other services may also use this standard widely in the future. The legacy systems integration can benefit from this standard through integration with Web services. However both public and private legacy artifacts will require further development and analysis.

References

- [1] Lam W., Investigating success factors in enterprise application integration: a case-driven analysis, *European Journal of Information Systems*, Vol. 14 No. 2, pp 175-187, June 2005.
- [2] Ganti N. and Brayman W., *The Transition of Legacy Systems to a Distributed Architecture*, John Wiley & Sons, 1995.
- [3] Upakare R. and Sasikumar M., *Converting Legacy Application to Web Services using an Open Source Toolkit: A Case Study*, Workshop on Free and Open Source Learning Environment and Tools (FOSLET'06), 2006.
- [4] Michelis, G.D., et al. A three-faceted view of information systems. *Communication of The ACM* Vol. 41, No. 12, pp 64-70, 1998.
- [5] *The Essential Component for Enterprise, Client/Server Applications*, <http://www.systemsgroupi.net/>, International Systems Group, Inc 1997.
- [6] Chari K., and Seshadri S., Demystifying integration, *Communications of the ACM*, Vol. 47 No. 7, pp 58-63, 2004.
- [7] XML-RPC Specification, <http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto-spec.html> and <http://www.xmlrpc.com/spec>
- [8] Newcomer, E. *Understanding Web Services XML SOAP UDDI & WSDL*, Addison Wesley 2002, ISBN:978-0-201-75081-2.
- [9] Douglas K. B., *Web Services and Service-Oriented Architectures: The Savvy Manager's Guide*, Morgan Kaufmann Publishers, ISBN 1-55860-906-7.
- [10] Samtani G., and Sadhwani D., *Enterprise Application Integration and Web Services*, International Conference on Internet Computing, Date: 2002

- [11] Service Oriented Architecture – A field Guide to Integrating XML and Web Services, By Thomas Erl, Prentice Hall 2004, ISBN-10: 0131428985.
- [12] Hasselbring W., Reussner R., “The Dublo Architecture Pattern for Smooth Migration of Business Information Systems”, International Conference of Software Engineering, Pages: 117-126,2004.
- [13] Dossick Stephen E., and Kaiser Gail E. “WWW access to legacy client/server applications”, Computer Networks and ISDN Systems, The International Journal of Computer and Telecommunications Networking, 28(7-11):931-940, Elsevier Science B.V., May 1996.
- [14] Kim H. S. and Bieman J., Migrating Legacy Software Systems to CORBA based Distributed Environments through an Automatic wrapper Generation Technique, Proc. Joint meeting of the 4th World Multiconference on Systemic, Cybernetics and Informatics (SCI'2000) and the 6th International Conference on Information Systems Analysis and Synthesis (ISAS'2000)
- [15] Miller J., Mukerji J., et.al., Model Driven Architecture - A Technical Perspective, Object Management Group (OMG), <http://www.omg.org/mda/> (Document number ormsc/2001-07-01).

Dr. S. Kami Makki has earned his Ph.D. from the University of Queensland in 1997. He has held a number of research and academic positions prior to joining the department of Electrical Engineering and Computer Science at the University of Toledo. He is an active researcher in the fields of distributed systems, databases, mobile, wireless networks and information systems. He has numerous publications in peer-reviewed international journals and conferences, and co-edited and contributed to several books. Prof. Makki is an editor of International Journal of Infonomics and International Journal of Communications. He has also served as a technical program chair and technical program committee member and reviewer for a number of IEEE and ACM sponsored conferences. He has received a number of achievement awards such as 2003 Achievement Award from the World Academy of Sciences.