Integrating enterprise application using message-oriented middleware and J2EE technologies

S. L. Ooi, M. T. Su
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur, Malaysia

Abstract—Enterprise systems integration allows applications to be connected to one another in intra- as well as inter-organizational settings. Despite providing tremendous benefits, enterprise-wise integration is complicated. Several integration technologies are available ranging from electronic data interchange, enterprise resource planning, enterprise application integration (EAI) and web service. This study focuses on enterprise applications integration approach using Message-Oriented Middleware (MOM) and Java 2 Enterprise Edition (J2EE) technologies. The middleware serves as a bridge between the different applications involved in system integration. Using a common interface layer rather than point-to-point integration enables all the involved applications to communicate freely with each other. Moreover, EAI approach allows the reuse of existing applications besides allowing new applications and data to be incorporated. A prototype of the proposed EAI architecture is implemented in a banking environment scenario.

I. INTRODUCTION

Enterprise systems integration allows applications to be connected to one another in intra- as well as inter-organizational settings. The benefits of integrating systems include enabling easier data access, achieving data consistency, providing more efficient interoperability between multiple organizations as well as allowing new and legacy systems to co-exist and complement each another [1]. Despite the numerous benefits, integrating enterprise systems is complicated. Systems whether new or old run on diverse environments and platforms, making them difficult to be integrated without tremendous effort in achieving data consistency. Furthermore, the issue of reliability is of utmost important. Integrated system need to exchange critical and valuable business information internally in a reliable, virtually traceable way. Other than these, the inherent complexity of translating between different systems’ data structures and business processes, the availability of a wide range of integration technologies and unsatisfactory performance caused by varying degrees of support for integration [1], also contribute to the complexity in integrating systems.

The approaches of enterprise applications integration range from file and document exchange, database integration and federation, message and document translation, distributed object interaction, to workflow and business process management [1] and so on. This study focuses on enterprise applications integration approach using Message-Oriented Middleware (MOM) and Java 2 Enterprise Edition (J2EE) technologies.

Estimation showed that as much as 80% of business information is shared within and between organizations using File Transfer Protocol (FTP) technology. Despite its popularity, the use of FTP leads to high possibility of exposing valuable business information to risk. To make the matter worst, since transfers are not transparent, user may not even know when things went wrong [2].

Using MOM as the middleware in between the systems is able to increase the reliability of the integrated system when compared to using FTP technology since data logging is incorporated in MOM. The proposed integration approach is also able to support a broad set of platforms and application environments. This study aims to build a workable prototype as a proof of concept of the proposed Enterprise Application Integration (EAI) architecture and implementation.

II. INTEGRATION TECHNOLOGIES

Enterprise integration refers to the capability of integrating the functionalities of a variety of systems [3]. It covers two distinct aspects which are technical integration and behavioral integration. Technical integration refers to integrating software and hardware while behavioral integration refers to redistribution of roles and responsibilities among the members of organization.

A. Electronic Data Interchange

In the past many organizations had used Electronic Data Interchange (EDI) approach to exchange business documents in an integrated way [4]. EDI refers to the various standards that describe the format, content, and structure of the data to be exchanged between computers to permit data to be transferred to an application in another location without re-keying. EDI occurs when organizations exchange point-to-point information between their computers through private telecommunication networks [5]. The benefits of EDI can be found in [5]. EDI manages to achieve data integration but not process integration. It also does not provide the required flexibility and maintainability [4].

B. Enterprise Resource Planning

Enterprise Resource Planning (ERP) historically focuses on integration of internal business functions (internalization) [3]. Fully-integrated ERP system is capable of processing commercial functions distributed throughout the company’s components that are dispersed geographically [6]. As packaged software solution, the major benefit of ERP system is that it provides a common integrated software platform for business processes [6]. It promises to tackle inconsistency of data, incompatible format and
Enterprise Application Integration

With the advancement in Web technology, new economy, such as Electronic Commerce (e-commerce) had mushroomed. Companies can no longer refrain from expanding their business processes to their partners, suppliers, distributors, and customers. In the mid of 1990s Enterprise Application Integration (EAI) emerged to enable systems integration that is more affordable and require lesser programming. EAI refers to plans, methods, and tools aimed at modernizing, consolidating, and coordinating the overall computer functionality in an enterprise [3].

Enterprise Application Integration (EAI) had emerged to overcome the limitation of ERP by providing an integrated organizational infrastructure. It provides a solution to intra- and inter-organizational systems and process integration [4]. EAI results in a more organized business process, achieves return on investment, increases collaboration among partners, achieves data, object and process integration and reduces cost.

Earlier integration of different systems required that codes be rewritten on the target systems, which is time consuming and costly. On the other hand, EAI is related to middleware technologies such as message-oriented middleware (MOM), and data representation technologies such as XML [8]. The special middleware serves as a bridge between the different applications involved in system integration. Using a common interface layer rather than point-to-point integration enables all the involved applications to communicate freely with each other. Moreover, EAI approach allows the reuse of existing applications besides allowing new applications and data to be incorporated. The difference between EAI and traditional integration approaches is illustrated by Fig. 1.

EAI can be divided into four major categories [9]. Firstly, database linking where databases share information and duplicate information as needed. The second category is application linking where enterprise shares business processes and data between applications. The next category is data warehousing where data are extracted from a variety of data sources and channeled into a specific database for analysis. The final category is common virtual system which is the pinnacle of EAI whereby all the aspects of enterprise computing are connected to appear as a unified application.

D. Web Services

Web services approach is a relatively new integration technology. It is a distributed computing paradigm which allows applications written in diverse languages and running on multiple platforms to interoperate and integrate more easily and less expensively than other traditional integration methods [4]. The Web service protocols are built on top of application-level network protocols such as hypertext transfer protocol, simple mail transfer protocol, and file transfer protocol.

The advantages and disadvantages of the above integration technologies can be found in [4].

III. MESSAGING SYSTEM

Messaging system is an event-driven communication layer that allows enterprise applications to transfer information securely and at a pace the interacting systems can maintain [10]. It supports loose-coupling between message consumers and message producers. The resulting integrated systems are dynamic, reliable, and flexible where modification of the ensemble of sub-applications can be done without affecting the rest of the system [11]. Other advantages of messaging systems include high scalability, easy integration into heterogeneous networks and reliability [11].

The most common messaging models are point-to-point (PTP) and publish/subscribe (Pub/Sub) [12]. PTP messaging is built around the concept of queue; sender post messages to a queue and receiver retrieve these messages from queue or from another queue which is linked to this queue. Pub/Sub messaging is built around the concept of topic where message producers send messages to a specific topic and all message consumers who have registered their interest in this topic will receive a copy of the related messages. Both queue and topic are called destinations for messaging applications [12]. A destination encapsulates the address information used by messaging product to locate the end point for the delivery of message. Destination is not the endpoint itself [10]. There is a third messaging model namely Request-Reply Messaging which is often defined as a subset of one of the other two models [13]. This model is used when an application sends a message and expects to receive a message in return.
This study makes use of PTP messaging using message queue. PTP messaging model is expected to be supported by all MOM providers. On the other hand, Pub/Sub messaging model is not suitable to be implemented in this study since it is used to publish the same message to multiple applications. In the proposed EAI architecture, only one single application acts as the receiver of the message sent by the sender. Request-reply messaging model is the standard synchronous object-messaging format [13]. Contrarily in the proposed EAI approach, asynchronous messaging technology is utilized.

IV. PROPOSED APPROACH

The architecture of the proposed enterprise applications integration approach is outlined in Fig. 2 using the banking environment as an example. The figure shows the integration of ePayment system of Bank A with applications running on other banks using Java Message Services (JMS) and Message-Oriented Middleware (MOM). The ePayment system of Bank A provides payment service for its customers to pay the respective merchant over the web. These merchants which range from insurance, telecommunication, entertainment companies to e-Shop have to first of all register themselves with the bank.

![Fig. 2. Proposed EAI architecture](image)

This approach allows ePayment system of Bank A to carry out both local transactions as well as remote transactions. Local transaction refers to transaction that will be processed locally within its environment. Contrarily, remote transaction refers to transaction that will cross the JMS interface and MOM communication layer in order to interact with applications running on other parties. In the context of the banking environment used in this study, local transactions are those transactions that will be processed within ePayment System of Bank A, whereas remote transactions are those transactions that need to interact with applications running on Bank B or Bank C. To demonstrate the proposed approach, the following are developed: a simple ePayment system for Bank A, the communication layer which consists of JMS interfaces, MOM and its data logging and Message Routing System, and two applications on the remote banks. The function of Message Routing System is to route message retrieved from MOM to the specified destination which is either the application on Bank B or Bank C.

As mentioned earlier, the ePayment System of Bank A carries out both local transaction and remote transaction. For local transaction, HTTP request is sent from Web browser to Apache Web Server and forwarded to Application Server in order to produce the user interfaces of ePayment System using Java Server Pages (JSPs). The business tier is encapsulated in the Spring Framework (J2EE Open Source framework) where the business objects and its implementation are managed by lightweight container. Spring Framework reduces the complexity of creating the application when compared to using Enterprise Java Beans (EJB) [14]. It is an increasingly important integration technology and its benefits can be found in [15]. The result of the transaction will be stored at the originating application’s database, in this case Bank A’s database.

For remote transaction, the first part of the process is similar. In addition to that, the lightweight container sends object (data representation) to JMS interface. The JMS interface constructs an Extensible Markup Language (XML-) encoded message that stores the information of the transaction based on the received object. Then the JMS interface sends the XML-encoded message to MOM Server. The XML-encoded message is queued in MOM server which will be retrieved by the Message Routing System’s listener, which is also a JMS interface. In addition, the delivery of the transaction over MOM is recorded (referred as data logging in Fig. 2). When the JMS interface which is attached to the Message Routing System retrieve the XML-encoded message from MOM Server, it transform the XML-encoded message to an object and pass it to the Message Routing System.

The Message Routing System routes the transaction message to its destination which is either Bank B or Bank C’s application. The result of the transaction will be stored at the database of the destined bank. If the transaction is completed successfully, the receiving application will return a success message in the form of an object back to the Message Routing System. The Message Routing System then passes the message back to its JMS listener and constructs it into an XML-encoded message which is queued in MOM server to be retrieved by the ePayment System’s JMS interface. Once ePayment System’s JMS interface received the XML-encoded message from the MOM Server, it transforms the XML-encoded message into an object and passes it to ePayment System as the response for the transaction.

The JMS interface is used to access the messaging system. It defines a common enterprise messaging API that is designed to be easily supported by a wide range of enterprise messaging products [16]. Moreover, the JMS interface can be extended transparently across diverse platforms and networks. However, the JMS interface needs to be customized in order to use it to read data from other system because each system employs specific data format.
V. IMPLEMENTATION

The implementation of the prototype of the EAI architecture is described here. Fig. 3 shows the transaction screen of the ePayment prototype system. Once a transaction is submitted, transaction details will be sent to MQ Request Queue. This transaction will be treated as a message and MQ Server will generate a message ID for it (Fig. 4). Subsequently, this message will be retrieved by message router program to be sent to MQ Response Queue. During this process, message ID will be copied over as correlation ID. This enables the ePayment system to retrieve the correct message as the reply for the initial message (where message ID equals correlation ID).

![Fig. 3. ePayment System transaction screen](image1)

![Fig. 4. MQ Request Queue](image2)

![Fig. 5. MQ Response Queue](image3)

![Fig. 6. MQ Message Details](image4)

VI. CONCLUSION

Application integration within an enterprise is unavoidable nowadays. It helps an enterprise to achieve business competitiveness. Despite its importance, application integration does not come easily. It is expensive and requiring a lot of resources. Hence, many integration approaches had been explored to maximize on the return on investment.

In this study JMS (which is platform-independent), asynchronous communication API and MOM are used in integrating applications within an enterprise. The proposed Enterprise Application Integration (EAI) architecture is implemented in a banking environment scenario. The ePayment system of one bank is integrated with applications running on other banks using Java Message Services (JMS) and Message-Oriented Middleware (MOM). The JMS interface is used to access the messaging system (MOM). The benefit of JMS is that it defines a common enterprise messaging API which is supported by a wide range of MOM. Moreover, the JMS interface can be extended transparently across diverse platforms and networks. In addition, the delivery of the transaction over MOM is automatically logged. Logging the transaction data improves message delivery and fault tolerance since in the event of any failure, the data log can be examined and message can be re-delivered. To
conclude, the proposed EAI architecture serves as a viable alternative for enterprise application integration and is useful for supporting Business-to-Business (B2B) e-commerce.

REFERENCES


