Redesign the Integration of Movement Control and Maintenance Control Systems of Ethiopian Airline to Improve Reliability

Berhanu Bezualem
Ethiopian Airline Enterprise
berhanubezualem@yahoo.com

Mesfin Kifle
Department of Computer Science, Addis Ababa University, Ethiopia
kiflemestir95@gmail.com

Abstract

In general, as an enterprise grows, its IT challenges grow with it. One of these challenges is the integration of different legacy applications. Modern enterprise environment consists of numerous applications and systems that vary in size, complexity and age. Integrating legacy applications saves time, expense and development cost.

The same is true in the airline industry where the business is more and more dependent on IT technologies. There is a high demand of integrating applications. However, the integration solution is not without a challenge. In Ethiopian Airlines, there are two critical operational systems; movement control and maintenance systems that have to be integrated. The former is a legacy system whereas the latter is a web-based system. To integrate the two systems, interface software was developed in house and deployed for use but was not reliable as required; missing data and frequent failure. This work studied, analysed and recommended an alternative technology for improving the reliability problem of the existing integration software used to integrate these two applications by evaluating the different contemporary and current application integration technologies.

This paper discusses in detail the most common distributed technologies by considering major factors such as interoperability, coupling, serialization, reliability, ease of implementation and other factors deemed necessary. More emphasis was given to Web Service technology.

Having studied these different technologies, a mixed technology of MS MQ and Web service were selected as better solution to meet the business goal which is achieving reliability. Moreover, prototype was developed using the selected technologies to demonstrate the integration. Though WS RM is a best solution to meet this goal it was not selected in this study since the specification is relatively new and not yet implemented and tested for use in the real industry.

Keywords: Coupling; Integration; Interoperability; Message Queue; Operational systems; Reliable Messaging; Web service

1. Introduction

In general as an enterprise grows, its Information Technology (IT) challenges grow with it. One of these challenges is the integration of different applications. Virtually all enterprises have legacy applications and databases. Connecting to the legacy applications saves the time and expense of having to migrate them and provides a mechanism for tying together fragmented IT platforms. The importance of integration was very closely related with the sensitiveness of the data or application [1]. Furthermore, the distributions and analysis techniques that are commonly used are fairly specific to lifetime data. Different integration problems require different solutions, and it is important to use the right one. To address the diversity of applications and data that must be connected, different organizations based on their business targets can produce a range of integration products and technologies.

The airline industry is one of the major transport service providers. Its major business focus is providing air transport services for passengers and cargo. In most cases, it is expected that systems have to talk with each other. In addition, since products acquired from different software vendors, the integration task mainly falls on the shoulder of the internal IT team of the airlines. The existence of this problem was a reality that was also observed in Ethiopian Airline. Among the applications that have to integrate are the two main operational systems: the Maintenance Control (MaC) and Movement Control (MoC) systems.

Ethiopian Airlines (EA) is one of the best airlines operating in Africa with its hub at Addis Ababa, Ethiopia [2]. EA has a number of systems which are mainly airline specific and other generic systems like System Application Program Enterprise Resource Planning (SAP ERP) to support its business. In addition, it has also in house developed applications mainly for the purpose of integration of various systems acquired from different vendors.

Different airlines use different products and the way they use the products differ according to the business model the airline follows, for example, some MoC products include MaC application as one module; however this module is recommended for airlines that do not have a big maintenance facility [3]. In EA the maintenance facility is too wide so this module does not support the business hence they need a complete package maintenance management solution. The two major operational systems in any airline operations are the Flight Following system also called the MoC and the MaC systems. MoC deals with tracking of the position of all aircrafts of the airline in real time while the major function of the MaC is developing maintenance schedule, producing maintenance tasks, and tracking of aircraft parts wear and tear. The latter system depends on the former system flight data feed like schedule origin or destination cities, schedule departure date and time and so on. These two systems use different technologies to exchange information. MoC provides only strictly flat file for data exchange while MaC avails Web service technology is for data exchange with the other system. It is within this technology constraint that the integration software was developed and deployed for use.

Because the newly developed integration software does not provide complete data feed, daily monitoring and manual intervention is required to maintain the completeness and accuracy of the data mainly for cost and safety reasons. The integration software developed in house by EA to enable operational data exchange between the two mission critical software solutions was not reliable as required. This is because as the daily monitoring of this interface software shows there were significant numbers of messages that were rejected by the other system and in addition the software frequently fails and needs restarting again and again.

Major problem is observed on the amount of messages failed or missed from the total amount of message sent in this mission critical business. The accumulated data shows that there is problem with the current interfacing application of EA.

The points of failure of these messages were Web service, Oracle and mapped drive. These missing data or incompleteness means the accumulated usage value assigned to a given aircraft parts do not show the real situation until it is corrected manually. Here, one can easily understand how much the manual tasks are cumbersome. Furthermore, the integration application has also a problem of non-functional requirement - reliability - which is the probability of the software executing without failure for a specific period of time.

The main task of this research was studying and analyzing in-depth, assess contemporary application integration technology architectures and recommend the best technology by considering major factors that drives the direction of IT. Furthermore, software quality attributes such as performance, reliability and scalability were evaluated in terms of business goals for design decisions.

The following methods were used to meet the objectives of this study. 1) Document review and analysis 2) Experiment with several scenarios to discover how the software behaves in real time situations. 3) Literature review on remote procedure call architectures, web service integration framework and software quality attributes. 4) Prototyping: Since the main objective of this study was to improve reliability, a prototype was developed and used in order to validate the reliability improvement of the interfacing application by the use of selected integration technology for the business we have at hand.

There were no any modifications done on the two systems of MoC and MaC. In general, the scope of this study was integrating the two systems to improve the reliability problems that the EA was facing nowadays and recommending better integration technologies using different factors.

2. Literature Review

2.1 Major Distributed Computing Technologies Used for Application Integration

As pointed out in [5, 6] Web service, Distributed Component Object Model (DCOM), Java version of Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA) and socket based technologies are the major distributed technologies used for application integration. As described in [6], currently Web service technology is the one that the IT industry embraces. The main difference between the Web services approach and traditional approaches (for example, distributed object technologies such as the OMG CORBA or Microsoft Distributed Component Object Model (DCOM) lies in the loose coupling aspects of the architecture. Coupling refers to the act of joining things together, such as the links of a chain: the degree to which software components depend [4]. In a distributed environment, to determine the degree of coupling in a system, one needs to look at different levels.

Instead of building applications that result in tightly integrated collections of components, the whole approach in modern distributed computing technologies (like in Web services) is much more dynamic and very adaptable to changes in general. Another key difference is that through Web services, the IT industry is tackling the problems using technology and specifications that are being developed in an open way.

As elaborated in [5] Web services is a term that can be understood in many different ways - some argue that it means the services built using new XML based standards and services like Simple Object Access Protocol (SOAP) and others consider it to mean a broader communication process or the new Web based service phenomenon. In either way, the idea behind Web services is a clear one. Until now, a majority of Web-based services have been targeted to users accessing them with Web browsers. The next logical steps are services provided by applications for other applications. Web services is an architecture that enables the building of loosely coupled distributed systems using technology based on open standards that do not force to lock-down to a particular programming language, component model or computing platform.

In this paper, it is tried to compare and contrast different competing technologies like CORBA, RMI and Web Services. As stated in [7], for example, RMI has significant features that CORBA doesn't possess - most notably the ability to send new objects across a network, and for foreign virtual machines to seamlessly handle the new objects. As stated in [8] RMI has a lot of potential, because of the flexibility of remote method invocation, it has become an important tool for Java developers when writing distributed systems. CORBA is gaining strong support from developers because of its ease of use, functionality, and portability across languages and platforms [9]. Before completing the comparisons with web services here, let us see it with DCOM too. Table 1 [10] summarizes the fact.

As shown in Table 1 Web services solve many of the problems in other technologies. It can be used to integrate the existing distributed technologies CORBA and DCOM. This role can be argued by considering the following arguments: 1) The nature of Web Services: it is XML, SOAP and HTTP based 2) Web Services support and endorse to the specification 3) Technologies integration, not replacement.

Table 1: Comparisons among CORBA, DCOM and Web Services

Characteristics	*CORBA*	*DCOM*	*Web Services*
Specification	Specification	Implementation/specification	Specification
Platform Support	Platform-Independent	Mainly Windows, but it can support other platforms	Platform independent
Data model	Object Oriented	Object Oriented Model	Message exchange
Client-Server Coupling	Tight-Coupling	Tight-Coupling	Loose -coupling
Language Support	Any language with an IDL binding	Any language with an IDL binding	Any language
Wire Protocol	Internet Inter-ORB Protocol (IIOP)	Object Remote Procedure Call (ORPC)	SOAP and (HTTP)
Internet friendly	No	No	Yes
Firewall traversal	Not firewall friendly	Not firewall friendly	Firewall friendly

2.2 Limitations of Existing Distributed Technologies

In addition to the tight coupling, the problem with the current technologies used in creating distributed software applications is the lack of interoperability. The other problem with applying these technologies is caused by the corporate firewalls: binary serialization unlike XML serialization and most of these technologies are not future proven. The difference between binary and XML serialization is that in XML serialization it is possible to change any common language runtime objects into XML documents or streams and vice versa. The XML serialization enables it to convert XML documents into such a meaningful format that the programming languages can process the converted documents with ease. But binary serialization converts the files into a binary format which is not a human readable format. Enabling communication with Web services, possibly dynamically discovered, requires loose coupling between a requester and the service to be used [11]. Web services bring many benefits to enterprise users who are willing to capitalize on the potential they provide. They can be used to reduce highly the application development complexity and costs by providing standard external interfaces to applications and whole systems [5]. In general, Web services have the following advantages: 1) Loose coupling 2) Interoperability 3) XML serialization (plain text), not blocked by firewall, and 4) Future proof.

2.3 Reliable Messaging Technologies

a. Message Queue

The Message Queue (MQ) technology offers so many advantages than the traditional technologies. In our case we use it because of the following two important benefits and others that really fit to solve the problem that we have at hand: 1) Avoid unnecessary reading and writing (see Figures 1 and 2) Avoid sorting of files: by doing so, we improve reliability.

b. Web Service Reliable Messaging

Web Service Reliable Messaging (WSRM) describes a protocol that facilitates the reliable delivery of messages between two web service endpoints in the presence of component, system or network failure [12]. The specification outlines two distinct roles viz. a source and a sink. WSRM provides support for various delivery modes such as exactly-once and at least once.

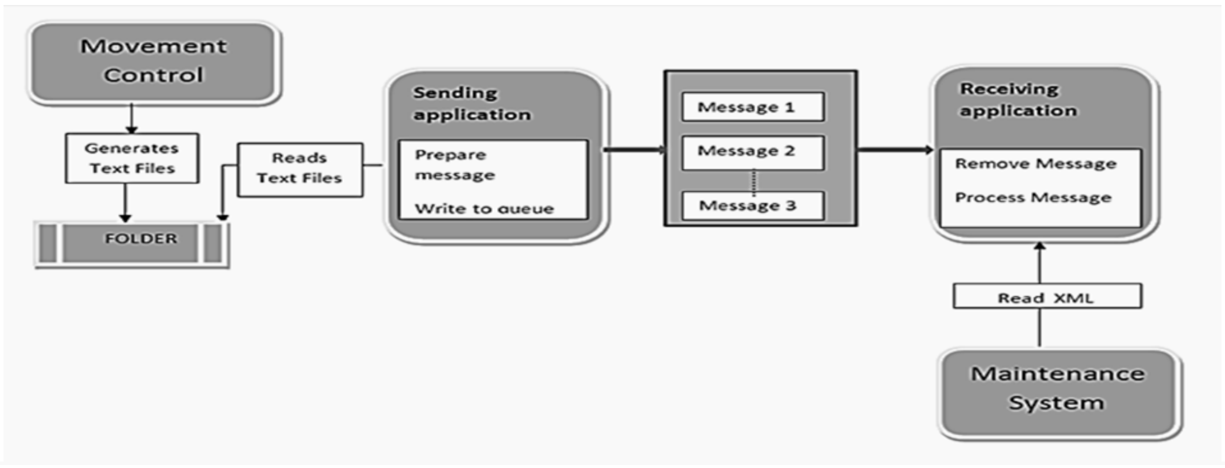

Figure 1: Microsoft Messaging Queue (MSMQ) Model in Web service

The delivery guarantees are valid over a group of messages, which is referred to as a sequence. Messages that are sent may never arrive, arrive too late, arrive multiple times or arrive out of order. After messages arrive, the service may crash and loose some messages. Application code can handle these failures with extra code, sometimes needing the cooperation of the remote endpoints using an enhanced message exchange pattern [13]. However, this approach can significantly increase the investment in the application as well as its time to market. It can also make the application more difficult and time-consuming to adapt to its changing requirements in the future. As an alternative, reliable messaging (RM) can be used as part of overall application failure recovery to improve application programmer productivity and time to market.

In general WS RM has the following major strengths: 1) Perform a task asynchronously, meaning it executes the next task without waiting for a result; 2) Performs in disconnected or connected environment (store and forward protocol) which means we don't need a dedicated connection. This improves the reliability in terms of communication which is the main aim of this study; 3) Message sequences are tracked and maintained; and 4) Interoperable which means it can perform in different platforms. However, RM is not yet fully tested and implemented.

3. Findings of MoC to MaC Applications Integration

3.1 Application Description

The current working system of the business is relying on the two operational systems; the MoC and the MaC systems [2]. MoC is the system where setting up the message format is performed. This is done specifically on Movement Control Administrator (MCA). In the existing business setting we have different message types which are generated from the MoC: a) Planned (NEW) Messages: There are two scenarios for these messages to get generated: when the flight schedule is processed and when there is a Tail change from one aircraft to another or when the tail change is from an aircraft with Tail to with no Tail (dummy Tail). b) Movement (MVT) Messages: These messages get generated when there is actual time insert or change in the Movement Control called OOOI and respectively to mean OUT, OFF, ON, and IN. c) Cancel (CNL) Messages: These messages get generated when the flight is cancelled or deleted in MoC. The second one is the MaC system which uses Web service.

3.2 Extent of the Weakness of the Existing Integration Software

In order to show the extent of the problem in the current integration interface, we try to collect data from system log. Table 2 shows this concisely.

Table 2: Summarized data collected from existing system log file

Date	*Total message Sent*	*Number of Undelivered Messages*	*% Failure Rate*	*Major Failure Reason*
Nov 11	17796	206	1	Web service Transport
Dec 11	20176	553	3	Web service Transport
Jan 12	17937	558	3	Web service Transport
Feb 12	12389	173	1	Web service Transport
Mar 12	14975	435	3	Web service Transport
Apr 12	14706	268	2	Web service Transport
May 12	18199	290	2	Web service Transport
Jun 12	10086	161	2	Oracle Connection
Jul 12	20228	535	3	Web service Transport
Aug 12	49970	575	1	Web service Transport
Sep 12	53342	1157	2	Web service Transport
Oct 12	53373	2853	5	Web service Transport
Nov 12	63446	2054	3	Web service Transport
Dec 12	64236	2418	4	Web service Transport
Jan 13	59948	2788	5	Web service Transport
Feb 13	69669	10509	15	Web service Transport

The data was collected from the log file of the existing integration software and from its technical documents. The software logs errors happening in each day. Moreover direct observation on the software was conducted to observe how the system behaves in a real time. This research was conducted by analyzing the data collected from the existing system log file. This log file was used to assess the negative impact it has indirectly on the business. In general the problem was found to be too difficult and the average 17 months (about 11/2 years) failure rate is 2.5% for this mission critical business which is directly related with safety and cost.

3.3 Findings and Analysis

The root causes of these failures were identified as: 1) Web service data communication, 2) Oracle connection: During the study there were few messages identified left undelivered, 3) Extensive string manipulation to handle message ordering and XML mapping. The other finding is that the software uses extensive string manipulation to sort data coming from source application in a sequential manner, and 4) Extensive data writing or reading tool from the database: This has a direct negative impact for messages not be delivered to the destination. To solve the problem the appropriate technology was identified by the study. The nature of the business requires reliability. According to the opinion of experts in the field, the error tolerance is zero for this business. However, the observed data indicates that the error rate is very high and needs a significant improvement on the existing integration software.

4. Solution Design

4.1 Factors in Integration Technology

The goal of integration here is to connect diverse pieces of software of EA. However, developing a comprehensive integration solution to integrate the applications from the two vendors by providing a unified solution to improve reliability is the main target although giving complete solution which is fully tested using today's latest integration technique like reliable messaging is a challenging one. Currently, the available product mappings for the integration application and extended enterprise patterns focus on the use of the following technology options (see Table 3 [14]).

Table 3: Technology options across the patterns

Business Drivers	*Direct Connection*	*Broker*	*Serial Process*	*Parallel Process*
XML	✓	✓	✓	✓
Web services	✓	✓	✓	✓
J2EE Connector	✓			
Java Message Service (JMS)	✓	✓	✓	✓
Message-oriented Middleware		✓	✓	✓
Flow languages			✓	✓

Different data types are used to solve according to the problem at hand. Because this study was primarily analysis on the effect of lack of appropriate applications integration, selecting the right application integration technology and data was done in order to improve reliability. The data source for this research was from the movement control system of EA in the planned or NEW Messages.

To send a message to a queue requires only a few steps: First, obtain a reference to the appropriate message queue (using the queue's path, format name, or label), and then use the Message Queue object's Send method. Generally, that is all about specifying: 1) a queue by its path, 2) a queue by format name, and 3) a queue by label. In general, there are three main tasks that have to be done in order to implement MQ. These are: Creating Queues, Deleting Queues, and Sending Messages. For a message to be sent from application A to application B, there are two main tasks or procedures. Figure 2 illustrates message communication between two applications using the window service of MQ. The two main procedures are: 1) The Web Server of window service receives the messages from application A and makes queues (see Figure 2), and 2) We wrote a script called XML mapping that converts the text message into XML schema so that application B's Web service.

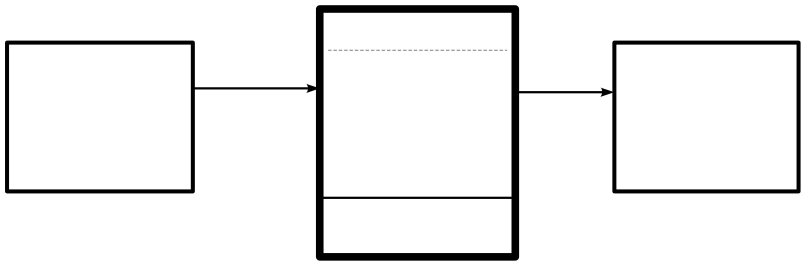

Figure 2: Communication of two Applications Using the Window Service of the MS MQ

4.2 Evaluation

The prototype was designed and developed based on the business decisions. Accordingly, the mixed technologies MQ and Web services were used to implement the prototype. This software was tested based on a copy of 24 hours actual data. The general observation was that: 1) It successfully works as message exchange software, 2) It stores the messages sent by application A (the source) until application B (destination) is up and running, 3) The software works without failure for 24 hours and this is the indication that the software can manage the processing of a day volume intensive task without any problem. This was achieved mainly unlike the existing software which uses a database extensive write or read manipulation for handling data exchange management, the prototype software uses the MQ built in functionality of data exchange. This is also an additional contribution for achieving reliability.

5. Conclusion

The current working system of the airline business operation is relying on the two operational systems. In EA's case the MoC system uses Web service implemented in the MaC system to interact synchronously using Web service messaging technology. There are a lot of points to consider while deciding on the integration technology between applications like the current infrastructure, time to market, future expansion plans, reliability and transaction support.

In this paper, we tried to asses different software integration technologies. Accordingly, there are two broad categories of integration technologies: the traditional and current ones. From the current technology, we selected the two technologies: Web service and MQ. Using Web service and MQ for implementation, a prototype was developed using the script written in VB.Net in order to demonstrate the concept in integration. Moreover, the role of the database management technology was removed and the mapping of the message format from text to XML schema was managed without the need of the data management system.

In the future: 1) We recommend integration technology tasks should consider WS RM as a better integration technology option because it has a number of features which address the shortcomings of the current integration technologies and 2) Thoroughly assessing the complex features of MQ and Web Services as integration technologies also has to be conducted.

References

[1] Microsoft Corporation, Understanding Microsoft Integration Technologies, September, 2005, http://technet.microsoft.com/en-us/library.

[2] Ethiopian Airline Business Process, Movement Control and Maintenance system, November, 2012

[3] Sabre, Sabre Maintenance Control System, Texas USA, 2007.

[4] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson, "Web Services Platform Architecture", Crawfordsville, Prentice Hall PTR, 2005.

[5] Ron Schmelzer, Travis Vandersypen, Jason Bloomberg, Madhu Siddalingaiah, Michael D. Qualls, Sam Hunting, Chád Darby, David Houlding, and Dianne Kennedy, "XML and Web Services Unleashed", Sams Publishing, 2002.

[6] Timo Ahokas, "Using XML in Making Legacy Application Data Accessible Through Web Services", Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Telecommunications Software and Multimedia, 2003.

[7] An Introduction to Complex Event Processing in Distributed Enterprise Systems, 2000, http://www.informit.com/articles/article.aspx

[8] Reilly, D., "Mobile Agents - Process migration and its implications", November 1998, http://www.davidreilly.com/topics/software_agents/mobile_agents

[9] Reilly, D., "Introduction to Remote Method Invocation", October 1998, http://www.davidreilly.com/jcb/articles/javarmi/javarmi

[10] Morgan, B., "CORBA meets Java", Java World, October 1998, http://www.javaworld.com/jw-10-1997/jw-10-corbajava.

[11] Gunjan Samtani, "Using Web Services for Application Integration", 2002.

[12] Shridee Pallickara, Geoffrey Fox, Beytullah Yildiz, Sangmi Lee Pallickara, Sima Patel, and Damodar Yemme, "An Analysis of the Costs for Reliable Messaging in Web or Grid Service Environments", Indiana University.

[13] George Copeland, "Web Service Reliable Message Programming", April 2008.

[14] Technology options for Application Integration and Extended Enterprise patterns. IBM Corporation, 2004.