An Interoperability Study of ESB for C4I Systems

Abdullah Alghamdi, Muhammad Nasir, Iftikhar Ahmad
Department of Software Engineering, College of Computer & Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Kingdom of Saudi Arabia.
{(ghamdi,mnasir,iftahmad)@ksu.edu.sa}

Khalid A. Nafjan
Computer Technology Department,
Riyadh College of Technology, Riyadh, 11543, Kingdom of Saudi Arabia.
{khnafjan@rct.edu.sa}

Abstract—**Ever since the inception of the idea of collaborating the enterprise systems, the need of an Enterprise Service Bus (ESB) has been a relentless need of the market, the bigger the systems get after collaboration the failures of the ESB's was inevitable. Things moved to more gravity when the bulkiest of the systems like Defense architectures came into picture, with the advent of this not of the efficiency but also the factors like stability, reliability, resource utilization were also of pivotal importance. This paper reviews a critical and comparative analysis of the current ESB's keeping in view the C4I System as a base, so as to ascertain which ESB fulfills the requirements of the system of systems. In comparison we try to analyze Mule ESB, GlassFish ESB and Fuse ESB with respect to interoperability. This study demonstrates that Mule is more feasible to C4I systems because it is simple, easy to integrate, no adopter requirement and flexible.**

Keywords-C4I, Enterprise Service Bus (ESB), Fuse ESB, Glass Fish ESB, Mule ESB, Service Oriented Architecture (SOA), Interoperability

I. INTRODUCTION AND MOTIVATION

Enterprise Service Bus (ESB) is a fundamental constituent of Service Oriented Architecture (SOA). An ESB provides secure message transfer service between applications and interoperability using web services and related technologies. ESB provides loosely coupled services. ESB can be used to connect different army wing's systems to communicate with each other and share certain information. The applications communicate with each other by services invoking in a location independent fashion using ESB [1].

Command Control Communication Computer and Intelligence (C4I) provide the army commanders situational awareness, information about friendly forces, location and status of enemy forces. The army commander then takes decision on the basis of this information. But the commander should have relevant knowledge and always have good experience and take some stress. After taking decision about his forces, friendly forces and also enemy forces commander have to convey it to his and friendly force. For doing this commander needs to be supported by tools to enable and accelerate decision making and planning for war strategy. The process of C4I technology is to expedite the chain links and through that targeting information should pass to weapons.

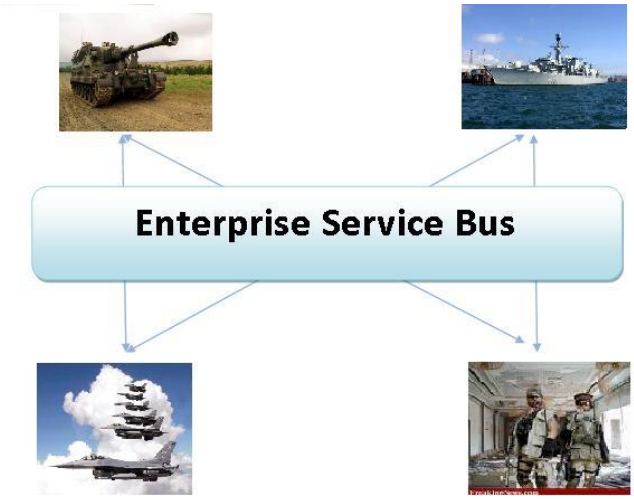

Figure 1. Enterprise service bus support interoperability between different C4I systems of various forces

A better and reliable tool can provide a better option to commander to take decision. To interconnect different systems of defense for proper exchange of data especially during a war, ESB plays an important role. For this regard defense systems required loosely coupled systems that can work together and as well as independent. Defense systems require a middleware that can interconnect their systems with each other and should be reliable and strong in interoperability and data transferring.

The paper is structured as follows; background, related work, enterprise service buses, methodology and conclusion.

II. BACKGROUND

Service Oriented Architecture (SOA) provides loosely coupled services that are Operating System or programming language independent. Further, it adds this facility through web services just like Simple Object Access Protocol (SOAP), Representational State Transfer (REST), Remote Procedure Call (RPC), Common Object Request Broker Architecture (CORBA) etc. SOA provides facility for creation of applications using services that can be organized in different ways to make novel applications [2].

SOA has emerged new criteria for software development and system integration through existing web services over the

internet. Presently, web services are being developed rapidly. SOA has become an effective way in enabling different applications to share data and work together over wide area networks [3].

ESB assists as an infrastructure backbone for SOA applications and services and facilitate enterprise integration. ESB especially reduces cost and time to create new processes through reutilization of existing applications and data. ESB is considered more reliable for delivering messaging across services even over hardware layer, and in critical circumstances like network or software failure, the shot messages are buffered and secured by the ESB's and delivered when the system is up and running again.

A feature that makes ESB attractive to users is that, its ability to exploit configuration more than codification. There is nothing wrong with writing codes. However, there are plenty of codes to be written elsewhere that does not have anything to do with interdependencies between applications and services, thus making the applications capable and difficult to manage [4].

For object level of composite applications an ESB provides composition of services, communication between objects and service deployment functions. Composite application development methods also identify the state and transitions a document goes through as each supplementary service is called, while processing an overall process [5].

Without proper communication the desire target cannot be achieved. The technology used to transport data between systems especially in Defense Information Network System (DINS) consist of information services and transfer system. Information services provides secured, unsecured voice, data electronic mail, video conferencing and images etc to users with user owned equipments. All the activities in C4I system is depend on telecommunication and computing support. Defense Information Systems Agency (DISA) is responsible for planning, developing and providing information services to war fighters [6].

To make a secure defense system is a great deal in its true sense; this is because of tremendous rise in threats in day-to-day world. Many systems and methods are being used to make secure army defense systems. For example Network Intelligence (NI) which needs Processes: Department's member of intelligence bodies established to implement technologies which is more valuable and useable for everyone. National Intelligence analyzed their capabilities as to which technologies they have and what else is needed to ensure the security of the defense systems [7].

At this point of time many ESB are available to connect different system and synchronize them so that they can easily communicate with each other. Different companies are providing their ESBs. It is very difficult to choose an ESB in accordance with the set parameters and requirements. Hence, we are going to evaluate Fuse, GlassFish and Mule ESBs.

III. RELATED WORK

To evaluate ESBs with respect to user needs is a challenging task. There are many acceptable criteria for evaluating ESBs. Acceptable criteria are those that lead closely to a particular ESB that fulfill the requirements of SOA application. Different researchers apply many criteria to conduct the evaluation. Researchers usually compare general ESBs, open source ESBs or commercial ESBs. Every researcher imposes his own list of criteria to conduct his evaluation and the most commonly base is cost. Cost is an important factor but it turns ineffective when open source ESB are compared.

Woolley applied Vollmer and Gilpin's evaluation criteria to two open sources ESBs, such as Apache Service Mix and Mule Source Mule. They included current offering, strategy, market pressure and integration into the list of criteria. Woolley suggested that Mule ESB is the best and after this Fiorono ESB. Other ESBs were BEA System Equalogic Service Bus, IBM WebSphere Enterprise Service Bus and Apache ServiceMix [8].

Desmet et al. compared two open sources ESBs such as Apache ServiceMix and Mule Source Mule, and also two commercial ESBs like IBM WebSphere Enterprise Service Bus and BEA Systems Aqualogic Service Bus. This research was on performance. Because of the flexibility ESBs may turn into bottleneck if complicated messages use it with many processes. In this worst case any business or defense process might be paralyzed. Hence, the performance is an important criterion for evaluation. Desmet et al. rated Open ESBs first and commercial ESBs after them. ESB rates were based on the performance test results [9].

Vollmer and Gilpin conducted evaluation on eight commercial ESBs with hundred criteria, which were in three grouped as market pressure, current offering and strategy. They rated Cape Clear first and second to BEA Aqalogic Service Bus. Other ESBs were IBM WebSphere ESB, Fiorano, IONA Artix, PolarLake, Software AG and Sonic. ESB rates were based on surveys and briefings [10].

Vittie also evaluated commercial ESBs. He used integration, price and core bus feature as evaluation criteria. He rated BEA Aqualogic Service Bus first and second to Oracle SOA Suite. The others were Fiorano, Cape Clear, Tibco Software, IBM Websphare Enterprise Service Bus, Sonic and Software AG. This is based on information provides by the consumers or was taken from the previous studies [11].

IV. ENTERPRISE SERVICES BUSES

a) MULE ESB

Mule ESB offer simple development model and lightweight architecture, so integrating, interoperability and creating services are easy and fast. Mule ESB needs low CPU (Central Processing Unit) and memory and simplify deployment and maintenance. Mule ESB does not need to replace or change existing system it can easily work with any existing infrastructure. It can easily deploy in any topology with or without an application container. Mule ESB also provide same performance and reliability challenges that are required for even large SOA implementations.

Mule provides pluggable connectivity and common transports such as JMS, HTTP, SMTP, FTP, POP3, and XMPP are supported natively, as are web services. Messages transferred through MULE ESB along one of these protocols can behave like synchronously or request-response [12].

Messaging system that is typically used in Mule ESB is JMS but any other messaging server can also be implemented such as Microsoft Messaging Queuing (MSMQ), IBM WebSphere MQ or TIBCO Rendezvous. There are no specific rules for integration service layer when using Mule ESB. We can connect mainframe applications, web services, messaging, sockets etc and interact with them consistently.

Mule is lightweight integration platform and service container that allow quick and easy interoperability to applications. It is Java based messaging framework that allow quick and easy connectivity of application and enable exchange of data between them. Plug-in architecture of Mule provides the facility for building block facility. Mule use SOA that integrate existing system easily. Regardless of the different technologies the applications use, including JMS, SOAP, REST, MQ, JBI, AQ, caching, JavaSpaces, GigaSpaces, Email, IM, JCA, AS400 Data Queues, System I/OWeb Services, JDBC, HTTP, and more, Mule seamlessly handles interactions among them all [13]. It can be use easily with any application server or as standalone. Mule components can be any type and can easily integrate anything from a Plain Old Java Object (POJO) as a part of any other framework.

Mule ESB does not require any specific programmatic code Application Programming Interface (API) to run its components. It provides facility of connectivity over several protocols just like HTTP, SOAP, JMS, SMTP, FTP etc. Mule also provides support of integration with Spring Framework and Business Process Management (BPM).

Mule handles interaction among technologies that applications use, including JMS, Web Services, Hyper Text Transfer Protocol, Java Database Connectivity and many more. Mule has capacity to manage all interactions between applications and components transparently. No matter, whether they are on same machine or over internet. In Mule, no specific messaging format, it can be in any format from SOAP to binary files. Mule relies on JMS for the support of high availability. Mule has no prescribed message format. It supports XML, CVS, Binary, Streams, Record and Java object etc. It provides the facility of zero code intrusion. Objects are fully portable without any Mule specific API on service object.

Mule provides messaging framework that reads, transforms and send data as message between applications that are not able to read or process data coming from another application [13].When source applications connect to Mule and want to share data with other target applications, it reads data from one former, change it completely as needed so that can be read by other application, and then sends it to the later. This functionality of Mule enables to integrate all types of applications even that are not built for integration [14].

The main advantage of Mule ESB is it allows different applications to communicate with each other within intranet or over the internet. Mule has an advantage that it can convert data as needed but other ESBs have to create an adapter for every application and convert the data into single common messaging format. In Mule no need for any kind of adapters to connect applications and not required a common messaging format. Information sent on any communication channel, such as HTTP or JMS, and is translated as needed along the way.

b) GlassFish ESB

GlassFish ESB provides lightweight integration platform with fast development tools and deploy SOA components with free dependencies and flexibility. GlassFish ESB is easy to integrate and provides interoperability. It contains GlassFish application server, NetBeans tooling, JBI runtime for deploying solutions, integration engines, adapters for external systems, and simple installer. GlassFish ESB provides JBI container that support components and includes a Normalized Message Router (NMR) to locate appropriate service providers [15].

Interoperability option provides facility to communicate heterogeneous systems. Make easy to develop secure, cross platform web services that are reliable and faster that will operate heterogeneous environments.

GlassFish ESB is reliable and high performance infrastructure. It increases interoperability and scalability for different systems with different architecture and provides secure interoperability for exchange of information related to any defense wing. Glass Fish is highly integrated, scalable application integration solution for SOA adapters. It contains Glass Fish application server, Net Bean s tooling, Java Business Integration (JBI) runtime for deploying solutions, integration engines, adapter for external system, and simple installer [16].

GlassFish is Open Source ESB it provides lightweight interoperable tool and flexible and without dependencies. GlassFish provides pluggable architecture, through these components and services that can be interoperable, allow users and vendors to plug and play.

GlassFish ESB is based on Open ESB that delivers a platform for integration, Enterprise Architecture Integration (EAI) and Service Oriented Architecture (SOA). Based on large number of standards, such as JBI, Java EE and SOAP and so on, allows enterprises to build flexible, healthy solutions for integration their system using a large number of components including binding components (adaptors) and service engines (processors) [17].

GlassFish ESB use JBI component architecture's asynchronous and decupled designed model that allows vertical and horizontal scalability. Advantage of Staged Event-Driven Architecture (SEDA) can be taken because of its synchronous, message based nature, and this provides minimizing blocking threads, associated memory requirements and scalability applications without explicit code.

c) FUSE ESB

FUSE ESB can easily be embedded at endpoints that allow distributed systems to intelligently interact without mandating a centralized server. FUSE ESB has a pluggable architecture

and work with other integration components already being used just like OSGi, JMS, JCA and JMX etc. FUSE ESB supports JBI and OSGi architectures that allow using their preferred service solutions in their SOA [18].

FUSE ESB based on Apache ServiceMix and is a fully standard base and open source interoperability platform for enterprise information technology organizations. FUSE ESB allows organizations to use their service solution in their SOA with pluggable architecture. It is lightweight in interoperability so, all FUSE components provide the solution; can easily setup at endpoints. In FUSE without mandating centralized server allow distributed systems to interact intelligently.

FUSE ESB is part of application integration and messaging components based on apache projects that also includes FUSE Message Broker, FUSE services framework and FUSE mediation router. FUSE ESB is one of a family of components that includes FUSE HQ, FUSE Message Broker, FUSE Services Framework, and FUSE Mediation Router. The FUSE components are tested for interoperability, certified, and supported to combine the speed and innovation of open source software with the reliability and expertise of commercially provided enterprise services [19].

FUSE ESB provides facility to use their preferred service solution in their SOA with pluggable Java Business Integration (JBI) and Open Service Gateway initiative (OSGi) architecture. It also provides the support for Spring Framework, which is lightweight container for application components. The advantage of Spring Framework is provides advantage to write light weight JBI components using POJO. FUSE ESB also support interoperability through Web services to complex and distributed services or standalone service.

To locate appropriate service provider JBI container support components and includes Normalized Message Router (NMR). FUSE ESB JBI container and FUSE ESB components deployed with any standard JBI-compliant Service Engine or with Binding components. NMR provides the interface for connectivity between service providers for many-to-many connectivity. In JBI Components Service Engine provide some the logic needed to provide services for massage transformation, composition or advance message routing, inside of the JBI environment. They can only communicate with other components inside of the JBI [20].

In JBI components Binding Components provide access via a particular protocol outside the JBI environment to services. They implement the logic needed to connect to a transport and receive a message through that transport. Binding components are also responsible to normalize the message for JBI environment.

The NMR uses Web Service Definition Language (WSDL) based messaging model to act through the message exchange between JBI components. WSDL based model provides insurance that the JBI components are fully decoupled. The WSDL model defines operation between service provider and service consumer through message exchange [21].

V. METHODLOGY

The procedure or methodology introduced in this work consists of steps such as selection of goal, determine evaluation scale, determine evaluation criteria, assignment of priorities, result and comparative analysis and final decision as shown in the figure below.

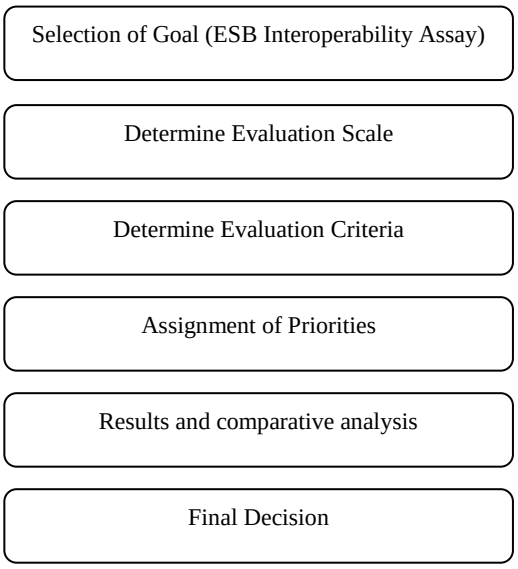

Fig. 2 Evaluation procedure

a) Selection of Goal

The first step of methodology is selection of goal. The goal is ESB interoperability analysis for C4I system. The main aim of this work is to analyze semantic, syntactic and network interoperability for C4I architecture framework.

b) Determine Evaluation Scale

The scale used in this work is fifteen point scale. The scale contains numbers 1 to 15. So, we use this scale for measuring different types of interoperability aspects for C4I systems.

c) Determine Evaluation Criteria

The interoperability study is divided into criteria such as semantic interoperability, syntactic interoperability and network interoperability. Therefore, we analyze interoperability on the basis of three aspects like semantic, syntactic and network.

Semantic interoperability provides exchange of data in messaging format between systems. It is necessary to use both messaging standard and coding of messaging data with a vocabulary standard so that receiving system can easily interpret the data being exchange. In C4I system semantic interoperability is critical in exchanging information between different wings of forces to ensure good decision about any critical situation especially during war. Without semantic interoperability systems will create islands of forces

information which can be accessed by only subset of that wing. Missing of semantic interoperability will lead towards redundant of data entry, unnecessary duplicate testing etc.

Syntactic interoperability allows detecting syntax errors and also allows to receiving system to request for resending the message that is not received properly or misrepresent. Without syntactic interoperability proper communication is not possible. In C4I systems every wing has its own system and different syntax, so syntactic interoperability plays a major role to make better and proper communication between all systems properly and without errors.

Network interoperability is the continuous ability to send and receive data between interconnected networks providing the level of quality expected by the end user customer without any negative impact to the sending and or receiving networks. Network interoperability plays a major role in C4I system. In C4I system every wing of force is working separately and connectivity of every system is possible only if proper network interoperability exist. It provides a common platform for every system either they homogeneous or heterogeneous.

d) Assignment of Priorities

The priorities are assigned to each EBS such as Mule, Fuse and GlassFish based on knowledge and experience from reviewed research as mentioned in related work and enterprise services busses' literature. We rate 6 to Mule ESB in semantic interoperability because it provides very simple and no specific message format, it can be any type from SOAP to binary files. After Mule ESB we rate GlassFish ESB 5 and FUSE ESB 4, both provide bit more complicated messaging format as compare to Mule ESB. For syntactic interoperability we rate 6 to Mule, in it no need to create adopter for multiple application or systems to communicate. Information can be transfer through any communication channel such like HTTP or JMS and is translated as needed along the way. After Mule ESB we rate FUSE ESB 5 and GlassFish 4 for syntactic interoperability. Both need adopters for exchanging information. For network interoperability we rate 6 to GlassFish ESB because its provide dependency free and flexible integration of systems. It also provides more secure and reliable integration between forces as compare to Fuse ESB and Mule ESB.

e) Results and comparative analysis

Results derived using above method are shown in the Table1. The total weights of Mule, Fuse and GlassFish are 16, 14 and 15 respectively. Therefore, the use of Mule ESB in defense architecture framework is more appropriate as compared to others enterprise service busses.

TABLE I. COMPARATIVE ANALYSIS OF ESB

Criterion	Mule ESB	Fuse ESB	GlassFish ESB
Semantic Interoperability	6	4	5
Syntactic Interoperability	6	5	4
Network Interoperability	4	5	6
Total	**16**	**14**	**15**

f) Final Decision

The results and comparative analysis indicate that the use of Mule ESB in the architecture of C4I system is suitable because it has more features and strengths in semantic and syntactic interoperability. Further, it will help more to tackle interoperability issues faced among different C4I systems of various forces such as army, air and naval forces. The GlassFish and Fuse buses have secondary rating in this assay process.

V. CONCLUSION

This paper described a brief introduction of enterprise services buses keeping in view the C4I System as a base, so as to ascertain which ESB fulfills the requirements of the system of systems (SOS). Further a comparative analysis of three main Enterprise Service Buses (ESBs) for instance Mule, GlassFish and Fuse is made. We also assessed their feasibility to defense system applications such as C4I systems. The comparative analysis of ESBs such as Mule, GlassFish and Fuse is made using weight assignment on the bases of literature surveyed. This study described that Mule is more feasible to C4I systems as compared to Fuse and Glassfish because it is simple, easy to integrate, no adopter requirement and flexible.

Acknowledgment

Special thanks to all ASERLAB members for their valuable suggestions throughout this work.

REFERENCES

[1] Martin Keen, Amit Acharya, Susan Bishop, Alan Hopkins, Sven Milinski, Chris Nott, Rick Robinson, Jonathan Adams and Paul Verschueren, "Patterns: Implementing an SOA using Enterprise Service Bus", IBM Redbooks, SG24-6346-00 , July 5, 2004. http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf

[2] Luis Garces-Erice, "Building an Enterprise Service Bus for Real-Time SOA: A Messaging Middleware Stack", 33rd Annual IEEE International Computer Software and Applications Conference, pp. 79-84, 2009, doi: 10.1109/COMPSAC.2009.119

[3] James Bean "SOA and Web Services Interface Design", Morgan Kaufmann, ISBN13: 978-0-12-374891-1, Oct 2009. http://www.elsevier.com/wps/find/bookdescription.cws_home/717477

[4] Antonio J. Rao, Valverde and Jose F. Aldana Montes, "Extending ESB for Semantic Web Services", On the Move to Meaningful Internet System: OTM 2008 Workshops, Springer Berlin/Heideberg, Volume 5333/2010, ISBN: 978-3-540-88874-1 , DOI: 10.1007/978-3-540-88875-8_122 , pp 957-964, Nov 19, 2008, http://www.springerlink.com/content/v62727537h6751v3/fulltext.pdf

[5] Frank Cohen, "Fast SOA", Morgan Kaufmann, ISBN-13: 978-0-12-369513-0, ISBN-10: 0-12-369513-9, Chapter 2: Managing the XML Explosion, Nov, 2006. http://www.elsevier.com/wps/find/bookdescription.cws_home/710099

[6] Abdullah S. Alghamdi, "Evaluating Defense Architecture Framwork for C4I System using Analytic Hirarchy Process", Journal of Computer Science 5(12): 1075-1081, 2009.

[7] Saurabh Mittal, Bernard Zeigler, Jose L. Risco Martin, Ferat Sahin and Mo Jamshidi, "Modeling and Simulation for systems of systems Engineering", Chap 5, Wiley [Imprint], Inc. 2008. http://acims.arizona.edu/PUBLICATIONS/PDF/Mo_Chapt_5_MittalZeiglerJoseFeratV4.pdf

[8] Robert Woolley, "Enterprise Service Bus (ESB) Product Evaluation Comparisons", Utah Department of Technology Services, Oct 18, 2006;

http://dts.utah.gov/techresearch/researchservices/researchanalysis/resources/esbCompare061018.pdf: Webpage accessed on Dec 15, 2009.

[9] Stein. Desmet, Bruno Volckaert, Steven Van Assche, Dietrich Van Der Weken, Bart Dhoedt, Filip De Turck, "Throughput Evaluation of Different Enterprise Service Bus Approaches", Conference on Software Engineering Research and Practice; Jun 25-28, 2007, ISBN: 1-60132-033-7, 1-60132-034-5 (1-60132-035-3), CSREA Press, pp. 378-384. http://www.ibcn.intec.ugent.be/papers/3032.pdf

[10] Ken Vollmer and Mike Gilpin, "The Forrester Wave: Enterprise Service Bus, Q2 2006", BEA Systems, Nov 17, 2006, http://whitepapers.zdnet.co.uk/0,1000000651,260256988p,00.htm

[11] Lori MacVittie, "Review: ESB Suites", Networking Computing, CMP Media LLC, March 10, 2006. http://www.networkcomputing.com/wireless/review-esb-suites.php

[12] Peter Delia, Antoine Borg, Ricston Lts "MULE 2: A Developer Guide to ESB and Integration Plaeform", Professional and Applied Computing, Apress, ISBN: 978-1-4302-0981-2 (Print) 978-1-4302-0982-9 (Online), DOI 10.1007/978-1-4302-0982-9, chapter 1, pp 1-28, Feb 7, 2009. http://www.springerlink.com/content/978-1-4302-0981-2

[13] http://www.mulesoft.org/display/MULE/Home: Webpage accessed on Sep 10, 2009

[14] Kruessmann T., Koshel A., Murphy M., Trenaman A., and Astrova I, "High availability: Evaluating Open Source Enterprise Service Bus", Information Technology Interfaces, Proceedings of the ITI 2009 31st International Conference , Jun 22-25, 2009, pp 615-620. DOI: 10.1109/ITI.2009.5196157 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05196157

[15] Vasiliev and Yuli, "Beginning Database-Driven Application Development in Java EE Using GlassFish", Apress, ISBN: 978-1-4302-0963-8 (Print) 978-1-4302-0964-5 (Online), Springer Link Date: Apr 21, 2009, DOI: 10.1007/978-1-4302-0964-5. http://www.springerlink.com/content/978-1-4302-0963-8

[16] Sun MicroSystems, "Sun GlassFish Enterprise Service Bus", 2009 http://www.sun.com/software/javaenterprisesystem/javacaps/glassfish-esb-ds.pdf: Webpage accessed on Dec 30, 2009

[17] Michael Shephared, "Semantic Interoperability", CHPSTP Workshop, Dalhousie University, May 25-27, 2005, http://web.his.uvic.ca/chpstp/RLE01/RLE%20Shepherdv2.pdf: Webpage accessed on Nov 30, 2009

[18] http://en.wikipedia.org/wiki/FUSE_ESB: Webpage accessed on Jan 5, 2010.

[19] http://www.fusesource.com: Webpage accessed on Sep 1, 2009

[20] Adam Badura,Bartosz Sakowicz and Dariusz Makowski, "Integration of Management protocols based on Apche ServiceMix JBI platform", CADSM 2009. 10th International Conference, pp: 381-384, Feb 2009, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04839857

[21] Tijs Rademakers and Jos Dirksen, "Open Source ESBs in Action", Manning Publications, ISBN 1933988215, Sample Chapt 1, Sep 2008 http://www.manning.com/rademakers/sample_ch01_ESB.pdf