Selecting the Best Alternative SOA Service Bus for C4I systems Using Multi-criteria Decision Making Technique

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

***Abstract*— A Service Bus is a distributed infrastructure used for enterprise integration. It resolves the problem of connecting heterogeneous C4I systems among various defense forces. The current work focuses on describing a detailed comparative analysis of three SOA service buses; Mule, Fuse and GlassFish for C4I systems using Multi-criteria Decision Making Technique (MCDM). In this paper, the phenomenon of selecting the best alternative is based on two criteria; main criteria and sub-criteria. The presented results may be used for integrating different C4I systems and can further assist enterprises in selecting a service bus in an optimized way.**

***Keywords- Service Oriented Architecture (SOA)*, Command, Control, Communications, Computers and Intelligence (C4I) system, Multi Criteria Decision Making (MCDM), Service Bus**

I. INTRODUCTION AND MOTIVATION

The increasing uses of C4I systems in defense and civil areas had made it more vital and smart. Therefore, this justifies that the researchers, analysts, designers and developers are taking much interest in C4I systems. Presently there are many issues in the integration of heterogeneous C4I systems that may be minimized using SOA service buses. There are many service buses available in the market today but the problem is which one is more suitable. Therefore selecting a SOA service bus has become a difficult task because many factors have to be considered. This paper describes an assessing mechanism of three SOA service buses namely Mule, Glassfish and Fuse keeping in view the C4I System as a base, so as to ascertain which SOA service bus fulfills the requirements of the system of systems. The assessing mechanism consists of two criteria; main criteria and sub-criteria. We evaluated and rated the SOA service buses on the bases of main criteria and sub-criteria which are further processed by assigning priorities, and calculating weights. The C4I systems are used in various departments such as; defense, police, investigation, road, rail, airports, oil and gas where command and control scenarios exist. The main focus of these systems is in defense applications. C4I systems consist of people, procedures, technology, doctrine and authority and play a growing role in information management, data fusion, and dissemination [1-2].

A service bus is a fundamental constituent of Service Oriented Architecture (SOA). An SOA service bus provides secure message transfer service between applications and interoperability using web services and related technologies. SOA service bus provides loosely coupled services. Service bus 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 service bus. A service bus assists as an infrastructure backbone for SOA applications and services and ease enterprise integration. A SOA service bus notably reduces cost and time to create new processes through reutilization of existing applications and data. Service bus is considered much 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 service bus and delivered when the system is up and running again [Different companies are providing their service buses. It is very difficult to choose a service bus in accordance with the set parameters and requirements [3-5].

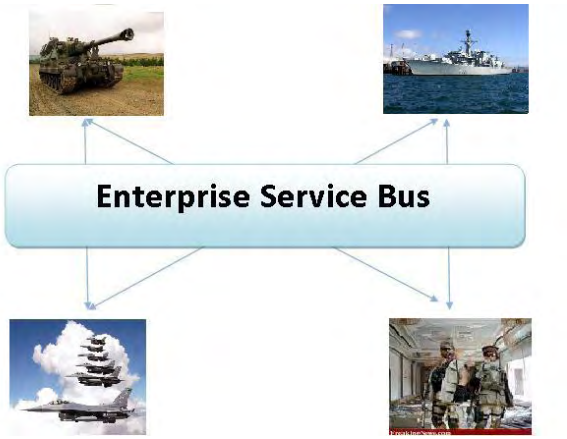

Fig. 1. Enterprise service bus support interoperability, availability& messaging between various C4I systems

978-1-4244-7626-8/10/$26.00 ©2010 IEEE

The rest of the paper is organized into the following sections related work, methodology and implementation, results, and conclusion.

II. RELATED WORK

Much work has been done in the area of ESBs evaluation with respect to user needs because it is a challenging task. Many researchers used different mechanism to compare and evaluate them based on certain criteria. But the important criteria are those that lead closely to a particular ESB that fulfill the requirements of SOA application. Researchers usually compare general ESBs, open source ESBs or commercial ESBs. Every researcher imposes their own list of criteria to conduct their evaluation and the most commonly base is price. Price is an important factor but it turns futile when open source ESB are compared.

One of the distinct works is done by Woolley who applied Vollmer and Gilpin‟s evaluation criteria to two open sources ESBs, such as Apache Service Mix and Mule Source Mule. He 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 [6].

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 [7].

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 [8].

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 [9].

Tobias et al. evaluated open sources enterprise services buses such as Fuse, Mule and OpenESB on the basis of criteria like stateless, stateful, extensibility and failover. They rated Fuse as first and Mule as second and OpenESB as third. Their study revealed the need to identify critical information resources and expose them through loosely coupled, reusable, and composable services for successful composition into workflows. Without the basic raw material of workflow, the information consumed by the business process, the value of ESB orchestration would be severely limited. Without access to information freed of business process presumptions, the re-combination of information by the orchestration engine in a new process would be difficult to achieve [10].

Interoperability is an important issue in designing and development process of C4I systems. Alghamdi presented an approach to evaluate different architecture framework for C4I system using MCDM such as AHP [11].

III. ENTERPRISE SERVICE BUSES (ESBs)

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 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 and 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 along one of these protocols can behave like synchronously or request-response [12].

Messaging system that is typically use 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. 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 Service Oriented Architecture (SOA) that integrate existing system easily. 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 [13].

Mule ESB does not require any specific programmatic code Application Programming Interface (API) to run its components. Mule also provides support of integration with Spring Framework and Business Process Management (BPM). Mule has capacity to manage all interactions between applications and components transparently. No

matter, whether they are on same machine or over internet. 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 [14].

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. 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 [15].

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 [16].

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 Beans tooling, Java Business Integration (JBI) runtime for deploying solutions, integration engines, adapter for external system, and simple installer [17].

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, robust solutions for integration their system using a large number of components including binding components (adaptors) and service engines (processors).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 [18].

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 has pluggable architecture that supports JBI and OSGi architectures that allows using their preferred service solutions in their SOA. 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. In Fuse without mandating centralized server allow distributed systems to interact intelligently. Architecture of Fuse ESB and GlassFish ESB is same. Fuse ESB is part of a family 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. JBI container support components and includes Normalized Message Router (NMR) to locate appropriate service providers. 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, orchestration 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].

IV. METHODOLOGY AND IMPLEMEMTATION

The methodology incorporated in this evaluation consists of goal selection, decision of criteria; determine the alternatives, building hierarchy, assignment of priorities, calculation of weights, and consistency test as shown in Figure 2. Further, this work is implemented using multi-criteria decision making software.

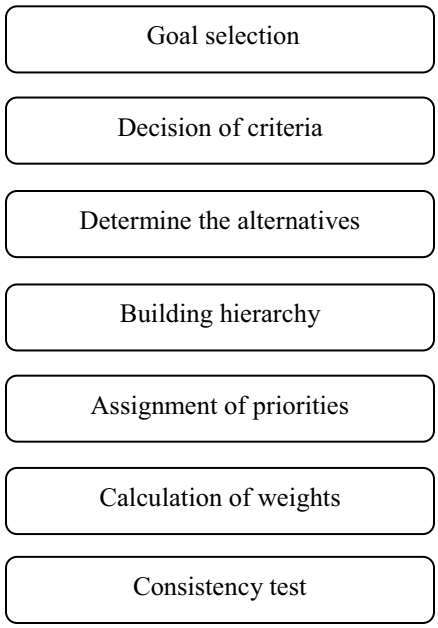

Fig. 2. Methodology Process

A- Goal Selection; First of all, we selected a goal for this work. The goal is best selection of service bus for C4I system. Three service buses namely Mule, GlassFish and Fuse are selected for analysis purpose.

B- Decision of Criteria; Secondly, we decided criteria and sub-criteria. The main criteria consist of Interoperability", „Extensibility", „Messaging", „Easiness", and „Availability". The main criteria are further divided into sub-criteria. The criterion „Interoperability" is divided into sub-criteria namely „Syntactic", „Semantic" and „Network". In the same way, the criterion „Messaging" is divided into „Reliability", „Security" and „Speed". The „Availability" is further divided into sub-criteria such as „State less", „State full" and „Failover". The selection of criteria and sub-criteria is based on the works as done by many other researchers [2, 6, 10, 11].

C- Determine the alternatives; Thirdly, we determined the alternatives such as Mule, GlassFish and Fuse. These alternatives are the focus of this work.

D- Building Hierarchy; The hierarchy is built on the bases of criteria, sub-criteria and alternatives as shown in Figure 3. The goal "the best selection of service bus for C4I systems" is at top of the hierarchy. The criteria and sub-criteria are shown in the middle. The alternatives are at bottom of the hierarchy but these are not shown due complexity in the diagram.

E- Assignment of Priorities; The assignment of priorities is based on the information obtained from previous works [2, 6, 10, 11]. The scale used for pairwise comparison is nine points scale as shown as Table 1.

TABLE I.
PERIORTY TABLE

Intensity	Definition
1	Equal importance
2	Weak importance
3	Moderate importance
4	Moderate importance plus
5	Strong importance
6	Strong importance plus
7	Very strong importance
8	Very strong importance plus
9	Extreme importance

F- Calculation of Weights; The weights of each node (criteria, and sub-criteria) are calculated on the bases of assigned priorities as shown in Table 2 to Table 5.

G- Consistency Test; The consistency ratio is calculated based on the weights. If the consistency ratio is less than 10 percent, the inconsistency is acceptable. Otherwise, we need to revise the subjective judgment.

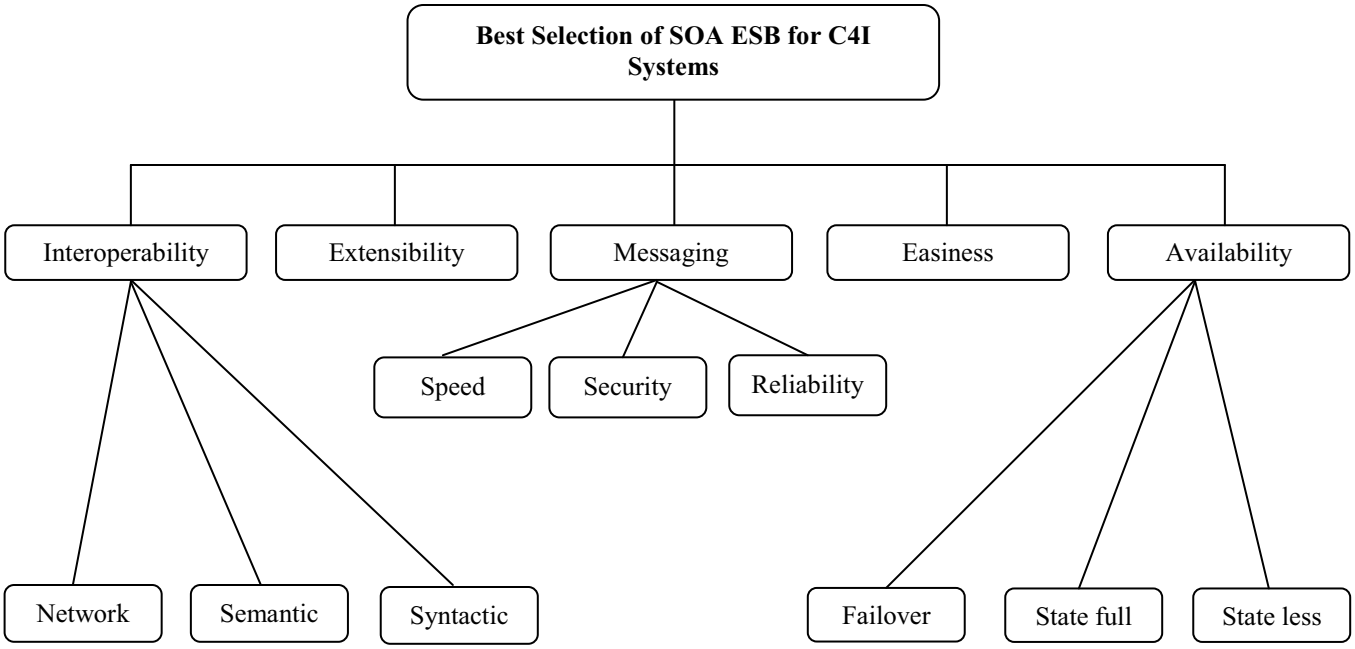

Figure 3: Hierarchy consist of goal, criteria and sub-criteria

TABLE II.
CRITERIA WEIGHT

Weights	Interoperability	Extensibility	Messaging	Easiness	Availability	Total
Local	0.38	0.09	0.23	0.12	0.18	1.00
Global	0.38	0.09	0.23	0.12	0.18	1.00

TABLE III.
INTEROPERABILITY SUB CRITERIA WEIGHTS

Weights	Syntactic	Semantic	Network	Total
Local	0.33	0.33	0.34	1.00
Global	0.12	0.12	0.14	0.38

TABLE IV.
INTEROPERABILITY SUB CRITERIA WEIGHTS

Weights	Reliability	Security	Speed	Total
Local	0.32	0.21	0.47	1.00
Global	0.07	0.05	0.11	0.23

TABLE V.
AVAILABILITY SUB CRITERIA WEIGHTS

Weights	State less	Sate full	Failover	Total
Local	0.33	0.33	0.34	1.00
Global	0.06	0.06	0.06	0.18

Table 2 shows weights of main criteria. The sum of local weights is equal to 1 and same for global weights. Interoperability sub-criterion weights are shown in Table3. The sum of interoperability local criteria‟s weights is equal to 1 and sum of global weights is equal to 0.38 that is its global weight. Same is the case with sub-criterion Messaging and Availability.

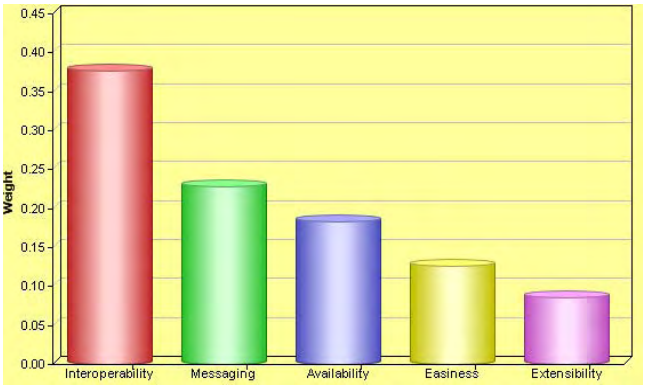

Fig. 4. CRITEIA‟S ANALYSIS

V. RESULTS

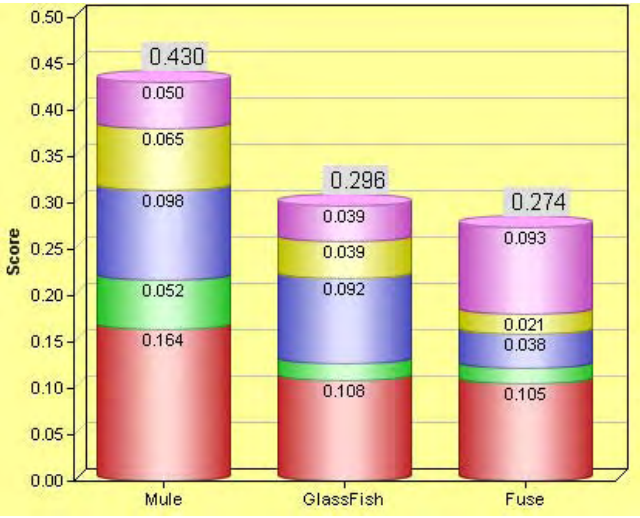

Fig. 5. ALTERNATIVE ANALYSIS OF MULE, GLASSFISH & FUSE

Figure 4 shows criteria ranking such as interoperability, messaging, availability, easiness and extensibility. Results in Figure 5 shows that Mule is best service bus in the application of C4I system. The GlassFish is rated as second and Fuse as third in this work of assessment. Further, Fuse is best in case of extensibility. However, Mule is preferable as compared to other service buses like GlassFish and Fuse.

VI. CONCLUSION

The multi-criteria decision making technique has been used to evaluate three SOA service buses; Mule, GlassFish and Fuse. The evaluation process consists of main criteria and sub-criteria. According to our study, we have concluded that among all the service buses, the Mule is most suitable to tackle the current issues of C4I systems such as interoperability, messaging, availability, and easiness.

REFERENCES

[1] Lean Weng YEOH, Ming Chun NG, "Architecting C4I Systems", Second International Symposium on engineering Systems MIT, Cambridge, Massachusetts, June 2009.

[2] Abdullah S Alghamdi, Iftikhar Ahmad, Muhammad Nasir, "Evaluating ESB for C4I Architecture Framework Using Analytic Hierarchy Process", 9th International Conference on Software Engineering Research and Practice (SERP10), Las vegas,Nevada, USA, in press.

[3] 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.

[4] 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 2004. http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf

[5] 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_Mittal ZeiglerJoseFeratV4.pdf

[6] Robert Woolley, "Enterprise Service Bus (ESB) Product Evaluation Comparisons", Utah Department of Technology Services, Oct 2006. http://dts.utah.gov/techresearch/researchservices/researchanalysis/resources/esbCompare061018.pdf: Webpage accessed on Dec 15, 2009.

[7] 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, pp. 378-384, Jun 2007. http://www.ibcn.intec.ugent.be/papers/3032.pdf

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

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

[10] Tobias Kruessmann, Arne koschel, Martin Murphy, Adrian Trenaman, Irina Astrova, "High Availability: Evaluating Open Source Enterprise Services Buses", Proceedings of the ITI 2009 31th Int. Conf. on information Technology Interface, June 2009, Cavtat, Croatia.

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

[12] Peter Delia, Antoine Borg, Ricston Lts "MULE 2: A Developer Guide to ESB and Integration Plaeform", Professional and Applied Computing, Apress, DOI 10.1007/978-1-4302-0982-9, chapter 1, pp 1-28, Feb 2009.

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

[14] Vina Ermagan, Ingolf H. Krüger and Massimiliano Menarini, "Aspect-oriented modeling approach to define routing in enterprise service bus architectures", ACM New York, pp. 15-20, 2008. Bookmark DOI: http://doi.acm.org/10.1145/1370731.1370735.

[15] Ricky E. Sward and Kelly J. Whiteacre, "A Multi-Language Service Oriented Architecture Using an Enterprise Service Bus", ACM New York, Volum 28, pp. 85-90, Issue 3, December 2008, DOI: http://doi.acm.org/10.1145/1454497.1454489,

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

[17] 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

[18] Mike Somekh, Mark Foster, Rastislav Kannocz "GlassFisg ESB High Availability and Clustring, Sun Micro Systems", White paper, Chapter 1 Introduction, Dec 2009. https://www.sun.com/offers/docs/glassfish_esb_ha_wp.pdf

[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