Bachelor Thesis in Software Engineering
June 2012

Overcoming the Initial Obstacles With the Enterprise Service Bus

Johannes Björk and Fredrik Olsson

Contact Information:
Authors:
Johannes Björk
johannes@johannesbjork.se

Fredrik Olsson
fredrik.m.olsson90@gmail.com

University advisor:
Nina D. Fogelström

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet : www.bth.se/com
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract. This bachelor thesis looks into the use of the Enterprise Service Bus(ESB) as the central hub in system integration, which is currently the newest standard for large-scale system integration. The thesis is focused on issues with ESB integration and solutions to these problems, as the positive aspects have already been covered in other reports. There is not much available information about the issues that can occur when using an ESB architecture and how to resolve these issues. The contribution of this thesis is gathering known issues from both academic papers and from the industry and providing solutions to these issues. The solutions we found can be used by new ESB developers to avoid common mistakes and also might also assist by providing a solution in case an issue already had occurred.

Keywords: ESB, Enterprise Service Bus, ESB issues, Service Bus problems

Table of Contents

1 Introduction

A lot of companies have an increasing need to integrate different software to create systems that is able to automatically handle large parts of business logic. Systems that are integrated with each other can save both time and money, as they can cut out time consuming, manual handling of data transfer.

The current de facto standard for software integration is called Enterprise Service Bus (ESB)[9]. The ESB handles all communication between different individual systems (nodes) that are to be integrated in the finished system.

The ESB needs to be able to provide different interfaces for nodes to communicate and also needs the capability to transform data so that all nodes receives data that it can understand and work with.

But how easy is it to start up with integration using an ESB?

The goal of this paper is to find as many issues as possible concerning software integration with an Enterprise Service Bus (ESB), and also try to find solutions to these issues.

The purpose is to make integration with ESB easier by pinpointing the available solutions to known issues and if no solutions are available, try to find our own methods to deal with those problems.

The aim is to provide some kind of solution to every issue that we encounter during the literature review section.

Our methods consist of two parts:

1. Literature review - Information will be collected from research databases available via BTH and the collected articles will give a base of issues that needs to be addressed.
2. Interviews/Surveys - Based on the result from the literature review, a survey will be created where we will try to confirm if the found issues are also experienced by the practitioners. If so, do they have a possible solution of their own and if not, how are the issues handled.

The result of this paper will hopefully provide a guideline to developers that are new to integration with ESB. A resource that will help them get past the most common issues and give them a quicker start with development.

1.1 State of the art

Enterprise Service Bus: A Performance Evaluation [15] provides performance comparisons between different ESB solutions. It compares ServiceMix[2], Mule ESB[1] and WSO2 ESB[3] by stress testing the core functionality of an ESB for each solution. Getting on Board the Enterprise Service [11] brings up various problems relating to the implementation cost of an ESB and also the complexity of migration and older integration solutions into an ESB. It brings up the problems of what functionality goes in the nodes if possible and what goes in the ESB to get the best performance. An Integration Strategy for large Enterprises [6] discusses pros and cons when choosing an integration solutions. For example

that using SOA when dealing with homogeneous system is not the most cost effective solution because of the additional overhead that is not required when integrating non heterogeneous systems. A High Performance Enterprise Service Bus Platform for Complex Event Processing [7] suggests improvement of algorithms to increase performance and backs it up with mathematical equations. A Scalable and Dynamic Service Oriented Workflow Model [18]. Discusses the problems with the UDDI and takes up improvements for a better UDDI where services picks the best service for the task depending if it is online if not it auto picks another from the choices the UDDI provides.

What we have concluded from the available literature we see there is a lack of information concerning the problems and shortcomings of ESB integration. Most of the papers we read discussed the advantages of ESB compared to older integration solutions like Hub-and-Spoke and Point-to-Point. Many of the papers focused on comparisons of different ESB solutions, those do not provide information on disadvantages of an ESB solutions just performance data on various ESB products. Because of these conclusions we decided to focus our thesis on the disadvantages of the ESB integration architecture.

2 Theory

In this section we will provide the background knowledge that can be useful to be able to understand what we are investigating. The techniques and terms explained are:

- Point-to-Point Integration
- Hub-and-Spoke Integration
- Enterprise Service Bus (ESB)
- Service Oriented Architecture (SOA)
- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description Discovery and Integration (UDDI)
- Representational State Transfer (REST)
- Client-server
- Stateless
- Cache
- Uniform interface
- Layered system
- Code-on-demand
- a description of our current software engineering project

2.1 Point-to-Point Integration

The existing integration architecture is the result of the Point-to-Point integration pattern. Initially not much thought was given to the integration because of software development was mostly done on a application level. Most of the

communication was an internal one. While this method of integration was done quickly and cheaply it became very hard and expensive to expand if new connectivity was required. Today's integration problem as a result of tightly fitted software coupling could be traced back to this early integration pattern. The topology makes it very hard add a node into this integration pattern. Because every node in the web needs to have connectivity with each other, increases each additional node the complexity of the system. The formula for calculating the number of connection for each additional node is (n(n-1))/2 [6] where n is the number of nodes that needs to be integrated. For example if 5 nodes needs to be integrated 10 additional connections are required.

This method is simple and may appeal when there is just a few nodes that needs to be integrated, but when the business changes and there is a need to add additional nodes the integration may become extremely costly and the result very brittle. If your application just contains 3 nodes or less and there are no plans for further expansion, then Point-to-Point is possibly a good solutions but otherwise a more complex integration pattern should be considered.

2.2 Hub-and-Spoke Integration

The Hub-and-Spoke[6, 13] method was the next step of integration patterns. In this pattern a central hub connects each node with a spoke. For each additional node only a single new connection needs to be defined. From this model much of the required components that exists today was laid out and defined: Message transformation, Message protocol standards, protocol security translation and so on. Once integration with multiple system was possible the need to share resources became great. The first method of integration with multiple systems came with the Remote Procedure Call (RPC) which was designed to function in a similar way as a local procedure call. The rise of distributed computing required a more modular system architecture and as a result required a more complex integration solution. The technologies of Remote procedure call and Distributed Object uses a method of tightly specified interfaces. As the system became more complex the and more changes to the interfaces as a result, it became very hard and costly to maintain and update those interfaces. Additional problems arose because of disparate system was not able to natively communicate with each other and resulted in many communication problems. These problems instagated in the rise of manny new communication technologies. One of those was Web Services, which is a point-to-point pattern with well defined and loosely coupled interfaces. The definition of loosely coupled is that the message can contain an undefined payload in either a text or a binary format.

The advantages of hub and spoke compared to point to point is that it has a reduced number of required connections. It also decouples the different nodes so they do not have to know about each other and makes it much easier to add and remove nodes. Because the only changes that have to be made is in the hub. As a result we have a system of independent distributed nodes that are interchangeable alto with the tight coupling that hub-and-spoke uses this is hard and time costly.

2.3 Enterprise Service Bus ESB

The Enterprise Service Bus is the newest iteration of Integration solutions. Today an Enterprise Service Bus is the most popular infrastructure for implementing Service-Oriented Architectures[9].The topology resembles that of the Hub-and-Spokes. The main difference of the ESB to the Hub-and-spoke is that ESB focus on open standards and manly on Service Oriented Architecture.

Some of the core features of an ESB is:

- Message transformation - To transform messages between different message protocols.
- Message Enrichment - the ability to enrich required missing data into a message.
- Message Routing - A message should be able to traverse the flow in number of ways according to standard patterns.
- Protocol translation - The possibility to send message on different types of transmission protocols.
- Security - To be able to translate and support different security protocols.
- Messaging- Support for asynchronous and synchronous messaging, request/response, send-and-forget, publish/subscribe.
- Transaction Management - The ability to support different types of transaction standards.
- Platform independent - The ESB should only use protocols and technologies that are platform independent.
- Audit - Monitoring status, logging, metrics, administration console.
- Legacy System - Support for legacy system protocols to allow integration with older system.
- Message Aggregation/Splitting - To be able to merge and split messages.
- Schema Validation - The possibility to verify the correctness of a message according to the message definition.
- Business rules - To support integration with a Business processing language.

2.4 Service Oriented Architecture SOA

Web services provide a standard means of interoperating between different software applications[5]. SOA is a framework for integration based on open defined standards. It focuses on services with loosely coupled well defined interfaces. As it uses platform independent standards it is a good fit for integration in heterogeneous environments. SOA exposes well defined interfaces for service and uses a stateless implementations to perform specific business tasks. The standards which SOA achieves its platform independence is Simple Object Access Protocol (SOAP) which is a message protocol defined in XML, Web Service Description Language (WSDL) which is a definition of what a SOAP message should contain and which SOAP operations exists, the Universal Description, Definition and Integration (UDDI) which contains which services a system exposes, Representational State Transfer (REST) is constraints which message and process should behave and WS-I which is guidelines to make disparate system interoperable.

2.5 Simple Object Access Protocol SOAP

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment[10]. SOAP is a XML based message protocol. It can be together with other application layer protocol but are mostly sent with HTTP. SOAP is platform independent and therefore much used for software integration. SOAP is the next generation XML-RPC but it still borrows some of its features. SOAP can be combined with WS-* standards like WS-Security which adds a layer of security to the protocol, by requiring username and password or other types of security tokens. The SOAP protocol has some advantages/disadvantages.

Advantages

- The SOAP protocol allows transportation on a various unspecific types of application and transportation layer protocols.

Disadvantages

- The protocol can be considered slow because of its XML format with its verbose nature is easy for humans to read but unnecessary large for a computer to process. Small messages can result in performance issues because of the large XML overhead that is required in SOAP. There is alternatives to SOAP where data is sent in binary form it reduces the size but makes it harder to correct errors.

2.6 Web Services Description Language WSDL

WSDL is a XML based schema languages used to define which operations a web service provides. A WSDL document defines services[8]. It provides detailed information of what the message shall contain, which data types the data shall be in, what operation shall be performed, which URL the request shall be sent to, what a successful and error message should look like and contain. WSDL works well with SOAP. The WSDL is published together with the service so that a SOAP message can be generated to whoever has access to the service. Additionally a incoming SOAP message can be validated according to the WSDL so that only messages that follows the standard will be processed. This feature can be expensive so it is usually turned off when the communication is internal, because it relies on that the connecting client follows the standard without having to verify it.

2.7 Universal Description, Discovery and Integration UDDI

UDDI is a XML standard that contains information about Web Service and its functionality[18]. it uses SOAP to communicate. It is used for identify what a service does and help a consumer find which service suits him best to perform a specific task. The register also contains technical information and links to the various WSDL.

2.8 Representational state transfer REST

REST is a standard for how a web standard like HTTP and URL should be used. To confirm to this standards is to be RESTful. The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system[14].The REST architecture consists of six constraints.

Client–Server Separating user interface with data storage. The server is not aware of the user interface it just provides the data and vice versa. This will improve the scalability allowing the clients to be interchangeable.

Stateless No client context should be stored on the server. Every request should contain all the required information. Session state should therefore be held on the client. This method will increase the scalability because of no context data will be required to shared over the system. There is also disadvantages like increased data transmission because of no data will be stored on the server. Resulting in repetitive data transmission of the user credentials.

Cache Having support for labeling message cacheable or uncacheable. Allowing the client to reuse response data later. Advantages of this method is that the response-time for request will be reduced. Disadvantages of this is that caching allows the possibility of the data becoming stale.

Uniform Interface Decoupling the interface from the service allowing the client and server application to be developed independently. The disadvantage is that the data need to be transformed into a general form before being transmitted instead of being in the form of what an specific application optimally wants it in.

Layered System Using a architecture hierarchy will result in improved system scalability, allowing load-balancing and shared caches. The client will not know if he is connected directly to the server or to a intermediary like a load balancer. This layer system style shields each layer so that each component are just aware of components it directly communicates with. This allows the component layers to be easily interchangeable.

Code-on-Demand This is an optional constraint. This allows the server to provide finished code that contains general functionality to be used in client. Allowing the server to change part of the client code. The code can be in a form of JavaScript or Java applets.

3 Research questions and research design

In this section we will present our research questions and also how we plan to find answers to these questions. We will provide the details on how we will collect published articles and how this will lead to the forming of a questionnaire to be used in further information gathering.

Research Questions We want to find as many ESB integration obstacles as possible and then provide solutions to these. Our aim with this is to produce one single document for developers who is to begin with ESB integration to help them avoid unnecessary cost and timeloss.

RQ1 What disadvantages are mentioned of ESB solutions and in what context in the literature?
RQ2 What disadvantages are experienced by the practitioners?
RQ2.1 Are there any available suggested solutions or improvements?

3.1 Methods for answering research questions

RQ1 Study of the result from the literature review.
RQ2 Interviews with participants from a leading telecom company in Europe and also developers from the project we are currently working in. The interviews will be conducted with a survey based on the result from *RQ1*.
RQ2.1 Study of the result from *RQ1* and answers given in *RQ2*. Besides this, we will gather information from the various communities connected to the different ESB solutions.

3.2 Literature review design

Information has been gathered from research databases using the following query strings:

- enterprise service bus
- esb problems
- esb issues
- enterprise service bus problems
- enterprise service bus issues
- esb advantages
- esb disadvantages
- enterprise service bus advantages
- enterprise service bus disadvantages

Searches were conducted on the library of Blekinge Institute of Technology and our preferred databases are *Engineering Village* and *IEEE*.

To limit our results, we decided to disregard articles that were published prior to 2005, as this field is in eternal progress and information gets old fast. Other criteria that needs to be met is a clear value to the purposes of this paper in the abstract, since time is of the essence we will not have time to read papers that we are not sure will give ous something.

3.3 Literature review results

The following papers were found during our literature review, and passed our selection criteria:

1. Enterprise Service Bus: A Performance Evaluation[15]. Published by Sanjay P. Ahuja, Amit Patel in 2011. This paper seemed useful because it contained a performance analysis between different types of ESBs.
2. SOAs & ESBs[12]. Published by J. Paisley in 2005. This article brings up some of the difficulties that can be encountered when implementing an ESB.
3. Getting on Board the Enterprise Service[11]. Published by S. Ortiz in 2007. This article brings up some of the challenges that can be encountered when implementing an ESB.
4. A high performance enterprise service bus platform for complex event processing[7]. Published by Deng Bo, Ding Kun and Zhang Xiaoyi in 2008. This paper brings up some of the challenges that can be encountered when implementing a complex ESB solution.
5. Service-Oriented Performance Modeling the MULE Enterprise Service Bus (ESB) Loan Broker Application[4]. Published by P. Brebner in 2009. This paper handles both scaling and performance in ESBs.
6. An integration strategy for large enterprises[6]. Published by Dejan Rismic in 2006. It provides information on important aspects of ESB. This is needed to be able to understand and improve the ESB.
7. Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus[17]. Published by Yan Liu, Ian Gorton and Liming Zhu in 2007. This paper handles important aspects of ESB performance. Needed to be able to find performance issues.

After we settled on the these specific documents, we began reading through them from top to bottom, noting everything that can be seen as an issue or flaw in the ESB architecture. Some of the included documents is not part of the literature to provide a source of issues, but instead serves as a source to be able to get better understanding of the ESB. Besides these, all documents have one common denominator: They handle aspects of the ESB that can be seen as issues or in need of improvement. The literature study gave us the following list with issues/flaws in ESB solutions:

1. It is hard to decide where all functionality goes. It has been mentioned in the literature that it is not always clear if functionality should be placed in the ESB or in one of the nodes in the system in the beginning of integration. A format transformation for example, could sometimes as easily be implemented in a node as in the ESB. [11].
2. There is a risk of slow performance, as the ESB prohibits direct communication between nodes, providing an additional step between nodes. According to the literature, the extra communication layer may present additional response time[15].

3. Migration of systems integrated with older architectures can be hard to do as older systems can have communication that goes everywhere and no clear way of sorting the flows.[11].
4. Support is sometimes limited and documentation and community answers is not always available without a lot of effort, when reading the literature we found that there could be some trouble to obtain relevant information and directions [16].

We thought that we would find more issues in this field and there is a slight possibility that we have missed articles that could provide more information. But we feel confident that the search strings we used, in the databases we searched, provided us with articles that together covers the issues that are involved with integration with an ESB architecture.

3.4 Interview survey design

The survey is created from the result of the literature review and is designed to accomplish two things.

1. Confirm that the known issues from the literature review is also present in the projects we are involved in. If this is not the case, show how the particular issue has been handled.
2. See if there are other issues that we have not found in previous research papers. We will also try to get the participants to try to think of possible solutions.

We chose to focus on issues and solutions to these. This is because a lot of the existing papers already have covered the positive parts of ESB integration and another list of good things would not give any real contribution. Our goal is to have a range between six to eight participants in this survey with varying experience in ESB Integration. We initially try to get as many professional ESB Integrators as possible then if they are not enough we will ask participants from our software project who has experience in ESB integration. The reason we decided on the number of participants was because of the limited time we had we thought it reasonable with that range of people. The challenge was to find people we the experience who also had time to spare to answer our questionnaire.

The survey will contain the following questions:

- Do you have any experience with software integrating using an ESB?
  - If so, what?
- Is it hard to decide which part of the system functionality really belongs in the ESB?
  - If so, please elaborate
  - If not, how do you decide?
- Do you experience an issue with slower communication between nodes because of the extra steps that comes with an ESB?
  - If so, please elaborate

    - If not, is it because the steps do not take any time or because the system does not require response?
- Have you ever migrated older, integrated systems to the ESB architecture?
    - If so, please describe the main issues encountered.
    - How did you do to resolve said issues?
- Are you able to get support or find information to resolve the issues that are encountered during integration?
    - If so, how do you find/get said help?
    - If not, how do you handle issues?
- Have you encountered any issues, apart from the ones mentioned in this questionnaire, when working with ESB integration?
- Have you been able to resolve any of these issues?
    - If so, please describe your solutions.

These questions was chosen to map the results in the literature review results.

4 Data collecting results

In this section we will present the results we got from the sampling that we performed.

4.1 Survey setup

We were involved in a student project together with 10 other students that worked together with a large, European telecom company in Karlskrona. The goal of this project was to integrate several of the companies existing products to create a new system. We were using an ESB solution from Mulesoft[1]. The experiences of the participants in this project was worth a lot. Partially because everyone had recently been involved in starting integration with an ESB and had the problems fresh in their memory ant partially because we, thanks to the companies position in the software development area, were able to get in personal contact with Mulesoft for guidance on how to progress with our development. During this guidance, we also got confirmation directly from Mulsesoft that there is known problems with the available documentation.

Our survey was filled out by seven individuals, all in some way connected to our current project. We chose participants based on their previous knowledge of system integration. Both with ESB and other techniques. Four of the participants are in our group and are currently working actively with integration using an ESB from Mulesoft, their value is explained in more detail in section 2.9. The other three are employees at the telecom company, which we have got in contact with while performing our project at the telecom company, the value of their answers is, as mentioned before, a lot of integration experience, both with ESB and other architectures, providing specific and generic knowledge. The surveys was conducted by handing the participant a printed survey with our previously stated questions in section 3.4. After this, the person had all the time he or

she needed to fill in their answers. All the participants had one of us present at all time in case something was unclear, badly formulated or in case they just wanted to say something that was not covered by the survey. The survey where done anonymously by all participants. Their roles and experience where recorded during the interviews. The participants roles and experience can be found in Appendix A.

4.2 Collected data

The collected answers from the survey can be found in appendix A.

4.3 Data analysis

To map our results to our research questions, we begin by rereading the issues we found in the literature review:

1. It is hard to decide where all functionality goes. Some things could be placed in the ESB but might as well be in one of the nodes in the system.
2. There is a risk of slow performance, as the ESB prohibits direct communication between nodes.
3. Migration of systems integrated with older architectures can be hard to do.
4. Support is sometimes limited and documentation and community answers is not always available without a lot of effort.

If we compare these initial issues one by one with the answers we got, we will see how well it maps with the development process, this may also provide a solution to the initial issues. This will later lead the way to an answer on research question 2 and 2.1

No.	Question	Yes	No
Q1	Do you have any experience with integration using an ESB?	7 (100%)	0 (0%)

Q1 The first question was an open ended question where the interviewee would describe their experience developing integration solutions using an ESB. To get a feel on how trustworthy and reliable their answers would be.

No.	Question	Yes	No
Q2	Is it hard to decide which part of the system functionality really belongs in the ESB?	4 (50%)	4 (50%)

Q2 On this question, we get a 50% distribution of agreement to this issue. This is due to one participant answering "yes and no". As it is not a unanimous agreement that the initial issue is in fact an obstacle, this provides possible solutions to the issue in question.

No.	Question	Yes	No
Q3	Do you experience an issue with slower communication between nodes because of the extra steps that comes with an ESB?	2 (29%)	5 (71%)

Q3 When trying to confirm this issue, we got 71.4% disagreement from the participants in the survey. As this is quite a large percentage it indicates that this is not a major issue. But still it's not 100%, so there may still be ways to resolve it.

No.	Question	Yes	No
Q4	Have you ever migrated older, integrated systems to the ESB architecture?	0 (%)	7 (100%)

Q4 Unfortunately we couldn't find anyone who had dealt with migration of older integrated systems to the ESB architecture. This prevents us from obtaining solutions to issues connected to this issue.

No.	Question	Yes	No
Q5	Are you able to get support or find information to resolve the issues that are encountered during integration?	2 (14%)	6 (86%)

Q5 Again, the 8 answers is due to one of the participants both agreeing and disagreeing with the statement. However, this is clearly an issue in the industry. 86% of the participants in our survey claim to have problem in finding the information and support they need. The remaining percentile said when asked, that he had not been that involved with integration that he had to find any information that wasn't already available within the group.

These results provides us with possible answers to both research question 2 and 2.1 on all points except the issue with migration of previously integrated systems.

Since research question 1 gets its answers through the literature survey, this is a satisfiable result.

5 Data synthesis and answer to research questions

We found possible solutions to the majority of the known issues we found, these solutions are not guaranteed to work for everyone and in any situation but it still may provide assistance for developers that are stuck in the process of either deciding on integration architecture or are in active development. From the survey we also found some additional issues concerning ESB integration that was not found in the available literature, some of these issues came with a solution and some came with hints on how to find a solution.

5.1 Issues

The provided solutions have not been put to any tests. They have all been provided from the performed sampling. A sampling that was answered by people that have experience with ESB integration. Both from a rookie perspective and the views of experienced developers. None of the solutions is guaranteed to work for everyone, but have been proved to work for active developers in a real project. Hence it should be possible to incorporate these solutions in other projects.

Table 1: This table contains issues found in the available documentation and various solutions found in the interview survey and the documentation.

No.	**Issue**	**Solution**	**Issue found in**	**Solution found in**
1.1	It is hard to decide where all functionality goes. Some things could be placed in the ESB but might as well be in one of the nodes in the system.	In real world implementation, it is important to set rules of communication. As can be seen in the answers to question 2.2 in appendix a, a firm rule of how communication needs to be applied when beginning ESB implementation.	Literature	Sampling
1.2	There is a risk of slow performance, as the ESB prohibits direct communication between nodes.	This should not be an issue as the ESB only performs actions that would need to be done by the individual nodes if the ESB was not there to sort it out. As long as the ESB only does what it needs to do, all nodes should get what it needs, when it needs it. Any bottlenecks is most likely to be found in connection or slow servers.	Literature	Sampling
1.3	Migration of systems integrated with older architectures can be hard to do.	This issue can not be either confirmed or dismissed. as we could not find any participant that had enough experience in integration to give any directions on the matter.	Literature	Not found
1.4	Support is sometimes limited and documentation and community answers is not always available without a lot of effort.	This is absolutely confirmed and a solution is not easy to give. The survey-takers that are involved in our project are quite privileged. Since they are involved in a project tied to a leading telecom company (a company that have a huge influence on the industry, they got an easy access to first class assistance from the developers of the Mule ESB. Otherwise it is confirmed that support and community response is hard to get by and since our survey-takers have been privileged enough, we can confirm that support is an area where vast improvements are needed.	Literature	Sampling

Table 2: This table contains issues and solutions found out from the interview survey.

No.	Issue	Solution	Issue found in	Solution found in
2.1	If the team developing the ESB falls behind, this affects all teams that utilize the ESB.	No solution is provided. However, we believe that this issue can be handled by the fact that you know that it exists. If a project is to be started and the project manager knows about the issues that may occur, education can be started at an early stage, preventing slow progress on implementation	Sampling	Sampling
2.2	Big changes between different versions of the ESB makes it hard when documentation isn't up to date.	This, as far as we have found out, had been handled by trial and error. Unfortunately this is the best answer to this issue at the moment. A more stable solution is to choose a version of an ESB and stick to it, ignoring possible improvements in order to keep the functionality already implemented.	Sampling	Sampling
2.3	There is no stated standard, or best practice, when it comes to ESB implementation.	This is still an issue and will keep on being an issue. Until the different providers of ESB solutions sit down and decide on a common standard for a minimum of functionality, there is no way of deciding on what needs to be there to be able to say that you've got an ESB integrated system.	Sampling	Not found
2.4	Problems when different nodes use different sessions. The ESB is usually stateless, meaning that it has no session id, this can be a problem if some nodes need their own sessions.	This is an issue that can quite easily be handled. Several different session ids can be stored in the ESB in order to keep connection with all the nodes. Another solution is to make the ESB perform a login and logout on each node, every time a request is made.	Sampling	Sampling
2.5	A lot of the ESB coding is done in XML which is not a proper programming language and this makes it hard to get a good overview of the code.	There are some XML editors with a GUI editor. These are not free from bugs, but combined with other tools they can provide an easier overview in the development process.	Sampling	Sampling
2.6	To get proper documentation, a enterprise license might be needed. This is expensive and might raise the projects budget.	This can be addressed in two ways. 1. Hope for the best. Meaning that the development relies on community and available documentation, with no additional economical cost for the project. 2. Purchase of enterprise license. Meaning a economical cost, but guarantees more documentation and support in the process. The first alternative would be advised for teams with experience of the platform and vice versa.	Sampling	Sampling

RQ1 The following issues can occur when implementing an ESB architecture solution in a software project (according to available documentation):

- **ISSUE1.1** It is hard to decide where all functionality goes. Some things could be placed in the ESB but might as well be in one of the nodes in the system.
- **ISSUE1.2** There is a risk of slow performance, as the ESB prohibits direct communication between nodes.
- **ISSUE1.3** Migration of systems integrated with older architectures can be hard to do.
- **ISSUE1.4** Support is sometimes limited and documentation and community answers is not always available without a lot of effort.

RQ2 The following issues were presented during the survey in addition to the issues found in the literature review:

- **ISSUE2.1** If the team developing the ESB falls behind, this affects all teams that utilize the ESB.
- **ISSUE2.2** Big changes between different versions of the ESB makes it hard when documentation isn't up to date.
- **ISSUE2.3** There is no stated standard, or best practice, when it comes to ESB implementation.
- **ISSUE2.4** Problems when different nodes use different sessions. The ESB is usually stateless, meaning that it has no session id, this can be a problem if some nodes need their own sessions.
- **ISSUE2.5** A lot of the ESB coding is done in XML which is not a proper programming language and this makes it hard to get a good overview of the code.
- **ISSUE2.6** To get proper documentation, a enterprise license might be needed. This is expensive and might raise the projects budget.

5.2 Solutions

RQ2.1 The provided, possible solutions to the encountered issues, these solutions have all come from the results in the sampling. None of the results have been tested, other than in the everyday development of the software integration project in which all the participants have been involved:

- **ISSUE1.1** In real world implementation, it is important to set rules of communication. As can be seen in the answers to question 2.2 in appendix a, a firm rule of how communication needs to be applied when beginning ESB implementation.
- **ISSUE1.2** This should not be an issue as the ESB only performs actions that would need to be done by the individual nodes if the ESB was not there to sort it out. As long as the ESB only does what it needs to do, all nodes should get what it needs, when it needs it. Any bottlenecks is most likely to be found in connection or slow servers.
- **ISSUE1.3** This issue can not be either confirmed or dismissed. as we could not find any participant that had enough experience in integration to give any directions on the matter.

ISSUE1.4 This is absolutely confirmed and a solution is not easy to give. The survey-takers that are involved in our project are quite privileged. Since they are involved in a project tied to a leading telecom company (a company that have a huge influence on the industry, they got an easy access to first class assistance from the developers of the Mule ESB. Otherwise it is confirmed that support and community response is hard to get by and since our survey-takers have been privileged enough, we can confirm that support is an area where vast improvements are needed.

ISSUE2.1 No solution is provided. However, we believe that this issue can be handled by the fact that you know that it exists. If a project is to be started and the project manager knows about the issues that may occur, education can be started at an early stage, preventing slow progress on implementation

ISSUE2.2 This, as far as we have found out, had been handled by trial and error. Unfortunately this is the best answer to this issue at the moment. A more stable solution is to choose a version of an ESB and stick to it, ignoring possible improvements in order to keep the functionality already implemented.

ISSUE2.3 This is still an issue and will keep in being an issue. Until the different providers of ESB solutions sit down and decide on a common standard for a minimum of functionality, there is no way of deciding on what needs to be there to be able to say that you've got an ESB integrated system.

ISSUE2.4 This is an issue that can quite easily be handled. Several different session ids can be stored in the ESB in order to keep connection with all the nodes. Another solution is to make the ESB perform a login and logout on each node, every time a request is made.

ISSUE2.5 There are some XML editors with a GUI editor. These are not free from bugs, but combined with other tools they can provide an easier overview in the development process.

ISSUE2.6 This can be addressed in two ways. 1. Hope for the best. Meaning that the development relies on community and available documentation, with no additional economical cost for the project. 2. Purchase of enterprise license. Meaning a economical cost, but guarantees more documentation and support in the process. The first alternative would be advised for teams with experience of the platform and vice versa.

6 Thesis conclusions and contribution

So the goal with this report was to find beginner problems with ESB integration and solve this issues. Have we succeeded? We have found important issues with ESB integration that was indeed encountered in the industry. The solutions that we got, was gathered from a survey that in turn was based on literature survey. The provided solutions have not been tested within the scope of the thesis, but they come from both experienced developers and developers that have 6 months

or less experience of ESB integration. But the last group have the advantage of direct guidance from MuleSoft. The issues and solutions can be found in the table in section 5.1. The issues that were not provided with a solution, at least comes with directions as how to handle. Providing this document should be enough to let a project planner make an informed decision on how to tackle ESB implementation. It provides information on what to consider when thinking about licenses and also information on how to handle flows within the system.

So are our methods comprehensive enough to give the result legitimacy? Compared to previously published work that has not given much focus on how to actually handle issues, this thesis is more focused on helping new practitioners to avoid the most common problems. Given the answers from our sampling, it is confirmed that the issues we found are also the issues that is encountered by practitioners. The number of performed interviews may be a little low, but we believe it still gave us the answers we needed. Since some of the participants have worked 6 months or less with ESB integration, we get the views of persons that has very recently been through all the problems with adapting to a new way of working. Furthermore, since these participants have been in direct contact with Mulesoft, they have been able to provide information that most developers will not be able to obtain. And with the more experienced integrationers from a leading European telecom company we got some insight from people who has more experience with various kinds of integration and because of this is able to provide more comparative information.

Previously available information (see references) fails to give a concluded image on ESB integration and issue handling, with this thesis all the necessary pieces should be incorporated so that an informed plan can be established.

If we would be to start over on this thesis, the main thing that we would change is that we would try to get a larger number of experienced practitioners as this would have increased the chance of getting tips on dealing with migration of older integrated systems. We would also have tried to perform some test of our own, to see if we could find problems with response times and such.

References

1. Mule ESB. http://mulesoft.org/.
2. ServiceMix. http://servicemix.apache.org/.
3. WSO2. http://wso2.com/products/enterprise-service-bus.
4. Brebner, P. Service-Oriented Performance Modeling the MULE Enterprise Service Bus (ESB) Loan Broker Application. *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference*, 2009.
5. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard. Web Services Architecture. http://www.w3.org/TR/ws-arch/, 11 February 2004.
6. Dejan Risimic. AN INTEGRATION STRATEGY FOR LARGE ENTERPRISES. *Yugoslav Journal of Operations Research*, 2006.
7. Deng Bo, Ding Kun, Zhang Xiaoyi. A high performance enterprise service bus platform for complex event processing. *Grid and Cooperative Computing, 2008. GCC '08. Seventh International Conference*, 2008.
8. Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl/, 15 March 2001.
9. Falko Menge. Enterprise Service Bus. 2007.
10. Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, Yves Lafon. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12-part1/, 27 April 2007.
11. Ortiz, S. Getting on Board the Enterprise Service Bus. *IEEE Computer Society Journa Volume 40 Issue 4, April 2007l*, 2007.
12. Paisley, J. SOAs & ESBs. *Dr. Dobb Journal Volume 30, Issue 2, No. 369*, February 2005.
13. Rodney Gleghorn. Enterprise ApplicationIntegration:A Manager's Perspective. *IT Professional*, 2005.
14. Roy Thomas Fielding. Representational state transfer (rest). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 2000.
15. Sanjay P. Ahuja and Amit Patelo. Enterprise Service Bus: A Performance Evaluation. *Communications and Network, Vol. 3 No. 3, pp. 133-140*, 2011.
16. Sturek, C. Open source SOA requires expertise. 2008.
17. Yan Liu Nat. and Sydney Gorton, I. and Liming Zhu. Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus. *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, 2007.
18. Yang Bo, Hu Guyu , Jiang Jinsong. A Scalable and Dynamic Service Oriented Workflow Model. *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference*, 2011.

Appendixes

ID		Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6	Participant 7
1	**Do you have any experience with integration using a ESB?**	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1.1	If so, what?	Several months of integration using Mule ESB	Some experience with Mule ESB (worked with it for approxametly 6 weeks)	Some experience of flows and also deploying the ESB on server	I have participated in integration of 2 nodes with Mule ESB	I have connected multiple nodes to the same ESB so that they don't need to know more about each other than that the ESB exists and takes care of everything. Transformation and connections if my main area of expertice.	I have some experience with software integration using Open ESB. I have been part of a team developing the integration logic for a project where many separate nodes had to communicate with each other.	I have been apart of project where a integration bus was used and i did some of the flows in the ESB. The reason we used a ESB was that we had in our software suit multiple systems that had to communicate with each other. The best way to achieve the communication in a structured manner was by using an ESB.
2	**Is it hard to decide which part of the system functionality really belongs in the ESB?**	Yes	Not in our project	Yes	No	Yes and no	No	Yes
2.1	If so, please elaborate.	There have been disscusions where functionality have been added to a node which are more suited for to a operation, instead of using existing functionality and enriching and/or removing uneccesary data in the ESB.	-	It's hard to get knowledge about it, it messy.	-	Basic functionality such as transformation and mediation seems like a "must have" to be able to call it a ESB, at the same time a twitter module, for instance, feels like a highly disposable function that indeed kan be nice to have, but nothing that should be placed within the scope of a ESB. I feel that, as of now, there is no clear standard that states what a ESB should be able to do		There have been some discussions whether to use existing functionality and aggregate and enrich the message or instead make more tailor suited functionality for each operations. We decided to use both methods.

2.2	If not, how do you decide?	-	In our project, we haven't allowed any direct communication between isolated nodes. All inter-node communication goes through the ESB	-	ESB is in charge of "traffic control" between nodes. System specific calculations is to be done in the nodes. So the decision of what is to be done in the ESB is "easy"	-	The API in each nodes where extensive so not much new functionality had to be added. Because of the functionality that was needed were already implemented.	-
3	**Do you experience an issue with slower communication between nodes because of the extra steps that comes with a ESB?**	I have not performed any benchmark tests	No	No	No	No	Yes	Yes
3.1	If so, please elaborate.	There is a lot of XSLT in our ESB which may degrade the performance	-	-	-		Most of our nodes are written in java so they are natively able to communicate with each other, so much of the protocol translation that are being done in the ESB could also as easily have been done in the separate nodes. The translation overhead will probably slow down the performance of the system.	We have allot of costly transformation in our ESB for example xml to object, object to xml and various xml parsing.
3.2	If not, is it because the extra steps do not take any time or because the system does not require response?	-	I haven't seen any actual performance comparisons, but Mule ESB appears to be a fairly fast ESB implementation. Responses are required in most of the flows.	You hardly notice any difference in performance	I have only been using simple and fast function calls through the ESB. These calls have not been enough to cause any time delay.	This depends on what is done in the ESB but in general it should not be an issue as the ESB only performs tasks that otherwise would have to be done by other nodes in the system.	-	-
4	**Have you ever migrated older, integrated systems to the ESB architecture?**	No	No	No	No	No	No	No
4.1	If so, please describe the main issues encountered.	-	-	-	-	-	-	-

4.2	What did you do to resolve said issues?	-	-	-	-	-	-	-
5	**Are you able to get support or find information to resolve the issues that are encountered during integration?**	We have had problems getting support and documentation	This can be difficult...	No	Yes	Yes and no	The documentation is a bit lacking and can be confusing.	There has been some problems getting information out of the documentation. We have had problems getting documentation for the latest version of the ESB the older versions are not compatible with the current version. Also much of the examples that exists does not work or are incomplete.
5.1	If so, how do you find/get said help?	We have mainly been reusing something we have gotten to work previously, not knowing if it is the correct method.	-	-	From forums and some help from experts. As work progresses, more and more knowledge is achieved by "trial and error"	If you get an enterprise license (or if your opinion of the product is really important), you might get in contact with the creators of the ESB. This expertis is superior to any available documentation, but not easy to obtain.	I usually just try till i have found the solution to a problem and if the same problem come up again i have documented how to solve it.	-
5.2	If not, how do you handle issues?	-	Especially when working with Mule ESB, because the documentation is severely lacking. To resolve these issues, we've had some contact with a company called Entiros and the developers of Mule, Mulesoft	Try to find information in the documentation or find somebody who can help me	-	Documentation on ESBs in general is scarce at best. If you don't fall into the the category mentioned above and hit an obstacle you either find another way to do it or create your own quick and dirty solution.	-	We had to guess much and if something worked we reused that.

6	**Have you encountered any issues, apart from the ones mentioned in this questionaire, when working with ESB integration?**	If the team developing the ESB is slow, other teams may be held up	-	No	Big changes between different versions of the ESB makes it hard when documentation isn't up to date	It can be hard to get a grip on how to get the most of the ESB (lack of "best practises"). Another confusing thing is when the ESB provides multiple environments. As with Mule for example, they have Mule standalone, MuleStudio and an alternative to build with maven. There are several functions that work with one or two of these version but not with the third. This creates even more confusion as documentation and community doesn't always clearly state which version is handled.	I have encountered problems when nodes that the ESB communicates with are not stateless meaning that they have a session id. ESBs are usually stateless so that they will easily scale over multiple servers without having to share sessions between them. The problem arise when a node require a session id to perform a operation. Do the ESB store sessions (reducing scalability), returning the session id to the calling node (reducing transparency) or performing a login and logout for each request (reducing performance by increasing communication overhead).	Much of the programming is done in XML. XML is not designed for use as a programming language and because of that it becomes hard the get an overview of the code. The ESB we used had an enterprise license cost so our budget became higher.
6.1	Have you been able to resolve any of these issues?	-		-	Yes	No, not really, as this is a issue for the creators of the ESBs	Yes	Yes

6.2	If so, please describe your solution.	-	-	-	Trial and error	The ESBs gets better and better all the time, along with that the creators need to step up in order to keep users. If documentation is not up to date, developers will either chose another alternative or flood support and forums.	We performed a login/logout for each request.	There are GUI editors to aid in creating this kind of XML code. But they are not bug free and does not display all options that are possible to do in the XML. So a combination of this methods were used.