Master Thesis
Software Engineering
Thesis no: MSE-2004-32
October 2004

Increasing availability in existing software systems

- An assessment of three-tier replication

David Granflo

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
David Granflo
Drottningatan 61
371 33 Karlskrona

External advisors:
Anders Larsson
Wireless Independent Provider AB
Address: Campus Gräsvik 5, Karlskrona, Sweden
Phone: +46 455 33 98 11

Per Werélius
Wireless Independent Provider AB
Address: Campus Gräsvik 5, Karlskrona, Sweden
Phone: +46 455 33 98 03

University advisor:
Dr. Michael Mattsson
TEK

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.tek.bth.se
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

Architectural design decisions are known to be crucial for the success of a system in development. An early design decision will most likely be expensive to change at a later stage of the development if the software engineers needs to get back to the drawing board. If a quality attribute has been neglected during architectural design it may ruin the entire project.

This work describes a way of increasing the availability of a software service, that have already been put into use, by adding a middle layer of replication logics represented by third party application servers and replicated Enterprise JavaBeans. The approach called Three-tier replication is assessed and compared to the origin architectural design by using a qualitative scenario based assessment. In addition, we have also implemented the new architectural design to be able to validate the assessment results. The validation is done by testing the scenarios on the two architectures.

The software service used is an industrial system for sending text messages from computers to mobile phones.

Keywords: availability, software architecture assessment, cluster, Enterprise JavaBeans.

CONTENTS

1 INTRODUCTION

1.1 Background

Software availability has become more and more recognized as a quality issue since business transactions have become computerised and human life has become, in some situations, dependent on working software systems. Companies or organizations that provide computerised services, both externally and internally, realise the importance of reducing the service outage[1]. Business critical systems that are not available for a period of time will lead to a loss of income. In some cases, human lives are dependent on software systems, which makes the availability of the system extremely important.

One way of increasing the service availability is to develop a clustered software solution that is located on two or several computers. By replicating the service on several computers, the service have a higher possibility to keep the service available. The cluster concept, which is further described in chapter 2, is nothing new to the academic world, nor to the industrial world. There are several reasons to why clusters are interesting to the industrial world and the availability aspect is one of them. [Pfister, 1998]

Creating clustered systems is considered to be a complex and time consuming activity. But according to Pfister [Pfister, 1998] it is still worth the bother of creating a clustered solution if the cluster concept fits the quality requirements. Even though it is worth the bother to develop a cluster it is still a costly process which small and low budget companies may not afford. Therefore it is interesting to investigate if and how, cheap or even free Commercial of the Shelf (COTS[2]) products can make the development of clusters, in purpose to increase the availability, less time consuming.

Enterprise Javabeans (EJB) [EJB, 2004] is a fairly new technique which requires some sort of application server. One of the interesting things about these application servers are their clustering features. All of them does not support clustering of EJBs, since it is not a part of the EJB specification, but many of them do and some of them are free.

The goal of this work is to investigate if and how the availability for existing software systems can be increased by adding a middle layer of replication logics between the client and the service server node. The replication logic is provided by an application server and EJBs.

1.2 Problem and Hypothesis

System architecture is the first design decision that describes the system's structure with it's components and the relationships between them. As a first design decision it has a big impact on the system's different quality attributes. It is generally very hard to make changes to the software architecture at a later stage of the development [Bengtsson, 2002]. I.e. if the availability is not prioritized during the architectural design, it is considered hard to increase the availability at a later stage.

Companies or organizations may not have realised the importance of availability until the system is in use and the first occasion of service outage. At that time it is usually not desirable to start all over with the architectural design and possibly change or update the entire system implementation. The work presented in this thesis investigates if there is a way of increasing the availability without changing the existing system.

[1] Service outage is a period of time when the software system is unavailable to it's users.

[2] COTS is also known as Complete of the Shelf since there are open source projects that provide products for free and are not commercial.

The investigation is focusing on one proposed solution of increasing the availability without changing the origin system code implementation. The approach that is used is called Three-tier replication [Baldoni et al, 2002] where a layer between the client and the service nodes provide the system with replication logics.

1.3 Methodology

The investigation of how to increase the availability of an existing software service was initiated by a company named Wireless Independent Provider AB (WIP) [WIP, 2004]. WIP develops software systems and services for making the communication between personal computers and mobile devices, such as cell phones and PDAs [3], easier. A service for sending text messages (SMS[4]) from computers to cell phones, and the other way around, was chosen since WIP discovered a need to put higher availability requirements on that service.

The work described is conducted as an action and evaluation research project. By assessing the origin architecture and a three-tier replication architecture we can tell if the latter is better suited for handling the new availability requirements or not.

Today the system's implementation does not support clustering. In other words: it can not be replicated and still keeping transparency towards the client. Three-tier replication approach [Baldoni et al, 2002] is used to transform the origin system into a clustered system. To create the replication logics between the client and the service nodes EJB is used in combination with an open source EJB server called JBoss [JBoss, 2004]. There are other EJB servers that provide a clustering feature but most of them are expensive and JBoss have a good reputation among software developers. One of the requirements from WIP was that it should not cost anything except from the development cost. I.e. no expensive software licenses or hardware could be bought.

The assessment is based on scenarios that intend to illustrate events that may interrupt the service in a manner that makes it unavailable. Since the clustered architecture is implemented the scenarios may, apart form being assessed theoretically, also be validated through testing.

1.4 Related work

The idea of using the Three-tier replication approach came from a research report written by Roberto Baldini, Carlo Marchetti and Alessandro Termini who investigated active software replication through a Three-tier Approach [Baldini et al, 2002]. Our approach differs from Baldini's work in the technique that is used. Baldini use a CORBA [CORBA, 2004] compliant infrastructure and we use EJB. However, the principle of letting a middle layer of replication logics between the clients and the server replicas are the same.

[Rushikesh et. al], 1999 have developed a model of creating replicated applications, called ShadowObjects. It resembles the way the EJB servers replicates the enterprise java beans in a way. Although their paper does not discuss the matter of replicating an end-tier, we do not see why it should not work.

[3] PDA is short for Personal Digital Assistant

[4] Short message service. Text messages from one cell phone to another. [GSM, 1996]

1.5 Outline

To give the reader an insight of the subject a couple of the major concepts are introduced in the next chapter (Chapter 2). Availability as a quality attribute, clusters and clustering EJBs with the application server JBoss [JBOSS, 2004] is presented.

In chapter 3, the origin architecture of the industrial system and the new clustered architecture are presented.

Chapter 4 describes the assessment method used and the execution of the assessment. After assessing the architecture a validation of the assessment is performed by running the scenarios on the two systems. The results from the validation are also present in this chapter.

The last chapter (Chapter 5) sums up the outcome of this work and discusses the conclusions that can be drawn and the flaws that are needed to be aware of to use the results of the assessment. Suggestions for future work within this subject are also presented.

1.6 Acknowledgements

We thank the advisors and the technicians at WIP for their helpful manner and their friendly every-day support.

We are also most thankful for the academic insight and guidance Dr. Michael Mattsson at Blekinge Institute of Technology has given us which have made this work possible.

2 CONCEPTS

2.1 Availability

There is actually no common agreement of what constitutes high availability [Pfister, 1998]. For some systems it might be acceptable with no more than a couple of seconds per year of non-operation time and for others it might be acceptable with a down time of several hours per year. It all depends on the context and purpose of the software system. For example developers of a real-time system for a space shuttle will probably have a less tolerant attitude towards system outage than a buyer of an office server system. Both of them will however most probably require a *High Available* system.

Attempts have been done to create a scale of the availability. One of them is Classes of Availability [Pfister, 1998] (also known as *number of nines'*) and gives a percentage precision of the availability (see Table 1). It does not say which class that is considered high, but the vendors may use it to give their buyers an idea of the expected system outage.

Availability	Total Outage per Year	Availability class
90%	**outage** > 30 days	1
99%	30 days > **outage** > 9 h	2
99,9%	9 h > **outage** > 1 h	3
99.99%	1 h > **outage** > 5 min	4
99.999%	5 min > **outage** > 30 sec	5
99,9999%	30 sec > **outage** > 3 sec	6

Table 1 Classes of Availability

When developing systems with the availability in mind, it is important to eliminate, or at least reduce, the *single points of failure* (SPOF). A SPOF is a piece of hardware or software that will bring down the entier system if it fails or breaks. As long as any piece of hardware or software can be replaced or repaired before the replicated piece breaks, the system is considered as highly available. [Pfister, 1998]

Wolf [Wolf, 2004] claims that a high available system must not have any SPOF. A SPOF can be anything from a small network switch to an entire data center as long as the system is dependent on that point. There are a lot of non-software issues that may result in a SPOF. For example power supplier switches.

If all SPOFs are eliminated, one single failure will not affect the users, but to do so will take an enormous effort from the architect and the people working with the system. The question is if it is cost-beneficial to go as deep as Wolf describes. Again it comes back to the availability requirements put on the system.

The causes for a system to become unavailable are numerous and many aspects needs to be considered when designing systems where high availability is requested. Power failure, hardware failure and software failure is a couple of the reasons that may cause outage of a software services.

Reasons may not only be unplanned. Examples of planned outage are software or hardware upgrades, different types of maintenance etc. According to Pfister [Pfister, 1998] planned outage stands for a much larger piece of the total outage than the unplanned. This is another problem when calculating a system's availability. Should the planned outage be included in an availability classification (i.e. Classes of Availability) or not? It is not always equally important [Pfister, 1998]. For example, if a service is only supposed to be available for a period of time per day (cf. stock-exchange systems), maintenance could be handled during the planned downtime. *Continues operations* is the term used, within the academic world, to describe the

situation when only planned outage are considered [Pfister, 1998]. When both planned and unplanned outage is considered, it is called *Continuous Availability*. In this work we take both planned and unplanned outage into account but from this point we only use the term *Availability*.

2.2 Clusters

A cluster is as a distributed or parallel system, consisting of two or more interconnected computers, which from the outside, is seen as a single unified computing resource [Pfister, 1998].

There are two different reasons for clustering a software system: increase performance and increase availability. Our work will focus completely on the availability aspect of clustering. The basic idea is to replicate the software on different physical locations to be able to keep the software accessible to the user even tough a computer of the cluster breaks or is shut down in maintenance purposes. In other words, decreasing the single points of failure.

As described in the definition above, a cluster is a distributed system. But to be able to tell the difference from a traditional distributed system we need to focus on the latter part of the definition – 'single unified computing resource'. The nodes in a traditional distributed system have their own internal identity and usually have different goals of their existence. Nodes in a cluster are, as defined, seen as a single unified computing resource without an individual identity and from the outside is seen as one system. Pfister [Pfister, 1998] does not separate the two concepts - distributed systems and clusters - but emphasises that clusters are a subspecies of distributed systems. A cluster may even be a node in a distributed system.

A cluster is seen as one computing resource, which means that the user or client code does not know which node it is communicating with or which node is performing the service. It should look as a singleton server from the clients point of view. To achieve this the result from each service node shall be the same for a certain request no matter which node that is processing the request [Baldoni et al. 2002].

2.2.1 JBoss clustering

A JBoss cluster consists of several instances of JBoss application servers. They can either be located on the same computer or spread over a local area network. The instances are members of what is called a partition. There are possibilities of having more than one cluster in a network which makes the partition deviation necessary for the different instances to know which cluster they are members of. The nodes in the partition keeps the consistency automatically and JBoss clusters support both stateless and statefull sessions. [Labourey et al, 2003]

The nodes in a cluster are broadcasting notifications of their existence over the network. This makes the addition and removal of cluster nodes very dynamic. The JBoss group calls this Automatic Node Discovery.

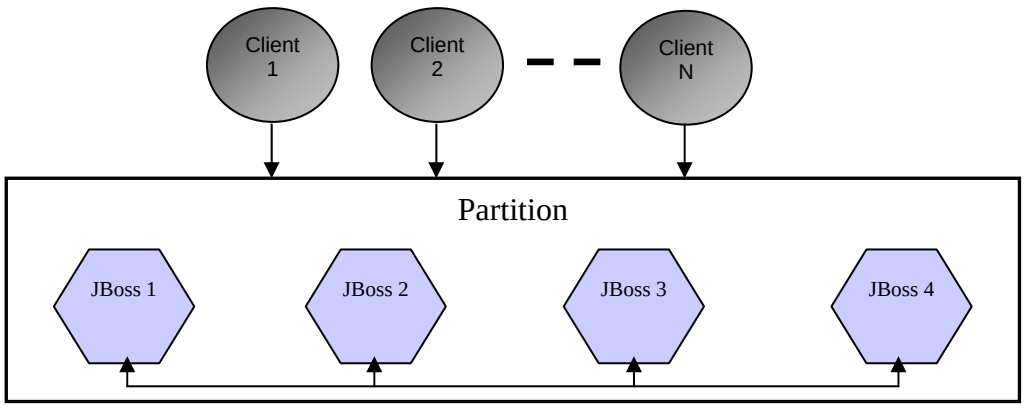

Figure 1, JBoss clustering [Labourey et al, 2003]

JBoss are using a technique called Smart proxies to keep the transparency without introducing a SPOF into the system. Since the communication between the clients and the nodes in a partition uses Remote Method Invocation (RMI) [RMI, 2004] there are possibilities for the client to download stubs containing the addresses to the different nodes. I.e. a software dispatcher is placed, during runtime, on the client side. The client code is unaware of this and for each response the client gets, the stub is containing an updated list of server nodes. All that is required is to know the address of one active node when the first call is made. This means that an eventual fail-over[5] are handled by the stub, provided by the server, on the client side. By doing this there will not be any single points of failures in the system (such as a dispatcher would be) and the clients' lifetime is strongly connected to the lifetime of the smart proxy. I.e. If the client fails, the built in dispatcher fails, but it would not be a service failure. [Labourey et al, 2003]

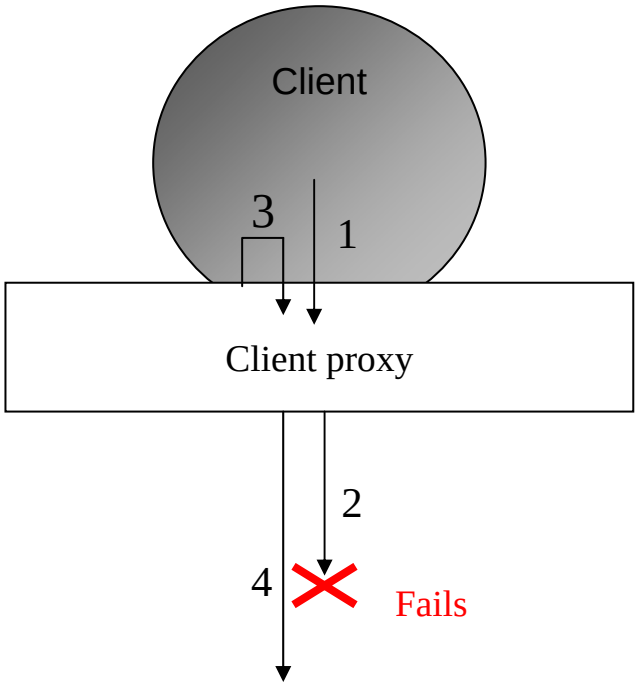

Figure 2, Smart proxy [Labourey et al, 2003]

2.3 Enterprise JavaBeans

Enterprise JavaBeans is provided by Sun Microsystems' and is defined as "... a component architecture for the development and deployment of component-based distributed business applications." [SUN, 2004].

Monson-Haefel [Monson-Haefel] gives a shorter version of the definition which mentions distributed objects. Distributed object looks like they reside on a different computer than it actually does [Monson-Haefel]. An object that resides on the middle tier looks like it is residing on the client side of the system. This is achievable through a *Remote Method Invocation* (RMI) [RMI, 2004] protocol that are using stubs and skeletons to communicate method invocations over a network (see Figure 3). Not only Java uses RMI but also CORBA [CORBA, 2004] and Microsoft .NET [MS, 2004]. They all have their own protocols.

[5] Moving the job/transaction to another node when the first node fails to respond.

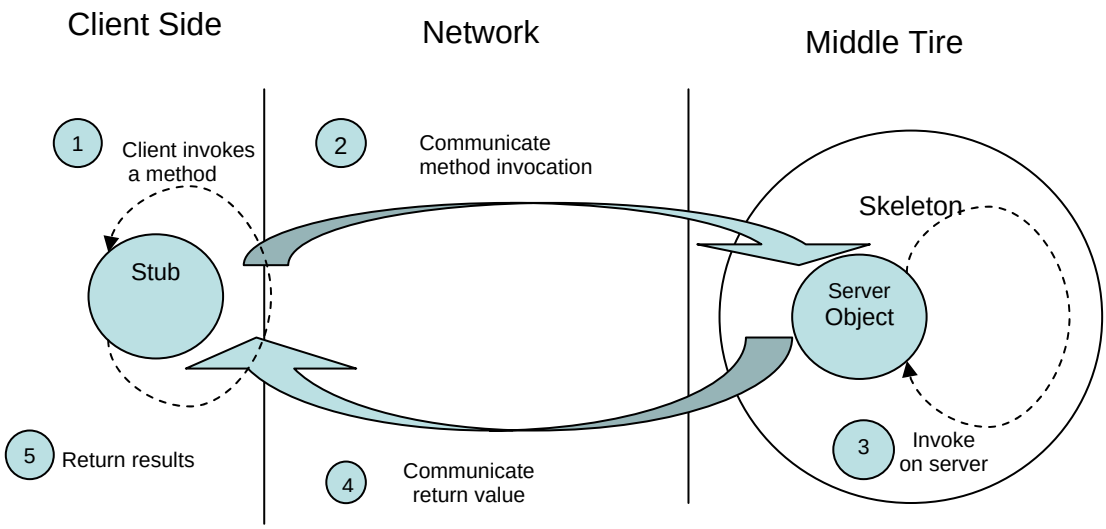

Figure 3, Java RMI [Monson-Haefel, 2001]

EJBs are dependent on an application server that supports the EJB specification. The main task for the application server is to host EJB applications. EJB applications are collections of components called beans and Meta information (xml documents) about the beans. An application is hosted by a container and the container is providing the services to the beans. It handles the communication between the client and the beans and the communication between the beans residing within the same container or between two different containers.

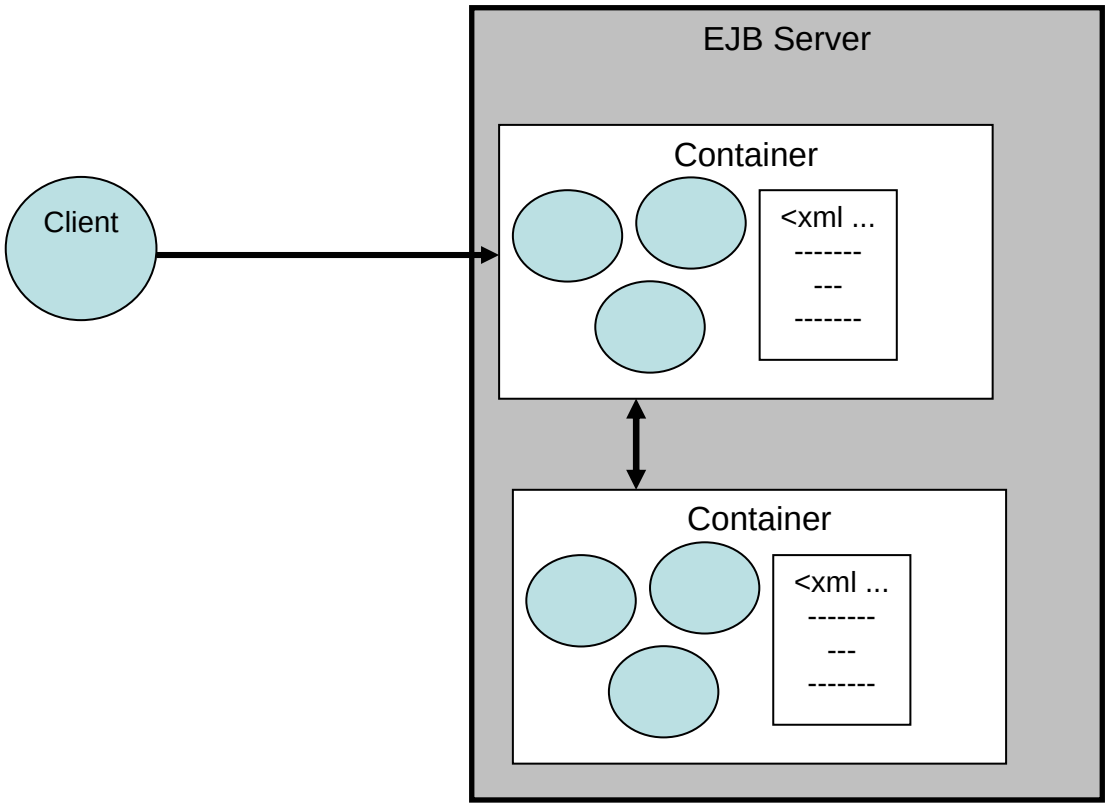

Figure 4, EJB Architecture [Monson-Haefel, 2001]

3 ARCHITECTURES

To investigate if and how the availability of an existing system can be increased by adding a middle layer of Enterprise Javabeans with replication logic, an industrial system was used. The system is a platform of mobile services such as sending text messages (SMS), Push SMS[6] (PPG), multimedia messages (MMS) and over the air configuration messages (OTA). To do so the service platform, Mobilis™ , is communicating with several mobile operators, e.g. Vodafone, Telia and Tele2.

The service this work is focusing on is the SMS service. A layer (POPE) between the client and the SMS server handles user accounts, authentication and authorization. It also provides the clients with the communication interface. Each user have its own account and a number of prepaid messages to send. POPE and the SMS server will be seen as one in this work and be referred to as *service node*. The different services are not dependent on each other and may therefor be deployed on computers separately.

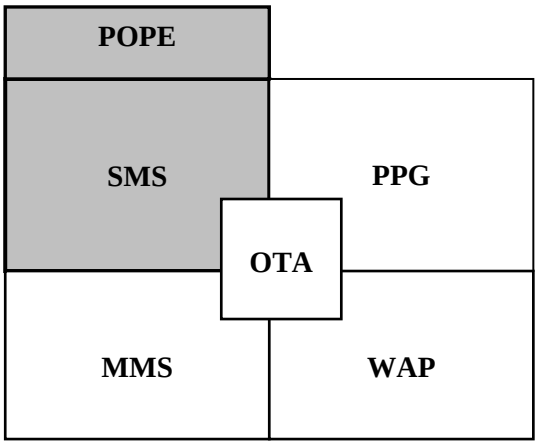

Figure 5, Mobilis™ – WIP's mobile service platform [WIP, 2004]

This SMS service together with POPE is used to send short messages (text messages) from personal computers to cell phones and the other way around. There are possibilities to send messages to multiple recipients and to receive replies from cell phones. Delivery reports (reports that indicates if and when the message reached its destination) from sent messages can be fetched. Communication between the clients and the server is synchronous and stateless. Synchronous in the sense of a request/response action before being able to send the next request and stateless in the sense of authentication, authorisation, account verification and sending messages is all in one service call from the client.

3.1 Origin Architecture

The origin architecture is a client-server system where multiple clients are able to invoke service actions on a remote server. There are no restrictions to replicate the server side of the architecture, but there are no current possibilities for the origin architecture to be replicated and at the same time keeping the transparency towards the client. If the server is replicated, the clients need to know the addresses to all servers, which puts the responsibility to manage eventual fail-over actions on the client code. There would also be a problem with consistency of the system data.

[6] Push SMS is a text message containing a link to a WAP page. PPG stands for Push Proxy Gateway.

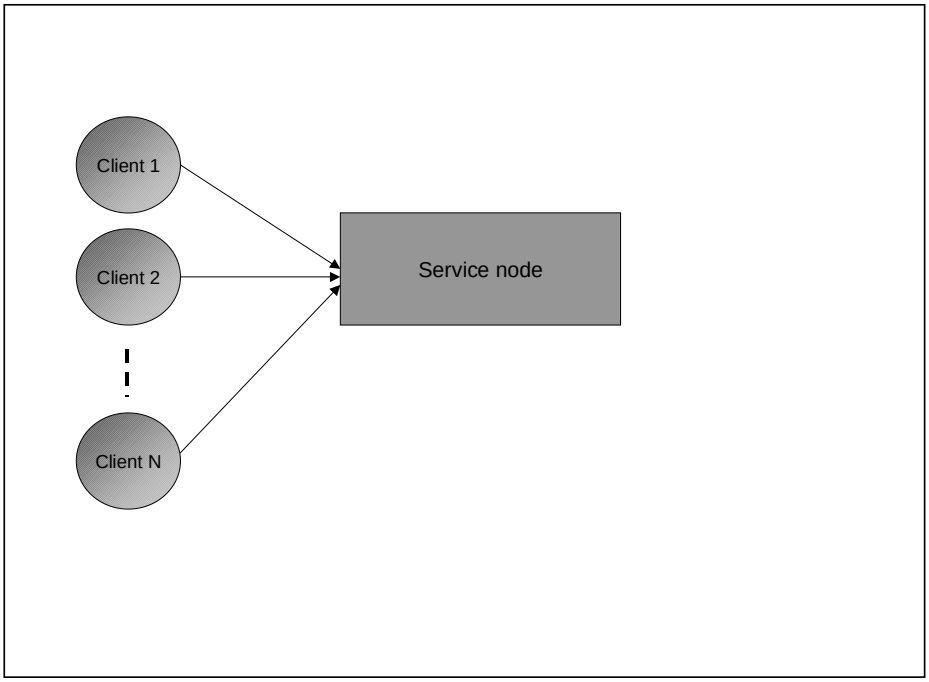

Figure 6, Origin architecture, Single service node

3.2 New Architecture

The reason to develop a new system architecture is as mentioned to increase the availability. To do so we need to decrease, or preferably eliminate, the number of single points of failures in the system. This means that we need to perform some sort of software replication[7] of the server side. When replicating a software system it is important to provide *linearizability consistency* to the service [Baldoni et. al, 2002]. Linearizability consistency means that all of the replicates perform the same task and respond in the same way to a client's request. If this is achieved we are one step closer to keep the transparency towards the client.

Since it is not desirable to change the existing system's implementation it is not possible to put the replication logics in the service node. Therefore a middle layer between the client and the service nodes will be responsible for the replication logics. This approach is called Three-tier software replication [Baldoni et. al, 2002].

Layer architecture approaches are not uncommon and accepted in other areas as well, such as Network communication architectures, database architecture. And to interfere as little as possible with the existing system, the three-tier approach was chosen to let the middle-tier work as an interface between the clients and the server system.

[7] Software replication is in this work considered as redundant server nodes that perform the same tasks according to its specification.

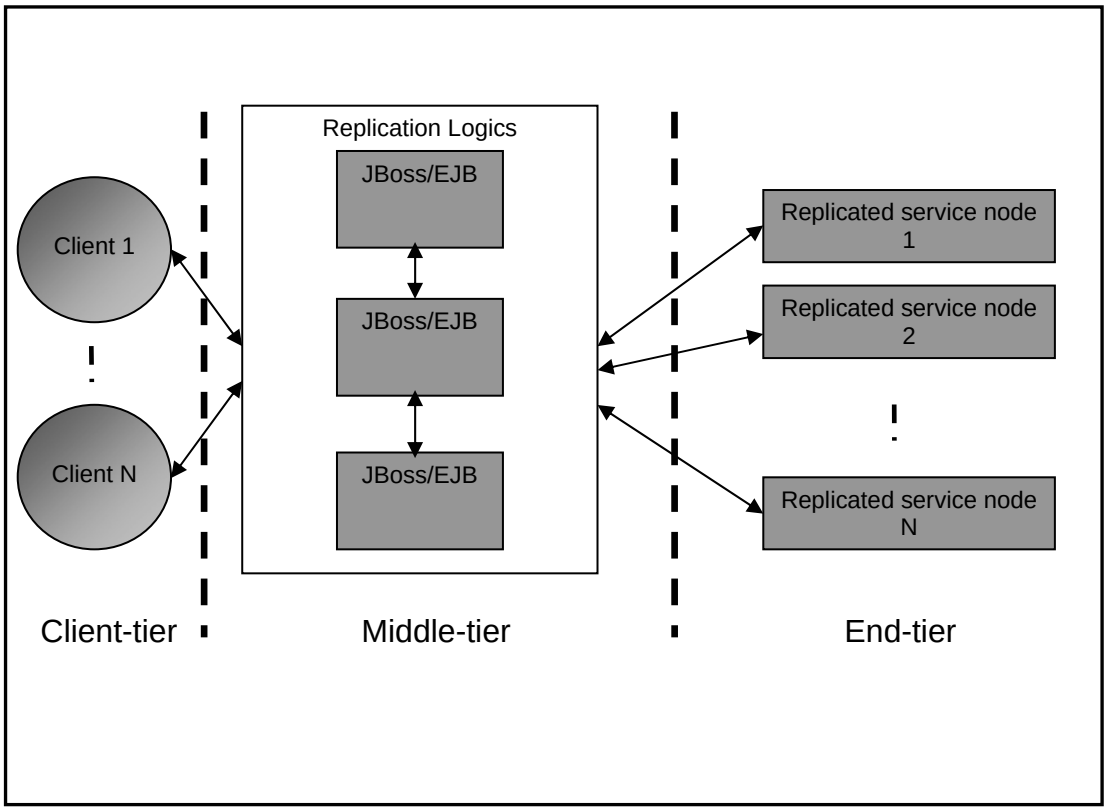

Figure 7, Three-tier software replication approach (JBoss/EJB solution)

The basic concept of Three-tier software replication is to let the middle layer receive requests from the clients and forward them to one of the replicate service nodes (end-tier). The middle layer is responsible for handling fail-over between the service nodes when the service node fails to process requests. In this case the middle layer is represented by application servers with deployed EJBs.

There is also a possibility of a middle-tier node failure. The JBoss application server together with EJBs provide the clients with a smart proxy, described in section 2.2.1 (Figure 2), that handles the fail-over actions between the clients and the middle-tier.

4 ARCHITECTURE ASSESSMENT

To evaluate and compare the two architectures described in chapter 4, scenario based assessment [Bosch, 2000] is used. Bosch describes different techniques of assessments – qualitative assessment, quantitative assessment and assessment of theoretical maximum – where the first is used to compare two architectures regarding one or more quality attributes. Quantitative assessment is used to make "… quantitative statements about the quality attributes of the software architecture." [Bosch, 2000] which is not of interest in our work since we do not want to put a value on the availability but compare two different approaches. Assessment of theoretical maximum is normally used to investigate if the current architecture is sufficient, for one quality attribute, or if it needs to be further evolved.

Considering the purposes of the different techniques, qualitative assessment is selected. The qualitative assessment gives a binary value of the two alternative architectures. I.e. it tells us which architecture is the better alternative for one quality attribute (in this case: availability). In traditional architecture assessment this may be a disadvantage since the best alternative may differ for different quality attributes, but in this case we are only interested in one attribute, which makes this technique most suitable.

Since the new architecture is also implemented, some of the scenarios may, in addition to the scenario based assessment, be tested to confirm or decline the results of the assessment.

4.1 Scenarios

Scenario based assessment is based on a set of scenarios. Normally an architecture assessment has many sets of scenarios since it is usually more than one quality attribute that is of interest. These sets are called profiles and can be specified in two ways; complete or selected [Bosch, 2000]. In a complete profile, as the name reveals, all relevant scenarios are included and in a selected profile, selected (preferably by random) scenarios are chosen to be a part of the profile's scenario set.

It would be preferable to use a complete profile when you are only evaluating the architectures from the perspective of one quality attribute, but to call the profile complete you need to cover all relevant scenarios. In almost every case this is considered as impossible [Bosch, 2000].

In this section the identified scenarios are listed and described. The scenarios are categorized depending on the causes. A traditional scenario profile would have weights assigned to them illustrating the likelihood of occurrence [Bosch, 2000] which was the plan for the scenarios listed in this profile as well. However we decided not to assign weights to the scenarios because of the lack of data. It would not be reliable weights since it all depends on the system context. WIP had no statistics of the outage duration and the systems from where we found data where too unlike to be useful as experience base for the weights of the scenarios in the profile.

4.2 Availability scenario profile

The profile is divided into two main groups. The first is for the unplanned scenarios and the second for the planned. Within the group of unplanned scenarios there are a couple of categories that are derived from the outage breakdown given by Pfister.

4.2.1 Unplanned outage

ID:	**U-1 CPU failure, service node**
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary service server node.

ID:	**U-2 CPU failure, Data source node**
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary data source node.

ID:	**U-3 Ethernet card failure, service node**
Category:	Network failure
Description:	An Ethernet card fails on an arbitrary service node. Meaning the hardware needs to be replaced.

ID:	**U-4 Network switch failure**
Category:	Network failure
Description:	The network switch breaks and prevents the nodes in the system to communicate with each other.

ID:	**U-5 Office power failure**
Category:	Power failure
Description:	Electricity is lost for the entire office.

ID:	**U-6 Power failure on system service node**
Category:	Power failure
Description:	An arbitrary system service node looses electrical power.

ID:	**U-7 Application server failure on system service node**
Category:	Software failure
Description:	The application server looses its ability to process requests on an arbitrary service node. I.e. the node will not be able to respond to that request.

ID:	**U-8 EJB failure on replication logic node**
Category:	Software failure
Description:	A request makes on of the EJBs to fail.

4.2.2 Planned outage

ID:	**P-1 Software upgrade on system service node**
Category:	Maintenance
Description:	Software upgrades on system service node what requires shutdown of the node's system software.

ID:	**P-2 Hardware upgrade on system service node**
Category:	Maintenance
Description:	Hardware upgrade on system service node that requires shutdown of the node.

ID:	**P-3 Removal of system service node**
Category:	Maintenance
Description:	A system service node is removed.

ID:	P-4 Addition of system service node
Category:	Maintenance
Description:	A system service node is added to be a part of the system.

ID:	P-5 Reconfiguring system
Category:	Regulations
Description:	System configuration needs to be changed/updated

4.3 Scenario execution

Scenario execution is a theoretical assessment of the architecture. The architecture is assessed on architectural level and not as an implemented system. Each scenario in the availability scenario profile is assessed and the impact on the architecture is documented. The availability is divided into three levels. One where the service availability is not affected at all, one where the scenario causes a very short time of outage and one where the service becomes unavailable for a longer time.

If the service is not affected at all, the availability value in the execution description will be 1. If the scenario causes a milder level of outage the value will be 0.5. And if the service is down for a longer period of time, the value of the availability is represent by the number 0.

4.3.1 Origin architecture

ID:	U-1 CPU failure, service node
Category:	Hardware failure
Description:	The CPU goes down on a arbitrary service node.
Impact:	Since the old architecture only support one service node in the system the service is considered unavailable. If the CPU goes down on that node it will not be able to process the request sent from the client and there are no other nodes to take care of the request.
Availability:	0

ID:	U-2 CPU failure, Data source node
Category:	Hardware failure
Description:	The CPU goes down on a arbitrary data source node.
Impact:	There is only one data source node in the origin system which the service server node is dependent on which makes it very important to the system. If it goes down the service is considered unavailable.
Availability:	0

ID:	U-3 Ethernet card failure, service node
Category:	Network failure
Description:	An Ethernet card fails on a arbitrary service node. Meaning the hardware needs to be replaced.
Impact:	The request sent from the client does not reach the service node. A time-out will be sent back as a reply without the service invocation being executed. System is considered as unavailable until the ethernet card is replaced.
Availability:	0

ID:	U-4 Network switch failure
Category:	Network failure
Description:	The network switch breaks and prevents the nodes in the system to communicate with each other.
Impact:	If the network switch breaks there will not be any communication between the different nodes in the system. I.e. datasource, service node. System is unavailable until the network switch can be replaced.
Availability:	0

ID:	U-5 Office power failure
Category:	Power failure
Description:	Electricity is lost for the entire office.
Impact:	All the nodes in the system is placed on a LAN[8] and is dependent on the same power supplier. The service is unavailable until the power is regained.
Availability:	0

ID:	U-6 Power failure on system service node
Category:	Power failure
Description:	An arbitrary system service node looses electrical power.
Impact:	Since the system only supports one service node there are no other nodes to take care of requests sent from the clients until the one node regains power.
Availability:	0

ID:	U-7 Application server failure on system service node
Category:	Software failure
Description:	The Application server looses its ability to process requests on an arbitrary service node. I.e. the node will not be able to respond to that request.
Impact:	If the software of the application server fails to run the business logic it will not be able to respond to any requests since there are no other servers to take care of the request.
Availability:	0

ID:	U-8 EJB failure on replication logic node
Category:	Software failure
Description:	A request makes on of the EJBs to fail.
Impact:	Not applicable to origin architecture

ID:	P-1 Software upgrade on system service node
Category:	Maintenance
Description:	Software upgrades on system service node what requires shutdown of the node's system software.
Impact:	If a software upgrade is performed, the system needs to be shut down and restarted with the new version. For that period of time the service is considered as unavailable since there are no other servers to take care of the request. The time is assessed to be rather short.
Availability:	0.5

[8] Local Area Network

ID:	P-2 Hardware upgrade on system service node
Category:	Maintenance
Description:	Hardware upgrade on system service node that requires shutdown of the node.
Impact:	A hardware upgrade will result in unavailability of the service since there is no other node to take care of incoming requests. The time of outage duration is considered as short which gives the availability level of 0.5
Availability:	0.5

ID:	P-3 Removal of system service node
Category:	Maintenance
Description:	A system service node is removed.
Impact:	Not really applicable to the origin architecture since there are no other node to take care of the request. It needs to be replaced and during that time the service is considered as unavailable.
Availability:	0

ID:	P-4 Addition of system service node
Category:	Maintenance
Description:	A system service node is added to be a part of the system.
Impact:	Not applicable to origin architecture.

ID:	P-5 Reconfigure system
Category:	Regulation
Description:	System configuration needs to be changed/updated
Impact:	The system reads configuration files and continues to operate.
Availability:	1

4.3.2 New Architecture

ID:	U-1 CPU failure, service node
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary service node.
Impact:	The middle layer detects the unavailable service node and redirects the request to one of the other service nodes. The service is considered available.
Availability	1

ID:	U-2 CPU failure, Data source node
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary data source node.
Impact:	There is still only one database source and it is a single point of failure. The new architecture will not be able to process the requests that require access to the data source.
Availability:	0

ID:	U-3 Ethernet card failure, service node
Category:	Network failure
Description:	An Ethernet card fails on an arbitrary service node. Meaning the hardware needs to be replaced.
Impact:	The node on which the broken ethernet card will be unable to process requests, but there are other that will be able to do so because of the smart-proxy on the client side.
Availability:	1

ID:	U-4 Network switch failure
Category:	Network failure
Description:	The network switch breaks and prevents the nodes in the system to communicate with each other.
Impact:	Since the cluster is located on the same LAN, a network switch is a single point of failure. The system will not be available if it breaks
Availability:	0

ID:	U-5 Office power failure
Category:	Power failure
Description:	Electricity power is lost for the entire office.
Impact:	All the nodes in the system is placed on a LAN[9] and is dependent on the same power supplier. The service is unavailable until the power is regained.
Availability:	0

ID:	U-6 Power failure on system service node
Category:	Power failure
Description:	An arbitrary system service node looses power.
Impact:	When one of the service nodes looses power there are other that will still be able to process requests from the client.
Availability:	1

ID:	U-7 Application server failure on system service node
Category:	Software failure
Description:	The Application server looses its ability to process requests on an arbitrary service node. I.e. the node will not be able to respond to that request.
Impact:	The replication logics will detect the failed node and redirect the request to an other of the replicated service nodes. The client will not be suffering from this failure except from lost performance.
Availability:	1

ID:	U-8 EJB failure on replication logic node
Category:	Software failure
Description:	A request makes on of the EJBs to fail.
Impact:	Since the EJBs are identical on the different replication logic nodes the request will be redirected to all the other nodes and make them fail as well. This will in bring the entire system down.
Availability:	0

[9] Local Area Network

ID:	P-1 Software upgrade on system service node
Category:	Maintenance
Description:	Software upgrades on system service node what requires shutdown of the node's system software.
Impact:	If a service node is shut down the replication logic layer will notice this, and will not send the requests to the non-functioning service node until it is up and running again. The clients' requests will still be processed by the system.
Availability:	1

ID:	P-2 Hardware upgrade on system service node
Category:	Maintenance
Description:	Hardware upgrade on system service node that requires shutdown of the node.
Impact:	The system will still be able to respond to clients' requests since there are other service nodes.
Availability:	1

ID:	P-3 Removal of system service node
Category:	Maintenance
Description:	A system service node is removed.
Impact:	There will still be service nodes to process the clients' requests. Service is available.
Availability:	1

ID:	P-4 Addition of system service node
Category:	Maintenance
Description:	A system service node is added to be a part of the system.
Impact:	After notifying the middle layer of a new replica, the service node is a part of the system and it will not affect the availability.
Availability:	1

ID:	P-5 Reconfigure system
Category:	Regulation
Description:	System configuration needs to be changed/updated
Impact:	The system service nodes read the configuration files and continue to operate. A problem of inconsistency will occur. This breaks the cluster rule of linearizability consistency, but the service will still be available.
Availability:	1

4.4 Scenario validation

As a complement to the assessment on architectural level the new architecture was implemented to be able to validate the correctness of the architecture assessment. The scenarios in the profile were used as a form of test cases to confirm the scenario based assessment.

ID:	**U-1 CPU failure, service node**
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary service node.
Validation OA[10]:	There is no node to take care of incoming requests from the clients.
Validation NA[11]:	The implemented system inactivates the service node and stop forwarding request to the broken node. The service is considered available. When the CPU is replaced the node needs to be manually activated.

ID:	**U-2 CPU failure, Data source node**
Category:	Hardware failure
Description:	The CPU goes down on an arbitrary data source node.
Validation OA:	The service is unavailable due to lacking authorization possibilities.
Validation NA:	When the data source becomes unavailable the user authorization fails and no requests are able to be processed.

ID:	**U-3 Ethernet card failure, service node**
Category:	Network failure
Description:	An Ethernet card fails on an arbitrary service node. Meaning the hardware needs to be replaced.
Validation OA:	Tested by detaching the ethernet card. Requests do not reach the destination. No more nodes to handle the requests
Validation NA:	Tested by detaching the ethernet card. The call from the replication logic layer was timed out and redirected to another service node. There was a delay in handling the request, but it was processed.

ID:	**U-4 Network switch failure**
Category:	Network failure
Description:	The network switch breaks and prevents the nodes in the system to communicate with each other.
Validation OA:	No requests reached their destination
Validation NA:	No requests reached their destination.

[10] Origin Architecture
[11] New Architecture

ID:	U-5 Office power failure
Category:	Power failure
Description:	Electricity power is lost for the entire office.
Validation OA:	Not tested.
Validation NA:	Not tested.

ID:	U-6 Power failure on system service node
Category:	Power failure
Description:	An arbitrary system service node looses power.
Validation OA:	The one service node lost power and was not able to process any requests.
Validation NA:	The EJBs noticed the failed service node and stopped forwarding requests to that node.

ID:	U-7 Application server failure on system service node
Category:	Software failure
Description:	The Application server looses its ability to process requests on an arbitrary service node. I.e. the node will not be able to respond to that request.
Validation OA:	Not tested.
Validation NA:	The JBoss partition recognized the failed JBoss instance and excluded it from the cluster. An updated list without the failed node was sent to the client proxy. The request was re-routed to another node without the client knowing about it.

ID:	U-8 EJB failure on replication logic node
Category:	Software failure
Description:	A request makes one of the EJBs to fail.
Validation OA:	Not applicable
Validation NA:	The request was not redirected as the assessment indicated. An exception was thrown and the client received an error message. When resending the request (to another node) the same thing happened. But the service was still available for other requests. The value of availability is therefore changed to 0.5.

ID:	P-1 Software upgrade on system service node
Category:	Maintenance
Description:	Software upgrades on system service node what requires shutdown of the node's system software.
Validation OA:	The service becomes unavailable for a period of time. The time is outage is quite short and a user might not even notice.
Validation NA:	The EJBs detected an unavailable service node and stopped forwarding requests to that node. When the service node was started again, it manually had to be activated in the EJB configuration.

ID:	P-2 Hardware upgrade on system service node
Category:	Maintenance
Description:	Hardware upgrade on system service node that requires shutdown of the node.
Validation OA:	Not tested.
Validation NA:	Not tested.

ID:	P-3 Removal of system service node
Category:	Maintenance
Description:	A system service node is removed.
Validation OA:	There are no other nodes to take care of the requests. Service is unavailable.
Validation NA:	EJBs noticed the removed service node and stopped forwarding requests to that node.

ID:	P-4 Addition of system service node
Category:	Maintenance
Description:	A system service node is added to be a part of the system.
Validation OA:	Not tested.
Validation NA:	After starting up the new service node and manually configured the EJBs, it started to forward the requests to the new node.

ID:	P-5 Reconfigure system
Category:	Regulation
Description:	System configuration needs to be changed/updated
Validation OA:	The service node was reconfigured without any requests failing.
Validation NA:	The service nodes where reconfigured manually one after another. No requests were left without a response.

4.5 Results

To sum up the scenario based assessment of the two architectures we present the result from a category perspective. The scenarios are divided into five categories all together: hardware failure, network failure, power failure, software failure and maintenance/regulation.

Taking into account both assessment and validation, the new architecture has most impact on the maintenance scenarios. The new architecture handled almost all the maintenance scenarios, compared to the origin architecture which did not manage to keep the service available for more than one of them with no outage, and two scenarios with minor outage.

Unplanned scenarios were also handled better with the Three-tier replication architecture. As far as the scenario did not affect the entire network or infrastructure, the service was kept available. But since the architecture does not support communication over separated networks, the service became unavailable when the scenarios affected the entire office space. For example power failure for the entire building or a broken network switch were circumstances that made the service unavailable even for the new architecture.

	New Architecture	Origin Architecture
Hardware failure	1/2	0/2
Network failure	1/2	0/2
Power failure	1/2	0/2
Software failure	1/2	0/1
Planned	4.5/5	1 + 0.5 + 0.5 / 4

Table 2, Handled scenarios

According to Pfister the unplanned causes of outage duration only stands for 8 % (see Figure 8) of the total amount which means that the planned scenarios should be more important to handle. On the other hand power failure for the entire office building stands for most of the unplanned outage time which the new architecture did not handle.

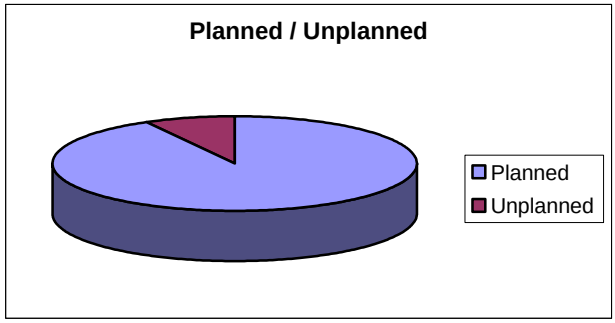

Figure 8, Planned / Unplanned outage duration [Pfister, 1998]

The summary of the assessment and validation (Table 2) let us know that the new architecture manage to keep the service available for more scenarios than the origin architecture could manage. However the new architecture does not handle all scenarios that would cause the service to become unavailable, using the origin architecture, but it manage more of the assessed scenarios.

5 SUMMARY

This section discusses the weaknesses of JBoss clustering EJBs and the assessment. Here we also describe the advantages of the implementation of the new architecture by using third party clustering logics such as JBoss.

5.1 Discussion

When Java RMI is mentioned in the context of communication between computers, many software developers becomes skeptical. And there are reasons to become just that since it is hard to accomplish, and considered insecure, when the communicating computers are located on different networks. This makes the solution of clustered EJBs somewhat reduced when it comes to availability, since it would be preferable to spread the cluster over a wider area to reduce the single points of failure. All the outage reasons that affect the network, e.g. building power failure, will make the cluster unavailable. In addition the redundant service nodes are all working with a single shared data source which makes it impossible to eliminate the single points of failure since no changes could be made on the code of the service nodes. If the data source fails, the entire cluster will be useless.

According to [Wolf, 2004], the availability of the service can not be considered as high if there are any single points of failure in the system. But the assessment described in this work was not meant to state whether the system was highly available or not, but to assess if it had a higher availability than the origin architecture.

The results of this work may be questioned since it is only based on one sort of system. If the system would be handling states, it would probably have become more expensive to develop since the replication logics would have had to take that under consideration. The application server used in this assessment, JBoss application server, is according to its specification supposed to handle statefull transactions, but it is not evaluated in this work.

Another uncertainty of the results of this work is the scenario elicitation. Several scenarios where not assessed since they would have the same impact on the system as another scenario that was already in the profile. A couple of those scenarios could make the results differ from the result presented in this work. It would also have been better if the scenarios could be weighted in a manner that would have been reliable. If any data on a similar system was available to use as an experience bas for the weighting of the scenarios a more quantitative result could have been presented.

As mentioned in the methodology section, WIP wanted to spend as little money as possible for the architectural change. And by using a COTS application server that is provided under the LGPL[12] license WIP may distribute the server as a third party system without any royalties or license fees. The cost of the clustered architecture is purely from the development of the replication logics that is summed up to 400 man-hours. There was only one software developer, with no experiences of EJB, working with the development which gives an indication of how little effort it takes to develop a cluster of EJB servers that affect the availability in a positive manner.

5.2 Conclusions

By using an existing software system described in this work, and applying a three-tier replication approach using application servers supporting clustering and EJBs, we have proven, with the flaws discussed in section 5.1, that the availability of existing software systems can be increased for a relatively low cost.

[12] Lesser General Public License

No changes had to be made to the origin system code implementation. This may however be specific to this system. The changes that had to be made was on the client tier since the remote objects was changed.

There are as discussed weaknesses to this approach, mainly since Java RMI restricts the distribution of the system. This means that the LAN will become a single point of failure and is by that a sore spot of the system. However the cost is fairly low which makes this approach interesting for low budget organizations that wants to increase there software service availability.

5.3 Future work

Quality attributes are not uncommonly contradicting each other which make it interesting to assess the new architecture from another perspective. For instance the performance of the service will most likely be affected by adding a middle tier of replication logics.

To evaluate the Three-tier replication approach with EJB further the communication over different networks is an interesting aspect. If the cluster could be spread over a wider physical area, the availability of the service would definitely be higher. The server room would not be seen as a single point of failure.

There are some other techniques to use in this area which are of interest of investigating further. Three-tier replication increases the availability of a service but another technique may increase it even more and may also be less costly.

6 REFERENCES

[Baldoni et al, 2002]
Robert Baldoni, Carlo Marchetti and Allesandro Termini, "Active Software Replication through a Three-tier Approach", 2002, IEEE, Reliable Distributed Systems, 21st IEEE Symposium

[Bengtsson, 2002]
PerOlof Bengtsson, "Architecture-Level Modifiability Analysis", 2002, Blekinge Institute of Technology, ISBN 91-7295-007-2

[Bosch, 2000]
Jan Bosch, "Design & Use of Software Architectures", 2000, Pearson Education Limited, ISBN 0-201-67494-7

[CORBA, 2004]
http://www.corba.org/ , September 26, 2004

[GSM, 1996]
European Telecommunications Standards Institute, "Global System for Mobile Communications-Digital cellular telecommunications system (Phase 2) technical realization of the Short Message Service (SMS)", 1996, ICS 33.060.50

[RMI, 2004] Java RMI Specification
http://java.sun.com/j2se/1.5.0/docs/guide/rmi/spec/rmiTOC.html, September 26, 2004

[JBoss, 2004]
JBoss Open Source, September 26, 2004, http://jboss.org

[Labourey et al, 2003]
Sacha Labourey and Bill Burke (The JBoss Group), "JBoss Clustering (edition 4)", 2003, JBoss Group, LCC Atlanta USA, JBoss documentation

[Monson-Haefel]
Richard Monson-Haefel, "Enterprise JavaBeans, 3rd Edition", 2001, O'Reilly & Associates Inc., ISBN: 0-596-00226-2.

[MS, 2004]
http://www.microsoft.com/net/, September 26, 2004

[Pfister, 1998]
Gregory F. Pfister, "In search of Clusters", 1998, Prentice Hall, ISBN 0-13-899709-8

[Rushikesh et. al, 1999]
Rushikesh K. Joshi, O. Ramakrishna and D. Janaki Ram, "ShadowObjects – A Programming Model for Service Replication in Distributed systems", Journal of Parallel and Distributed Computing 59, pp. 1-12.

[SUN, 2004]
EJB Specification, http://java.sun.com/products/ejb/, September 26, 2004

[WIP, 2004]
Wireless Independent Provider AB, http://www.wip.se, September 26, 2004.

[Wolf, 2004]
Laurence J. Wolf, “Preventing Failure: The Value of Performing a Single Point of Failure Analysis for Critical Applications and Systems”, March/April 2004, Auerbach Publications Inc, Information Systems Security