VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Tony Micael Alexander Sandelin

DESIGNING AND IMPLEMENTING ARCHITECTURAL CHANGES FOR LIFE CYCLE TRACKING (LCT)

Technology and Communication
2010

VAASAN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Tony Sandelin
Opinnäytetyön nimi	Arkkitehtuuristen muutosten suunnittelu ja toteutus tuotteiden jäljitettävyys järjestelmään (LCT)
Vuosi	2010
Kieli	englanti
Sivumäärä	50
Ohjaaja	Pirjo Prosi

Tämän opinnäytetyön tarkoitus on ollut tuottaa ABB:n tietojärjestelmäprojektiin liittyvä määrittelyvaiheen dokumentaatio sekä määrittää projektin johtamiseen liittyvä teoriapohja.

Tietojärjestelmäprojektin tavoite oli suunnitella sekä toteuttaa tuotteiden jäljitettävyysjärjestelmälle uusi arkkitehtuuri. Uuden arkkitehtuurin kolmitasorakenne helpottaa järjestelmän ylläpitoa, parantaa ulostulodatan laatua sekä alentaa järjestelmän ylläpitokustannuksia.

Jotta järjestelmän suorituskyky vastaisi paremmin tulevaisuuden haasteisiin, myös järjestelmässä käytetyt teknologiat päivitettiin. Tiedonsiirrot eri tasojen välillä toteutettiin SOAP-viesteillä, jotka lähetetään HTTP:n yli käyttäen verkon ohjelmistorajapintoina web-palveluita. Liiketoiminnalliset prosessit automatisoitiin hyödyntäen Business Process Management (BPM) -palvelinta ja verkon turvallisuutta parannettiin käyttämällä ulkoverkkoon liikennöitäessä demilitarisoitua palvelinta, jossa on tuplapalomuuriarkkitehtuuri.

Tätä lopputyötä kirjoitettaessa uuden arkkitehtuurin kehittämisvaihe on valmis ja uusi arkkitehtuuri on parhaillaan testivaiheessa.

Asiasanat	suunnittelu, toteutus, arkkitehtuuri, määrittelyvaihe, dokumentaatio, SOAP, HTTP, WWW-palvelu, BPM, DMZ

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma

ABSTRACT

Author	Tony Sandelin
Title	Designing and Implementing Architectural Changes for Life Cycle Tracking (LCT)
Year	2010
Language	English
Pages	50
Name of Supervisor	Pirjo Prosi

The goal of this thesis has been to produce the project definition phase documents to an information system project at ABB and at the same time provide a theoretical background for project management.

The project designed and implemented a new architecture to an existing data collection system. The new three-tier architecture facilitates maintenance tasks, enhances the quality of the output data and at the same time makes the maintenance of the system financially more economical.

To face the requirements of future, the technologies used were also updated to correspond to the demands of modern business processes. Data transfers between the tiers are implemented using SOAP over HTTP and using web services as web APIs. Business processes were automated with a BPM server, and the network security was enhanced through usage of a DMZ with dual-firewall architecture.

At this point the new architecture has been developed and is currently in its testing phase.

Keywords	Design, Implementation, Architecture, Definition phase, Documents, SOAP, HTTP, Web Service, BPM, DMZ

FOREWORD

This thesis has been commissioned by ABB Distribution Automation unit in Vaasa and it will be my final-year project for Vaasan Ammattikorkeakoulu, University of Applied Sciences, before obtaining a degree in Information Technology.

Even though this task has been very challenging and time consuming, I have enjoyed working with it from the beginning until the end. I am sure I could have chosen an easier topic for the thesis, but an easier topic would surely not have been as rewarding and educational as this project has been.

First of all, I would like to thank my ABB supervisor Mathias Grädler and my manager Kaj Andtbacka for guiding and supporting me along the way. I would also like to thank my previous manager Mari Lintula for introducing me to Mathias and making this possible in the first place.

Additionally I would like to thank my supervisor at the University of Applied Sciences, Pirjo Prosi, for guidance and support as well as all the other people that have supported the project's progress in any way.

Finally yet importantly, I would like to thank my fiancée Päivi and my son Luca for their loving support and patience during this hectic season we have had, while writing this thesis.

Tony Micael Alexander Sandelin

Vaasa, February 19, 2010

ABBREVIATIONS

API	Application Programming Interface
BEEP	Blocks Extensible Exchange Protocol
BPM	Business Process Management
B2B	Business-to-Business
DA	Distribution Automation
DMZ	Demilitarized Zone
DWH	Data Warehouse
ERP	Enterprise Resource Planning
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IED	Intelligent Electronic Device
IS	Information System
LCST	Life Cycle Service Tool
LCT	Life Cycle Tracking
MES	Manufacturing Execution System
MS	Microsoft Corporation
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery, and Integration
WSDL	Web Service Description Language

WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language
RPC	Remote Procedure Call

CONTENTS

1 INTRODUCTION

This chapter provides a general introduction to the thesis; introducing the subject, explaining the objectives and describing the structure.

1.1 LCT Concept

As the competition in the business market tightens every day, efficiency plays a great role. To increase efficiency, products are managed all the way through their lifecycles in the most effective way. Management continues from when the product is merely an idea until it is decommissioned. This business activity is called Product Lifecycle Management (PLM). /21/ PLM can be defined as "a systematic, controlled concept for managing and developing products and product related information" /19/.

To make the PLM even more efficient, a supplementing concept has been introduced; Life Cycle Traceability (LCT). LCT focuses on collecting information on the product's life after it has been delivered to the end customer. It keeps track of its geographical locations, owners and possible service history. After the product has been delivered to the end customer, the information has to be collected remotely. To increase the reliability of the data and to make sure that the data is up to date, an automated data collection system can be used to eliminate the human factor.

Such management concepts improve the operational efficiency of a company, by providing coherent and easy information retrieval and advanced information traceability. By using web based systems, companies can easily connect their globally dispersed facilities with each other. Using these product management concepts benefits all the groups working across the value chain of a company, saving both resources and valuable time. /19/

1.2 The Motivation and Objectives of the Thesis

During spring 2009, the author of this thesis was given the opportunity to take over the responsibility of an Information System (IS) project at ABB's Distribution Automation unit in Vaasa, and follow the project through as project manager. As this was the author's first project, the writing of a thesis supporting the project was considered to be worthwhile. It also ensured that the author got an in-depth view of project management and project documentation, forming a solid foundation for the first project.

The purpose of this thesis was to produce the fundamental documents needed in the project. Those documents define how the project should progress and what it should achieve. The following documents are the outcome of this thesis:

- Project plan, including an activity plan
- Requirements specification
- Functional specification
- Test plan

As a crucial part of this thesis, the actual output documents are also evaluated during the assessment of this thesis. However, they are not included in the public release due to the classified content owned by the case company.

The thesis and the project support each other in many ways. The following statements concentrate on describing how the thesis supports the project:

Theory of project management provides a foundation for managing the project in general and for documenting the project. This is considered to be indirectly beneficial to the project output.

Defines the requirements set for the project in the form of specification documents (Requirements Specification and Functional Specification) needed in the project.

Provides a project plan describing in detail how the project should be carried out, what its risks are and an estimation of the required resources (time, labour and budget).

Prepares a test plan on how the system should be tested before being considered to be stable and moved to production.

Proposes future enhancements on what could still be improved in the system after the project has concluded.

The figure below illustrates the interaction between the thesis and the project and shows which parts of the project are particularly supported by the thesis (Figure 1).

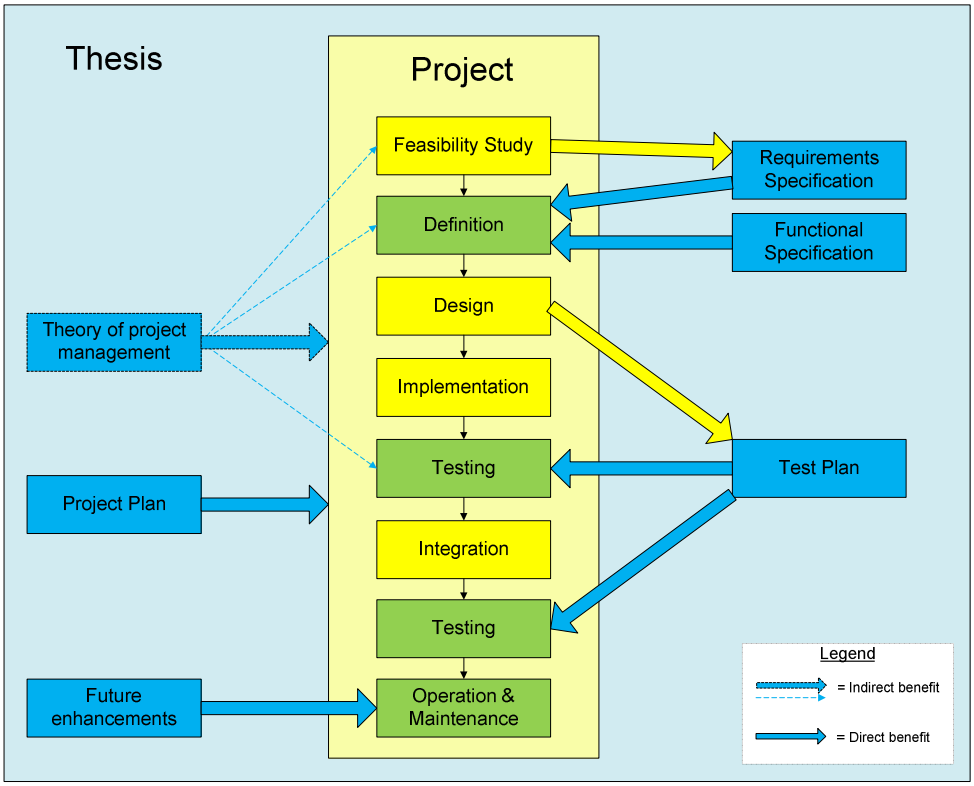

Figure 1. How the thesis and the project support each other.

1.3 The Structure of the Thesis

The thesis is divided into seven chapters.

Chapter one is an introduction to the thesis; introducing the subject and defining the motivation and structure of the thesis.

Chapter two covers the theoretical frame of reference of the thesis having its main focus on project management, the technologies used in the implementation as well as the testing method used.

Chapter three introduces the project that this thesis supports. It specifies the goals of the project, introduces the case company and examines the business case behind the project. Chapter thee also introduces the core functionality of the LCT.

Chapter four describes how the definition phase of the project was executed, what was produced during this phase as well as presents the main use cases of the system.

Chapter five presents the design and functionality of the architecture, at the same time describing how the technologies used have been adapted to the system.

Chapter six focuses on describing how the testing of the new architecture was carried out and at what point the tests were executed.

The seventh and final chapter evaluates the success of the project as a whole. Possible future improvements are provided for the case company and the overall success of the study is evaluated by the author.

2 THE THEORETICAL FRAME OF REFERENCE

This chapter covers the theory behind the project management and technologies used in the implementation. However, the theory section of this thesis focuses only on the aspects that are relevant for the scope of this thesis.

2.1 The Definition of a Project

A project is a promise of a solution to a problem. It is temporary by nature, always unique and contains uncertainty to some extent. Even if it usually culminates in the creation of one or several end products, it consists of a well-defined collection of tasks. A project also always consumes resources in the form of time, money and labour. /6/

2.2 Introduction to IS Projects

The lifecycle of an Information System (IS) project can be simplified to specification, design and implementation. As a general rule, a specification should answer the question what will be done and a design to the question how it will be done. An IS project also has several supporting operations, of which the most significant are quality assurance, product management and documentation. These supporting operations follow the project and the final product over its whole lifecycle. /5/

2.3 Project Execution Methods

A project can be approached in many different ways. The best approach is often a question of opinion and preference, but may also depend on the nature of the project or dependencies to other projects or systems. /4/ The next two subchapters introduce the execution methods that have been used in the project that this thesis is related to.

2.3.1 System Development Methodology

Methodologies were originally introduced to obtain control of large software development projects. Methodologies impose a disciplined process for project execution, aiming for a better predictability and increased efficiency within the project. The two mainstreams within the methodologies are

- *Plan-driven methodologies*, having a strong emphasis on planning, thus, often very bureaucratic. Following these methodologies might create so much additional work that the whole development pace slows down. /4/

- *Agile methodologies,* which compromise between no methodologies used and plan-driven methodologies. These methodologies are less document-oriented, are more adaptable to changes, have short iterations and behave in a more people-oriented way. /4/

As the project that this thesis is related to contained many unknown areas and resource dependencies, a more agile approach was considered to be more efficient. Even if most of the unknowns were identified immediately during the first steps of the project, they were of such a nature, it was considered to be a risk to delay attempting to compose complete specifications of the project before implementation.

2.3.2 Work Phasing

The development work or even the whole lifecycle of a project or a final product can be divided into phases according to a systematic work phasing model. The most commonly known phasing model is the waterfall model (Figure 2). This model was also used in the project related to this thesis, by adapting it to the agile approach. /5/

Even if it seems simple to just follow a scripted model, an IS project can never be run exactly according to the waterfall model in practice. The requirements keep changing and some of the requirements only appear during the project. It still

serves as a good stepping stone to the fundamental principles of project management. /5/

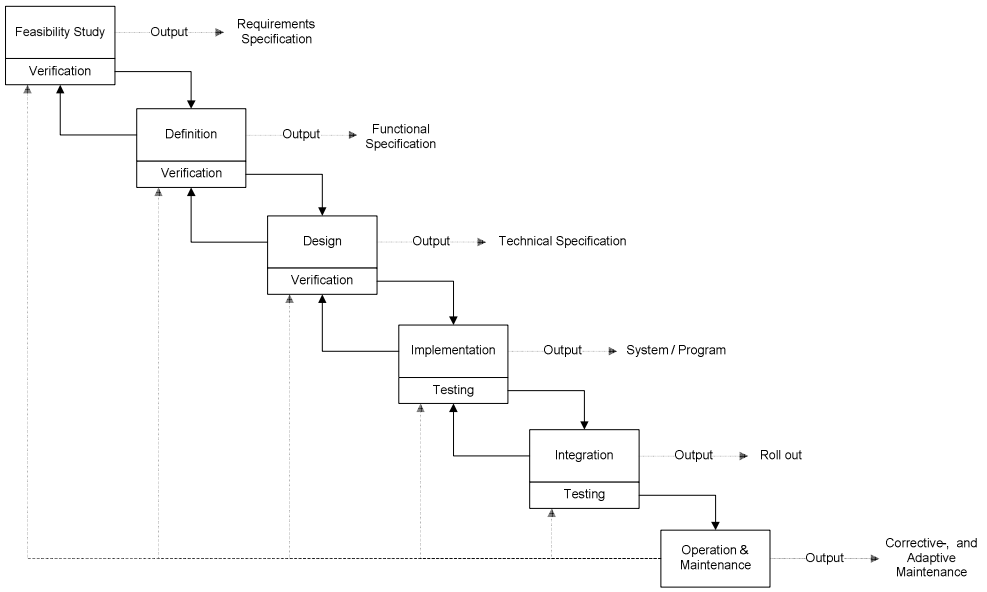

Figure 2. The development lifecycle according to the waterfall model.

All phases in the waterfall model include quality assurance measures that are used to eliminate the flaws from the system in as early a phase as possible. The phases are:

Feasibility Study: Collect the business requirements; the general requirements on the system level defining the needs without contributing to how it will be fulfilled. This phase describes why the system is made.

Definition: Analyzing the collected business requirements, evolving them into functional specifications and defining the system to be implemented. This phase describes what the system does.

Design: Designing the implementation of the functions. Consist of two levels, architectural-, and module design. In architectural design, the system is divided into independent parts, modules. While in the module design, the internal structure of a module is designed. A technical specification is the outcome of this phase that describes how the system is implemented.

Implementation: Implementing the system according to the specification documents.

Integration: Integrating the system (or program) into a production environment.

Testing: Both the implementation-, and integration phases include testing. Testing is handled on several layers; unit testing, integration testing and system testing. The unit testing concentrates on the smallest testable part in the programming code, integration testing tests the cooperation of a group of units and system testing focuses on the system as a whole and its performance.

Operation and Maintenance: Can be split into corrective-, and perfective maintenance. In corrective maintenance, the system is patched to better correspond to the requirements, whereas in perfective maintenance, the system is improved by modifying or extending its functionalities. This phase goes on through the whole lifecycle of the system until it is decommissioned. /5/

2.4 Project Definition Phase

During the project information is typically collected into different kinds of documents. There might be dozens of documents related to a project, but as a minimum there should be a project plan, functional specification, technical specification and a test plan.

2.4.1 Project Plan

A project plan is a description of how the final result will be reached with the defined resources and schedule. It is also a follow-up tool for the project, by which any possible slippages in the schedule or resources can be noticed as early as possible. /5/

Even if the project would be very small when considering its resources or timeline, the importance of a project plan should not be underestimated. Without a project plan, a project cannot be controlled. Controlling the project is exercised by comparing two points - the one where you are supposed to be, defined in the

project plan, and the one where you are at the moment. If there is deviation between these two points, corrective actions should be taken. /11/

At a minimum, a project plan should contain the

- Objective(s)
- Author
- Time schedule
- Realization
- Approval

But since a project plan is one of the fundamental documents in the project, it is recommendable to also include the

- Risks related to the project
- Resources needed
- Supporting operations
- Technology used
- Budget needed
- Constraints and dependencies
- Follow-up activities /5/

Making an estimate of the required work amount is one of the hardest things when drafting a project plan. The estimation can however be eased by dividing the project into as small tasks as possible and by comparing the project to previous, similar projects. Having several people perform the estimation and calculating an average estimate also increases the reliability of the truthfulness of the estimations. The main reasons for its problematic nature are

- *Complexity:* Information systems are complex by nature. If the problem to be solved is complicated, so is the solution.

- *Invisibility:* The current status of an unfinished information system is hard to tell.

- *Modifiability:* An apparently easy modifiability creates a pressure of change throughout the whole lifecycle of the project.

- *Uniqueness:* An IS project is always unique to some extent, which increases the risk of facing new and unpredictable problems.

- *Unscalability:* Old project methodologies might not work if the size or the tools of the project change.

- *Discontinuation:* Even if an information system works at a certain time in a certain environment, there is no guarantee that it will always work. /14/

2.4.2 Specifications in General

Specifications describe what the project should produce and how it should be produced in detail. In the definition phase, the project requirements and functionalities are specified in documents. These two documents describe the goals of the project in order to assure the necessity and feasibility of the project. /5/

Even if good specifications are the key to a successful project, it is still the weakest link in IS projects. When the project advances, there is a risk that the specifications will not be updated, due to a busy schedule and repeating modifications. If this happens, the specification gradually turns out to be completely useless. The characteristics of a good specification are

- *Perfectness:* Defines all the and only the required matters.

- *Accuracy and flawlessness:* Does not contain errors.

- *Understandable:* Easy to understand.
- *Testability:* Can be compared and verified unambiguously, if the project output is equivalent to the specification.
- *Traceability:* All functionalities can be traced all the way to their original requirement.

A specification, which fulfils all of the previously listed demands, is in practice impossible to produce as every specification contains errors, has ambiguities and lacks information. /5/

2.4.3 Requirements Specification

A requirement describes a single customer requirement that is necessary for the project output, in order to have a value for the customer. The requirements reflect the needs of both the customer and the users, providing a basis for a common understanding between the parties. /22/

2.4.4 Functional Specification

A functional specification describes how a requirement will be fulfilled in order to meet the requirements of the stakeholders /15/. It is a design document reflecting how the finished system will work. Its main task is to bridge the gap of comprehension between the different parties, being technical enough, yet understandable by parties not familiar with the coding language. It should also detail all the non-programming requirements. /10/

2.4.5 Test Plan

The test plan defines how the testing will be implemented and scheduled. It describes the various types and levels of testing to be done and explains the objective of each of the tests. It also defines the resources executing the tests, as well as the expected test results. /16/

2.5 Implementation Technologies

This chapter introduces the technologies on which the implementation of the system is based on.

2.5.1 XML

The extensive markup language is a standard for a document markup, endorsed by W3C. The data in XML documents are text strings that are surrounded by text markups, which describe the data. It is very flexible and extensible, allowing the developer to invent the elements and tags that they require and develop it in as extended a manner as required. It is simple and easy to read with any text editor and most importantly, it enables the possibility to truly cross-platform. /7/

As using the XML format for creating an interface between two systems, the permitted markups need to be decided and documented in a schema, to which the document instances can be compared to. The schema can be done with a specific XML schema language, of which the most commonly used is the W3C XML Schema language. Documents matching the schema are called valid. /7/

2.5.2 Web Service

A web service is a service available over the Internet, not tied to a certain operating system or programming language and uses a standardized XML messaging system. The architecture of a web service can be divided into four main layers. /3/

The service transport layer transports messages between applications. It supports the HTTP, SMTP, FTP and BEEP protocols, with HTTP being the most common protocol used to communicate over the WWW. /3/

The XML messaging layer encodes messages in a common XML format that is understood by both ends and it supports the XML-RPC and SOAP protocols /3/. Of these, the most commonly used is SOAP, a protocol based entirely on XML, making it uniquely platform- and language-independent. It is used for exchanging

information between computers, having its main focus on remote procedure calls transported via its most popular transport protocol, HTTP. SOAP is able to use both XML namespaces and XML schemas, and is considered to be a cornerstone of the web service architecture. As can be seen in figure 3, every SOAP message must have a root envelope element and a body, leaving the header and fault information as optional. /3/

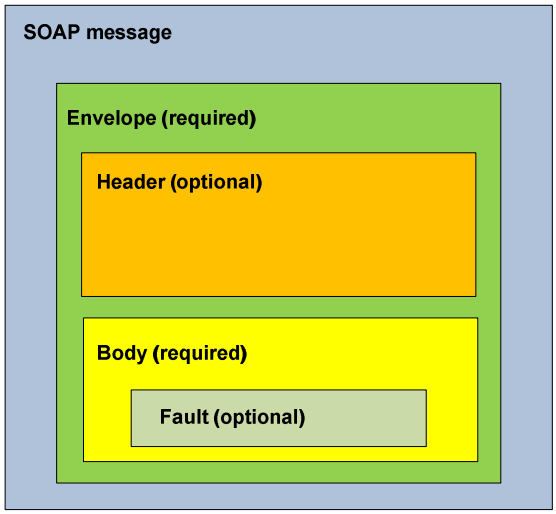

Figure 3. The structure of an XML SOAP message

The service description layer describes the public interface to a specific web service via WSDL /3/. WSDL is an XML based language used for describing and publishing a web service in a standard way, assuring that both the sender and receiver understand the process. WSDL typically uses one or more XML schemas to describe the data. The WSDL elements also contain a description on the operations to be performed on the data and a binding to a protocol defining how the message should be sent. By using WSDL, a client can locate a web service and invoke any of the public functions. WSDL is typically used with SOAP, thus the WSDL specification is equipped with a preexisting SOAP binding. /17/

The service discovery layer centralizes services into a common register, UDDI, from which they can be easily found. /3/ UDDI is an XML based registry providing access to WSDL documents, thus, a significant part of the web service

infrastructure. It is a service for publishing and searching the address for a web service, as well as information describing the business. A UDDI service can be private or public. The public UDDI service, which is available, is hosted by multiple vendors and is free of charge for all users. Private UDDI services are often used to store information on the internal web services of businesses. /17/

2.5.3 Network Security with a DMZ

In order to maintain the security of a company's internal network, no external systems or users should be able to access it. However, in many cases, it is essential from a business point of view to enable data to be uploaded or downloaded to/from an internal network via an external network, either by employees or by customers. One solution for this is to use a demilitarized zone (DMZ), where a server acting as an intermediary handles the traffic between an internal and external network. /13/

Any system that can be directly contacted by an external system or user should be placed in a DMZ, as these are the most likely to be attacked. A system located on a DMZ should have restricted access to systems on the internal network. Where possible, the internal systems should initiate the connection to the system on the DMZ instead of the other way around (Figure 4). To further improve the security, the web server contacted from the external network can be set to communicate with a third system that is housing the application that actually communicates with the database in the internal network (Figure 4). /13/

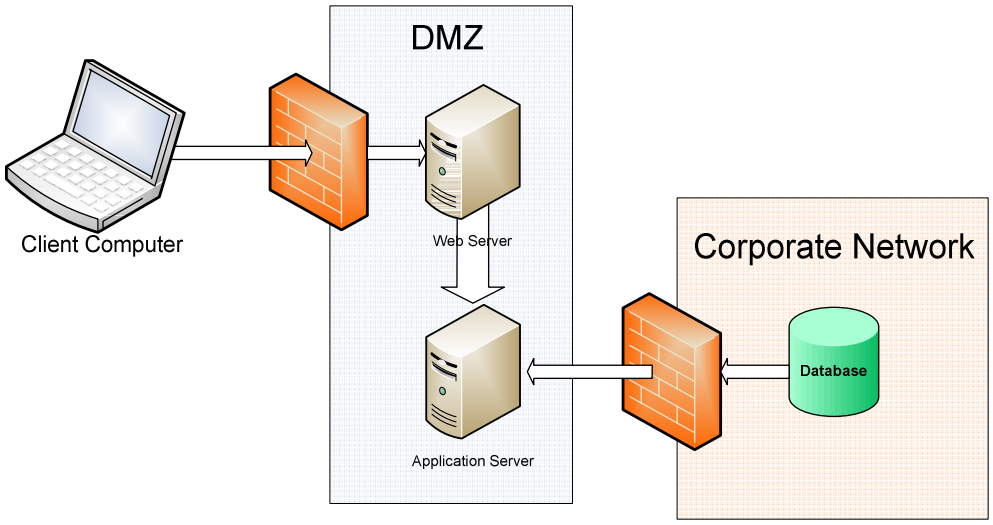

Figure 4. An example of a DMZ solution with dual-firewall architecture.

Additionally, dual-firewall architecture may be configured to the DMZ (Figure 4). In this architecture, the DMZ is isolated from both networks with two separate firewalls. The first one is set between the DMZ and the external network and the second one is between the DMZ and the internal network. This solution raises the overall security, since a single attack is unlikely to pass both firewalls. /13/

2.5.4 BPM Server

A Business Process Management (BPM) server facilitates the exchange of information between information systems running on different hardware and software platforms. It enables the automation of business processes and provides tools to design, develop, deploy and manage the processes. Nevertheless, most importantly, it offers a possibility to integrate processes to information systems both located within the organization and between other organizations, at the same time. Data within the BPM server is handled in XML format by a built-in message engine. /9/

2.6 Testing Method

The testing of a system or application is performed according to a test plan prepared during the definition phase of the project. The testing should be executed in an ascending order, moving from testing the smallest units possible towards testing the complete system from a user perspective. The testing of an information system can be divided in four main categories: /12/

Unit testing: concentrates on the smallest testable unit in a program or system. Each unit is executed in order to verify that it performs its assigned function. A unit can be, for example, a GUI component, functions, an online program or a stored procedure. Unit testing also makes it mathematically possible to fully test the code's logic using fewer tests. /12/

Integration testing tests the collaboration of the units as they have been integrated together as groups of units, modules. As every unit has already been tested during unit testing, the integration testing can only concentrate on module interfaces. When the development then evolves towards system testing and the modules are integrated together, incremental integration testing may also be performed. It is done by combining the modules in steps, where in-between, testing is performed concentrating on the newly added module. /12/

System testing focuses on the system as a whole, executed through performing the possible use case scenarios defined in the requirements of the system. System testing verifies that the functions are carried out correctly and the output meets the requirements. During system testing, non-functional characteristics should also be tested, such as usability, performance, stress tolerance and compatibility. /12/

Acceptance testing is a similar test to the system test, only that it concentrates strictly on the end user operation. This test is usually performed by the stakeholder of the system, especially if the stakeholder has not been involved in the development process or system testing. /12/

3 PROJECT INTRODUCTION

This chapter introduces the project, to which this thesis is related to and the case company, by whom this thesis has been commissioned.

3.1 The Case Company

This thesis has been commissioned by ABB's Distribution Automation unit in Vaasa. The following subchapters provide a brief presentation of the case company.

3.1.1 ABB Generally

The history of the ABB Group stretches all the way back to 1889, when the then called Strömberg was established. These days, ABB is a global leader in power and automation technologies and has its headquarters located in Zürich, Switzerland. The turnover of the ABB Group in 2008 was calculated to be as high as 34.9 billion US Dollars and having approximately 120 000 employees in over 100 different countries. In Finland, ABB employs over 6 650 people in 40 different locations, of which Vaasa and Helsinki are the largest. /1/

3.1.2 Distribution Automation

The Distribution Automation (DA) Business Unit operates in multi-national B2B business; developing, manufacturing, marketing and delivering protection and automation products for the distribution of electrical power all around the world. The main customers are electrical utilities, power intensive process industries and offshore operators, mainly ships and oil production. The Units are located in six different countries, with its centre in Vaasa, Finland.

The Unit in Vaasa employs around 260 people, consisting mostly of white-collar workers, leaving only a share of 8 percent to production personnel. It is the global market leader in selling protection relays, having over 90 percent of the manufactured products exported to more than 70 countries. The international

working environment is comprised of employees from 12 different countries, making English the common language. /2/

3.2 LCT Business Case in a Nutshell

The LCT concept provides a software infrastructure to automatically collect, store and provide the lifecycle relevant data of ABB products. One of its main objectives is to localize ABB products world-wide, but it also enables the usage of proactive service strategies and concepts, as well as provides an interface for market analysis. Thus, the LCT concept opens several possibilities for increasing the market share of ABB. /8/

3.3 Project Goals

The old architecture of LCT was dispersed in an unfavourable way for the case company, making the maintenance of the system troublesome and time consuming. In the old architecture, only data with a value at the Group level was stored in a usable format, leaving data with the value only at a business unit level unused. From a financial aspect, the old solution was also not as economical as it could be as one part of the architecture was installed on a server leased from a third party.

The goal of the project was to redesign the architecture of the existing data collection system. It was to be redesigned in such a way that it would operate in a new environment using new data transfer methods and at the same time, improve its internal logic to enhance its output. The architectural changes of the LCT were not to remarkably affect the LCT user experience. The transition to a new architecture was to be as smooth as possible, requiring no interaction of the user of the LCT.

3.4 LCT Cornerstones and Project Constraints

The LCT concept follows the product through its whole lifecycle, from its manufacturing to its decommissioning. All of the collected data should be stored into the Business Units Global database where it can be processed in order to be

served to the ABB Installed Base System, which acts as the frontend of the LCT. As a majority of product sales reaches the end-customers through channels, the final location of the product is often unknown at the time of manufacturing. The following subchapters give a more detailed introduction to the cornerstones of the LCT concept, which will remain as unchanged as possible during the execution of the project. Thus, also describing the main constraints set for the architectural changes made in the project.

3.4.1 The Data Collection Process

After the product has been manufactured, tested and shipped from the factory, the first data snapshot is collected to the LCT system through internal systems. This snapshot represents the original composition of the product. For the older products that do not have built-in support for automated data collection functionality, LCT, this often remains the only snapshot of the composition itself.

Now, as the product has left the factory and is out of reach, following its life becomes much more difficult. This is where the LCT kicks in. By connecting the product to a computer with the appropriate Configuration Tool installed, LCT updates can be sent either manually by the user or as an automated process running in the background of the Configuration Tool. If the automated alternative is selected, the software automatically notices any changes in composition, configuration, physical location and end customer information, which then triggers an LCT update.

3.4.2 The ABB Installed Base System

The ABB Installed Base System keeps track of all the information concerning ABB products and systems at a customer's site. The ABB Installed Base System should be kept up-to-date in a structured and consistent way, so that it can provide reliable information of the whole life cycle of a product or system. The main benefits of this information are

- the service history of a product can be easily obtained by service personnel

- the time-to-live -time of a product can be easily estimated by sales personnel
- the service- and installation tasks can be concentrated on a whole area at the time
- the reliability of a product can be measured easily.

By obtaining this information, the organization can provide their customers with a more proactive support. An offer of a service or alternatively a completely new product may be given to the customer before the customer even knows that it is needed. This kind of preventive maintenance not only strengthens the ABB brand, but also ensures a satisfied customer, as they manage to avoid a system failure that may have expensive consequences. /18/

4 ANALYSES AND PROJECT SETUP

The following chapter describes how the definition phase in the project was executed.

4.1 Use Cases

Possible use cases for the concept were clarified based on the customer requirements received in the project handover and the information obtained in the feasibility study phase of the project. Each use case was made up of a set of possible sequences of interactions between the system and the user. The use cases contain all the system activities that have significance to the user. A use case can be thought of as a collection of possible scenarios related to a particular goal.

It was discovered that the LCT concept had two main use cases. The first use case provides the ABB Installed Base System manufacturing information and the second use case provides it with updated information from the field. These two use cases are presented in detail in the next subchapters.

4.1.1 Factory Information to the ABB Installed Base System

This use case represents how the information of a manufactured-, and ordered product becomes available for ABB users (Figure 5). Even if the scope of the project included only factory information from Finland to be sent to the ABB Installed Base System, the global roll out was to be considered during the whole project.

As figure 5 shows, data from four different factories will be collected to the Global DWH. From there, the finalized data is replicated to the ABB Installed Base System via a Generic Interface. The data in the ABB Installed Base System is then accessible by all ABB users through its web based frontend. In the frontend, the users can view in addition to product- and module information, also site- and customer information as well as service events containing composition history.

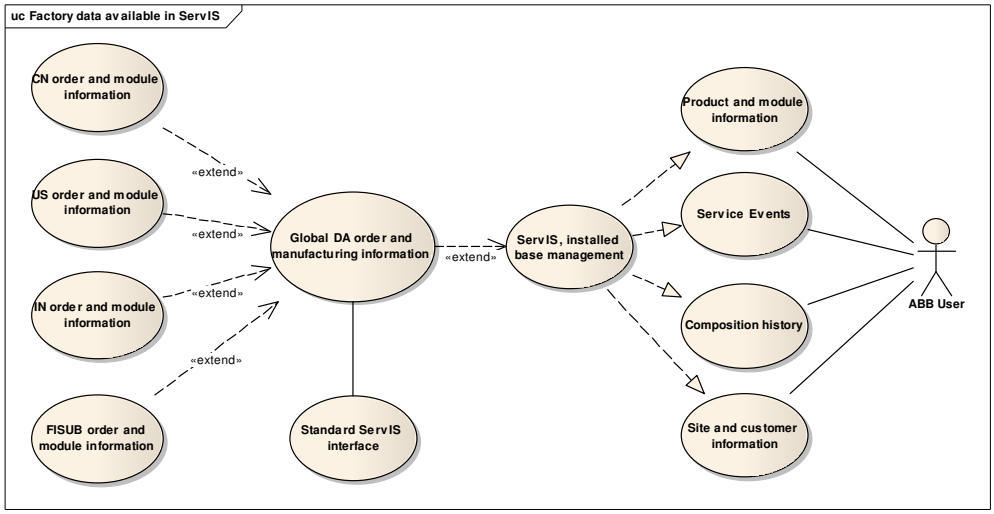

Figure 5. UC-1 Factory data available in the ABB Installed Base System.

4.1.2 Update Information to the ABB Installed Base System

This use case shows how the information of a change made to a product or its site becomes available for ABB users (Figure 6). This data is collected automatically from a product located on the customer's site, via the Internet.

This use case may be launched by three different types of changes in the product's information; customer- or site update, hardware- or software change or by a configuration change. As any of these changes occur, the data in the product is replicated and sent to the Global DWH, where the data is compared against the previous composition, collecting the noted changes into a service event. The finalized data is then replicated to the ABB Installed Base System via a Generic Interface. The data in the ABB Installed Base System is accessible by all ABB users through its web based frontend. In the frontend, the users can view in addition to product-, and module information, also site-, and customer information as well as service events containing the composition history.

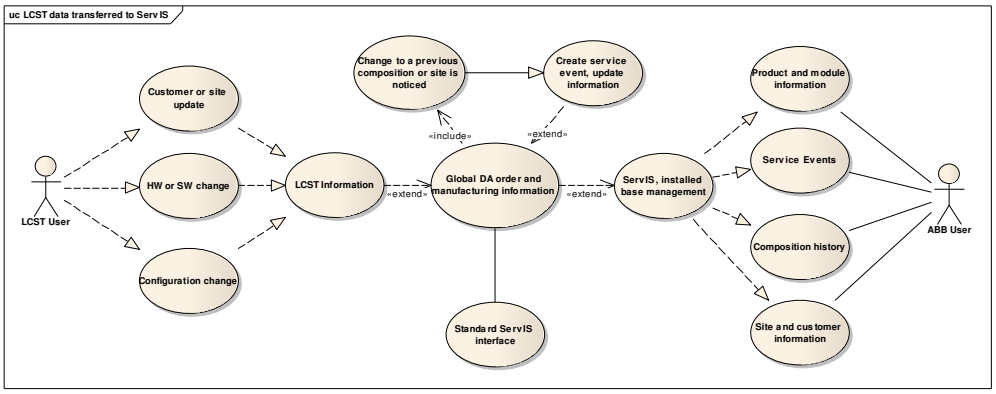

Figure 6. UC-2 Update data available in the ABB Installed Base System.

4.2 Project Plan

The planning of the project execution began as the use cases and the customer requirements for the project deliverables were clear and a preliminary architecture plan had been prepared. The project plan comprised of two different parts. The project plan, documented with MS Word and the activity plan, documented with MS Project. Both of the parts were verified by relevant stakeholders as they were done. The actual project- and activity plan for the project was produced as an outcome of this thesis.

4.2.1 Project Plan (MS Word)

The project plan was made according to a MS Word template provided by the case company. It contains all the information that is considered relevant for project members, stakeholders and other related people, such as the

- Objectives
- Milestones
- Financials
- Organization and roles
- Communication

- Deliverables and dependencies
- Assumptions and constraints
- Reference to the activity plan

4.2.2 Activity Plan (MS Project)

The activity plan contains the detailed chain of events, or tasks, to be executed in order to accomplish the project goals. It was made by first listing all the main tasks, which were then split into as small subtasks as possible, and finally set into a logical execution order. By providing the tasks with estimated durations and linking them together in MS Project, the estimated total duration for the project was obtained.

To obtain estimations that were as reliable as possible for the time required for the tasks, several people were used to make an estimate of which an average was taken. During the project, the activity plan was updated constantly according to the realization, in order to maintain control over the project progress. The progress of the project was followed up in review meetings, which were held on a biweekly basis.

4.3 Requirements

The main requirements of the project were given to the author in the project handover. The detailed requirements were prepared by the author based on constraints of the environment and ABB policy. The requirements have been documented on a template provided by the case company and the document has been approved by all relevant stakeholders. The requirements specification document was produced as an outcome of this thesis.

4.4 Functional Specification

The specification document provides a complete overview of the architecture's functionalities and the way it has been implemented, referencing to more detailed specifications on required areas. As the project was executed with an agile

approach, the functional specification of the architecture was documented along with the development of the architecture. As a result this document also covers the matters usually documented separately in a technical specification. The functionalities were documented on a template provided by the case company and the document has been approved by all relevant stakeholders. The functional specification document was produced as an outcome of this thesis.

4.5 Test Plan

The test plan for the implemented system has been prepared, based on the possible use cases of the system. As the case company did not have any template, it was documented on a template made by the author. If considering the lowest level in the test plan to be the smallest component possible to be tested and the highest level being the system as a whole, it can be stated that the test plan was designed to be carried out in a ascending order; moving from the smallest independent unit towards complete data acquisition processes. The test plan was produced as an outcome of this thesis.

5 DESIGN AND FUNCTIONALITY

This chapter introduces the design and functionality of the architecture, as well as the technologies used.

5.1 Three-Tier Architecture

The architecture was designed as a three-tier architecture consisting of a backend, a middle layer and a frontend (Figure 7). Contrary to standard practices, in this context as Backend is considered the data entry points, as Middle Layer the data storage and processing of the data and as Frontend the GUI that brings finalized data to the user. All these tiers consist of independent systems integrated to one another by a BPM server, making them work together seamlessly as if they were one big system.

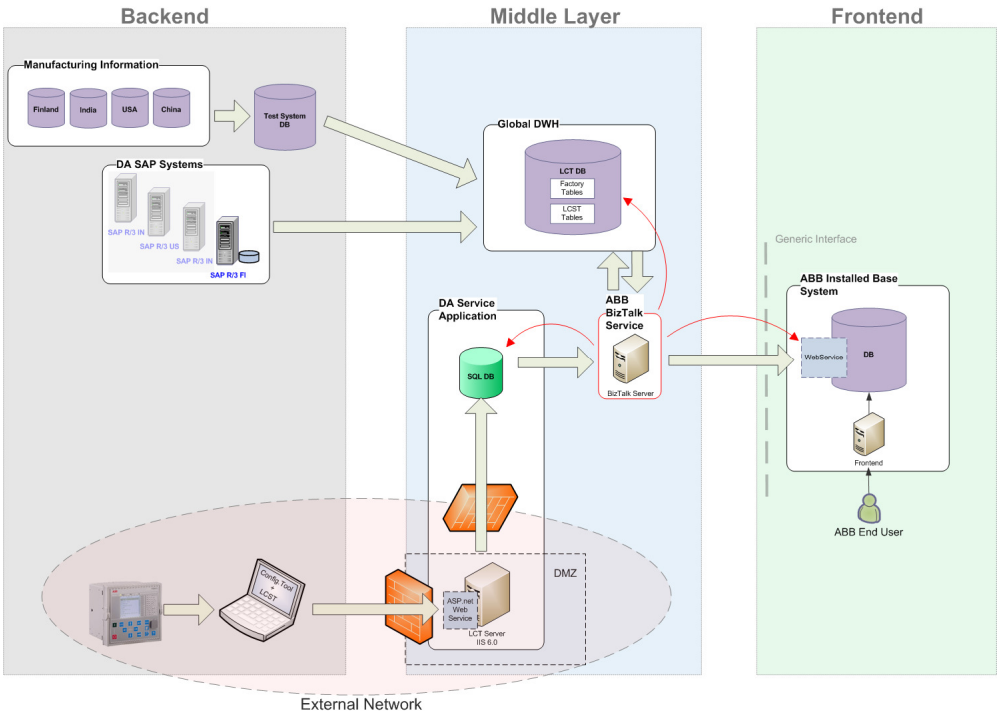

Figure 7. The architecture was designed as three-tier architecture.

5.2 Backends

The Backend is where the data is entered into the LCT system. In the LCT concept, the data is entered through two different entry points; the first snapshot is collected from the factory and the updating snapshots are collected from the Intelligent Electronic Device (IED) itself.

5.2.1 Order and Manufacturing Information

The first data snapshot of an IED is entered to the LCT system's Backend as the IED has been shipped to the customer. This first snapshot, consisting of the original composition of the device, is compiled of two different data sets. The first part consists of ordering information collected from the ERP system and the second part of manufacturing information collected from the Test System. As the architecture has been stated to be stable in production use and rolled out globally, these snapshots will be collected from all DA factories and stored to a common DWH, the Global DWH.

The data for the first snapshot is collected by predefined SQL stored procedures, which verifies that the data for the IED is available in both the above-mentioned sources before replicating it to the LCT database in the Global DWH. As this had already been set up earlier for the old architecture, it required no other actions by this project other than the updating of the data content replicated to the LCT database.

5.2.2 IED and Configuration Tool

In order to collect updating snapshots from an IED, it needs to be connected to a computer with a network connection and the appropriate Configuration Tool software has to be installed. This Configuration Tool automatically polls for changes in the device and can be set to send updates to the LCT system either automatically, without any interaction by the user, or to be done manually by the user. The updating snapshots can only be collected from newer IEDs, which have a built-in LCT functionality. The functionality of the IED's built-in LCT software,

as well as the functionality of the Configuration Tool software, was to be left unchanged, as they were out of scope of this project.

The hardware interface between the IED and the Configuration tool is established via either a serial or Ethernet connection. The software module in the Configuration Tool that sends the data snapshot to the LCT system, called LCST, uses the following software interfaces for communicating to the destination server in the Middle Layer:

- *LCST – Web service on DMZ:* SOAP messages over the Internet, in other words XML over HTTPS. Used to verify and register the source of the data.

- *LCST – Update File storage on Server Machine:* HTTPS over Internet. Used to remotely update the LCST setting if required.

- *LCST – LCT server on DMZ:* A TCP socket opening over the Internet. Used to transfer the data snapshot to the LCT system.

Since the functionality of the Configuration Tool was not changed by this project, the interfaces between the Backend and the Middle Layer were to remain the same. /20/

Before uploading data to the LCT system, the data is encrypted with a public RSA encryption key. As the LCST then initiates a connection to the web service in the LCT system's DMZ located in the Middle Layer, its unique ID is verified. In order to successfully upload the data consisting of a binary- and an XML file, the XML file also needs to pass its verification criteria. HTTPS is used as a network transport protocol for sending data through the external network.

5.3 Middle Layer

The Middle Layer is where all the logic is located and where the data is collected and stored. It receives its input data from the Backends, processes it and forwards the finalized output data to the Frontend. As already described in chapter 3.4, this

is mainly the tier the architectural changes of this project concerns, as the Frontend and Backend was to remain unchanged as far as possible.

The Middle Layer consists of three different systems serving different needs, in order to fulfil its tasks. These systems are the Global DWH, containing all the data, a BPM server taking care of the data processing and data transfers, and a DMZ server with a web service enabling data to be received from the external network.

5.3.1 Global DWH

The Global DWH is the master database of the LCT, where all the data is stored into a relational SQL database, organized in a logical way for reuse. It also contains SQL stored procedures providing the predefined database queries required to process the data, which are then called by the BPM server. The data from the Backends is stored into two different sets of tables in the LCT database, one for each Backend.

The stored procedures verify that the IED sending the updating snapshot has actually been manufactured, notices changes made since the last update, stores data received from the DA Service Application into the database as well as provides required data to the BPM server.

5.3.2 DA Service Application

This is the data collection server on a DMZ that receives data from an external network. It acts as an intermediary between the Backend, located in the external network, and the Global DWH. The project only covered the development of a dedicated database as well as a modification of the data collection server to work against a different vendor's database software. The data collection server's functionalities were already designed and developed earlier for the old architecture.

This application consists of an XML based web service implemented with .Net/C# programming language, as well as a dedicated database and some SQL stored

procedures. The web service calls for two different methods depending on the case. The first one is for registering a new LCST as it connects for the first time and the second one is for receiving data uploads. The database contains one table for each method, one for registered LCSTs and one for uploaded data. The stored procedures then integrate the functionality between the web service and the database as well as provide data to the BPM server as it is being called.

As a data upload is received, the sender is verified and the data content is decrypted with the private RSA encryption key. If the uploaded data passes these security checks, it is stored in the database for the BPM server to pick it up for processing.

5.3.3 BPM Server

The BPM server is what integrates all the systems within the Middle Layer together, making it a fully operational system. As the system has two different business processes, it thus also has two orchestrations, one for each Backend. Both, the input and output data of the BPM server is in XML. Microsoft BizTalk Server was used as the integration software.

The first orchestration serves the Backend providing the first data snapshot, introduced in chapter 5.2.1. The tables in the Global DWH specified for this snapshot, are being polled by the BPM server by calling an SQL stored procedure with a certain time interval. As the stored procedure detects a new entry, it provides it to the BPM server as a return result. The BPM server then organizes the new entry into its internal memory according to an XML schema similar to the table structure. From this schema, the data is mapped to another XML schema, which is used by the system providing the Frontend, and uploaded as an XML message enveloped in SOAP format to the Frontends web service. HTTP is used as the transport protocol. The return result received is stored by calling a stored procedure that stores the result to the original line of entry in the LCT database in Global DWH, to for it specified columns.

The second orchestration serves the other Backend that provides the updating snapshots, which were introduced in chapter 5.2.2. As the uploaded data is stored in the DA Service Application's database, it is picked up in the same way as in the first orchestration; by calling a stored procedure with a certain time interval. When the stored procedure detects a new entry, it provides it to the BPM server as a return result, making it possible for the orchestration to move on to the next process. As the composition of the IED is in this snapshot provided as an XML file, the new entry is organized into the BPM server's internal memory according to an XML schema similar to the structure of the XML file.

After that, the BPM server calls for a stored procedure in order to notice added and removed modules by comparing the newly obtained data snapshot to the previous updating snapshot, or preferably to the first snapshot. This procedure also verifies that the IED has really been manufactured, as if no previous snapshot is found for comparison, the execution of the orchestration is cancelled and no data is stored in either the Global DWH or the Frontend. As the BPM server has gathered all the required data it stores it to the LCT database in the Global DWH, to the tables specified for this process.

Yet, the BPM server still keeps the data in its internal memory. In order to provide the Frontend with complete snapshots, instead of incremental ones, an SQL stored procedure is called. This stored procedure picks up such data that is not provided in the updating snapshot, from the first snapshot stored in the Global DWH. This data is then used to fill empty elements in the XML schema, as the data is mapped to the schema provided by the Frontend. Finally, the data is then uploaded as an XML message enveloped in SOAP format to the Frontends web service using HTTP as its transport protocol. The return result received is stored by calling a stored procedure that stores the result to the original line of entry in the LCT database in Global DWH, to specified columns.

5.4 Frontend

The Frontend is the tier that brings the output of the system to the user via an application layer. The Frontend in the LCT system is based on an existing information system, the ABB Installed Base System. The system is developed using Oracle tools and it is already integrated with other vital ABB systems required for user access management, data clean-up and product identification. The data in the ABB Installed Base System is organized according to a logical tree structure hierarchy.

The objectives of the project regarding this tier, was to integrate the Middle Layer to the Frontend as seamlessly as possible. The Middle Layer was to act as a slave to the Frontend, providing it with finalized and the most up-to-date data through its Generic Interface. With the Frontend system already being widely used, it was able to provide complete user management and a GUI for the LCT system.

5.4.1 The ABB Installed Base System's Frontend (GUI)

The ABB Installed Base System's frontend runs inside a Web Browser, providing easy access from any computer inside the corporate network. It has several drill-down dimensions, providing the user with the possibility to choose a dimension fitting him/her the best, as moving between levels of the hierarchy when viewing data. The data in the dimensions is organized in a logical tree structure. This way, the requirements of different users can be better met, the target group of the ABB Installed Base System's frontend being extensive.

5.4.2 The ABB Installed Base System's Generic Interface

The ABB Installed Base System's Generic Interface has been deployed with the SOA approach. All of the services that it provides to external systems are implemented as web services. The data exchange is realized by web services implemented on the sending and/or receiving system. As the data exchange format, it uses an XML structure defined in an XML schema. The new LCT architecture has been designed considering integration to this interface, as it was one of the fundamental requirements of the project.

As this project was an early adopter of this interface, many problems arose during the project progress. Being a global IS tool, any changes to the ABB Installed Base System's data model or its Generic Interface affect all the other systems using it. Thus, achieving satisfactory solutions required both teamwork and flexibility from all parties. During the integration, it was striven to follow the general rule and best practice in XML data exchange, to be as restrictive as possible on the sender side and as tolerant as possible on the receiving side.

At this stage, the updating of the data in the ABB Installed Base System can only be done incrementally. Extending the system to accept complete updates was considered too time consuming, delaying the release of the new architecture even further. However, it might be extended to support this functionality at a later point.

6 TESTING

This chapter describes how the new architecture was tested as the development work of the project was finalized.

6.1 Unit Testing

The unit testing was executed in parallel with the development of the system, by the developer. This way, most of the flaws were detected and fixed, in as early a state as possible. In order to pull off such an agile testing method successfully, the developer was to be kept as up-to-date as possible on the systems requirements, restrictions and dependencies.

6.2 Integration Testing

As a major entity of the system was complete, its functionality was integration tested. This was done in order to verify that all the units worked seamlessly together and that the entity provided its expected output. As the LCT system is based on several already existing systems, a representative from every system included in the entity to be tested, needed to participate to the testing.

The testing was first executed in the development environment of the system. As the system was moved to its testing environment, the functionality of the main entities were double checked by rerunning the integration tests to avoid drawbacks during system testing. The testing was done with a simple application simulating an IED, as the Backends were not yet integrated to the Middle Layer at this point.

6.3 System Testing

This was the final test run to the system before moving it to its production environment and going live with it. In this test, the system was tested as a whole, simulating all possible use cases that the system may face during its lifecycle. The system's compliance with its requirements was evaluated and all bugs were documented in order to be fixed, either before going live or at a later point through an updating release.

The complete and integrated system was tested in both the test environment and the production environment. This way, all flaws could be revealed as early as possible, still not leaving the impact of a different hardware environment unobserved. The testing was done with a real IED, to which real hardware and software changes were made, validating the output of the system from the Frontends GUI.

6.4 Acceptance Testing

The acceptance testing of the system will be carried out in two phases. The first phase is a demo meeting. During this phase the functionality of the system is described to the stakeholders. The second phase is a user survey, which will be held after the system has been running for a while. After both of these phases have been completed successfully, the system will be rolled out globally.

7 CONCLUSION

This chapter contains the reflections of the author, regarding the project as well as reviews the results achieved at the time of writing this thesis.

7.1 Personal Review

After managing the project for 9 months and in parallel writing this thesis, an end is finally in sight. The project turned out to be more challenging than estimated, extending the original project running time by more than 200 percent. This delay also affected the completion of the thesis, as I was occupied working with the project itself and as I wanted to depict the architecture as close to the final implementation as possible in this thesis.

On the bright side, my lack of experience regarding project management that I had at the beginning of this project is surely gone for good. If everything would have gone exactly as planned, which almost never happens in IS projects, I would now lack many experiences. In fact, I feel as if I have learnt more along this project than during the four years I spent in school, preparing myself for such demanding and responsible tasks.

Even though I usually tend to downplay my efforts, I can proudly say that I have dedicated myself completely to this project, sacrificing all other interests in order to complete the project and thesis with honours. Yet, it would not have been possible without the great team spirit within the project team group and the support of my other co-workers that have helped me reach my goals.

7.2 Review of Results

At the time of writing, the project is still in its testing phase, where the system is being tested and finalized in order to be released. Considering the circumstances, I think the goals have been achieved rather well despite the delays caused by the constraints that occurred during the project. The most important aspect, in my opinion, is that the new architecture has really been designed with a thought of a future-proof foundation. To obtain this, looking the other way, even when facing

the smallest problem, is never an option. The main challenges that I met during the project were the following:

- Matching the inconsistencies in the LCT's and the ABB Installed Base System's data models, of which some were not discovered until the project had progressed to a certain point.
- The ABB Installed Base System requiring more accurate data than the LCT system could provide.
- Lack of functionalities in the ABB Installed Base System's Generic Interface in order to support LCT.
- Restrictions set by the security policy of the case company.
- Encryption methods to be used in sending/receiving data via external network.
- Resource problems, especially during the holiday seasons.
- LCT's dependency on data models used in the Backends, as they were designed for the old architecture.
- Project definition phase was concluded based on inadequate information, as all use cases had not been identified.

If I could start the project over, there would not be much I would do differently, as I have really pushed myself to the limit on this one. Nevertheless, what I would do differently is the project's definition phase, which I would try to do even more thoroughly. I cannot emphasize enough how important the definition phase in a project is. Making the smallest mistake in this phase might ruin the whole project. Still, it is probably the most underestimated phase in a project.

Overall, I think the LCT system's functionality is only at a grass roots level of what it could be. I however truly believe that the new architecture of the LCT

concept is now on a much more solid foundation, ready to be developed to the extent it has once been intended to.

7.3 Improvement Proposal

In order to improve the system even further, I have prepared a list of tasks that could be implemented in the future to obtain even more reliable data content and to further extend the benefit of the system. However, it is not included in the public release of this thesis due to the classified content owned by the case company.

LITERATURE FOR THESIS WRITING

/1/ ABB Group 2009. Group's global www home page [Referenced 8.9.2009]. Available in world wide web at: <URL:http://www.abb.com/cawp/abbzh251/76465d8d53273699c12571920030dbef.aspx>

/2/ Ahonen, Tiina 2009. General Presentation of Distribution Automation, Finland 2009. ABB Group. Power Products division Medium Voltage/Distribution Automation business unit of Finland. Microsoft Power Point presentation. Available for ABB employees in internal corporate network.

/3/ Cerami, Ethan 2002. Web services essentials. 1st Edition. Sebastopol, CA. O'Reilly & Associates, Inc.

/4/ Fowler, Martin 2000. The New Methodology. Web article released in July 2000. Updated in December 2005 [Referenced 24.11.2009]. Available in www-form: <URL:http://martinfowler.com/articles/newMethodology.html#FromNothingToMonumentalToAgile>

/5/ Haikala, Ilkka – Märijärvi, Jukka 2004. 10. painos. Ohjelmistotuotanto. Helsinki. Talentum.

/6/ Heerkens, Gary R. 2002. Project Management. Wisconsin. CWL Publishing Enterprises.

/7/ Harold, Elliotte Rusty, Means, W. Scott 2004. XML in a nutshell. 3rd Edition. Sebastopol, CA. O'Reilly Media, Inc.

/8/ Kalliomaa, Tero 2006. Life Cycle Traceability for Medium Voltage Products - General Presentation. ABB Group. Updated in June 2007 by Suovarinaho, Mika [Referenced 31.1.2010]. Available for ABB employees in internal corporate network.

/9/ Kurbel, Karl Eugen 2008. The Making of Information Systems: Software

Engineering and Management in a Globalized World. Berlin. Springer.

/10/ Lecky-Thompson, Guy W. 2005. Corporate software project management. Massachusetts. Charles River Media, Inc.

/11/ Lewis, James P. 2007. Fundamentals of Project Management. 3rd Edition. New York. AMACOM.

/12/ Lewis, William E., Veerapillai, Gunasekaran 2005. Software testing and continuous quality improvement. 2nd Edition. Boca Raton, FL. Auerbach Publications.

/13/ Maiwald, Eric 2003. Network security: a beginner's guide. 2nd Edition. Emeryville, CA. McGraw Hill Companies.

/14/ Markkanen, Jouko 1999. Ohjelmistotuotanto ja uusmedia. Tietovalta. Microsoft Power Point presentation [Referenced 3.8.2009]. Available in world wide web at: <URL:http://users.evtek.fi/~erkkir/multimedia/markkanen.ppt>

/15/ Muller, Robert J. 1998. Muller Productive objects: an applied software project management framework. San Francisco. Morgan Kaufmann Publishers, Inc.

/16/ Murch, Richard 2001. Project management: best practices for IT professionals. Upper Saddle River, NJ. Prentice Hall, Inc.

/17/ Newcomer, Eric 2002. Understanding Web services: XML, WSDL, SOAP, and UDDI. 3rd Printing. Indianapolis, IN. Addison-Wesley Professional.

/18/ Petersen, Heiko 2005. ServIS Requirements Specification. ABB Group. Updated in June 2005 [Referenced 3.8.2009]. Available for ABB employees in internal corporate network.

/19/ Saaksvuori, Antti – Immonen, Anselmi 2005. Product lifecycle management. 2nd Edition. Berlin. Springer.

/20/ Saari, Kari 2006. Functional Specification for Data Collector. ABB Group. Updated in November 2006 [Referenced 30.1.2010]. Available for ABB employees in

internal corporate network.

/21/ Stark, John 2005. Version 1.7. Making Progress With PLM: A Manual to support PLM Initiatives and PLM Projects. A Manual that can be bought via the world wide web at: <URL:http://www.johnstark.com/prgrs.html>

/22/ Young, Ralph Rowland 2006. Project requirements: a guide to best practices. Vienna. Management Concepts, Inc.