Design and Implementation of Requirement Handler over the Farkle and Optima

ARGHAVAN MAJIDI

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2014

TRITA-ICT-EX-2014:30

Design and Implementation of Requirement Handler over the Farkle and Optima

Arghavan Majidi

August 2013

Supervisors:
Fredrik Kilander, KTH
Barbro Claesson, ENEA
Detlef Scholle, ENEA

Abstract

This Master thesis is a part of an ongoing project called industrial Framework for Embedded Systems Tools, iFEST. IFEST is an EU founded project for developing a tool integration framework in order to facilitate both hardware and software co-design and life-cycle aspects for the development of embedded systems. This leads to reducing the engineering life-cycle costs and time-to-market factor for complex embedded system projects.

Test Manager is a part of the testing framework that invokes some test cases for testing functionalities of other components of the system which are “Optima” and “Farkle”. When the test is done the Test Manager will get the test results from the system and return it to the tester user. The final implementation and integration of Test Manager is not within the scope of this thesis work. However, a pilot version of Test Manager was implemented as a working prototype to get stakeholders’ feedback and validate the initial requirement and design. After iterating on requirement factors and finding criteria for the optimum design, different design alternatives went through an AHP[1] decision making process to come up with an ideal design model.

The aforementioned process is followed by four different aspects of our design model; the integration models, the choice of programming language, the choice between web and desktop user interface, and the choice of database system. For each of these four choices, different options are presented during in the literature study. The final design model is the outcome of the AHP analysis.

[1] *Analytic Hierarchy Process*

Sammanfattning

Detta examensarbete är en del i ett pågående projekt som kallas industriell ram för inbyggda systemverktyg: iFEST. IFEST är ett EU-grundat projekt för att utveckla ett ramverk för verktygsintegration i syfte att underlätta samtidig design av både hårdvara och mjukvara, samt livscykelaspekter för utveckling av inbyggda system. Detta leder till att minska de tekniska livscykelkostnaderna och time-to-market i komplexa projekt för inbyggda system.

Test Manager är en del av en testningsram som anropar tester för att testa funktionerna i andra komponenter i systemet, som "Optima" och "Farkle". När testet är gjort kommer Test Manager att få testresultaten från systemet och returnera dem till den mänskliga testaren. Den slutliga genomförandet och integrationen av Test Manager är inte inom ramen för detta examensarbete. Emellertid har en pilotversion av Test Manager implementerats som en fungerande prototyp för att få intressenternas synpunkter och validera ursprungliga krav och design. Efter iteration av kravfaktorer och sökande efter kriterier för optimal utformning, gick olika designalternativ genom en AHP-baserad beslutsprocess för att komma till en ideal designmodell.

Den tidigare nämnda processen följdes av fyra olika aspekter på designmodellen; integrationsmodeller, valet av programmeringsspråk, valet mellan webben eller särskilt användargränssnitt, och valet av databassystem. För vart och ett av dessa fyra aspekter, presenteras olika alternativ i litteraturstudien. Den slutliga utformningen av modellen är resultatet av AHP-analysen.

Contents

Content of figures

Content of tables

Chapter 1

Introduction

1.1 Background

This master thesis was conducted in collaboration with Royal Institute of Technology (KTH) and ENEA [1]. ENEA is an international information technology company that is famous for developing an operating system known as OSE[2]. This thesis work was part of the design and implementation of an industrial Framework for Embedded Systems Tools in an EU funded project called Artemis/iFEST [3] [2] within ENEA. IFEST is a research project approved by the European Union in 2009/2010. The main purpose of iFEST is to develop a tool chain framework for embedded systems in order to decrease the time and cost of development by up to 20% in the engineering life cycle.

ENEA has already developed a couple of tools including a debugging tool called “Optima”, and a test and verification tool called “Farkle”. The purpose of the whole system is to develop an application that integrates several development tools such as Farke and Optima. This thesis is the continuation of a previous thesis work by Daneil Digerås [23]. In the previous work, the primary focus was on the integration of Farkle and Optima, and the hardware devices. Eclipse was used as the supporting platform. The components of the project were implemented as Eclipse plug-ins.

[2] Operating System Embedded

[3] industrial Framework for Embedded Systems Tools

1.2 Problem Statement

The main focus of this thesis work was to choose a collection of technologies and tools that met the requirements and delimitations of the project. These decisions were considered as a pre-study for a final product. The final product was to be implemented as an application which facilitates the testing process for testers. The implementation of the Test Manager application with its database, intermediate layer and the user interface had to follow the stakeholder's requirements and delimitations. The suitable features and attributes that were derived from the requirements and delimitation are discussed in next chapter.

The questions addressed in this thesis are:

Q1. What are the suitable technologies for implementing the three different layers of the application according to the requirements and delimitations? Technologies include language programming, database management systems, and user interface tools.

Q2. What are the alternative integration models in order to integrate Test Manager with the rest of the system? Which alternative method should be used with respect to the requirements and delimitations?

1.3 Purpose

The main purpose of this thesis is to find a series of optimum technology choices in order to make the application design model as efficient as possible with respect to the company's requirements and limitations. The relevant layers for technology selection are: the user interface, business logic and database. The final product is meant to provide the tester with information about the test specification and the test result.

One of the most important steps in the implementation phase of the application is to develop a working prototype of the Test Manager that operates as an interface between the system and the user. The test Manager runs the test cases and shows the results to the user. Another component to be developed was a database to store the results of the test cases for future analysis and data mining purposes.

1.4 Method

The thesis consists of two different parts. First is the theoretical part including the literature study and design of the Test Manager software, and the second part is the practical part during which I implemented the pilot version of the Test Manager. The theoretical part includes a literature study on previous works within the iFEST project and the previous thesis work that needs to be continued in the current thesis work and also brainstorming sessions that were held to determine the relevant properties of the suitable technology with respect to the projects requirements. The outcome of meetings is presented in Appendix 4.

This work also contains some research on different programming languages, alternative development environments and tools, and integration options for the rest of the iFEST components. Additionally, an investigation into requirement evaluation techniques and decision making processes was required in order to make sure that the final design met with the original requirements. The theoretical part's outcome provided a basic definition of requirements and design, which was implemented in a prototype version in the development phase, followed by an integration of the application with the rest of the system. Then validation of the implemented specifications was checked against the original requirements in a presentation of the Test Manager prototype. Continuous improvement of the software quality was achieved by getting feedback from multiple presentation meetings throughout the development of the Test Manager prototype. After presenting the Test Manager prototype to the stakeholders, the initial requirements were validated and the design was reviewed. Based on their feedback and ideas, the ultimate design decisions were made. Decisions included choosing the proper technology for the database layer, intermediate layer and the user interface layer, as well as choosing the proper integration model. All of these decisions went through the proper decision making process.

1.5 Delimitation

ENEA considers a few options of technologies and tools to be used in each layer that could be compared with each other. For example, for the business logic layer which interacts with previously implemented software, Java and Python are considered. For the database layer Text File, MySQL, Casandra and CouchDB were considered as alternative options.

1.6 Thesis Overview

You find in this chapter as an introduction of the iFEST project, as a bigger project which Test Manager is a part of. And also the way Test Manager relates to other components of iFEST. So the scope and functionality of Test Manager and its interfaces with other components of the project are described in this chapter. In addition to that, the scope and the phases of the thesis work is also discussed. Phases include: requirement handling process, implementing the prototype project, reviewing the prototype with the stakeholders, and decision making process for the optimum technologies to be used in the final design model. Problem statement is introduced as choosing proper technologies and tools to meet requirements and delimitation from the stakeholders, as well as finding out proper level of integration that Test Manager should have with the rest of the component within iFEST. Later the methodology for performing the thesis work phases and some delimitation as initial options for technology choices is introduced.

In chapter two methods that help for decision making process are introduced. According to the type of problem and availability of information, we can choose different decision making methods. By considering the type of information that is available, 13 practical methods are described. I also described categories are suitable for problem statements in my thesis and finally selecting the best method which is Analytic Hierarchy Process.

Chapter three was started with an introduction to iFEST and the scope of this thesis work within the iFEST project. Previous works that are done in the project is fully described and also the components that our Test Manager needs to be interfacing is described to have a clear understanding of the role that it will have among other components of the system. Then I described a series of problems that needs to be addressed by making the right decision about them. These problems are mainly choosing the right method for design the application.

User interface technology as the firs problem is a decision between desktop applications of web applications. In this section I described the advantages and disadvantages of each separately. When it comes to programming languages I compared Java and Python. The choice of data storage system is the third problem to choose between text file, SQL based, No-SQL, document-style storage, key-value storage, graphical databases, or column based storage systems. The other issue was to find the best way to integrate Test manager with the rest of the components. To make the integration process as optimum as possible I described

different integration levels like: platform integration, presentation integration, data integration, process integration and control integration. For each of the problems I described different alternatives with pros and cons. For example Data integration level, as a problem, is one of the design decisions that are made in designing the test manager application. In chapter three I described difficulties regarding connectivity with data sources; interoperability, Non-redundancy, data consistency, data exchange, and Synchronization. Interoperability concerns about the data to have a coherent form when accessed by different tools. Non-redundancy of a data source has direct relation with having a high level of normality in the data source. Data consistency is important when we expect no inconsistencies in the data source even if the data is manipulated by several clients. Data exchange is important when tow live data bases needs to be in sync with each other so the data format that is exchanged needs to be understandable for both of them. And finally synchronization talks about the ability of the database system to be synchronized with other instances of the data that are working offline

In chapter four I present decisions made out of those alternative that were presented in the previous chapter. To make the decision I built a hierarchical decision trees for each decision. Such trees span from main objective from the first level to specific criterion in the second level. this tree is used in the AHP process to make the best decision according to mutual comparison of criterion. To make the mutual comparison we sat with stakeholder people to get their feedback on how do they priorities different criterion by mutually comparing criterion one by one.

In chapter five I discussed the optimum architecture design of the future implementation of Test Manager, by providing UML diagrams of the high level architecture and design. The Model, View, Controller, MVC, model is adopted as the main architectural design pattern. At the same type I used snapshots of the working prototype of the application to give a feeling of how it will look.

In last chapter I answered to the main research questions that I had in the beginning of the thesis work. Brief answer to the questions is that a set of suitable technologies to be used as programming language, database and user interface is selected based on iFEST requirements and constraints. Additionally five different levels of integration which are platform, presentation, data, process, and control were discussed as well as their advantages and disadvantages. Thereafter, the order of importance of comparison criteria was found out according to on the company’s requirements and constraints. As the future work the design

of the actual implementation of Test Manager could be a continuation of this thesis by following the ideal technologies which already discussed.

Chapter 2

Methodology

When it comes to design decisions, I had to take many factors and criteria into account and know the weight of each factor and how it affected the overall solution. To make sure that I followed a systematic approach I performed a literature study on decision making processes to find out the right method for comparing alternative design models using suitable criteria.

Decision making is one of the most challenging issues for managers and organizations. The number of criteria, and the various attributes that describe them, along with the importance of considering all these aspects simultaneously, makes the decision making process more complicated. Therefore, one needs a clear understanding of what the problem is and what are the effects of various attributes in advance in order to make effective decisions. Moreover, complicated decisions require a systematic approach.

Design decisions are critical for the success of a software project in the process of developing an information system, and are great examples of the decisions that need a systematic approach to be solved. The more information we have, the less uncertainty we will have while making design decisions. The kind of information that we gather from the business domain of the information system plays a significant role in making the right decision, which directly affects the level of validation and verification of the system. With more complete and up-to-date data, better decisions are possible.

This section considers the methods that were used in the decision making process of this thesis concerning the selection for four design aspects: the level of integration, the kind of user interface, the programming language, and the kind of database.

In order to select an option, we need to consider a number of criteria at the same time. Multi Criteria Decision Making (MCDM) is one branch of study which concerns the aforementioned circumstances. MCDM is defined as “the study of methods and procedures, by which concerns about multiple conflicting criteria can be formally incorporated into the management planning process” [3]. MCDM is divided into a two main groups, Multiple Objective Decision Making (MODM) (Yoon 1980 [4]) and Multiple Attributes Decision Making (Saaty.T 1980). MODM considers multiple goals in solving a problem and is more suitable for situations where we need to make a decision to satisfy multiple goals at the same time. MODM tries to find an optimal solution using mathematical methods to satisfy various constraints. When it

comes to MADM, there are multiple alternatives among which we just want to choose one. In this method we do not have an objective function as in the MODM approach. MADM approaches are categorized in different ways. In a general categorization of MADM approaches, the decision model with one goal and different parameters is divided into three types:

1. Certainty models where all parameters are specified.
2. Risk models where parameters are not specified but they follow an accumulative distributed function, and are possible to estimate.
3. Uncertainty models where we have no information about the parameters and accumulative distributed functions.

Hwang and Yoon (1981) in detail categorize a series of 17 methods by considering type and important features. A modified taxonomy of 13 of 17 which is more practical is shown in figure 2.1 [4].

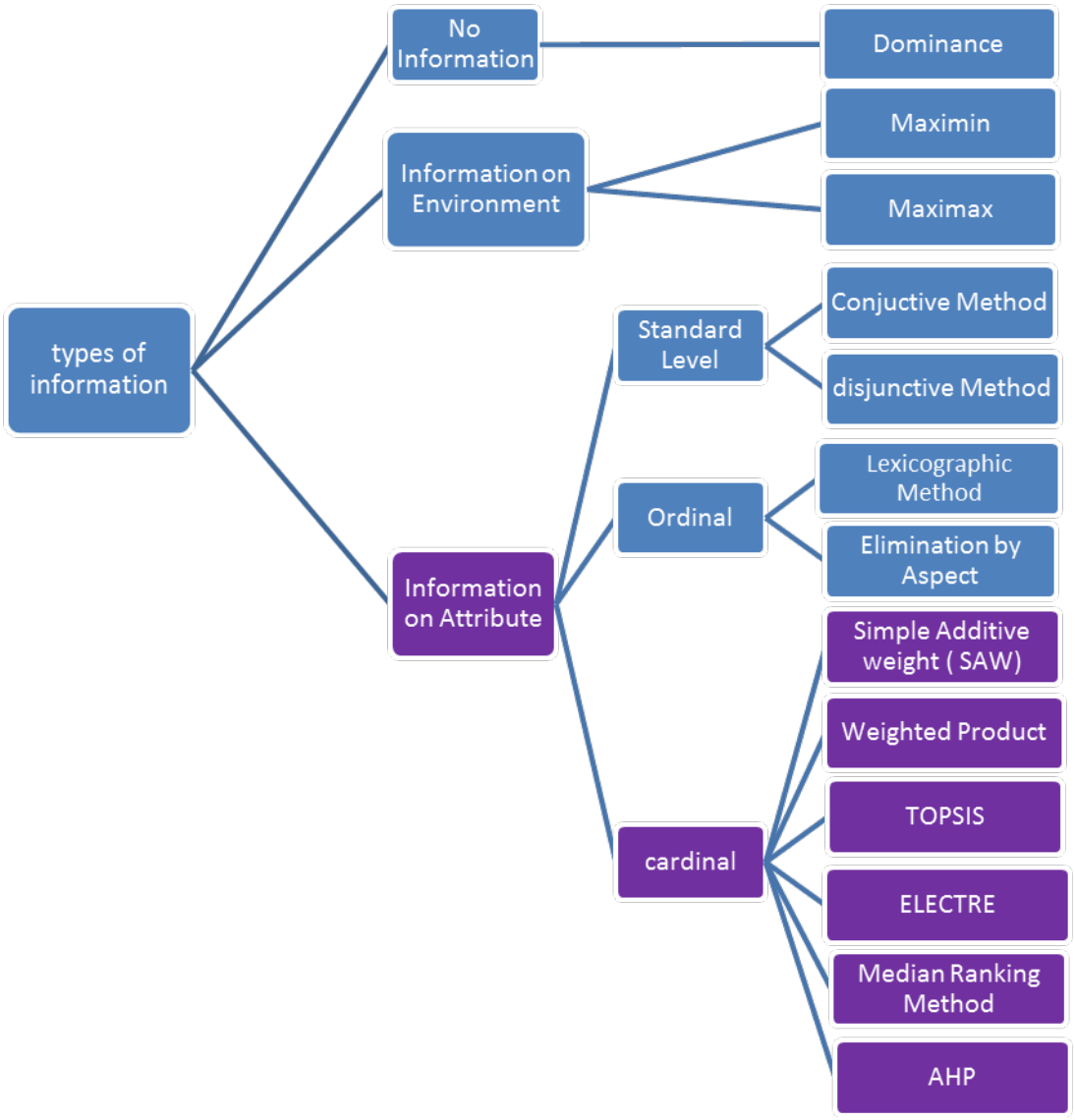

Figure 1: Taxonomy of MADM by Hwang and Yoon categorization [4]

Since I already had gathered information about the attributes of the business domain, there was fortunately no need for estimation. Therefore, the design decisions that were made in this thesis work fall primarily into the group where there is enough 'Information on Attributes' available (figure 2.1). **Information on attribute** is further categorized into three subcategories, which are Standard Level, Ordinal and Cardinal.

Standard level; Standard attributes should have the same unit of measurement. For example if we are comparing different products and attributes of the physical weight of product parts, all of the attributes are expressed in kilograms. The amounts for each attribute are supposed to be measurable by figures of the same measurement unit. Therefore

descriptive values like “good” or “bad” cannot be analyzed in this method in the Standard level.

Ordinal; Ordinal attributes are used to rank the alternatives with sequential or ordinal attributes like first, second, and third.

Cardinal; finally cardinal attributes cover the descriptive values as well. Cardinal attributes can convert these attributes to a measurable form.

I had different descriptive ranges of attributes. For example in comparison between programming languages some attributes like reliability, security are cardinal variable so cardinal methods are the suited for this problem. Here is an explanation of the most common methods from Cardinal models.

2.1 Simple Additive Weight

Simple Additive Weight is the simplest and most widely used method. This method considers all attribute values of an alternative and calculates the total value of each as follows:

$$V_i = \sum_{j=1}^{n} (w_j r_{ij}) \quad \text{i=1,2,…}$$

Where w is weights, j the attribute and is the value of response of alternative “i” on attribute “j”. The limitation of this method is that all the attributes should be numerical and comparable. [4]

2.2 Weighted product

The idea behind Weighted Product method is the same as Simple additive Weight. The only difference is in calculation of the values. Multiplication between attributes value is the way that this method uses [12].

$$V_i = \prod_{j=1}^{n} {x_{ij}}^{w_j}$$

A limitation of this method is that all the attributes should have the same type of unit.

2.3 TOPSIS

The TOPSIS method was originally proposed by Hwang and Yoon in 1981, which stands for Technique for Order of Preference by Similarity. The idea behind this method is to convert alternatives to geometrics distance, where the best solution is the one which has the shortest geometric distance from the positive ideal solution and the longest geometric distance from the negative ideal solution[4].

2.4 AHP

The “Analytic Hierarchy Process” is a multi-criteria decision-making method which is introduced by Saaty (1977 and 1994). AHP is a means which is used to analyze complex decisions. This method has a multi-level hierarchical structure of objectives, criteria, sub-criteria and alternatives [17]. AHP enables decision makers to simultaneously determine the interaction between many complex factors. All attributes compares in a pair-wise manner by experts and sorts them based on their preferences so this method is very close to humans approach analysis.

The detail of this method will be discussed in chapter 4. This method is originally from Saaty in 1980 [13]. This is a widely use method which considers both qualitative and quantitative data. Saaty demonstrated the advantages of this method with a number of case studies [13].

2.5 Method of Choice

Simple Additive Weight and Weighted Products are two methods that are not suitable for the problem statement because the types of data that we work with are not numeric and they do not have the same unit. TOPSIS also includes complicated calculations which are not easy to use. The AHP method was a suitable candidate since opinions of the project supervisor can be taken into account by allowing preferences to be stated for different criteria based on projects requirements.

2.6 Summary

In this chapter methods that help for decision making process are introduced. According to the type of problem and availability of information, we can choose different decision making methods. By considering the type of information that is available, 13 practical methods are described. I also described categories which are suitable for problem statements in my thesis and finally selecting the best method which is AHP.

Chapter 3

Extended Background

3.1 Previous works

3.1.1 iFEST

This thesis work was a part of a much bigger project which called (industrial Framework for Embedded Systems Tools), iFEST. Funded by the European Union, it tries to define a standard for fast developing and more productive industrial embedded systems. The framework that is going to be built by iFEST will be used for integrating different tools and devices in multi-core and embedded systems, by enabling standard tools and devices to be added or replaced in a system without the need for extensive integration efforts. This overview is taken from the iFEST official website [2].

Although iFEST is a standard, a data streaming application and an industrial control application is being designed as pilot implementations of iFEST standards. IFEST starts with abstract models to describe the behavior of a system independently from actual platforms. Then, these abstract models will be translated to platform dependent models. As a result, iFEST enables designers and stakeholders to design the desired system according to business requirements prior to actual detailed design. As described in figure 3.1, what the integration platform does can be summarized as follows: The business process is illustrated using a process model. Each activity of the process model will be replaced by a tool. Each tool supports importing and exporting of data with other activities as suggested by the process model. The integration platform will join the tools and supports communication between these tools.

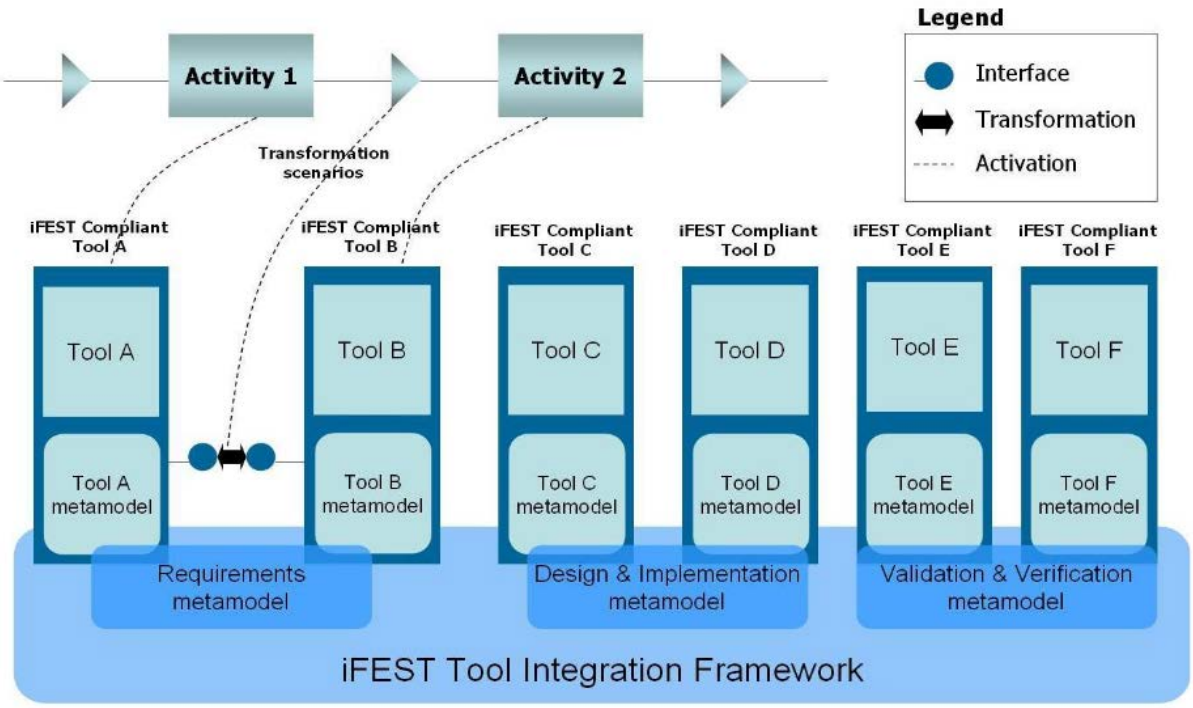

Figure 2: iFEST Tool Integration Framework [2]

IFEST is supported by major players in the embedded systems industry like Siemens, Honeywell, ABB, Thales, SELEX, and Galilio. So any kind of performance improvement or decrease in production costs is likely to lead to major economic benefits.

3.1.2 Optima

Optima is a debugging tool that has been developed by ENEA as part of the iFEST project to monitor and debug processes that are being run on ENEA's operating system, OSE. Optima is in fact a plug-in tool in the Eclipse CDT[4] environment. CDT adds some plugins inside Eclipse which facilitate the development of C/C++ software. Optima interacts with the OSE[5] real time operating system through a TCP/IP connection. The general functionality of Optima can be classified as a Debugger and Log Analyzer.

The debugger can have access to target processes that are being run on the OSE real time operating system. When the connections are established it is possible to select a target

4 CDT - C/C++ Development Tooling for the Eclipse Platform

5 Operating System ENEA

process and define the type of events to be traced. Any incoming and outgoing signal between the target process and other processes can be traced and logged.

By having the logs, the Log Analyzer is able to analyze the data based on event dump files. Log Analyzer will present detailed information about running processes and information about what signals are sent and when. The result could be shown in different ways like timelines, Gantt charts and text logs [14].

3.1.3 Farkle

Farkle is another test tool which is developed by ENEA as part of iFEST project. Farkle tests different applications which are running on a target device which refers to a flash file system. Test devices run OSE. Farkle is implemented in Python and Ruby. Farkle's main purpose is to facilitate sending and receiving signals. This communication happens through the OSE Gateway. For this to happen, Farkle classes will convert C-code from OSE to a Python class. There is also an off-line tool in the Farkle package which is called Sigpa. Sigpa is responsible for creating physical signals.

3.1.4 Integration between Optima and Farkle

The Test Manager needed to have interaction with Optima and Farkle via the connect layer. Optima and Farkle were already integrated. The overall design of this communication is shown in figure 3.2.

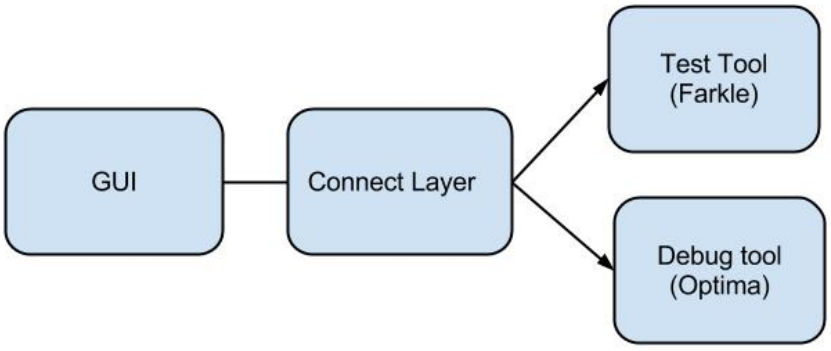

Figure 3: Test and Debug tool integration

Here is a short description of the major design components:

The connect layer is responsible for establishing a communication. For establishing the communication it has knowledge about signals, tests, requirements and test results. It keeps all these information inside itself. It also keeps an artifact called test suite which includes a combination of certain signals, results and requirements that are associated by a certain test.

GUI, Graphical User Interface, or the Test Manager is the component that is designed in this thesis work. It interacts with the user and enables him/her to start the tests and control the test results.

3.2 Technologies

In this section important technologies that are used in this project are explained. First I focus on general important points on Integrated Development Environments and then useful technologies in three different sections which are the user interface layer, business logic layer and database layer based on project delimitation of choices. The remainder of this section is dedicated to different models of integration and their advantages and disadvantages.

3.2.1 Integrated Development Environment

An Integrated Development Environment, IDE, is a platform that helps developers to accomplish their tasks in a more organized and convenient way. It provides developers with tools to easily create a user interface, write more readable code and in better format and integrates with other components of the system. Ease of access to a database is another advantage of an IDE [5][17].

There are some famous IDEs such as Intelij, Netbeans and Eclipse. Each of them has their own advantages and disadvantages. For example, Netbeans provide a very friendly user interface for developers while Eclipse is a better tool for integration between tools [6].

Working within the Eclipse environment and the use of pre-developed plugins were one of the conditions of the project. Eclipse is a multipurpose IDE that can be used as a container environment for many other development tools and programming languages like Java and Python. When it comes to integration, Eclipse is a platform that could be considered as one of the strongest IDEs for facilitating the integration process, basically because it supports a wide variety of languages and tools. “Eclipse is an IDE for anything and for nothing in

particular"[5]. An Eclipse plug-in is designed for the purpose of providing the developers a means to build a variety of tools for different vendors. It is the ease of creating a plug-in and their availability that makes it scalable and popular. One of the positive aspects of Eclipse is its portability [5]. It can be run on different operating systems like Linux, Windows, etc. Eclipse also can provide support for different content types such as HTML and XML. The Eclipse platform is designed based on specific mechanisms to facilitate building of new tools. The overall architecture of Eclipse is shown in figure 3.3.

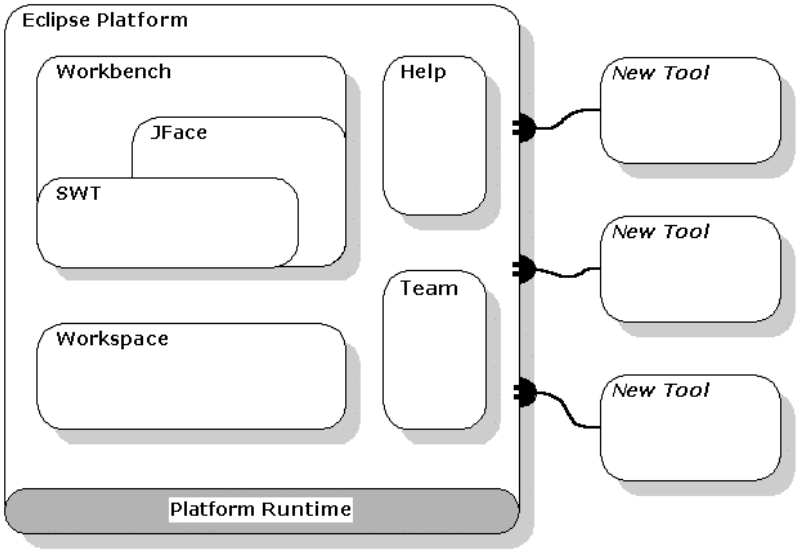

Figure 4: Eclipse platform architectural overview [5]

As you can see this platform consists of Workbench, Workspace, Help, Team support and Plug-ins. below you will get some general information about each part.

Workbench

All Graphical User Interface components are built inside workbench. The workbench's structure is based on two important toolkits which are SWT and JFace.

Workspace

The Workspace is the place in which all plug-in tools act. Files are grouped into projects and one or more projects together form the workspace. Each project is bound to a specific location in the file system and they are bound to sibling subdirectories of a single workspace directory [5].

Help

The help mechanism in the Eclipse environment could be considered as a central online book that any other tool can contribute to. The raw content is based on HTML while it shows up as XML language in the book. This conversion does not need any modifications.

Team support

Eclipse facilitates teams to work on a workspace under a specific version and configuration management by providing a common repository that is allocated to all the team members. Adding and rewriting on the repository, updating files, and comparing different versions are a few of the functionalities that are a supported by Eclipse.

Plug-ins

The smallest part of the Eclipse platform is called the plug-in. Depending on the complexity of the project, a plug-in could be implemented as one or several plug-ins. Java is the required implementation language. Each plug-in declares any number of extension points, and any number of extensions to other plug-ins for establishing the connection. When a plug-in runs, it uses a plug-in registry to look up extensions which are related to its extension points. These actions are required to activate the plug-in.

3.2.2 User Interface layer

Since there was no delimitation on whether to use a web or desktop user interface from the project owners, I performed a comparison between desktop and web applications, before selecting a user interface for the application [7].

1) Web applications could be used anywhere that has a connection to the Internet and anytime, while desktop applications confine users to the systems where the application has been installed.

2) Desktop applications look better than web-application from a look and feel aspect. In desktop applications, the designer has more powerful tools to create almost anything they want. Web application user interfaces are based on HTML and this will cause that interface to be displayed differently in different web browsers.

Although client-side scripting languages like Javascript can be included in a web interface, these technologies are expensive with regards to time and effort. Additionally my intended interface was not customer facing and it does not need fancy UI components.

3) If performance is required, then a desktop application is preferable. In a web application, both data and interface is transmitted every time an HTML page is requested. This could

severely affect the performance. In some applications, such as computer games, it is necessary to have heavy interactions with the machine's hardware devices and video card. Therefore, in this situation using desktop application has a really better performance. [7][22]

4) In case the interface interacts with some other devices and applications, then the best option would be a desktop interface. Because web page runs within a limited browser environment which limits functionalities like interacting with a POS device or writing into office documents [7].

5) One of the problems of the desktop interface is updating the clients with the latest version. This could be challenging if the number of users are large, while this is not a problem for web-based interfaces at all. Although the number users for our application are limited, a web-based user interface can make it scalable for even more users in other places for the future.

3.2.3 Programming Language

The best way of selecting a language and associated technologies is to pick them based on the project's need instead of preference of the programmer. So as mentioned before, a few technologies were already selected by the company and selecting the best one was a part of the thesis criteria. Java and Python were two alternative languages from ENEA to be compared. Based on four brainstorming sessions with the project owners some important criteria were agreed on. Thereafter based on these criteria I compared Java and Python through an Analytic Hierarchy Process method.

For comparing the performance of these two languages I had implemented a comparative program in both Python and Java to benchmark the performance of the two languages. The task was to map letters to specific digits in order to encode telephone numbers with some specific functional requirements. Then the program would list all possible encoding combinations of a sample phone number. The output of this activity helped me in comparing the performance of Java and Python in the comparison table in appendix 1.

Although Java is more structured, it is easier for a programmer to write code in Python. This is mostly due to the benefits and strengths of its dynamic type system. Not having to consistently worry about types gives you more time to actually concentrate on finding a solution to a given problem. It brings you some sense of freedom. Also when operating on data structures in Python, it seems a lot more powerful and versatile than its Java equivalent. Examples would be the ease of operating on dictionaries, for example retrieving keys and values, or the possibility to merge lists just by using concatenation. There are some

additional powerful tools in Python which provide a number of potential benefits, the very least being the reduction of the amount of code. While we are on the subject of amount of code, Python really is a clear winner if you're interested in having as little code as possible. It is not unusual to be able to squeeze in one line what would have taken multiple lines in Java. However, there are some drawbacks in Python when compared to Java. One is the execution time of the implementation. Another potential problem is the lack of blocks and thus the heavy reliance on proper indentation. This can sometimes make your code hard to debug. While in Java we can have auto format that fixes the indentations (which does not and cannot exist in Python) the only exception in Java brackets is when the “if” or “loop” blocks are a single line and can be written without curly brackets. The detailed comparison of these two languages is shown in chapter 5. I did my best to cover as many criteria as possible to compare these languages; however, I also used additional article which compared these languages [21].

The lessons learned from that comparison task were fed into the AHP process. As a result of the AHP process, Python was finally selected, although the Test Manager prototype was written in Java. This was the limitation from the company to finish the prototype rapidly so the prototype was done before the research result is achieved.

3.2.4 Database

One of the requirements of this project was to facilitate the recording of data. Data could be the result of test cases or it could be static, pre-stored information. Pre-stored information could include information about a test's functionality, for example what a specific test is doing and what the target object of the test is. From the stakeholders' perspective, these are the three categories of data storage technologies that may be used: SQL based databases, No SQL [18] databases, and text files. The text below describes the comparison between these candidates.

Text file

A text file is the easiest way to store data. It does not need any special design and implementation. A text file is readable by using specific operating system calls. It is easy to edit. It does not need to be installed and there is no need for learning any extra database software. In another words it is very easy to use. Another positive point is that text files do not require lots of disk space.

SQL

A SQL Database has great security, where having a database is a good way of protecting data from external manipulation. Moreover, a large amount of data can be stored and it is very easy to retrieve data. The level of accessing to the data can be customized.

No-SQL

NoSQL is a term which was used in 1998 by Carlo Strozzi[26]. NoSQL covers all databases which do not follow a popular Relational Database Management System (RDBMS). NoSQL databases deals with non-relational data and it avoids using some popular operation like JOIN in SQL, which will lead to an increase the speed of online transaction processing.[18] Flexibility and performance are the most important features for thinking about NoSQL models. NoSQL models also scales well with a large number of users and data, and is thus adopted by many social networks like Facebook and Twitter. Another strength of this model is that it deals with complex data, so it is useful for business and commercial applications. There are four different sub-categories in NoSQL databases:

Document-Styles Storage

Data is stored as documents; data is in the form of key-value pairs and they are accessible in a recursive manner. CouchDB is an example of this model.

Key-Value Storage

The general structure of Key Value Store is very similar to Hash maps in Java. We have a keyValueStore(key)=value Global structure. These models do not provide an automatic method for indexing but to access the key-value we can create another structure. One of the important examples of this model is XML database.

Column-Based

This model is based on the Table or Column model. Casandra is a famous example of this model. This model allows cells to have more than one value inside it. The number of columns and the type of columns in a row is flexible and can vary from row to row.

Graphical DB

This is a graph based model and the interaction between each node and the relationships between nodes are established by key-value elements. This model is able to represent a complex network. Neo4j is one of the examples of this model [24].

3.3 Tool Integration

3.3.1 Definition of tool integration

B. Thomas and I. Nejmeh in “Definition of tool integration for environments,” [8] mentioned that “Integration is not a property of a single tool, but its relationships with other elements in the environment, mainly with other tools, platforms, or processes”. In other words, Integration means that things function as members of a coherent whole”. Since these relationships could occur in different levels, Anthony Wasserman has defined five different levels oh integration of tools which will be discussed more in this chapter [10].

In another categorization, software tool integration is divided into two classes, horizontal and vertical integration. Horizontal integration is about integrating all activities that occur during the project life cycle, and vertical focuses on specific parts of the project [9].

3.3.2 Different perspective on tool integration

Discussion about integration can be considered from two different perspectives. The first perspective is from the user's point of view. It means that the environment should be shaped in way that users accept that they are working with an integrated tool. The latter is the designer's view in which they consider the amount of time and resources that should be assigned to provide users with an integrated tool.

3.3.3 Tool Integration levels and their properties

The following categorization is extracted from "Tool integration in software engineering environments" by Wasserman [10].

1. **Platform Integration**

This kind of integration requires the two components to be on the same platform and to have common tools to operate with each other. Tools could be run on the same computer or the same operating system while in different locations such as in a distributed network. They can even be run on different machines with the same operating systems. Having the same setup for tools is not required.

Advantages and Disadvantages

Information can be exchanged through the platform in a real-time manner. In many cases the efficiency of organizations which use the same platform for their systems is more than the efficiency of organizations that avoid this. On the other hand, platform Integration facilitates the development and maintenance of systems. In spite of these advantages integration development is costly for initial development.

2. **Presentation Integration**

Presentation Integration deals with user interface consistency. "Look and feel" similarity is a feature that increases the quality of the presentation integration, or in other words, presentation integration provides the same look and feel across all the interfaces. The same look and feel makes the learning process of different tools in a specific environment much easier for users. Presentation integration helps us to create new user interface classes that look like old ones while it can be supported by different tools in the background. Appearance, behavior, and interaction paradigms are properties that are important in this kind of

integration. If there is an appearance similarity between two tools and if a user learns one tool, users are able to learn to transfer this knowledge.

Advantages and Disadvantages

Re-usability of the user interface components are the first and foremost benefit of presentation integration. In presentation Integration there is no need to create new interfaces, we just can reuse what has already been designed. So this process leads to a quick implementation and thus reduces the implementation costs. Regardless of the business logic of applications, presentation integration can be used in different applications or in other words “It works with monolithic applications” [11]. Since validation of the application is tested through the business logic, presentation Integration will reduce the business logic risk in this level of integration [11].

Presentation Integration is too fragile, however, since changing the user interface occurs very frequently during the development process. For example, presentation integration depends on the specific position of data fields on a user interface in a such way that even small changes to data structure will lead to a break of the data presentation [11]. Another feature to be considered is the lack of direct interconnection between interface and database in systems that follow the MVC design pattern. Limited access to data will however cause problems in extracting information from internal data sources. Since any access to the data source should go through a mediation layer, any change in the data access rules implies changes on mediation components as well.

There are currently numerous types of devices that access the Internet, where screen size can vary significantly. The extracted information will be presented to users with a variety of screen sizes. Therefore it would be a good idea to have the presented data to be split into multiple smaller screens. This requires multiple requests and as a result this will slow down the transactions' performance.

3. Data Integration

Data integration takes care of difficulties that arise in the connectivity with data sources. Based on [8], a data integration tools have similarity in their view of data. For example, to establish interaction between different tools, data should have a common format otherwise a conversion program is required to translate it to the destination format. Challenges will come up when several databases must be merged into one. Also the more tools that share a common data source, the more complications will arise. To address all these issues, five properties have been defined for data integration in [8]:

Interoperability

Data needs to be manipulated to have a coherent form in order for different tools to be able to use them. This requires work and effort in the data source integration phase. The required amount of work has a direct relationship with the interoperability of the integrated tools. This means that the more interoperability is needed, the more data manipulation is required.

Non-redundancy

The non-redundancy property recognizes the normality of the data which has been stored by different tools. The less redundant data, the more well-integrated are the tools.

Data consistency

Data Consistency concerns the case when different tools, with semantic relations to each other, have manipulation access to the same data source. In this case the tools should cooperate in their manipulation of data in order to avoid inconsistency in the common data source. There should be ACID[6] compatibility for the database to be considered as consistent.

Data exchange

In order to keep two data sources synchronized with each other, one party should export while the other should import data updates. Problems arise when there is a mismatch between how the data is formatted in different systems. In this case there should be efforts to transform the data to the desired form, such as writing a script that does the transform operation. This facilitates the process of exchanging data between two tools when importing and exporting data between them. These facilitation tasks are necessary because data may differ in various aspects, such as form and semantics.

Measuring the effort needed to facilitate data exchange is the main subject in this field. The difference between data exchange and interoperability is that the former works on non-persistent data while the latter works on database content.

To be able to transform data in an efficient way, we need to have access to the meta-data that describes the data structure and documents data entities and the logical relations among them. So any data integration tool has to have access to enough meta-data in order to facilitate the transformation between different applications.

Synchronization

This property deals with the synchronization and the manipulation of non-persistent data between cooperating tools.

[6] Atomicity, Consistency, Isolation, Durability

Advantages and Disadvantages

Here are some benefits of data integration. A data integrated tool supports multiple users that work on one or many distributed databases at the same time. This can create many data update transactions on the network, often requiring high bandwidth access to database servers. Data Integration is widely used in e-business applications. Because e-business applications need to have access to data in real-time and there should be an environment that supports the transformation of dynamic data between different modules.

Challenges

Data integration is sometimes needed for integrating applications within their own presentation and business logic layer. Having access to the data layer has the benefit that users are able to access to raw data, while the drawback of this issue is no encapsulation of application operations. As a result there would be a malfunction in the system because each component has the right to update directly. That is why the majority of application vendors tend to hide their data model in order to reserve the right to change the data model at any time without changing the external user interface or even the APIs and services.

The other challenging issue with data integration is “semantic dissonance”. Distinguishing between field names that are semantically close to each other is not straightforward. Consider, for example, distinguishing between “dates of birth” and “age”, if different data types and formats are used. In these cases, interaction between these databases requires a translation to a unique form.

4. Control Integration

Control integration is a way to facilitate communication between tools in order to notify each other about their latest state and recent changes. Without this kind of integration users must get information about the tools' jobs manually, whereas by using control integration notification this could be done automatically. This integration method is similar to data integration. Data integration covers data representation, conversation and storage task while this level fulfills the functionality of the system. It could be said that control and data integration tools complete the process of data integration.

Provision and Use are the main properties of control integration tools. Provision property means that in order to reach a well-controlled integration state, a tool will provide the other tools with some services. The amount of these services defines the provision value. Use

property means that when services are provided, other tools will use them. Use refers to what extent a tool uses another tool's services.

Advantages and Disadvantages

During the development of an integrated solution, control integration helps in testing the functionality of each component along the integration chain. So the most important benefit of control integration is that testing of the component occurs early in the development process. This helps developers to get early feedback, including warnings about code malfunctioning, code changes, unit tests results, etc. Although it is positive to know about malfunctions, it takes some time and cost to do the required settings for the control integrations to work.

5. Process integration

Process integration considers the functionality of the entire system. Process model integration provides the users with tools to follow the sequence of all steps in the process. Based on business requirements, each step in the process chain can be automated, semi-automated or manual. Three requirement aspects should be considered in order to achieve well integrated process system. These aspects include process, event and constraint.

Process: In order to accomplish a certain chain of activities, a number of physical activities and/or software tools need to be combined. There are a variety of methods for orchestrating tools, such as using a Business Process Management System or direct integration of tools with each other. The business requirements determine which method is used.

Event: An event is a specific point in time when a specific step of the process needs to run or reach a common point in order for a specific thing to happen. Tools should be ready to respond when another tool indicates an event [8].

Constraint: Constraints are specific requirements or limitations of the project that are imposed by the stakeholders. They impose the way tools must behave in the process.

Advantages and Disadvantages

Creating a unique sequence of tasks in a process increases the maintenance efforts that are necessary for each task in the process. However, each task has the ability to be reused in one or many processes. A process can be integrated with other processes. This ability will bring up some drawbacks because each process is involved with complexities like

concurrency, checkpoint, etc. which make the integration process even more complex. Integration needs a lot of information about process configuration which needs to be provided by the process manager. If process integration is done through a business process management tool, it can provide reports about the current state of the process. This also enables us to monitor process instances and their respective statuses.

There are also some challenges with process integration. When a central process runs several other processes, it is probable that bottlenecks appear somewhere in between. This is due to running lots of processes in a central process. Such circumstances need to be optimized to avoid bottlenecks in the process. So overusing of process integration could bring up a number of challenges. Although it might be thought that process integration could be used in any integration solution, it is not always a good solution.

Chapter 4

Outcome of decision making process

4.1 Selection of tool integration model

In order to make a decision on what integration method to be used, first step is to build the hierarchical tree. In order to create the hierarchical tree following steps should be followed:

1. Selecting an objective for the decision making process which is “Selection of tool integration model”
2. Recognition of the criteria; the criteria are speed, cost and effort, reusability, data integrity and scalability. These criteria were defined by the project’s stakeholders.
3. Choosing the alternatives. Alternatives are: platform, presentation, data, process and control integration.

Here is the hierarchical tree of the above steps:

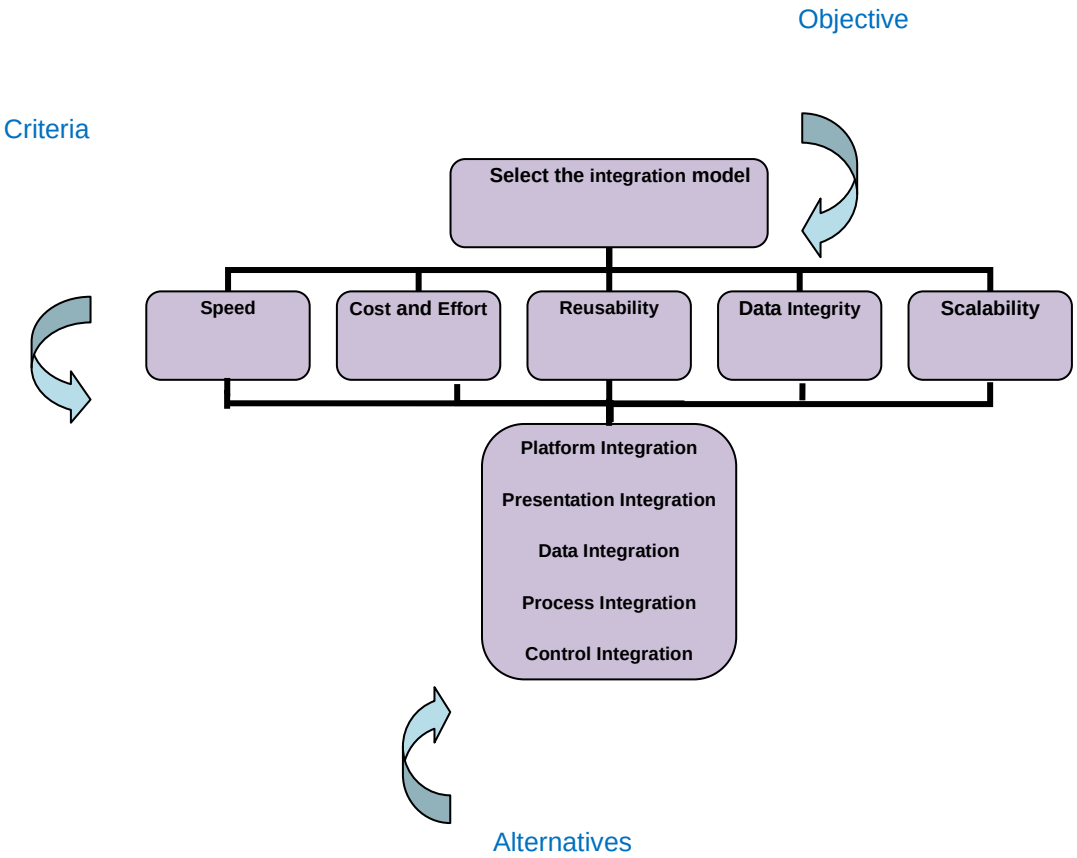

Figure 5: Hierarchical tree based on integration model

The second step is to create a pairwise comparison matrix, which expresses the relative importance of each criterion and preference over another criterion. A pairwise matrix should be created for both all criteria and all alternatives based on each criterion. To create the pairwise matrix all the criteria are compared with each other. This comparison has been done by the project's stakeholders. For example it was said that "Speed" is 2 times as important as "Cost and Effort" or "Speed" and "Data Integrity" have the same importance so we have digit 1 in this matrix. Table 4.1 shows the detailed comparisons between all the criteria, followed by the corresponding matrix.

Criteria/Criteria	Speed	Cost and Effort	Reuse	Data Integrity	Scalability
Speed	1	0,50	2,00	1,00	3,00
Cost and Effort	2,00	1	3,00	2,00	4,00
Reuse	0,50	0,33	1	0,50	1,00
Data Integrity	1,00	0,50	2,00	1	2,00
Scalability	0,33	0,25	1,00	0,50	1

Table 1: Comparision data integration criteria

Here is the matrix extracted from above table:

$$\begin{bmatrix} 1 & 0{,}50 & 2 & 1 & 3 \\ 2 & 1 & 3 & 2 & 4 \\ 0{,}5 & 0{,}33 & 1 & 0{,}5 & 1 \\ 1 & 0{,}5 & 2 & 1 & 2 \\ 0{,}33 & 0{,}25 & 1 & 0{,}5 & 1 \end{bmatrix}$$

Now we can obtain the ranking of priorities from pairwise matrix. In order to obtain priority vector there are several methods like Eigenvector method, weighted least-squares method, Logarithmic least-squares method, Logarithmic goal programming method and Fuzzy preference programming method [15]. Eigenvector gives good approximation of the priorities vector [15]. This method is easy to follow for readers and there are not many differences between this method and the others so Eigenvector is the method that has been used in this thesis. Here is the short way to compute the eigenvector in three steps:

1. Obtain squared matrix of pairwise matrix.

2. Calculate sum of the rows and then normalize the rows.

3. Repeat step 1 and 2 until consecutive calculations of the sums produce a sufficiently small differences.

Step1:

$$\begin{bmatrix} 1 & 0{,}50 & 2 & 1 & 3 \\ 2 & 1 & 3 & 2 & 4 \\ 0{,}5 & 0{,}33 & 1 & 0{,}5 & 1 \\ 1 & 0{,}5 & 2 & 1 & 2 \\ 0{,}33 & 0{,}25 & 1 & 0{,}5 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0{,}50 & 2 & 1 & 3 \\ 2 & 1 & 3 & 2 & 4 \\ 0{,}5 & 0{,}33 & 1 & 0{,}5 & 1 \\ 1 & 0{,}5 & 2 & 1 & 2 \\ 0{,}33 & 0{,}25 & 1 & 0{,}5 & 1 \end{bmatrix} = \begin{bmatrix} 4{,}99 & 2{,}91 & 10{,}5 & 5{,}5 & 12 \\ 8{,}82 & 4{,}99 & 18 & 9{,}5 & 21 \\ 2{,}49 & 1{,}41 & 4{,}99 & 2{,}66 & 5{,}82 \\ 4{,}66 & 2{,}66 & 9{,}5 & 5 & 11 \\ 2{,}16 & 1{,}245 & 4{,}41 & 2{,}33 & 4{,}99 \end{bmatrix}$$

Step 2: Sum of the rows

$$\begin{bmatrix} 4{,}99+2{,}91+10{,}5+5{,}5+12 \\ 8{,}82+4{,}99+18+9{,}5+21 \\ 2{,}49+1{,}41+4{,}99+2{,}66+5{,}82 \\ 4{,}66+2{,}66+9{,}5+5+11 \\ 2{,}16+1{,}245+4{,}41+2{,}33+4{,}99 \end{bmatrix} = \begin{bmatrix} 35{,}9 \\ 62{,}31 \\ 17{,}37 \\ 32{,}82 \\ 15{,}135 \end{bmatrix}$$

For normalization all the rows should be divided by the sum of the rows:

35, 9+62, 31+17, 37+32, 82+15,135= **163,535**

$$\begin{bmatrix} 35{,}9/163{,}535 \\ 62{,}31/163{,}535 \\ 17{,}37/163{,}535 \\ 32{,}82/163{,}535 \\ 15{,}135/163{,}535 \end{bmatrix} = \begin{bmatrix} 0{,}219524872 \\ 0{,}381019354 \\ 0{,}106215795 \\ 0{,}200690984 \\ 0{,}092548996 \end{bmatrix}$$

Step 3:

By doing step 1 and 2 again:

$$\begin{bmatrix} 4{,}99 & 2{,}91 & 10{,}5 & 5{,}5 & 12 \\ 8{,}82 & 4{,}99 & 18 & 9{,}5 & 21 \\ 2{,}49 & 1{,}41 & 4{,}99 & 2{,}66 & 5{,}82 \\ 4{,}66 & 2{,}66 & 9{,}5 & 5 & 11 \\ 2{,}16 & 1{,}245 & 4{,}41 & 2{,}33 & 4{,}99 \end{bmatrix} * \begin{bmatrix} 4{,}99 & 2{,}91 & 10{,}5 & 5{,}5 & 12 \\ 8{,}82 & 4{,}99 & 18 & 9{,}5 & 21 \\ 2{,}49 & 1{,}41 & 4{,}99 & 2{,}66 & 5{,}82 \\ 4{,}66 & 2{,}66 & 9{,}5 & 5 & 11 \\ 2{,}16 & 1{,}245 & 4{,}41 & 2{,}33 & 4{,}99 \end{bmatrix} =$$

$$\begin{bmatrix} 128{,}26 & 73{,}41 & 262{,}34 & 138{,}48 & 302{,}48 \\ 222{,}47 & 127{,}36 & 455{,}11 & 240{,}22 & 524{,}68 \\ 62{,}25 & 35{,}63 & 127{,}36 & 67{,}22 & 146{,}83 \\ 117{,}42 & 67{,}224 & 240{,}225 & 126{,}8 & 276{,}96 \\ 54{,}37 & 31{,}12 & 111{,}23 & 58{,}71 & 128{,}26 \end{bmatrix}$$

Sum of the rows:

$$\begin{bmatrix} 904{,}97 \\ 1569{,}85 \\ 439{,}31 \\ 827{,}63 \\ 383{,}71 \end{bmatrix}$$

For normalization all the rows should be divided by the sum of the rows:
$904{,}97+1569{,}85+439{,}31+827{,}63+383{,}71= 4126{,}494$

$$\begin{bmatrix} 904{,}97/4126{,}494 \\ 1569{,}85/4126{,}494 \\ 439{,}31/4126{,}494 \\ 827{,}63/4126{,}494 \\ 383{,}71/4126{,}494 \end{bmatrix} = \begin{bmatrix} 0{,}219309211 \\ 0{,}380431906 \\ 0{,}106461156 \\ 0{,}200809365 \\ 0{,}092988362 \end{bmatrix}$$

And by comparing with the result from previous step:

$$\begin{bmatrix} 0{,}219309211 \\ 0{,}380431906 \\ 0{,}106461156 \\ 0{,}200809365 \\ 0{,}092988362 \end{bmatrix} - \begin{bmatrix} 0{,}219524872 \\ 0{,}381019354 \\ 0{,}106215795 \\ 0{,}200690984 \\ 0{,}092548996 \end{bmatrix} = \begin{bmatrix} -0{,}0002 \\ -0{,}0005 \\ 0{,}0002 \\ 0{,}0001 \\ 0{,}0004 \end{bmatrix}$$

As you see there is not too much difference (Four decimal places is a good stop point [25]). So the result is acceptable and by sorting these values we have:

0,381019354 > 0,219524872 > 0,200690984 > 0,106215795 > 0,092548996

Or in other words the relative importance of the criteria is ordered as follows:

Cost and Effort > Speed> Data Integrity> Reuse> Scalability

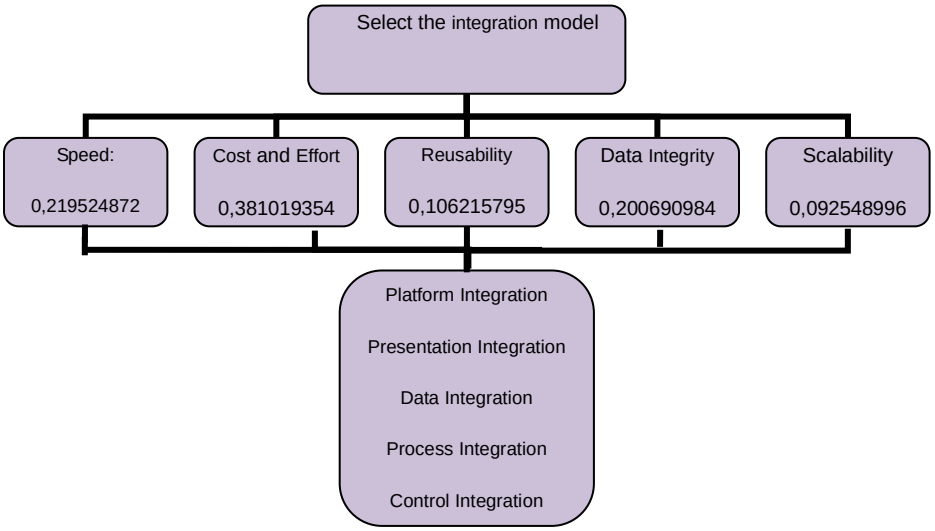

Figure 6: Relative importance of criteria based on integration model

It is time to calculate the pairwise matrix for alternatives based on each and every criterion. We need to know for example how important platform integration is in term of speed in this project. To find out this question was asked from project stakeholders. They answered this question by using these terms very good, bad, and excellent and etc. Since mutual comparisons between these criteria are based on personal judgment, these values are not quantitative. To have measurable values based on AHP model, values from 1 to 7 has been allocated as follows. T(Terrible)=1, VB(Very Bad)=2, Bad(3), M(Medium)= 4, G(Good)=5, VG (Very Good)=6 and E(Excellent)=7.

Preferences	Numeric Value
Excellent	**7**
Very Good	**6**
Good	**5**
Medium	**4**
Bad	**3**
Very Bad	**2**
Terrible	**1**

Table 2: The qualitative values to measurable number

Integration/ Criteria	Speed	cost and effort	reuse	data integrity	scalability
Platform	E(7)	VG(6)	M(4)	VG(6)	M(4)
Presentation	B(3)	G(5)	G(5)	VB(2)	VG(6)
Data	B(3)	VB(2)	G(5)	E(7)	B(3)
Process	B(3)	M(4)	E(7)	B(3)	E(7)
Control	M(4)	M(4)	M(4)	B(3)	G(5)

Table 3: Integration model and criteria's

The next step is comparing between the integration models using each criterion with the help of table 4.2. It means that, for example, based on speed criteria in table 4.1, platform has an E(excellent) value and presentation has an B(Bad) value. So for comparing platform and presentation in term of speed Excellent (7) divided Bad(2) is equal 7/3=2.33.

Based on speed	Platform	Presentation	Data	Pocess	Control	Average
Platform	1,00	2,33	2,33	2,33	1,75	0,349839908
Presentation	0,43	1,00	1,00	1,00	0,75	0,150160092
Data	0,43	1,00	1,00	1,00	0,75	0,150160092
Process	0,43	1,00	1,00	1,00	0,75	0,150160092
Control	0,57	1,33	1,33	1,33	1,00	0,199679817

Cost and effort	Platform	Presentation	Data	Process	Control	Average
Platform	1,00	1,20	3,00	1,50	1,50	0,285714286
Presentation	0,83	1,00	2,50	1,25	1,25	0,237904731
Data	0,33	0,40	1,00	0,50	0,50	0,095047589
Process	0,67	0,80	2,00	1,00	1,00	0,190666697
Control	0,67	0,80	2,00	1,00	1,00	0,190666697

Based on reuse	Platform	Presentation	Data	Pocess	Control	Average
Platform	1,00	0,80	0,80	0,57	1,00	0,149720348
Presentation	1,25	1,00	1,00	0,71	1,25	0,187019303
Data	1,25	1,00	1,00	0,71	1,25	0,187019303
Process	1,75	1,40	1,40	1,00	1,75	0,26214174
Control	1,00	0,80	0,80	0,57	1,00	0,149720348

Based on data integrity	Platform	Presentation	Data	Pocess	Control	Average
Platform	1,00	3,00	0,86	2,00	2,00	0,309763095
Presentation	0,33	1,00	0,29	0,67	0,67	0,103494891
Data	1,17	3,50	1,00	2,33	2,33	0,361149751
Process	0,50	1,50	0,43	1,00	1,00	0,154881547
Control	0,50	1,50	0,43	1,00	1,00	0,154881547

scalability	Platform	Presentation	Data	Pocess	Control	Average
Platform	1,00	0,67	1,33	0,67	0,80	0,162054654
Presentation	1,50	1,00	2,00	0,86	1,20	0,235254303
Data	0,75	0,50	1,00	0,43	0,60	0,117627151
Process	1,50	1,17	2,33	1,00	1,40	0,266335088
Control	1,25	0,83	1,67	0,71	1,00	0,195612217

Table 4: Comparisons of alternatives

The final step is to calculate the matrix product below. The left matrix is the normalized values which are calculated in table 4.3 and the right matrix is the criteria's weight.

$$\begin{bmatrix} 0{,}349839908 & 0{,}285714286 & 0{,}149720348 & 0{,}309763095 & 0{,}162054654 \\ 0{,}150160092 & 0{,}237904731 & 0{,}187019303 & 0{,}103494891 & 0{,}235254303 \\ 0{,}150160092 & 0{,}095047589 & 0{,}187019303 & 0{,}361149751 & 0{,}117627151 \\ 0{,}150160092 & 0{,}190666697 & 0{,}26214174 & 0{,}154881547 & 0{,}266335088 \\ 0{,}199679817 & 0{,}190666697 & 0{,}149720348 & 0{,}154881547 & 0{,}195612217 \end{bmatrix} *$$

$$\begin{bmatrix} 0{,}2195249 \\ 0{,}3810194 \\ 0{,}1062158 \\ 0{,}200691 \\ 0{,}092549 \end{bmatrix} = \begin{bmatrix} 0{,}278728555 \\ 0{,}186017627 \\ 0{,}172409024 \\ 0{,}189187545 \\ 0{,}181572098 \end{bmatrix}$$

Which means the order of weight importance is as follows:

0,278728555> 0,189187545> 0,186017627>0,181572098>0,172409024

or in the other words if we bind the above values to their corresponding alternatives we come up with the following order of alternatives:

Platform>Process> Presentation>Control>Data

4.2 Selection of programming language

Java and Python are the two programming languages that are compared in this thesis with respect to the project requirements. The relevant factors for comparison that arose during the brainstorming session are mentioned in table 4.4. The number of factors is 16 and many of these 16 criteria had almost very close meanings and it would made it difficult for stakeholders to priorities in mutual comparison therefore I decided to group them into a smaller set of criteria to avoid confusion and meaning overlap between criteria. As a result to make the calculations possible, factors are categorized into 5 categories as reflected in the left column of the table below.

Category	Important Criteria
Efficiency	Compiler Efficiency
Efficiency	Memory usage
Resources	Documentation
Resources	Libraries(OO Libraries)
Resources	Available resources for learning
Scalability	Extensibility
Scalability	Portability / Machine-independence
Security	Reliability
Security	Security
Usability	Simplicity
Usability	Intended Purpose
Usability	Compiling
Usability	Debugging
Usability	Integrated Development Environments
Usability	Easy to use liberaries
Usability	Language Efficiency

Table 5: Important programming language criteria

The result of this analysis based on the AHP model is that Python is a better candidate than Java. The details of the calculations are accessible in appendix 1.

4.3 Selection of database

Text files, SQL database, Column storage, Document storage, and XML database are the alternative options we consider for storing for the test result information. Considering interoperability, non-redundancy, data consistency, data exchange and synchronization as important factors for such a storage system, a SQL based database was chosen as the best choice between the others as a result of an AHP process. The detailed calculation is in appendix 1.

4.4 Desktop or web application

By applying the AHP process for user interface requirements, a Web application proved to be a better choice than desktop application, at least with respect to the project requirements. The detailed calculation of AHP models can be found in appendix 1.

Chapter 5

Implementation

5.1 Architecture

As discussed in previous chapter Python was finally selected. But there was a limitation from the company to finish the prototype rapidly so the prototype was done before the research result is achieved. So the implementation and design of the Test Manager prototype was done in Java Standard Edition using object oriented methods. At this point I chose Java because I could rapidly develop a Java application for prototyping purposes. The Model, View, Controller (MVC) model is adopted as the main architectural design pattern which is a well-known design pattern for software engineering of scalable systems. An MVC application logically separates different aspects of the application into different components, while they keep a loosely coupled relationship between them [16].

The level design is as illustrated in the diagram in figure 5.1, consisting of several components which will be described separately. The smiley face represents the human user that interacts with the human interface component. Each box is a component of the system. The arrows from component A to B mean that component A calls a functionality of component B.

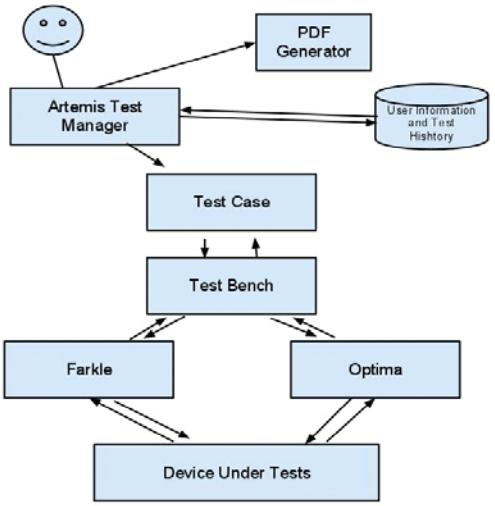

Figure 7: Design of Test Manager

Test Case Class:

The test cases are presented by relevant methods in the Test cases class. Since the real test cases have a lot of complexities to be directly integrated to the user interface, integration to the testing application is done via these intermediate test methods.

5.2 Application User Interface

The user interface is built using Java Swing libraries and consists of several forms. Each page is actually a JFrame class within the Swing framework. The relation between the forms is based on navigation links such as Home, Next. The main page of the application is where the user can see test categories and information about each category. Categories consist of: FTL[7], JEEF[8], and MDD[9]. These categories are described in section 5.3. By selecting each test category, the user is directed to the specific form of the category, where all the related tests are listed. Then the user can run the test case, by selecting a test case and providing

[7] File Translation Layer

[8]Journaling Extensible File system Format

[9] Memory Device Drivers

proper input to the test case. Figure 5.2 shows the landing page of the testing application where the user can read category descriptions and choose the intended test category.

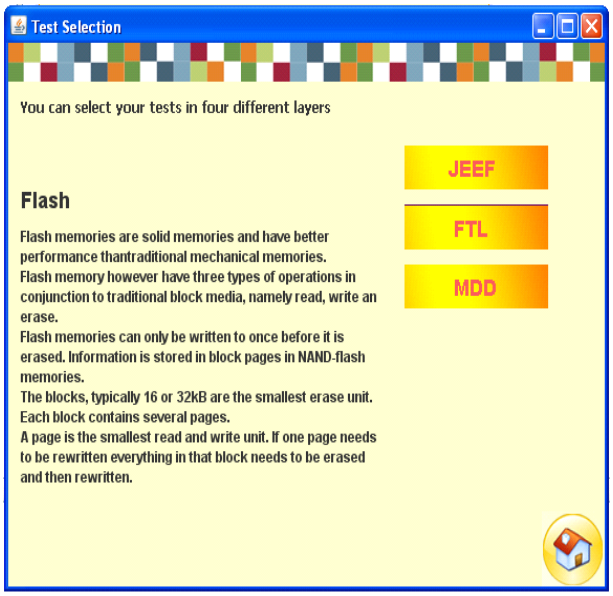

Figure 8: Landing page of the testing application with test category

By selecting a test category, the user is directed to the test page where all the tests of that category can be executed. As illustrated in figure 5.3, the user is able to run the test by selecting the intended test case and providing proper input parameters.
While running the test, a progress bar presents the progress of the test, and at the end the result is shown in the lower panel.

Figure 9: Test page runs the test cases and shows the results

MySQL Database: The MySQL database was used to store two sets of information: user credentials to be used at login time, and test case reports from running tests.

PDF Generator: The PDF generator was also written in Java and can produce PDF documents from the text information that is produced from running the test cases.

So far all aspects of the design are explained, now in figure 10 the sequence flow of different components are illustrated. This diagram shows the flow of control among different sections of the system in order to accomplish a test.

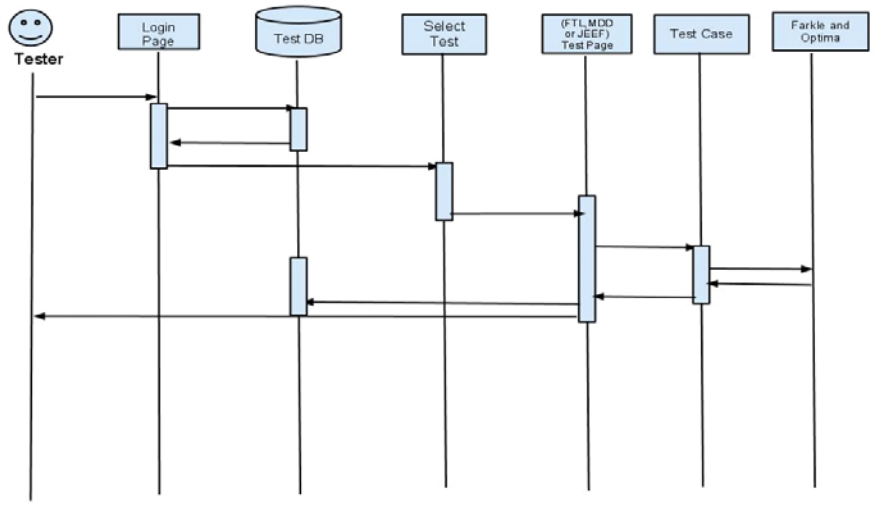

Figure 10: Flow of control among different sections of the systems in order to accomplish a test

5.3 Test Description

Three different layers have been tested in this project. The layers are FTL, MDD and JEEF, which are described later in this section. The information about all layers is in an ENEA internal document which is called EQos_Test_Description and I have used that instruction in the development process. Here is a sample of one of the mentioned layers.

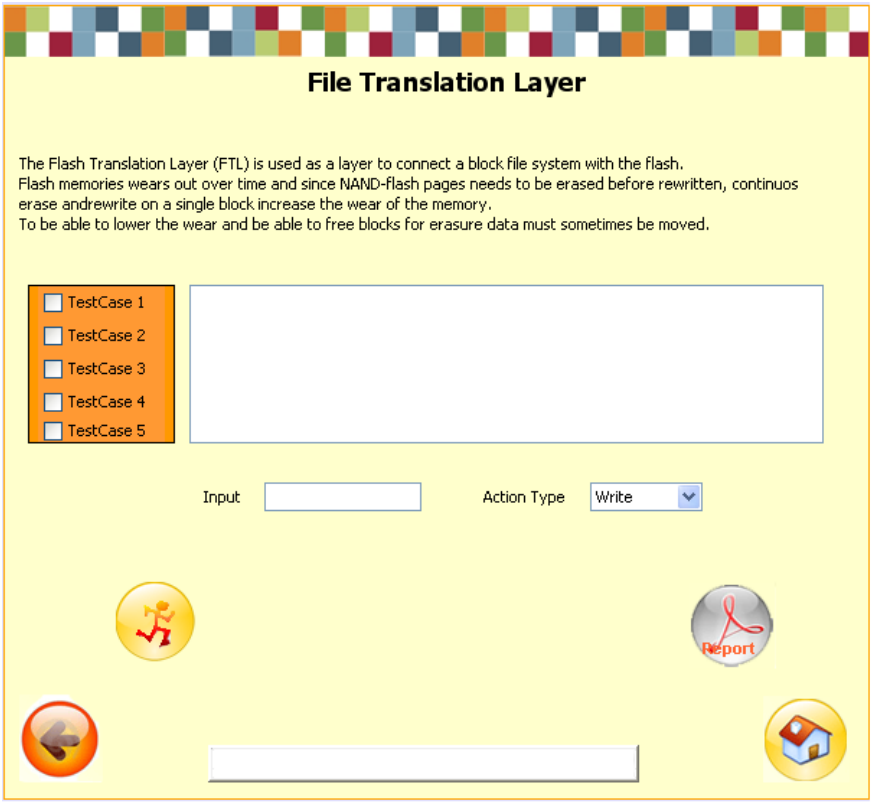

Figure 11: File Translation Layer Page

As you can see in the picture, this layer has five test cases which are related to each other. The test cases must be run in order, that is, test case three requires both test case one and two. The major task that is done by these test cases is to read, write and append operation on files. Each test case has specific limitation on the size of information to work with. The limitation on memory is 512 kb for test case 1. By selecting specific test case in the user interface and selecting a proper Action type, you can run the test case and see the result in the white panel and also you can have the result in PDF format.

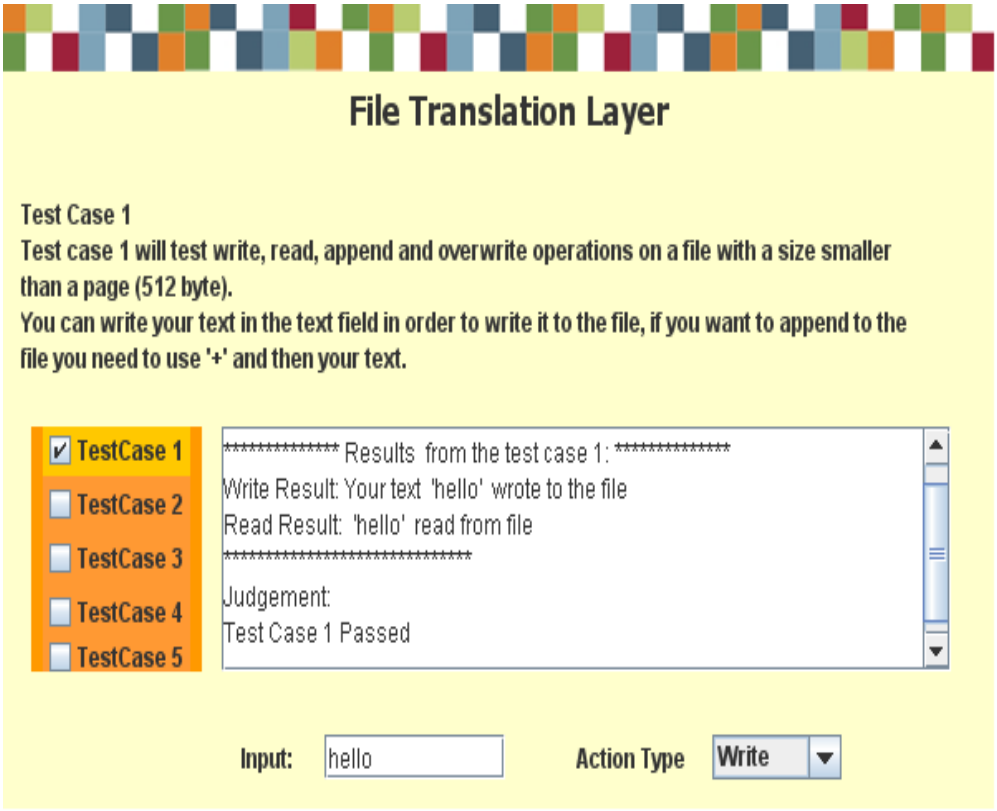

Figure 12: Execution of File Translation Layer Page

5.3.1 Flash translation layer

This test suite focuses on the functionality of the translation layer and the tests clarify if the implemented design meets the basic requirements of a translation layer. The results printouts from the terminal.

Test case 1 tests write, read, append and overwrite operations on a file with a size smaller than a page (<512 byte). Every test case is documented with prerequisites and steps. However I only include these details for the first test case for the purpose of familiarizing the reader with the structure of prerequisites and the detail of the steps.

Prerequisites: No prerequisites needed but if the exactly same results as depicted in the test are required, the flash needs to be restarted and formatted before the start of the test.

Steps: The following steps are manually executed once in chronological order.

Open test file tf1 (Parameter: w+)1
Write data to file
Read data from file
Close test file

1. Print content in file
2. Open test file (Parameter: a+)2
3. Append data to file
4. Read data from file
5. Close test file
6. Print content in file
7. Open test file (Parameter: w+)
8. Write data to file
9. Read data from file
10. Close test file
11. Print content in file

Test case 2 tests write and read operations with a size bigger than a page (>512 byte) and simultaneously makes a stress test with many consecutive read and write operations.

Test case 3 tests various file system commands and their effect on the flash memory.

Test case 4 shows the translation layer list and the garbage collection mechanism.

Test case 5 tests the flash scan function used at reboot.

5.3.2 File Data Caching in JEFF

The purpose of these tests is to verify that the modified version of JEFF, with the file data cache, consumes less energy than the original version. The following system configurations are used for tests:

A. Without file data caching, min cache size (4 blocks)
B. Without file data caching, default cache size (51 blocks)
C. With file data caching, min cache size (8 blocks)
D. With file data caching, default cache size (51 blocks)

Three single file tests (test case 1-3) are performed and evaluated with configuration A and C, i.e. with minimum cache size of 4 and 8 blocks respectively, where 4 blocks are required for metadata operations. These tests are performed with a file size of 4096 bytes, i.e. 8 partition blocks. This means that half of the file will fit in the cache at the same time, given that 4 cache blocks are allocated for metadata operations at all times. One multi file test (test case 4) is run with all configurations, involving two files of size 3072 bytes (6 blocks) and three files of size 2048 bytes (4 blocks). Hence configuration D reflects a case where all files fit into the cache at the same time.

The first test case involves writing sequences smaller than the partition block size sequentially to a file.

Test case 2 writes diverse sized sequences to random parts of a file.

Test case 3 writes sequences of even multiples of the partition block size sequentially to a file. As far as file data block writes goes, it will give the same outcome as if the entire file was written at once.

Test case 4 involves writing, reading and rewriting several files of different sizes randomly. This corresponds to writing each file in the same way as in test case 2, but might not have as good outcome for each individual file, since the writes to one particular block of the file can be interrupted by eviction for writing of another file.

5.3.3 Memory Device Drivers

Test Case 1 writes data to the RAM buffer. This test is important as the first step in writing to the flash is to transfer the data that is to be written to the RAM and then send it to the NAND Flash. After each write, the RAM is read and the data is then printed in the terminal, if an “a” letter is written, figure “61” will be shown, for “b” and “c” the numbers “62” and “63” will be printed respectively. These numbers are the representation of the letters in hexagonal form.

Test case 2 writes the data in the RAM buffer to a selected page. The user is asked to enter the page address to which the data is to be written to. When the user has entered the address the write is executed. After that the user will be asked to specify which page he wants to read. Now the user writes the same address as he stated when data was written in the above description. When the read is completed, the data that was first written to the page should be printed in the command prompt.

Test case 3 erases all data in a block, by replacing all previous information with FFh in all areas. This is tested by first reading data from pages in the same block. After the pages are read, the user will then enter the block that is going to be deleted. The block in which the pages he previous read are located are in is displayed in the prompt. After the erase is completed, the user reads the pages in the newly erased block. The expected result is that all pages that contained data before contains only FFh information.

Test case 4 gathers all the bad blocks, giving errors for those with more than 1-bit faults and saves their addresses in the IBT[10]. When the IBT is created the list of addresses is printed to the command prompt showing all the addresses of the invalid blocks. After that, the functionality of adding and searching in the IBT is tested; this is done by first asking the user

[10] Invalid Block Table

to add an address and secondly by asking him to enter a block address and then answering him if the block is good or bad.

Chapter 6

Conclusion and Discussion

The goal of this thesis was to make optimal technology choices in order to make the application design model as efficient as possible with respect to the company's requirements and limitations. Literature studies, implementation tasks, brainstorming sessions have been performed to address the research questions. In the following there will be a discussion of the results of the research study and then some discussion on future work that can follow this thesis.

6.1 Answering to the research questions

Q1. What are the suitable technologies for implementing the three different layers of the application according to the requirements and delimitations? Technologies are language programming, database management systems, and user interface layer.

Finally at the end of the thesis work a set of suitable technologies to be used as programming language, database and user interface is selected based on iFEST requirements and constraints. For the programming language, Java and Python were the options which are determined by the company. Comparison between these two languages was also based on the iFEST requirement and limitation in the form of a criteria list. The Analytic Hierarchy Process method, AHP, has been used for the comparison which is a muti criteria decision making process. In this way the projects' stakeholders' ideas were taken into account by getting their suggested weights for different criteria. This is achieved by giving weights to different criteria based on the project's requirements that are discussed in the requirement meetings. Finally, Python was proven to be the most suitable language for this project with such circumstances. A similar analysis has been performed for the database and user interface layer. The result of this analysis is that a SQL based database and a web-based interface are more effective in these circumstances.

Q2. What are the alterative integration levels in order to integrate to the rest of the system? In which level should the integration be done according to the requirements and delimitations?

During the thesis work, five different levels of integration which are platform, presentation, data, process, and control were discussed as well as their advantages and disadvantages. Thereafter, the order of importance of comparison criteria was found out according to on the company's requirements and constraints. In order to achieve this, we had requirement meetings with stakeholders to get their idea of what criteria are more important than other criteria. The importance levels of the criteria are afterwards fed into AHP process as weights for criteria when comparing different options. To make sure that the selected option from AHP is based on company's requirement. The important criterions for the company are cost and effort, speed, data integrity, reusability and scalability. By applying the AHP decision model and analysis of the options, the order of importance for different integration levels was figured out for this specific project, which is as follows:

Platform>Process>Control>Presentation>Data

6.2 Future Works

Although a user-interface is designed for Test Manager prototype, it is just a fast developed interface which requires more research on the way to show the test results and stored information to the interface. This interface can be used just as one of the interfaces that are shown to the tester for getting feedback while to reach to the ideal interface there should be few interfaces to know what exactly the tester needs.
So the design of the interface could be a continuation of this thesis by following the ideal technologies which already discussed.

Reference

[1] Enea Group ‘s software http://www.enea.se/software/products/, Retrieved 2011.

[2] ”Industrial Framework for embedded System Tools” http://www.artemis-ifest.eu/technicalscope, Retrieved 2012.

[3] “Multi-Criteria Decision-Making (MCDM)”, http://rfptemplates.technologyevaluation.com/multi-criteria-decision-making-mcdm.html, Retrieved 2013.

[4] HWANG, C. L., YOON, K., “Multiple Attribute Decision Making: Methods and Applications”, Berlin, Springer Verlag, 1981.

[5] Object Technology International, “Eclipse platform technical overview” http://www.eclipse.org/whitepapers/eclipse-overview.pdf, Retrieved 2011.

[6] Zhihui Yang, Jiang, M., “Using Eclipse as a Tool-Integration Platform for Software Development”, IEEE, vol. 24 no. 2, 2007.

[7] Microsooft Developer, http://msdn.microsoft.com/en-us/library/ms973831.aspx, Retrieved 2012.

[8] B. Thomas, I. Nejmeh, “Definition of tool integration for environments”, IEEE Software, vol. 9, no. 2,1992, pp. 29–35.

[9] A. Wasserman, “Software tools: past, present, and future”, IEEE, Software Methods and Tools, 2000. SMT 2000. Proceedings. International Conference on, 2000, pp. 3 –6.

[10] A. I. Wasserman, “Tool integration in software engineering environments in Software Engineering Environments, ed. F. Long. Berlin: Springer Verlag, 1990, pp. 137-149.

[11] Microsoft, “Presentation Integration”, http://msdn.microsoft.com/enus/library/ff649847.aspx, Retrieved 2012.

[12] Miller, D.W.; and M.K. Starr, “Executive Decisions and Operations Research”, Englewood Cliffs, NJ, U.S.A, 1969.

[13] Saaty, T.L., “The Analytic Hierarchy Process”, New York: McGraw Hill. International, Translated to Russian, Portuguese, and Chinese, Revised editions, Paperback (1996, 2000), Pittsburgh: RWS Publications, 1980.

[14] Enea, “Enea Optima Tools Suite User’s Guide”, Delivered with Optima, 2009.

[15] Bojan Srdjevic, “Combining diferent prioritization methods in the analytic hierarchy process synthesis”, Computers & Operations Research, 2005.

[16] Robert E, “Java SE Application Design With MVC”, http://www.oracle.com/technetwork/articles/javase/mvc-136693.html, Retrieved 2011.

[17] E.Triantaphyllou, H.Mann, "Using the analytic hierarchy process for decision making in engineering applications: Some challenges", Inter Journal of Industrial Engineering ,1995.

[18] B.G.Tudorica, C. Bucur, "A comparison between several NoSQL databases with comments and notes" IEEE, 2011.

[19] G.Papamarkos, L.Zamboulis, A.Poulovassilis. "XML Databases", School of Computer Science and Information Systems, Birkbeck College, University of London.

[20] S. Böttcher, A .Türling, "Transaction Synchronization for XML Data in Client-ServerWeb Applications".

[21] Antropova,O. Hollermann,L. Sharma,P. " Presentation Comparing Programming Languages"
http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/comparing/, Retrieved 2013.

[22] Paul Pop, "Comparing Web Applications with Desktop: An empirical study"
http://www.ida.liu.se/labs/eslab/publications/pap/db/hci.pdf, Retrieved 2012.

[23] Daniel Digerås, Integration between Optima and Farkle and verification with a use case about file storage stack integration in a quality of a service manager in OSE, Internal document at ENEA.

[24] Ch,Strauch. "NoSQL Databases"
http://oak.cs.ucla.edu/cs144/handouts/nosqldbs.pdf, Retrieved 2012.

[25] G,Coyle. "THE ANALYTIC HIERARCHY PROCESS",
http://www.booksites.net/download/coyle/student_files/AHP_Technique.pdf, Retrieved 2012.

[26] Strozzi, Carlo: "NoSQL A relational database management system",
http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page, Retrieved 2012.

Appendix 1 Comparison details of Java and Python

Programming Language/Criteria	Efficiency	Resources	Scalability	Security	Usability
Java	**M(4)**	**G(5)**	**G(5)**	**E(7)**	**G(5)**
Python	**VG(6)**	**G(5)**	**VG(6)**	**M(4)**	**VG(6)**

Criteria/Criteria	Efficiency	Resources	Scalability	Security	Usability
Efficiency	**1**	**2,00**	**1,00**	**1,50**	**0,50**
Resources	**0,50**	**1**	**0,50**	**1,50**	**0,33**
Scalability	**1,00**	**2,00**	**1**	**2,00**	**1,00**
Security	**0,67**	**0,67**	**0,50**	**1**	**0,50**
Usability	**2,00**	**3,00**	**1,00**	**2,00**	**1**

Or:

$$M=\begin{bmatrix} 1 & 2,0 & 1 & 1,5 & 0,5 \\ 0,5 & 1 & 0,5 & 1,5 & 0,33 \\ 1 & 2 & 1 & 2 & 1 \\ 0,67 & 0,67 & 0,5 & 1 & 0,5 \\ 2 & 3 & 1 & 2 & 1 \end{bmatrix}$$

Step1:

$$M*M=M^2$$

$$\begin{bmatrix} 1 & 2,0 & 1 & 1,5 & 0,5 \\ 0,5 & 1 & 0,5 & 1,5 & 0,33 \\ 1 & 2 & 1 & 2 & 1 \\ 0,67 & 0,67 & 0,5 & 1 & 0,5 \\ 2 & 3 & 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2,0 & 1 & 1,5 & 0,5 \\ 0,5 & 1 & 0,5 & 1,5 & 0,33 \\ 1 & 2 & 1 & 2 & 1 \\ 0,67 & 0,67 & 0,5 & 1 & 0,5 \\ 2 & 3 & 1 & 2 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 5,0 & 8,5 & 4,25 & 9 & 3,41 \\ 3,16 & 4,99 & 2,58 & 5,41 & 2,16 \\ 6,34 & 10,34 & 5 & 10,5 & 4,16 \\ 3,17 & 5,18 & 2,5 & 5,01 & 2,05 \\ 7,84 & 13,34 & 6,5 & 13,5 & 4,99 \end{bmatrix}$$

Step 2:

Sum of the rows:

$$\begin{bmatrix} 30{,}17 \\ 18{,}31 \\ 36{,}34 \\ 17{,}92 \\ 46{,}17 \end{bmatrix}$$

For normalization all the rows should be divided by the sum of the rows:

30, 17+18, 31+36, 34+17, 92+46,17= **148,91** so the result is:

$$\begin{bmatrix} 0{,}20 \\ 0{,}122 \\ 0{,}24 \\ 0{,}12 \\ 0{,}31 \end{bmatrix}$$

Step 3:

By doing step 1 and 2 again these result is acceptable(four decimal places), accurate Eigenvector obtained:

$$\begin{bmatrix} 0{,}20 \\ 0{,}123 \\ 0{,}24 \\ 0{,}12 \\ 0{,}30 \end{bmatrix}$$

w1=0,20, w2=0,123W3= 0,24, w4=0,120, w5=0,30

0,30>0,24>0,20>0,123>0,120

This means that:

Usability> Scalability> Efficiency> Resources> Security

Based on Efficiency	Java	Python	Eigenvector
Java	1,00	0,67	0,400599
Python	1,50	1,00	0,599401

Based on Resources	Java	Python	Eigenvector
Java	1,00	1,00	0,50
Python	1,00	1,00	0,50

Based on Scalabilty	Java	Python	Eigenvector
Java	1,00	0,83	0,454049
Python	1,20	1,00	0,545951

Based on Security	Java	Python	Eigenvector
Java	1,00	1,75	0,636653
Python	0,57	1,00	0,363347

Based on Usability	Java	Python	Eigenvector
Java	1,00	0,83	0,454049
Python	1,20	1,00	0,545951

$$\begin{bmatrix} 0,40 & 0,50 & 0,45 & 0,63 & 0,45 \\ 0,60 & 0,50 & 0,54 & 0,36 & 0,54 \end{bmatrix} * \begin{bmatrix} 0,20 \\ 0,12 \\ 0,24 \\ 0,12 \\ 0,30 \end{bmatrix} = \begin{bmatrix} 0,45 \\ 0,51 \end{bmatrix}$$

Finally

0,51>0,45 ⇨ Python>Java

Appendix 2 Comparison details of databases

References [19],[20] are used to compare these databases.

Database/Criteria	Interoperability	non-redundancy	Data consistency	Data exchange	Synchronization
Text File	T(1)	VB(2)	T(1)	M(4)	T(1)
MySql(SQL based)	G(5)	E(7)	E(7)	G(5)	G(5)
Casandra(Column Storage)	B(3)	B(3)	M(4)	B(3)	E(7)
CouchDB(Document Storage)	G(5)	B(3)	G(5)	B(3)	E(7)
XML Database (Key-Value)	G(5)	VB(2)	T(1)	G(5)	B(1)

Criteria/Criteria	Interoperability	non-redundancy	Data consistency	Data exchange	Synchronization
Interoperability	1	3	3	2	1
non-redundancy	0,33	1,00	1,00	0,50	0,50
Data consistency	0,33	1,00	1,00	0,50	0,33
Data exchange	0,50	2,00	2,00	1,00	1,00
Synchronization	1,00	2,00	3,00	1,00	1,00

$$M=\begin{bmatrix} 1 & 3,0 & 3 & 2 & 1 \\ 0,33 & 1 & 1 & 0,5 & 0,5 \\ 0,33 & 1 & 1 & 0,5 & 0,33 \\ 0,5 & 2 & 2 & 1 & 1 \\ 1 & 2 & 3 & 1 & 1 \end{bmatrix}$$

Step 1:

$$M*M=M^2$$

$$\begin{bmatrix} 1 & 3,0 & 3 & 2 & 1 \\ 0,33 & 1 & 1 & 0,5 & 0,5 \\ 0,33 & 1 & 1 & 0,5 & 0,33 \\ 0,5 & 2 & 2 & 1 & 1 \\ 1 & 2 & 3 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 3,0 & 3 & 2 & 1 \\ 0,33 & 1 & 1 & 0,5 & 0,5 \\ 0,33 & 1 & 1 & 0,5 & 0,33 \\ 0,5 & 2 & 2 & 1 & 1 \\ 1 & 2 & 3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4,98 & 15 & 16 & 8 & 6,49 \\ 1,74 & 4,99 & 5,49 & 2,66 & 2,16 \\ 1,57 & 4,65 & 4,98 & 2,49 & 1,99 \\ 3,32 & 9,5 & 10,5 & 5 & 4,16 \\ 4,15 & 12 & 13 & 6,5 & 4,99 \end{bmatrix} =$$

Step 2:

Sum of the rows:

$$\begin{bmatrix} 50,47 \\ 17,04 \\ 15,68 \\ 32,48 \\ 40,64 \end{bmatrix}$$

For normalization all the rows should be divided into sum of the rows:

50, 47+17, 04+15,68+32,48,+40,64= **156,31 so the result is:**

$$\begin{bmatrix} 0,3228 \\ 0,1090 \\ 0,1003 \\ 0,2077 \\ 0,2599 \end{bmatrix}$$

Step 3:

By doing step 1 and 2 again these result is acceptable (based on four decimal places), accurate Eigenvector obtained:

$$\begin{bmatrix} 0,322 \\ 0,109 \\ 0,100 \\ 0,207 \\ 0,259 \end{bmatrix}$$

W1=0,322>w5=0,259>w4=0,207>w2=109>w3=0,100 ⇨

Interoperability>Synchronization>Data exchange> Non-redundancy>Data Consistency

Based on Interoperability	Text File	MySql(SQL based)	Casandra(Column Storage)	CouchDB(Document Storage)	XML Database (Key-Vlue)	Eigenvector
Text File	1,00	0,20	0,33	0,20	0,20	0,05
MySql(SQL based)	5,00	1,00	1,67	1,00	1,00	0,26
Casandra(Column Storage)	3,00	0,60	1,00	0,60	0,60	0,157
CouchDB(Document Storage)	5,00	1,00	1,67	1,00	1,00	0,26
XML Database (Key-Value)	5,00	1,00	1,67	1,00	1,00	0,26

Based on non-redundancy	Text File	MySql(SQL based)	Casandra(Column Storage)	CouchDB(Document Storage)	XML Database (Key-Value)	Eigenvector
Text File	1,00	0,29	0,67	0,67	1,00	0,11
MySql(SQL based)	3,50	1,00	2,33	2,33	3,50	0,41
Casandra(Column Storage)	1,50	0,43	1,00	1,00	1,50	0,17
CouchDB(Document Storage)	1,50	0,43	1,00	1,00	1,50	0,17
XML Database (Key-Value)	1,00	0,29	0,67	0,67	1,00	0,11

Based on Data-consistency	Text File	MySql(SQL based)	Casandra(Column Storage)	CouchDB(Document Storage)	XML Database (Key-Value)	Eigenvector
Text File	1,00	0,14	0,25	0,20	1,00	0,07
MySql(SQL based)	7,00	1,00	1,75	1,40	7,00	0,53
Casandra(Column Storage)	4,00	0,57	1,00	0,80	7,00	0,34
CouchDB(Document Storage)	5,00	0,71	1,25	1,00	5,00	0,38
XML Database (Key-Value)	1,00	0,14	0,14	0,20	1,00	0,06

Based on Data-exchange	Text File	MySql(SQL based)	Casandra(Column Storage)	CouchDB(Document Storage)	XML Database (Key-Value)	Eigenvector
Text File	1,00	0,80	1,33	1,33	0,80	0,16
MySql(SQL based)	1,25	1,00	1,67	1,67	1,00	0,21
Casandra(Column Storage)	0,75	0,60	1,00	1,00	0,60	0,12
CouchDB(Document Storage)	0,75	0,60	1,00	1,00	0,60	0,12
XML Database (Key-Value)	1,25	1,00	1,67	1,67	1,00	0,21

Based on Synchronization	Text File	MySql(SQL based)	Casandra(Column Storage)	CouchDB(Document Storage)	XML Database (Key-Value)	Eigenvector
Text File	1,00	0,20	0,14	0,14	1,00	0,06
MySql(SQL based)	5,00	1,00	0,71	0,71	5,00	0,33
Casandra(Column Storage)	7,00	1,40	1,00	1,00	7,00	0,46
CouchDB(Document Storage)	7,00	1,40	1,00	1,00	7,00	0,46
XML Database (Key-Value)	1,00	0,20	0,14	0,14	1,00	0,06

Finally

$$\begin{bmatrix} 0,05 & 0,11 & 0,05 & 0,20 & 0,05 \\ 0,26 & 0,41 & 0,38 & 0,25 & 0,24 \\ 0,15 & 0,17 & 0,24 & 0,15 & 0,33 \\ 0,26 & 0,17 & 0,27 & 0,15 & 0,33 \\ 0,26 & 0,11 & 0,05 & 0,25 & 0,05 \end{bmatrix} * \begin{bmatrix} 0,322 \\ 0,109 \\ 0,100 \\ 0,207 \\ 0,259 \end{bmatrix} = \begin{bmatrix} 0,08 \\ 0,31 \\ 0,252 \\ 0,29 \\ 0,16 \end{bmatrix}$$

⇨

W2=0,31>w4=0,29>w3=0,252>w5=0,16>w1=0,08

SQL>CouchDB>Casandra>XML Database>Text File

Appendix 3 Comparison details of desktop and web based interface

	Ubiquitousity	Look & Feel	Performnance	Integration	Maintenance
Desktop	T(1)	G(5)	G(5)	M(4)	T(1)
Web	E(7)	N(4)	B(3)	VG(6)	G(5)

Criteria/Criteria	Ubiquitousity	Look & Feel	Performnance	Integration	Maintenance
Ubiquitousity	1	0,5	0,33333333	0,25	0,5
Look & Feel	2	1	0,5	0,33333333	0,5
Performnance	3	2	1	0,5	1
Integration	4	3	2	1	2
Maintenance	2	2	1	0,5	1

$$M=\begin{bmatrix} 1 & 0,5 & 0,33 & 0,25 & 0,5 \\ 2 & 1 & 0,5 & 0,33 & 0,5 \\ 3 & 2 & 1 & 0,5 & 1 \\ 4 & 3 & 2 & 1 & 2 \\ 2 & 2 & 1 & 0,5 & 1 \end{bmatrix}$$

Step 1:

$$\begin{bmatrix} 1 & 0,5 & 0,33 & 0,25 & 0,5 \\ 2 & 1 & 0,5 & 0,33 & 0,5 \\ 3 & 2 & 1 & 0,5 & 1 \\ 4 & 3 & 2 & 1 & 2 \\ 2 & 2 & 1 & 0,5 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0,5 & 0,33 & 0,25 & 0,5 \\ 2 & 1 & 0,5 & 0,33 & 0,5 \\ 3 & 2 & 1 & 0,5 & 1 \\ 4 & 3 & 2 & 1 & 2 \\ 2 & 2 & 1 & 0,5 & 1 \end{bmatrix} = \begin{bmatrix} 4,99 & 3,41 & 1,91 & 1,08 & 2,08 \\ 7,83 & 4,99 & 2,83 & 1,66 & 3,16 \\ 14 & 9 & 4,99 & 2,91 & 5,5 \\ 24 & 16 & 8,83 & 4,99 & 9,5 \\ 13 & 8,5 & 4,66 & 2,66 & 5 \end{bmatrix}$$

Step 2:

Sum of the rows:

$$\begin{bmatrix} 13,49 \\ 20,49 \\ 36,41 \\ 63,33 \\ 33,83 \end{bmatrix}$$

For normalization all the rows should be divided into sum of the rows:

13, 49+20, 49+36,41+63,33,+33,83= **167,58** so the result is:

$$\begin{bmatrix} 0{,}0805 \\ 0{,}1223 \\ 0{,}2173 \\ 0{,}3779 \\ 0{,}2018 \end{bmatrix}$$

Step 3:

By doing step 1 and 2 again these result is acceptable (four decimal places), accurate Eigenvector obtained:

$$\begin{bmatrix} 0{,}08 \\ 0{,}12 \\ 0{,}21 \\ 0{,}37 \\ 0{,}20 \end{bmatrix}$$

W4=0,37>w3=0,21>w5=0,20>w2=0,12>w1=0,08

Integration>Performance>Maintenance>Look and feel>Ubiquitously

Based on Usability	Desktop	Web	Eigenvector	Normalized
Desktop	1,00	0,14	0,123904	0,13
Web	7,00	1,00	0,876096	0,88

Based on Look and feel	Desktop	Web	Eigenvector	Normalized
Desktop	1,00	1,25	0,555556	0,56
Web	0,80	1,00	0,444444	0,44

Based on performance	Desktop	Web	Eigenvector	Normalized
Desktop	1,00	1,67	0,625234	0,63
Web	0,60	1,00	0,374766	0,38

Based on performance	Desktop	Web	Eigenvector	Normalized
Desktop	1,00	0,67	0,400599	0,40
Web	1,50	1,00	0,599401	0,60

Based on performance	Desktop	Web	Eigenvector	Normalized
Desktop	1,00	0,20	0,166667	0,17
Web	5,00	1,00	0,833333	0,83

$$\begin{bmatrix} 0{,}12 & 0{,}55 & 0{,}62 & 0{,}40 & 0{,}16 \\ 0{,}87 & 0{,}44 & 0{,}37 & 0{,}59 & 0{,}83 \end{bmatrix} * \begin{bmatrix} 0{,}08 \\ 0{,}12 \\ 0{,}21 \\ 0{,}37 \\ 0{,}20 \end{bmatrix} = \begin{bmatrix} 0{,}38 \\ 0{,}59 \end{bmatrix}$$

Web>Desktop

Appendix 4 Requirements

As mentioned in this report, choosing suitable technologies based on project requirements is the problem that was solved in this thesis. These requirements were obtained based on brain storming sessions. These requirements are described in this appendix.

Final requirements for integration models:

1- Speed
2- Cost and effort
3- Reuse
4- Data Integrity
5- Scalability

Final requirements for programming language:

1- Efficiency
2- Resources
3- Scalability
4- Security
5- Usability

Final requirements for database:

1- Interoperability
2- Non-redundancy
3- Data consistency
4- Data exchange
5- Synchronization

Final requirements for user interface:

1- Ubiquitousity

2- Look & Feel

3- Performance

4- Integration

5- Maintenance

TRITA-ICT-EX-2014:30

www.kth.se