Faculdade de Engenharia da Universidade do Porto

FATAdesk - Report Center

José Miguel Campos

Tese submetida no Âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major de Telecomunicações

Orientador: Luis Paulo Reis (Professor)

Julho de 2008

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"FATADesk - Report Center Module"

foi aprovada em provas realizadas 18/Julho/2008

o júri

Presidente Professor Doutor António Augusto de Sousa
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Fernando Augusto Cruz e Silva Mouta
Professor Coordenador do Instituto Superior de Engenharia do Porto

Professor Doutor Luis Paulo Gonçalves dos Reis
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - José Miguel Neves Leão de Campos

Faculdade de Engenharia da Universidade do Porto

Resumo

O projecto "FATAdesk" consiste num conjunto de programas de software, com o objectivo de auxiliar e automatizar várias tarefas do Test Program Engineers (*TPE*). Este projecto é necessário pois o workflow, dos *TPE*, contém várias fases repetitivas e demoradas. Assim também é possível melhorar o tempo de resposta aos pedidos de alterações aos testes.

O "Report Center" é uma aplicação web que irá integrar no projecto "FATAdesk". A sua principal função é auxiliar os *TPE* a analisar vários resultados de programas de teste. Esta aplicação pode ser dividida em módulos. O primeiro a ser desenvolvido foi o "Device Data". Este foi desenhado para analisar dados do ficheiro device data, que contêm resultados lógicos dos testes de um Flow, e criar relatórios, como por exemplo o "Flow Report".

O segundo módulo a ser desenvolvido foi o módulo "Transfer Letter". Quando são realizadas alterações nos Testes é implementada uma nova Versão, que consiste num conjunto de Flows. Ao transferir a Versão para a produção, é necessário acompanhar um documento, a "Transfer Letter". Este módulo irá auxiliar no preenchimento e na geração deste documento.

O último módulo a ser desenvolvido foi o "Test Times". O seu objectivo é permitir os *TPE* analisar e comparar mais facilmente, tempos de execução de Flows e Testes.

Depois da fase de desenvolvimento os engenheiros da "Qimonda Portugal" foram pedidos para responderem um inquérito, de forma a avaliarem e compararem as funcionalidades da aplicação com métodos de trabalho equivalentes. Depois de analisar os resultados, foi possível observar que as funcionalidades implementadas obtiveram um resultado positivo. Todas ultrapassaram os métodos anteriores em quase todas as propriedades. No entanto, e embora o trabalho desenvolvido seja de elevada qualidade, existe ainda melhorias a realizar.

Abstract

The "FATAdesk" is a project currently in developing in "Qimonda Portugal". It consists in several software programs, with a common purpose, to automate and help several tasks for the Test Product Engineers (*TPE*). This project is very important, since several steps of the *TPE* workflow are repetitive and time consuming. It will also increase the responsiveness to change requests.

The "Report Center" is a web application that will be included in "FATAdesk" project. Its main purpose is to aid the *TPE* analyze several program test results. This application can be broken down into modules. The first one to be developed was the "Device Data". It was designed to analyze the data from device data files, containing logic results of tests contained in a Flow, and create reports, such as "Flow Report".

The second module to be developed was the "Transfer Letter". When Test changes are made a new Version is implemented, containing several flows. When a Version is transferred to production, it must be accompanied by a document, the "Transfer Letter". This module will aid the *TPE* to complete this document and generate a PDF file.

The final module to be developed was the "Test Times". Its purpose is to allow the *TPE* to analyze and compare more easily, Flows and Tests execution times.

After the development stage the engineers from "Qimonda Portugal" were asked to answer a survey, in order to, evaluate and compare the application features against equivalent work methods. After analyzing the results from the survey, it is possible to observe that the implemented tools had a very positive feedback from its users. Almost all of its features surpassed the previous methods in the survey results. However, even though the developed work was evaluated as having a high quality, there is still room for improvements on several topics.

Acknowledgment

Although the work presented where was developed individually, there are several direct and indirect contributions that must not be overlooked. For that I wish to express my deepest thanks to Prof. Luís Paulo Reis, professor and my adviser from Faculdade de Engenharia da Universidade do Porto, for his guidance when writing this thesis.

Also I would like to thank José Machado, engineer and my adviser from Qimonda Portugal, without him this thesis would be incomplete.

My colleagues from CoC department for their incredible spirit and support, my thanks to them too.

And last but not least my deepest thanks and love to my family and friends for their amazing support and friendship.

José Miguel Campos

"The whole problem with the world is that fools and fanatics are always so certain of themselves, but wiser people so full of doubts. "

Bertrand Russell

Contents

List of Figures

List of Tables

Glossary

Devices - A part of electronic hardware, with the ability to store electronic data. These Devices can then be putted together and build what is commonly known as RAM modules.
Product - A product is an group of *Devices* with similar characteristics. These are characteristic are "Ram type" (DDR2), "Memory size" (512 Megabits) and "Chip size" (90 nm).
Organization - This term refers to the transfer size of a device. I can be 4, 8 or 16 bits.
Test - A software program, designed to test the quality of Devices.
Flow - A set of test with a common objective.
Version - A set of Flows.
Tester - Machine with the sole purpose to run Tests on several devices.
Parallelisms - This term refers to the number of trays a tester can test at the same time.
TPE - Test Product Engineer. The TPE are responsible for the creation/updates of test programs.
ORDBMS - Object-relational database management system.
SQL - Structured Query Language.
PL/pgSQL - Procedural Language/PostgreSQL Structured Query Language. Programming language that allows a much more procedural control than SQL.
View - A view is a virtual or logical table composed of the result set of a query.
Transactions - Transactions provide an "all-or-nothing" proposition stating that work units performed in a database must be completed in their entirety or take no effect whatsoever.
AJAX - Asynchronous JavaScript and XML.
URL - Uniform Resource Locator.
HTML - HyperText Markup Language.
XML - Extensible Markup Language.
CSS - Cascading Style Sheets.
DOM - Document Object Model.
TPCR - Test Program Change Request.
TPL - Test Program Logistic. Has the same purpose of a TPCR, but the change request affects the logistic part of a test.
Bug Fix - When a Bug is detected in a program test, a Bug fix must be implemented.

Chapter 1

Introduction

1.1 Qimonda

Qimonda is a leading global memory supplier with a broad diversified DRAM product portfolio. The company generated net sales of €3.61 billion in its 2007 financial year and has approximately 13,500 employees worldwide. Qimonda has access to five 300mm manufacturing sites on three continents and operates six major R&D facilities. The company provides DRAM products for a wide variety of applications, including in the computing, infrastructure, graphics, mobile and consumer areas, using its power saving technologies and designs.

The company in Portugal is one of the most advanced backend units. This is due to the flexible production and logic oriented to the clients needs. Since 7% of the worldwide *SDRAM* volume is produced in this site, it is one of the most important national exporters.

Although Qimonda has a work flow that enables these facts some steps can be very challenging. Presently Test Program Engineers (henceforth *TPE*) in the Center of Competence (*CoC*) department need to analyze ASCII formatted files, with enormous amount of information. To help the *TPE* overcome these stressful and error prone steps, a tool that can analyze the files and display reports is needed.

1.2 TPE Workflow

The *TPE* are required as to develop and maintain test programs for memories. These programs need to be very thorough and yet also run in the least amount of time.

They can only alter or create new tests, when they receive an approved request. They then proceed to start developing a new *Version* that implements one or more requests. Naturally, before it can be used in the production line, it must first be tested.

For the *TPE* to test if their new *Version* is working as intended, they must run it on previous selected *Devices*. This will output several ASCII formated files, that the *TPE* need to analyze (two of them can be seen in Chapters 1.2.1 and 1.2.2). These files include logic results, analogue results and test times. Since these file often have a very complex format, analyzing them thoroughly can be a difficult task.

Finally, after the *Version* has been tested, it is sent to the Production Department along with a document, the Transfer Letter. This document will contain several information, necessary for the *Version's* implementation in the production *Testers*.

1.2.1 Device Data File

The "Device Data" file contains logic results (Pass/Fail) of test program that were executed in the devices. This file contains all the information that involves the test execution conditions and, for each tested *Device*, register its identification and the results for each test.

In figure 1.1 an excerpt from this file, can be seen. The header, with logistic information for the tests execution, can be easily identified. The test names are written vertically, and aligned with it is the *Device* result for that test ("F" for fail and "." for pass). Further details can be seen in [1].

```
C0710
ZA123456.78
L1212345
T80 DD1   512M
A----
11---
EPFTES-38
M6771-75        ,M6771-76
#0004                 ,#FFFF
#0004:-----           ,#FFFF:-----
#D103:#D104:-----:-----,#FFFF:#FFFF:-----:-----
U6YH8XBS-3S     REV: V19

U6TC20
0----
TEMPERATURE: +095C
OFFSET1,OFFSET2: +3.750NS,+0.350NS
DATE: 20070621 TIME: 162812
barrosru
COMMENT
LOTTYPE
LOTSOURCE
LOTGRADE
0
000000000
000000000
00000000
------,------,T667S ,------,CONT-F,SPCL-S,PARA-F,OTHERs
TNO:                                                                 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                                                                     0000000001111111111222222222233333333334444444444555555555566666666667777777777888
                                                                     1234567890123456789012345678901234567890123456789012345678901234567890123456789012
CNO:                                                                 0000000000000000000000000000000000000000000000000000000000000000000000000000000110
                                                                     2221000111113333333444444414455555555555666666666667777777777788888888888999999999000
                                                                     1234549326783456789012345608901234567890123456789012345678901234567890123456789017
                                                                     CCCDILPNVIINIIIIIIIIIIIIIIBLHLLLLLHLLLIIIIIIIIIIIIIIOOLLLLLLLHHHHHHHLLLLHHHHRLDPTX
                                                                     HMMEDORCFLLCDDDDDDDDDDDDDDFSSMSMDSSMDDDDDDDDDDDDDDDDDDDQSASLLDQSASLLOOOOIIIIWDMOTC
                                                                     IPPVSTFW --SDDDDDDDDDDDDDDC44455566666MMMMMMMMMMMMMMTTQHCCCZZQHCCCZZSSHHSSHHDQLWRH
                                                                     PTTIOISA HLH0122233344556 7N33344455555012223334455676 6SFKNKNXSFKNKNX66666666XTKDSI
                                                                     IHLNRNRF IOR  PNQPPNRWBD   33344455545  PNQPPNRWBD   6PF6N6X66F6N6X6666666666YGGNRP
                                                                     DIOTTTTR CCT     01         33344555555     01        7D6 6 6   6 6 6  7S7S7S7S3L2 T
                                                                     0006655657757777777777777778888888888888888888888888888888888888888888888888888868
                                                                     0000000000000000000000000000000000000000000000000000000000000000000000000000000000
N     :TOTDT :DUT:R:CI:LOTID   :DESIGN      :WAFNUM:X :Y :PS:PLT:SR:CAT:RESULT                                                                           :IORESULT :E:EFINFO
000001:000032:001:0:00:AAE11161:512M T11 DD1G   :01:29:18:03:SMC:05:012:--------F.F.---------------FFFFFFFFFF--.---------.-FFFFFFFFFFFFFFFFFFFFFFFFFFFF--:#000007FF: :
000002:000032:002:0:00:AAE11458:512M T11 DD1G   :11:25:19:03:SMC:05:012:--------F...---------------FFFFFFFFFF--.---------.-FFFFFFFFFFFFFFFFFFFFFFFFFFFF--:#000007FF: :
000027:000032:027:0:00:3A707188:512M T80 02     :15:30:14:02:DRS:03:000:--------...---------------..........--.---------.-............................--:#00000000: :
000028:000032:028:0:00:3A549194:512M T90 12     :20:18:31:02:DRS:03:000:--------...---------------..........--.---------.-............................--:#00000000: :
```

Figure 1.1: Excerpt from a "Device Data" file

1.2.2 Test Time Files

The "Test Time Sums" and "Test Time Details" are two files that contain information about *Flows* and *Tests* execution time, respectively. The "Test Time Details" contains details of all the *Tests* execution times, of all the *Flows* of a *Version*. Further details can be seen in [1].

```
 PROGRAM        ; SUM TTIME(S)
U6TH4XBS        ;      9.79S
U6TH4XBS-4      ;     19.10S
U6TH8XBS        ;      7.26S
U6TH8XBS-4      ;     12.60S
U6TH6XBS        ;      6.33S
U6TH6XBS-4      ;      9.81S
```

Figure 1.2: Excerpt from a "Test Times Sums" file

```
 PROGRAM       ; NUMBER;  NAME;SUM TTIME(S);  INITIAL(S);   PG RUN(S);    FINAL(S);    TOTAL(S);MON;RT;ACT

U6TH4XBS       ; OVERHD;TDWINI;   560.00MS ;     0.00S  ;     0.00S  ;   560.00MS ;   560.00MS ;   ; 0;
U6TH4XBS       ; OVERHD;SEQINI;   720.00MS ;     0.00S  ;     0.00S  ;   160.00MS ;   160.00MS ;   ; 0;
U6TH4XBS       ;      1;CHIPID;     1.63S  ;   360.00MS ;   550.00MS ;     0.00S  ;   910.00MS ;   ; 0;  3
U6TH4XBS       ;      2;DEVINT;     1.65S  ;    20.00MS ;     0.00S  ;     0.00S  ;    20.00MS ;   ; 0;  0
U6TH4XBS       ;      3;IDSORT;     1.67S  ;    20.00MS ;     0.00S  ;     0.00S  ;    20.00MS ;   ; 0;  0
U6TH4XBS       ;      4;LOTINT;     1.69S  ;    20.00MS ;     0.00S  ;     0.00S  ;    20.00MS ;   ; 0;  0
U6TH4XBS       ;      5;DESINT;     1.72S  ;    20.00MS ;    10.00MS ;     0.00S  ;    30.00MS ;   ; 0;  3
U6TH4XBS       ;      6;NCWAFR;     1.74S  ;    20.00MS ;     0.00S  ;     0.00S  ;    20.00MS ;   ; 0;  0
U6TH4XBS       ;      7;VF    ;     2.14S  ;    50.00MS ;   330.00MS ;    20.00MS ;   400.00MS ;   ; 0;  3
U6TH4XBS       ;      8;IL-HIC;     2.56S  ;   140.00MS ;   250.00MS ;    30.00MS ;   420.00MS ;   ; 0;  3
U6TH4XBS       ;      9;IL-LOC;     2.94S  ;   100.00MS ;   250.00MS ;    30.00MS ;   380.00MS ;   ; 0;  3
U6TH4XBS       ;     10;NCSHRT;     3.07S  ;    30.00MS ;   100.00MS ;     0.00S  ;   130.00MS ;   ; 0;  3
U6TH4XBS       ;     11;BBTPO ;     3.63S  ;    70.00MS ;   470.00MS ;    20.00MS ;   560.00MS ;   ; 0;  3
U6TH4XBS       ;     13;LS4333;     8.41S  ;    60.00MS ;     4.70S  ;    20.00MS ;     4.78S  ;   ; 0;  3
U6TH4XBS       ;     14;LM4333;     8.61S  ;    60.00MS ;   120.00MS ;    20.00MS ;   200.00MS ;   ; 0;  3
U6TH4XBS       ;     15;LSCKN ;     8.77S  ;    60.00MS ;    80.00MS ;    20.00MS ;   160.00MS ;   ; 0;  3
U6TH4XBS       ;     16;LSCKX ;     8.95S  ;    60.00MS ;   100.00MS ;    20.00MS ;   180.00MS ;   ; 0;  3
U6TH4XBS       ;     17;ODT400;     9.13S  ;    60.00MS ;   100.00MS ;    20.00MS ;   180.00MS ;   ; 0;  3
U6TH4XBS       ;     18;HSCKN ;     9.27S  ;    60.00MS ;    60.00MS ;    20.00MS ;   140.00MS ;   ; 0;  3
U6TH4XBS       ;     19;HSCKX ;     9.42S  ;    60.00MS ;    70.00MS ;    20.00MS ;   150.00MS ;   ; 0;  3
U6TH4XBS       ;     20;XCHIP ;     9.44S  ;    20.00MS ;     0.00S  ;     0.00S  ;    20.00MS ;   ; 0;  0
U6TH4XBS       ; OVERHD;SORT  ;     9.79S  ;     0.00S  ;     0.00S  ;   350.00MS ;   350.00MS ;   ; 0;
U6TH4XBS       ; OVERHD;TBTIMT;     9.84S  ;     0.00S  ;     0.00S  ;    50.00MS ;    50.00MS ;   ; 0;
U6TH4XBS       ; OVERHD;DCLOG ;     9.84S  ;     0.00S  ;     0.00S  ;     0.00S  ;     0.00S  ;   ; 0;
U6TH4XBS       ; OVERHD;TDWEND;     9.94S  ;     0.00S  ;     0.00S  ;   100.00MS ;   100.00MS ;   ; 0;
```

Figure 1.3: Excerpt from a "Test Times Details" file

1.3 Report Center Purpose

The main purpose of the *Report Center* is to aid the *TPE* to analyze the files described in Chapter 1.2. This should be achieved through the creation of accessible and readable reports. Also by creating report with statistics and other informations, that can not be retrieved by merely analyzing the files, should aid even further the *TPE*.

To achieve the best outcome, the *Report Center* was developed with several concerns in mind.

- The user interface must as simple as possible.
- The performance of the tool must be in acceptable levels.
- The tool must have a very high level of maintainability.

1.4 Outline

This thesis is organized in five chapter. The first chapter contains an introduction to Qimonda Portugal, the Report Center and how it fits in the TPE workflow.

Chapter 2 contains a brief description of the all the technologies used and an explanation of why they were used.

A description of the implementation methods will be given in Chapter 3.

In Chapter 4 will present most of the implemented features.

Chapter 5 contains an evaluation will be made to the project. Also an explanation the evaluation project is presented.

Finally in Chapter 6 the conclusions from the work will be presented. A brief list of topics of future work is also suggested.

Chapter 2

Technology Review

In the development of the *Report Center* several technologies where used. As the backbone of any website a database and a web programming language are needed. They must be efficient, easy to maintain and also be freeware. The technologies used to fulfill these roles were *PostgreSQL* and *PHP*[2] (for further information see Chapters 2.1 and 2.2).

Also a technology that increases the interactivity between the user and the web site is required. The reasoning behind this requirement, is to create an user friendly interface and also increase its response time. *Javascript*[3] is used for this effect due to its easy manipulation of the *DOM* object and also because of its asynchronous communication capability (commonly known as *AJAX*).

2.1 PostgreSQL

PostgreSQL is an object-relational database management system (*ORDBMS*). An *ORDBMS* is a management system similar to a relational database that allows a developer the possibility to integrate their own data types and methods.

Since *PostgreSQL* is an *Open Source* software, it allows a large community to develop, maintain and correct it more thoroughly. Also, as shown in the case study of Web Commerce Group [4], considering its technology and total cost, *PostgreSQL* is a very competitive even against commercial *ORDBMS*.

It is also possible to extend PostgreSQL functionalities [5]. For instance, by creating functions in PL/pgSQL, its is possible to do SQL queries that return results that were otherwise impossible or extremely difficult. Also with its C and C++ API's, it allows the creation of applications that interact with PostgreSQL.

Advantages	Disadvantages
Performance - Its features and speed can be competitively compared to commercial databases,	**Slower than MySQL** - Since PostgreSQL is tweaked by default for compatibility
Freeware - PostgreSQL is a free software. Its usage in a commercial project has no cost.	
Expandable - Through add-ons and the use of PL/pgSQL new features can be added.	

Table 2.1: PostgreSQL Pros and Cons

2.2 PHP

PHP *(PHP: Hypertext Preprocessor)* [6] is a computer scripting language, originally designed for producing dynamic web pages due to its server-side scripting capability. It was originally created by Rasmus Lerdorf in 1994, but its main implementation is now produced by *The PHP Group* and serves as the de facto standard[1] for PHP as there is no formal specification.

PHP offers several features in it core build, such as database integration and FTP access [6]. It is possible to extend the number of features of *PHP* by dynamically load user made extensions, or compile *PHP* with these extensions.

Advantages	Disadvantages
Freeware - PHP is a free software. Unlike other frameworks, it brings no extra cost when developing an web page.	**Execution speed** - Since PHP is an interpreted language its execution speed is slower when compared to an compiled language.
Portability - When there is a need to change the web page to a different system, the only concern is to place it in the right directory.	**Open source** - Being an interpreted language anyone who has access to the host server can view the source code.

Table 2.2: PHP Pros and Cons

This project used this technology because of its cost, portability and its intuitive syntax. Also PHP is solid choice when developing an web page, since it is used by twenty two million web sites [7]. With such a huge community it is most likely that, for most problems, its solution has already been found and is available to everyone.

2.2.1 Smarty

Smarty is a PHP class that functions as an web template system. Smarty is primarily promoted as a tool for separation of concerns, helping in the creation of an layered application (further detail can be seen in Chapter 3).

[1] Standard (formal or informal) that has achieved a dominant and accepted position.

Smarty generates web content by the placement of special Smarty tags within a document. Here is on example of a possible template.

```
<html>
  <head>
    <title>{$title_text}</title>
  </head>
  <body>
    <p>{$body_text}</p>
  </body>
</html>
```

The template is then compiled by Smarty and these tags are substituted with PHP code. This compilation can be done in runtime, or the results can be cached, according to its configurations.

This technology is used in this project since the separation between the interface and the business logic bring enormous advantages. It is possible to create easily read templates, and thus, improves both maintainability and development of the project. It decreases development time due to its built-in functions. These function can generate several HTML source code with only a few Smarty tags. More information about Smarty's features can be seen in [8]

2.2.2 ePDF

The Ezpdf Class[9] is an PHP class that allows the user to build an object and output the data into a PDF file. This allows to dynamically create PDF files from data retrieve from a database.

This class was used instead of the FPDF Class[10][11] since its functions are more user friendly, and still enables full interaction with of the PDF object. Another advantage was the use of "Transactions" identical to their counterpart in SQL.

2.3 Javascript

JavaScript is a scripting language that is most often used in web development. It was developed by Breadan Eich with that intent. This is supported by the fact that it can be embedded or included in HTML and its interaction with the *Document Object Model* (DOM).

Advantages	Disadvantages
Responsive - Since JavaScript runs on the client side it make the web page more responsive.	**Compatibility** - Although ECMA-262[12] standard exist, there are browser that do not fully comply to it.
Good DOM[13] interaction - JavaScript allows an easy manipulation of the DOM, making possible to enhance the dynamics of a web page.	**Security** - JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the web.
AJAX - JavaScript can create an object that can communicate asynchronously with a server (see Chapter 2.3.1 for further detail).	

Table 2.3: JavaScript Pros and Cons

The use of this programing language was needed due to its "unique" ability to run on the client side. There is a similar language, *VBScript*[14] but it is only compatible with Internet Explorer. Also with JavaScript interaction with the *AJAX* object and *DOM*[13], a web developer can greatly increase a web page responsiveness and its interaction with the user.

2.3.1 AJAX

AJAX (Asynchronous JavaScript and XML) is a group of inter-related web development techniques used for creating interactive web applications. A primary feature its usage brings is the increased responsiveness and interactivity of web pages achieved by exchanging small amounts of data with the server "behind the scenes". This way the entire web pages does not have to be reloaded each time there is a need to fetch data from the server. This is intended to increase the web page's interactivity, speed, functionality and usability.

AJAX is asynchronous, in that extra data is requested from the server and loaded in the background without interfering with the display and behavior of the existing page (see figure 2.1).

The fact that *AJAX* alters the process of web browsing can lead to several advantages and disadvantages. Some of them can be seen in table 2.4

Although *AJAX* has some disadvantages most can be overcome with no liability to the end users. For instance when an *AJAX* script function is not instantaneous, an visual aid is shown so that the user can understand that the web application is processing the inputed instructions. Also the fact that, the usage of this technology can decrease bandwidth usage is a big advantage, since it also decreases the minimum requirements for an host server [15].

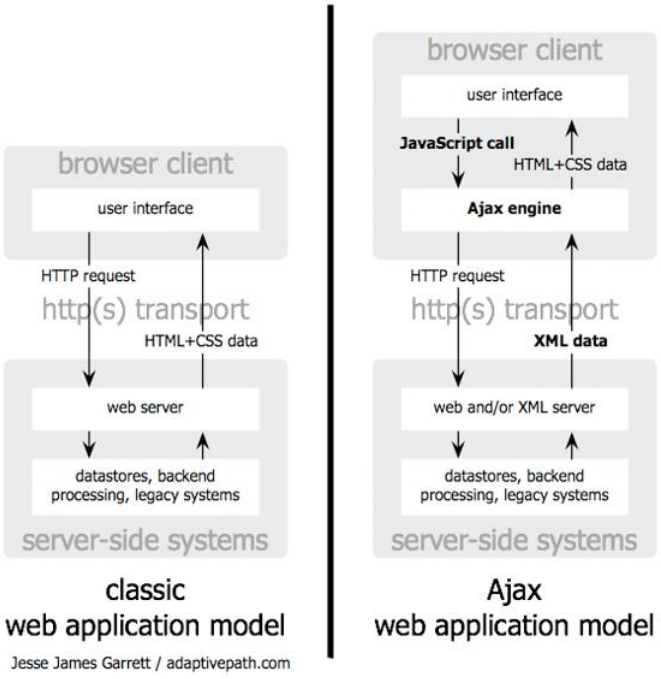


Figure 2.1: The traditional model for web applications (left) compared to the Ajax model (right).

Advantages	Disadvantages
Bandwidth usage - By generating the HTML locally within the browser, and only bringing down JavaScript calls and the actual data, Ajax web pages can appear to load relatively quickly since the payload coming down is much smaller in size, and the rest of the layout does not have to be redrawn on each update. An example of this technique is a large result set where multiple pages of data exist.	**Browser integration** - The dynamically created page does not register itself with the browser history engine, so triggering the "Back" function of the users' browser might not bring the desired result. Another issue is that dynamic web page updates make it difficult for a user to bookmark a particular state of the application.
Separation of data, format, style, and function - A less specific benefit of the Ajax approach is that it tends to encourage programmers to clearly separate the methods and formats used for the different aspects of information delivery via the web.	**Response-time concerns** - Network latency — or the interval between user request and server response — needs to be considered carefully during Ajax development. Without clear visual feedback to the users, they might experience delays in the interface of the web application, something which they might not expect or understand.
	Reliance on JavaScript and DOM - Ajax relies on JavaScript and the browser's Document Object Model (DOM), which are often implemented differently by different browsers or versions of a particular browser. Because of this, sites that use JavaScript may need to be tested in multiple browsers to check for compatibility issues.

Table 2.4: AJAX Pros and Cons

2.3.2 Prototype

The *Prototype JavaScript Framework* is a *JavaScript* framework created by Sam Stephenson which provides an Ajax framework and other utilities. *Prototype* aims to make it easier to work with *JavaScript*, offering a number of shortcuts for some of the most common uses[16].

Advantages	Disadvantages
Increases DOM/JavaScript Compatibility - By using the functions Prototype offers its is possible to achieve a code that is compatible with most Web Browsers.	**Performance** - Since the usage of Prototypes methods implies more commands, the execution time increases slightly.
Cleaner Source Code - Most of the implementation details are within Prototype's functions. Also with the ability to create a cleaner and simpler source code its maintainability increases.	

Table 2.5: Prototype Pros and Cons

The usage of this framework increases the development speed of JavaScript functions. It is possible to achieve this due to the compatibility issues will now be transparent to the developer. Also with a simples source code debugging tasks are simpler.

There are other similar framework, such as *JQuery*[17] and *Dojo Toolkit*[18]. Since the features of these frameworks are more aimed to deal with the visual part of the interface, Prototype was used instead.

2.4 Conclusion

All technologies mentioned above gave a positive contribute to all steps of the development process. As stated in the introduction of the current chapter and chapter 1.3 , a simple interface and good performance are required for this project to succeed.

PostgreSQL was used since its performance and features surpassed many of other free *ORDBMS*. Several of its features were crucial to the development of this project, *PL/pgSQL* for instance.

The reason behind the usage of *PHP*, as a server side language, was not only for its cost and its extensive functions list, but also due to the community behind it. With such a huge community it is most likely that, for most problems, its solution has been found and is available to everyone. Thus decreasing the development time.

Even though *JavaScript* has several compatibility problems there is no other language capable of performing the same task as *JavaScript*. Also with the use of Prototype framework, not only the compatibility issues became transparent for the developer, but also the source code becomes more clean.

Chapter 3

Implementation

In chapter 1.3 some general objectives were presented. For that reason a Logical and Physical Architecture were designed. Also the implementation methods, had these objectives taken in consideration.

In this Chapter an explanation of both Architectures will be given, and also how they were implemented using the technologies presented in Chapter 2.

3.1 Logical Architecture

In order to improve the organization of the information, and thus improving the maintainability, a layered architecture was used. This consists in separating the information by purpose, "Interface", "Business Logic" and "Database Communication". The "Interface" layer gathers all the information concerning what will be presented to the user, such as the *HTML* source code and simple functions involving it.

The "Business Logic" layer concerns algorithmic functions, that will handle the information exchange between the "Interface" and "Database Communication" layers.

Finally, the "Database Communication" layer is responsible for retrieving data, stored in a database. All logic behind the communication is attached to this layer.

In the following section, the implementation of these layers will be presented.

3.1.1 Interface

The interface source code was easily separated from the other layers, thanks to *Smarty*. By using *Smarty*, all the *HTML* was stored in *Smarty*'s templates, preventing it from being scattered between *PHP* and *JavaScript* functions.

JavaScript, as stated in section 2, can communicate asynchronously with the server, requesting data from a *PHP* script, and manipulate a web page. By creating *PHP* scripts that return a *Smarty* built *HTML* code, *JavaScript* only needs to insert the received data

directly into the web page. Thus keeping all the *HTML* generation to *Smarty* and also simplifying the *JavaScript* source code.

By using the *JavaScript* and *Smarty* synergy, building dynamic web pages becomes simpler. Also after analyzing the *DOM* object it was concluded that, it is possible to communicate between browsers windows. This is only possible when one window opens another, leaving a reference in the *DOM* of the "daughter" window.

3.1.2 Business Logic

The business logic was mostly separated from the database communication. Unfortunately there were few cases, that for performance and clearness reasons, the source code of both layers were mixed together. Also most of this layer was implemented in *PHP* and some specific parts were implemented in *JavaScript*.

The most challenging features were the *Comparisons* (present in the *Device Data* and *Test Times* modules) and also the Tree Menu wrapper class.

In the *Comparisons* script an *PL/pgSQL* function was used to aid it. This function returns a result set with the test names and its positions in each version. By arranging the data in this manner, the *Comparisons* business logic was made simpler.

The Tree Menu wrapper class, after receiving a database table name, its column names and web page name, automatically creates a menu from the data contained in the given view. It will have as many levels as the number of given column names, and link the last level to the given web page with an identification number as parameter.

3.1.3 Database Communication

Finally the database communication, was also implemented through *PHP*. In order to create a robust code, mostly *Views* and *PL/pgSQL Functions* were queried. This way, when changing the database format, by updating those *Views* and functions to output the same result set, the web application will still work.

Unfortunately, the *PL/pgSQL Functions* can cause some controversy ,since it can also be labeled as "Business Logic". But as stated previously, it can enhance the communication and create results sets, otherwise impossible. Also if the business logic performed by these functions were to pass to PHP, performance levels would decrease significantly.

3.2 Physical Architecture

The Physical Architecture purpose is to increase the separation, made by the Logical Architecture. By organizing the files from this project in folders, each with different purposes, a higher level of maintainability is obtained. The file structure can be seen in figure 3.1, and a brief explanation will be given in the following sections.

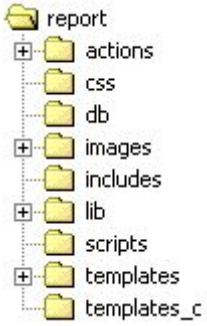


Figure 3.1: File structure in tree model

3.2.1 Root Directory (report)

The *Root Directory* contains all the directories of this project and the PHP script files accessible to the users. These files contain a simple business logic to process and assign data to the interface templates.

These files interact with the files contain in the *Data Base Access Directory*, to retrieve and process data, and *Smarty Templates Directory*, called by the Smarty class.

3.2.2 Smarty Templates Directory (templates)

This directory contains Smarty templates files. These files are essentially HTML files embedded with Smarty logic, that will be compiled and replaced with assigned data.

The templates include calls to JavaScript functions, stored in the *JavaScript Directory*, as to make full use of the AJAX technology.

Within this folder, module specific templates are separated in folders, in order to organize them.

3.2.3 JavaScript Directory (scripts)

The files from this folder contain all JavaScript functions declaration. These functions can use AJAX to call PHP actions from the *Actions Directory*, in order to retrieve structured information from the database.

3.2.4 Actions Directory (actions)

The *Actions Directory* contains exclusively PHP actions scripts. These actions are call with Javascript functions through AJAX. They retrieve information from the database using the classes declared in the files from *Data Base Access Directory*. The actions then call an template from the *Smarty Templates Directory* as to organize the returned data.

Within this folder, module specific actions are separated in folders, in order to organize them.

3.2.5 Data Base Access Directory (db)

This folder contains all the files with classes declaration that either do heavy business logic, such as comparison methods, or have database access logic.

3.2.6 CSS Directory (css)

Contains several CSS *(Cascade Style Sheet)* files. These files determine global aspects of the templates.

3.2.7 Images Directory (images)

All images of the web site are contained in this directory.

3.2.8 Configuration Directory (includes)

Configuration Directory contains several PHP configurations. The instructions in these files need to be run in all of the web pages of this project.

3.2.9 Libraries Directory (lib)

This directory contains several PHP libraries necessary for this project, such as the Smarty class definition.

3.2.10 Compiled Smarty Templates Directory (templates_c)

The directory *templates_c* contains the compiled templates. It is Smarty's default directory for its compiled templates.

3.3 Conclusion

All the work was developed with the objectives in chapter 1.3. It was due to these premises that many challenges were encountered but the project could have not be made any other

One of the common challenges was to develop a simple and efficient interface. Because for that to be possible, there must exist a kind of symbiosis between the technologies. This is specially true for *JavaScript* since it is responsible for the asynchronous communication and manipulating *DOM*, and thus increases the effectiveness of the interface.

As soon as the Interface was developed, the business logic and database communication behind it, were developed easily, except for a few features. As stated previously, a full separation of these layers was not possible due to performance issues.

Finally the development speed was increase, since the project files were carefully organized. In Appendix A it is possible to view the content of some files, and their locations,

that are responsible for the “View Test Times” feature (further detail about this feature can be found in Chapter 4.3.1).

Chapter 4

Project

In this chapter a presentation will be given, about the project, its objectives and its features. The chapter is organized in three sections, one for each module of the *Report Center*, and the conclusion.

Before starting to develop a module, a meeting with the *TPE* was made. The objective of this meeting is to know which features are fundamental for this project.

The implementation methods and architecture, described in Chapter 3, were used when developing all the features.

4.1 Device Data Module

In Chapter 1.2 it was shown that the *TPE* need to analyze several files, such as the *Device Data*. It contains several logistical information about the flow being ran, but most importantly it contains the logical results per device and per test contained in the flow (see Chapter 1.2.1 for further details).

The *TPE* need to analyze these files, to examine if the developed, or created, test are behaving as intended. Also, sometimes there is a need to compare flows from different versions.

As stated at the start of this chapter, a meeting was held to determine the features to be implemented. The features are :

- **Version Browser** - Allow the selection of a *Version*, in an ambiguous and intuitive manner.

- **Flow Report** - Show the users the results of a *Flow*. This must be done in such a way that, the user can analyze the results easily and quickly.

- **Version Report** - Show the users an overview of the *Version* results. This report must filter some data, according to the user, as to keep the report clear and easy to analyse.

- **Version Comparison** - Allow the user to compare two *Versions* from the same *Product*. This will allow the users to notice whether a *Flow* as been added or remove from said *Versions*.

- **Flow Comparison** - Allow the user to compare two *Flows* from different *Versions*, from the same *Product*. This report needs to show the user the difference between the tests contained in both *Flows*, and also compare the results between them.

This module was the first to be implemented, and thus was the most challenging. Also some features, such as the Comparisons (see Chapter 4.1.5), involved the development of complex algorithm.

Another challenging issue was *JavaScript*, due to its differences between browsers. Also *DOM* interaction is affected by these differences. Since at that time the Prototype framework was not being used, and due to some internal company rules, this problem had to be solved by debugging with Firefox© and find the equivalent code for Internet Explorer 6©.

4.1.1 Version Browser

The *Version Browser* was implemented has a tree menu, built from the database's information. The menu is built in such a way that the user will select attributes as to identify the intended *Product* and finally its *Version*. An example can be seen in figure 4.1.

After selecting the intended *Version* an option menu will become available to the user (see figure 4.2). These options will be explained in sections 4.1.2, 4.1.3 and 4.1.4.

Figure 4.1: Version Browser

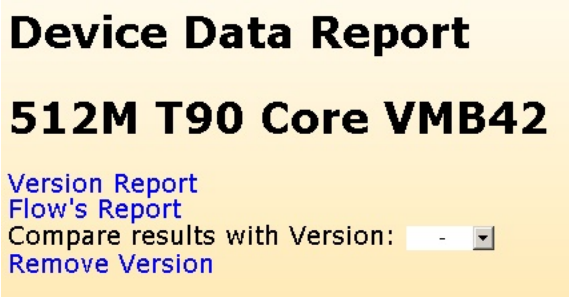


Figure 4.2: Version options menu

4.1.2 Flow Report

The *Flow Report* was implemented as a table with the results for each *Device* and *Test*. This report is presented in such a way that the users can find and view the information with little strain to the eye.

After the users select this option, a *JavaScript* function will retrieve a list of all the *Flows*, of the selected *Version* (see figure 4.3). The Report will be shown, in a new window, after the user selects the intended *Flow* from the above mentioned list.

This report has several option (as shown in figure 4.4). The first option, will toggle if the *Chip Ids*[1] are shown. The *"Hide Monitoring Tests"* will toggle, if the *Tests* with no logical results (in gray) are visible or not. It is also possible to input a filter, in order to show the *Tests* whose names are similar to it.

As to give a quick overview, a "Pass" (positive result) percentage of both devices and tests, is shown.

Flow List

Flow	Date	Remove
U6NI4XBS	2008-03-03 20:16:19+00	Remove
U6NI4XBS-25F	2008-03-03 20:22:26+00	Remove
U6NI4XBS-3S	2008-03-03 20:19:23+00	Remove
U6NI6XBS	2008-03-03 20:59:06+00	Remove
U6NI6XBS-25F	2008-03-03 21:03:03+00	Remove
U6NI6XBS-3S	2008-03-03 21:01:06+00	Remove
U6NI8XBS	2008-03-03 20:39:41+00	Remove
U6NI8XBS-25F	2008-03-03 20:44:51+00	Remove
U6NI8XBS-3S	2008-03-03 20:42:35+00	Remove

Figure 4.3: Flow list

[1] Refers to the rows "Tech_Size" , "Design", "LotRef", "Wafer","X" and "Y" in figure 4.4

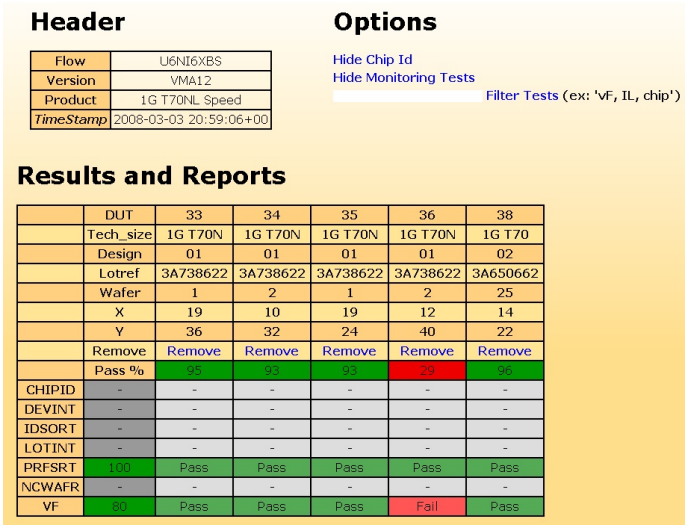


Figure 4.4: Part of a Flow Report

4.1.3 Version Report

The *Version Report* was implemented as a set of tables with several statistics. These tables are organized by *Flow*, and divided by *Tests* and *Devices*. The statistics shown in this report, are the same from the previously discussed statistics in the *Flow Report*.

In order to filter the massive amount of data, thresholds were implemented (see figure 4.5). According to these thresholds the results are either hidden, if their value is between the thresholds value, or shown, colored in red or green depending if the result is higher or lower then the thresholds value. For example in figure 4.2 we can see they are all set to 50%. Thus, all results below 50% are colored in red, those above will be colored green and no result was filtered since the thresholds are equal.

Since a *Version* can contain several *Flows* this report can be rather long. For that reason a "Navigation" tab (see figure 4.7) is shown, so that the users can easily redirect to a particular *Flow*.

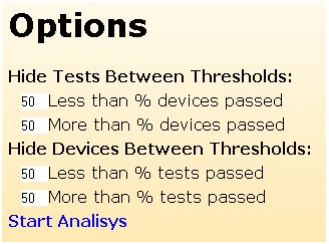
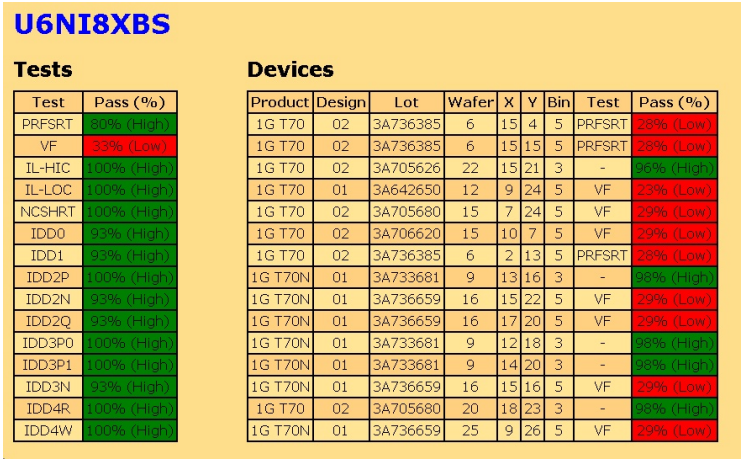


Figure 4.5: *Version Report* options



U6NI8XBS

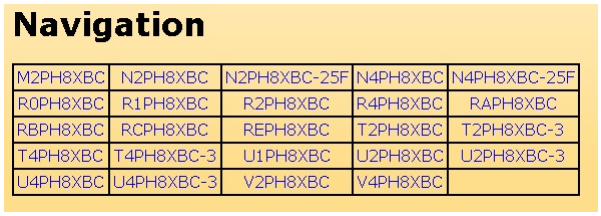
Tests

Test	Pass (%)
PRFSRT	80% (High)
VF	33% (Low)
IL-HIC	100% (High)
IL-LOC	100% (High)
NCSHRT	100% (High)
IDD0	93% (High)
IDD1	93% (High)
IDD2P	100% (High)
IDD2N	93% (High)
IDD2Q	93% (High)
IDD3P0	100% (High)
IDD3P1	100% (High)
IDD3N	93% (High)
IDD4R	100% (High)
IDD4W	100% (High)

Devices

Product	Design	Lot	Wafer	X	Y	Bin	Test	Pass (%)
1G T70	02	3A736385	6	15	4	5	PRFSRT	28% (Low)
1G T70	02	3A736385	6	15	15	5	PRFSRT	28% (Low)
1G T70	02	3A705626	22	15	21	3	-	96% (High)
1G T70	01	3A642650	12	9	24	5	VF	23% (Low)
1G T70	02	3A705680	15	7	24	5	VF	29% (Low)
1G T70	02	3A706620	15	10	7	5	VF	29% (Low)
1G T70	02	3A736385	6	2	13	5	PRFSRT	28% (Low)
1G T70N	01	3A733681	9	13	16	3	-	98% (High)
1G T70N	01	3A736659	16	15	22	5	VF	29% (Low)
1G T70N	01	3A736659	16	17	20	5	VF	29% (Low)
1G T70N	01	3A733681	9	12	18	3	-	98% (High)
1G T70N	01	3A733681	9	14	20	3	-	98% (High)
1G T70N	01	3A736659	16	15	16	5	VF	29% (Low)
1G T70	02	3A705680	20	18	23	3	-	98% (High)
1G T70N	01	3A736659	25	9	26	5	VF	29% (Low)

Figure 4.6: One *Flow* of the *Version Report*



Navigation

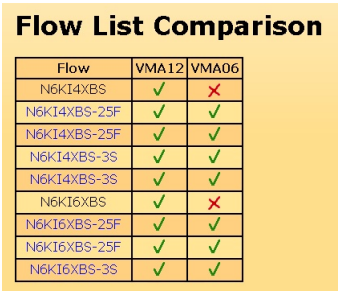
M2PH8XBC	N2PH8XBC	N2PH8XBC-25F	N4PH8XBC	N4PH8XBC-25F
R0PH8XBC	R1PH8XBC	R2PH8XBC	R4PH8XBC	RAPH8XBC
RBPH8XBC	RCPH8XBC	REPH8XBC	T2PH8XBC	T2PH8XBC-3
T4PH8XBC	T4PH8XBC-3	U1PH8XBC	U2PH8XBC	U2PH8XBC-3
U4PH8XBC	U4PH8XBC-3	V2PH8XBC	V4PH8XBC	

Figure 4.7: Flows in the *Version Report*

4.1.4 Version Comparison

The *Version Comparison* is report that will take two *Versions*, compares them and presents the user a *Flows* list. This list also gives the user the information of, in which *Version* the *Flow* existed.

The user can access this report by selecting another *Version* from the drop down box, seen in figure 4.2.



Flow List Comparison

Flow	VMA12	VMA06
N6KI4XBS	✓	✗
N6KI4XBS-25F	✓	✓
N6KI4XBS-25F	✓	✓
N6KI4XBS-3S	✓	✓
N6KI4XBS-3S	✓	✓
N6KI6XBS	✓	✗
N6KI6XBS-25F	✓	✓
N6KI6XBS-25F	✓	✓
N6KI6XBS-3S	✓	✓

Figure 4.8: Flow comparison between two Versions

4.1.5 Flow Comparison

The *Flow Comparison* is a report that shows *Test's* differences between two *Flows*. A table is presented with the information with the correct order the *Test* are ran, in which

Version they exist and whether they swapped places or not. Also the user may compare logical results, between *Devices* that were ran in either *Versions*. A list of *Devices* is available as seen in figure 4.9.

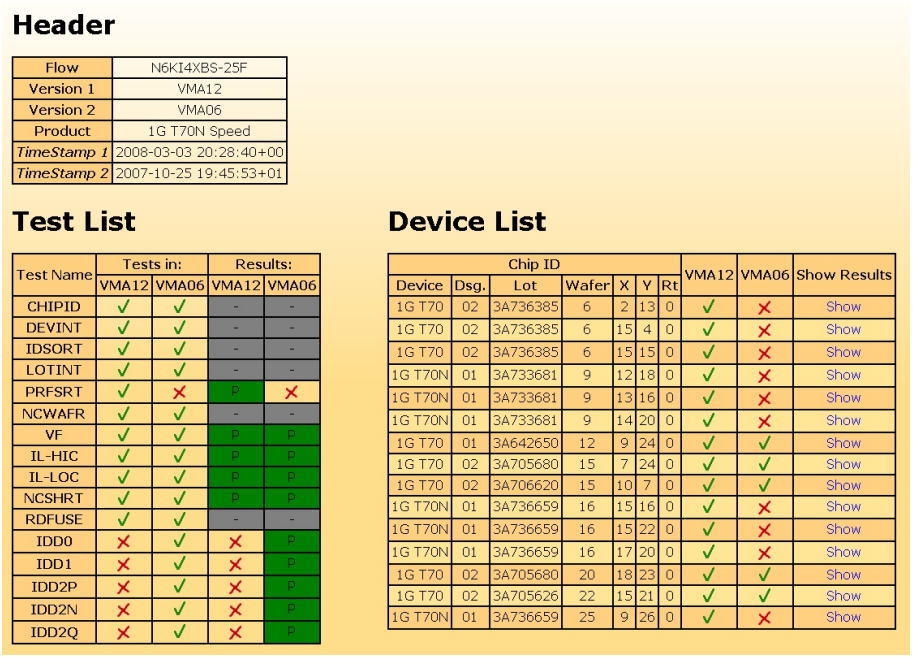


Figure 4.9: Tests and results comparison between two *Flows*

4.2 Transfer Letter Module

The *Transfer Letter* intent is for, the *TPE* give information about a *Version*, to the Production Department. It must contain all the information necessary, so that that department can implement the version on the production testers. This information also includes the *Test Program Change Request* (TPCR), *Test Program Logistic* (TPL) and *Bug fixes* implemented. Due to its complexity and extensiveness, manually creating the *Transfer Letter* is an effortful and error-prone task.

Since all the technical data from a *Version* is stored on the database, the following features were approved:

- **Input logistic information** - Allow the user to input information necessary for the generation of the *Transfer Letter*.
- **Add TPCR and TPL** - Allow the user to register, and associate, a *TPCR* and/or a *TPL*. The registration must be kept simple, and the application must allow the user to associate already registered *TPCRs* or *TPLs*.
- **Add Bug fix** - Allow the user to register, and associate, a *Bug fix*. The registration must be similar to the already existent tool.

- **Generate PDF** - The application must allow the user to generate a *Transfer Letter*, from the inputed logistic information and the stored technical data.

As stated in Chapter 1.3, the user interface must be kept simple and intuitive. That posed a challenge in this module, because of the registration and association of the *TPCRs*, *TPLs* and *Bug Fixes*. Also, creating a responsive interface was troublesome, since this Report has a significant amount of information. This means that the necessity to reload the entire web page, had to be reduce to a minimum.

4.2.1 Input Logistic Information

This feature was implemented in such way, that the need to reload web page was reduced to zero. Using *JavaScript* and *AJAX*, after the user changes any field, that data will be stored on the database. The user can then alter the several check boxes, drop down menus and text areas as to compliment the *Version's* technical data. Also the *Add TPCRs*, *Add TPLS* and *Add Bug Fix* features will be described in detail in sections 4.2.2 and 4.2.3.

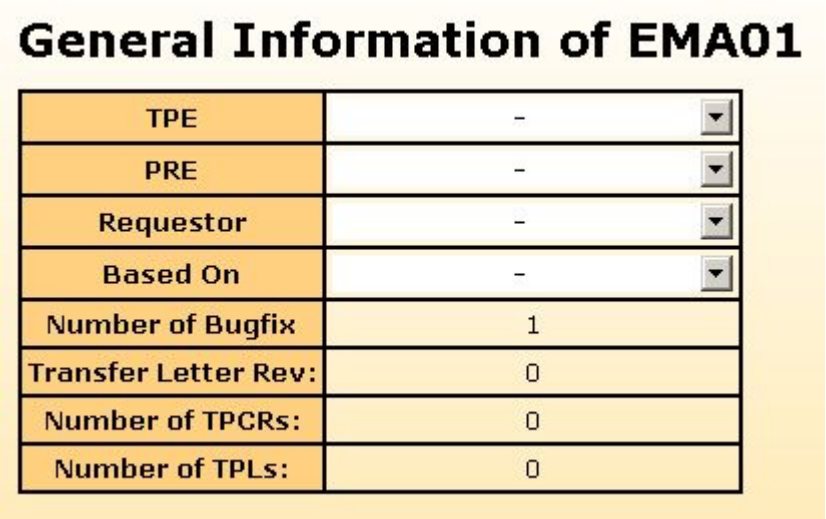


Figure 4.10: Available Options

Figure 4.11: Available Options

4.2.2 Add TPCRs and TPLs

The *Add TPCRs and TPLs* feature was implemented in a similar way as "Google Suggests"[19]. When the user inputs the *TPCR* (or *TPL*) number in the its respective text box, suggestions will appear, from already registered *TPCR* (or *TPL*). The user may then select one of the suggestions to associate that *TPCR*, or *TPL*, to the *Version*. Also, if it is not registered, the user may choose to register it by selecting the "New TPCR", or "New TPL" links. This will, not only register but also associate it to the *Version*. The user may alter their description, by double clicking the default description and a text area will appear.

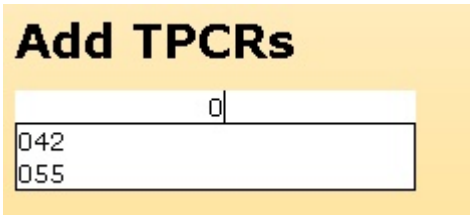


Figure 4.12: Add TPCR

Figure 4.13: Add TPL

4.2.3 Add Bug Fix

The *Add Bug Fix* feature was implemented as to emulate the behavior of similar method. By choosing to associate an *Bug Fix* a new window will be shown. The window

will contain a form to input all the information about the *Bug Fix*. This will register and associate it to the *Version*.

When the user is filling the "Summary" field, if another bug fix has a similar "Summary", a suggestion will be shown. The user may click one of the suggestions and the remaining fields will be filled with the suggested *Bug Fix* information.

Figure 4.14: Add bug fix

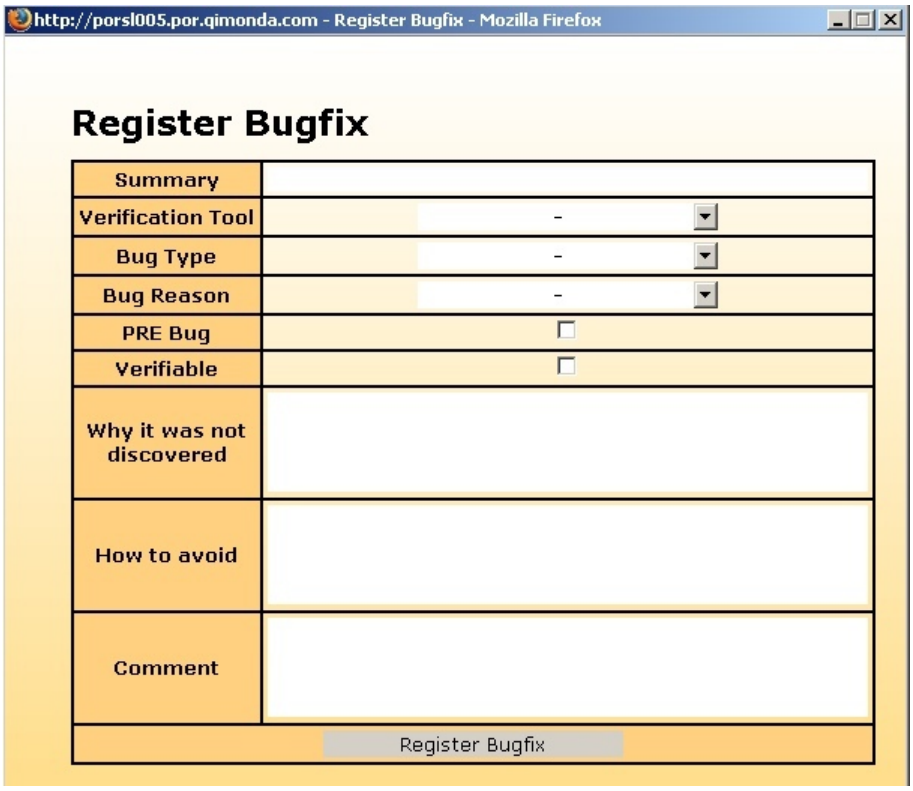


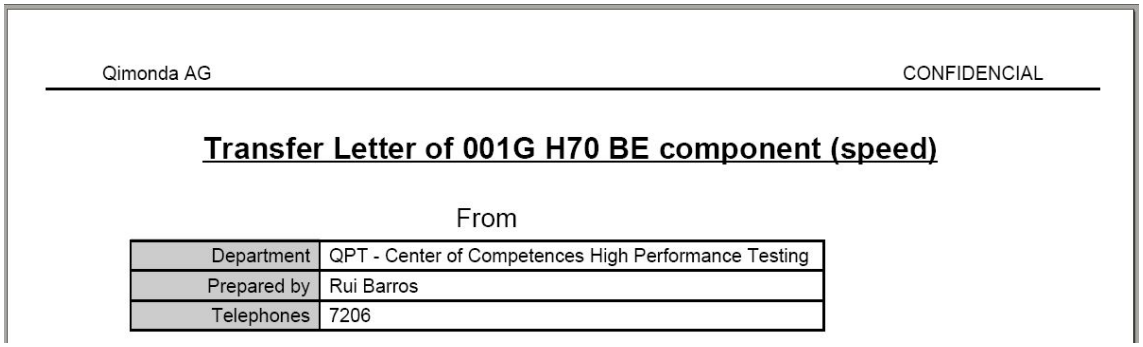
Figure 4.15: Bug fix window.

4.2.4 Generate a Transfer Letter PDF

This feature will generate a Transfer Letter in PDF format. The data is retrieved from the database and is inserted in the PDF file. This file will be stored and available in the *Report Center* server for one month.

To generate the Transfer Letter, the user must click the *Generate Transfer Letter* button and a new window will open, with the file's *URL*.

Figure 4.16: Generate PDF button



4.3 Test Times Module

The *Test Times Module* is intended to analyze two files, the "Test Times" and the "Test Time Sums" file. These contains the information of the times that a *Flow* or a *Test*, took to execute. The difference between these files is that the "Test Time Sums" contains the times of all the Version's Flows. On the other hand, the "Test Times" file contains the times of each *Test* of all the *Version*'s *Flow*, thus containing a lot of information. More details can be seen in Chapter 1.2.2.

After a change to a *Test*, the *TPE* must then check if that change had an bigger impact on the *Test's* execution time then expected. The analysis of these files are quite important due to the time impact on the production. The approved features were the following:

- **List Test Times sum** - Show the user all the *Version Flows* execution times.
- **List Test Times** - Show the user the execution times of all the *Flow Tests*.
- **Compare Test Times sum** - Allow the user to compare the "Test Times sums" from different *Versions*.
- **Compare Test Times** - Allow the user to compare the tests execution times from *Flows* of two different *Versions*.

The features were implemented without difficulties, due to the inclusion of the Prototype framework and the experience gained from the challenges in developing the previous modules. Apart from the features, this module also has the *Version Browser* feature (see Chapter 4.1.1).

Each Version can contain several results, since they can be ran in several *Testers* and include several product *Parallelisms*.

512M T90 Speed

Test Sum's

Version	Tester	Paralellism	Work Directory	Compare
VMA35	T55935	Single	/home/barrosru/Packages/512MT11S/VMA35/RESULTS/VMA35_CHTT_WORK93S_	-

Figure 4.17: Tester, Parallelism and Directory combination selection

4.3.1 List Test Times Sum

This feature was implemented by presenting a table, where each line has the times of a *Flow*. Since there are similar *Flows*, only differing which *Organization* they affect, they were grouped in one line. To show all the times, columns for each *Organization* where added.

The table (figure 4.18) will appear, with no page reload, after the user selects an *Testers* and *Parallelisms* pair (figure 4.17).

Selected Test Sum

Flow	Retest	Sum x4	Sum x8	Sum x16
U6THxXBS	No	12.2	7.9	6.3
U6THxXBS-3S	No	22.8	15.3	12.4
U6THxXBS-3SBIR	Yes	12	7.7	6
U6THxXBS-3SBIR	No	16	12.8	12.4
U6THxXBS-3SIB	No	25.2	17.9	15.5
U6THxXBS-3SIR	Yes	11.9	7.7	6
U6THxXBS-3SIR	No	13.6	10.2	9.2
U6THxXBS-4	No	12.4	7.9	6.3
U6THxXBS-4IB	No	14.7	10.7	9.6
U6THxXBS-4LP	No	23.6	15.8	12.9
U6THxXBS-4LPIR	Yes	11.9	7.7	6
U6THxXBS-4LPIR	No	14.3	10.6	9.6

Figure 4.18: List of test time sums

4.3.2 List Test Times

The *List Test Times* feature will present to the user the Test Times details of a selected *Flow*. This data will be organized in a table where each row has several times for a test. These time include the flow time line (Sum column), the test initialization, the actual run, finalization and the total time it ran.

To access this feature, the user only needs to select time from the table described in Chapter 4.3.1.

U6TH6XBS Details

#	Test	Retest	Sum	Initial	Run	Final	Total
OH	TDWINI	No	0.5	0	0	0.5	0.5
OH	SEQINI	No	0.6	0	0	0.1	0.1
5	IDSORT	No	1.3	0	0	0	0
1	CHIPID	No	1.3	0.3	0.4	0	0.7
4	DEVINT	No	1.3	0	0	0	0
7	PRFSRT	No	1.4	0	0	0	0
6	LOTINT	No	1.4	0.1	0	0	0.1
8	NCWAFR	No	1.4	0	0	0	0

Figure 4.19: List a Flow test times

4.3.3 Compare Test Times Sum

To access this feature the user only needs to select a *Version* from the drop down menu seen in figure 4.20.

This feature will show the user a table, that will set the *Flow's* times of both Versions side by side. Again the *Flows* are grouped by *Organization*, but to avoid cluttering the interface only one *Organization* is shown at a time. To access the times of other *Organizations* the user only needs to select the intended one, on the top of this table (see figure 4.20).

Selected Test Sum

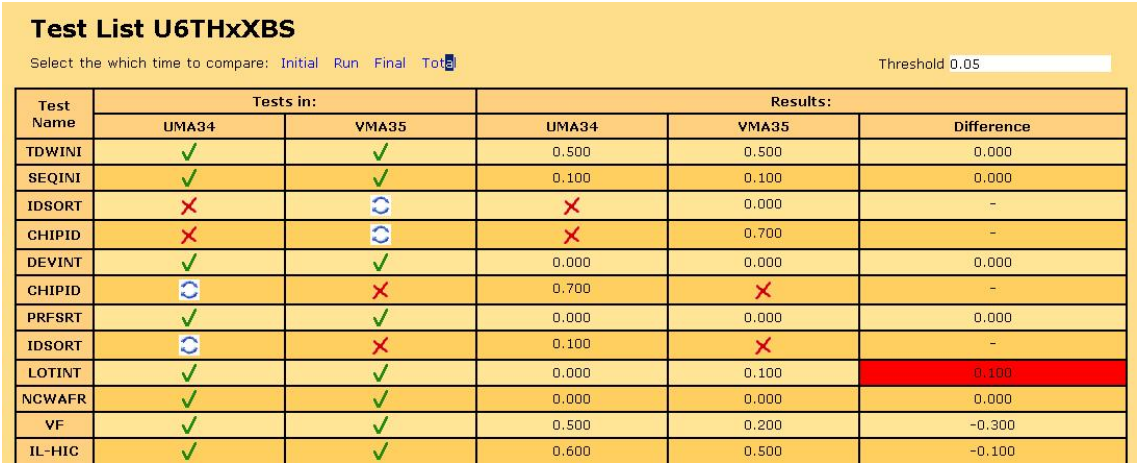
Select the organization: 4x 8x 16x

		VMA35	UMA34	Compare
Flow	Retest	Sum x16	Sum x16	
U6THxXBS	No	6.3	6	Compare
U6THxXBS-3S	No	12.4	15.8	Compare
U6THxXBS-3SBIR	No	12.4	16.6	Compare
U6THxXBS-3SBIR	Yes	6	5.7	Compare
U6THxXBS-3SIB	No	15.5	19	Compare
U6THxXBS-3SIR	No	9.2	13.5	Compare
U6THxXBS-3SIR	Yes	6	5.9	Compare
U6THxXBS-4	No	6.3	9.3	Compare
U6THxXBS-4IB	No	9.6	12.6	Compare
U6THxXBS-4LP	No	12.9	16.3	Compare
U6THxXBS-4LPIR	No	9.6	14.3	Compare
U6THxXBS-4LPIR	Yes	6	5.8	Compare

Figure 4.20: Compare test time sums

4.3.4 Compare Test Times

This feature will show the user a table that compares two *Version's Flows* details. This table will contain the list of all the test in both version, the information if whether they exist or not in a version, and if they exist if they swapped places. The user can select which times, available above the table, to view them (see figure 4.21). Also the user may choose to insert a threshold, as to identify more quickly which test has a difference higher than the threshold.



Test List U6THxXBS

Select the which time to compare: Initial Run Final Total Threshold 0.05

Test Name	Tests in:		Results:		
	UMA34	VMA35	UMA34	VMA35	Difference
TDWINI	✓	✓	0.500	0.500	0.000
SEQINI	✓	✓	0.100	0.100	0.000
IDSORT	✗	↻	✗	0.000	-
CHIPID	✗	↻	✗	0.700	-
DEVINT	✓	✓	0.000	0.000	0.000
CHIPID	↻	✗	0.700	✗	-
PRFSRT	✓	✓	0.000	0.000	0.000
IDSORT	↻	✗	0.100	✗	-
LOTINT	✓	✓	0.000	0.100	0.100
NCWAFR	✓	✓	0.000	0.000	0.000
VF	✓	✓	0.500	0.200	-0.300
IL-HIC	✓	✓	0.600	0.500	-0.100

Figure 4.21: Compare test times

4.4 Conclusion

All approved features in the *TPE* reunion were implemented. Screen shots of the features, with their respective description, were presented in this Chapter.

By using the implementation methods and architecture, described in Chapter 3, the development of the describe features, was made simpler. Again, the objectives in Chapter 1.3, were taken in consideration. Web page reloads were kept to minimum, to improve performance, and the interface was designed as simple and intuitive as possible.

Chapter 5

Evaluation

In this chapter an evaluation of the *Report Center* will be presented. For this purpose a survey [20] was done to the *TPE* from "Qimonda Portugal". The results will be shown and a conclusion will be made according to that feedback.

5.1 Survey

The survey can be seen in Appendix B. Its questions were grouped by feature and module as to organize its structure. The questions were made with two different purposes.

First to get the inquired to judge the modules and the Report Center. This feedback will show whether the work has sufficient quality to be released to all the *TPE* of "Qimonda". Unfortunately this does alone is not enough. It is also necessary to have reference to compare the *Report Center*.

The second purpose is to compare the *Report Center*s features to the method used previously. This means it is now possible to evaluate the quality of the *Report Center* but also to compare it to previous methods. If these comparisons have an outstanding result, then the *Report Center* will most likely replace those methods.

This survey was made in a population of fourteen *TPE*. The sample was eleven engineers since the remaining were absent or unable to answer at the time the survey was conducted.

5.2 Results

After all the TPE answered the survey, the following results were found. The results were converted into charts for an easier interpretation.

5.2.1 Device Data

The Device Data was got the results seen in figure 5.1. The inquired TPE were given four choices of answer, "Bad", "Sufficient", "Good" and "Very Good". From all the inquired 18% answered that the module was "Sufficient", while 36% and 46% answered "Good" and "Very Good".

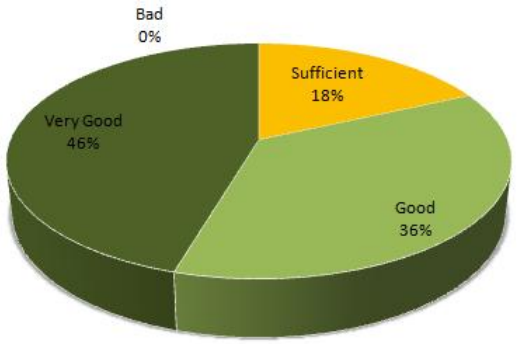


Figure 5.1: Survey Results: Device Data Module evaluation

As stated in the previous chapter, an absolute result is not enough for a proper evaluation. It is necessary to have a reference and evaluate all features according to that reference.

In order to increase the detail of this evaluation the TPE were asked to evaluate the module feature's in terms of "Readability", "Complexity", "Necessary Time", "Flexibility" and "Efficiency". Again the *TPE* had the same scale used in the module evaluation (four choices ranging "Bad" to "Very Good"). In order to achieve clear charts the scale was replaced by a numeric scale, where the number "1" is equal to "Bad" and number "4" to "Very Good".

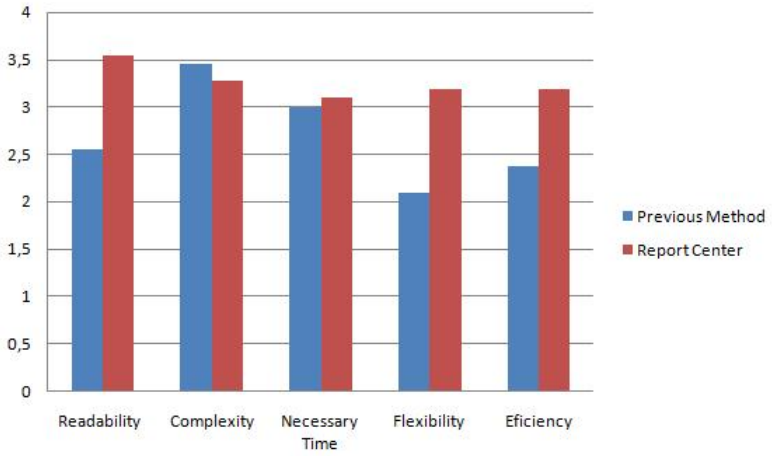


Figure 5.2: Survey Results: Device Data flow analysis

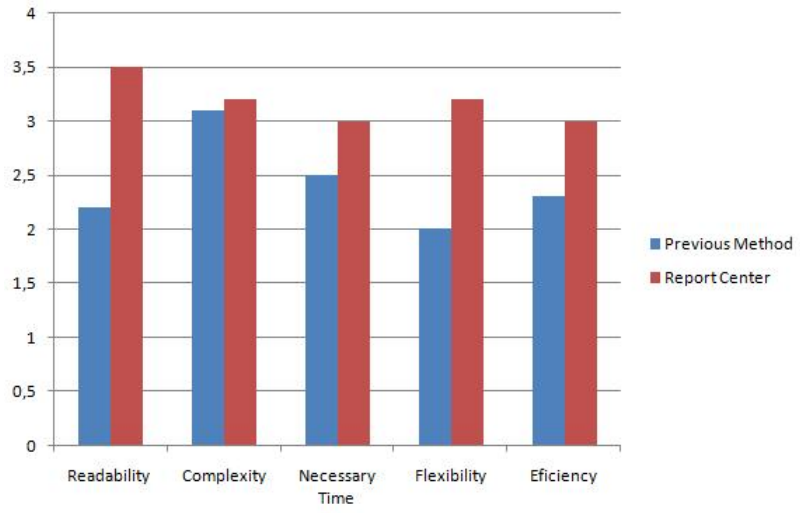


Figure 5.3: Survey Results: Device Data version analysis

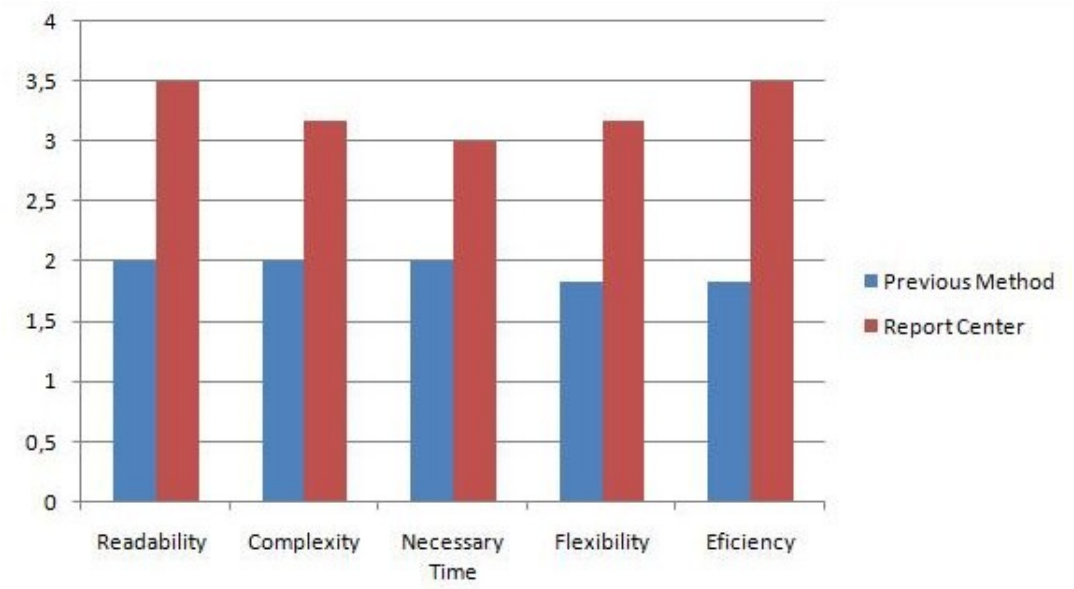


Figure 5.4: Survey Results: Device Data comparison

The results achieved surpassed all expectations. In comparison to equivalent method almost every evaluation topic in every feature. The only exception is the *Device Data* module complexity. Even though both methods have a very high mark, it can not be neglected that other methods are less complex than the *Device Data*.

In another matter, 45% of the inquired *TPE* did not evaluate the Comparison feature. This is due to the fact that, till the present time they never had to compare two Versions.

5.2.2 Tranfer Letter

In the *Transfer Letter* module evaluation, the methods used to evaluate the *Device Data* module were reused. This means that the scale is the same and also the topics of evaluation. In figure 5.5 can be seen the evaluation of the whole module and in figure 5.6 the chart that evaluates this module feature against other methods.

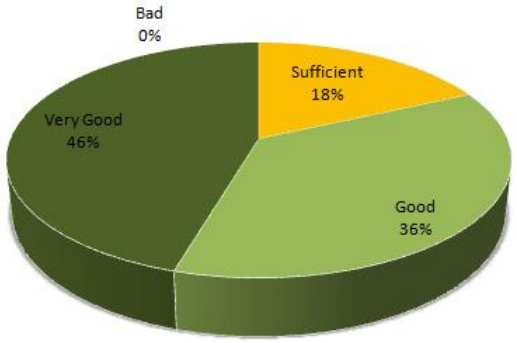


Figure 5.5: Survey Results: Transfer Letter Module evaluation

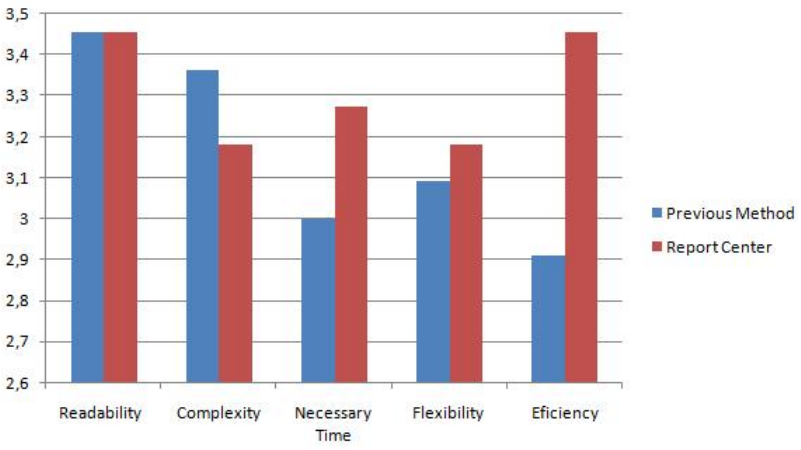


Figure 5.6: Survey Results: Transfer Letter Generation

Although this module has a good quality, since 80% of the inquired *TPE* gave a positive feedback, when compared to other methods the Transfer Letter generation brings only a slight improvement, and with increased complexity.

5.2.3 Test Times

Once again the method of evaluation in the Device Data Module is reused in Test Times Module. The overall evaluation can be seen in figure 5.7. In figures 5.8 and 5.9 the features "Test Time analysis" and "Test Time comparison", respectively, are being evaluated against their equivalent methods.

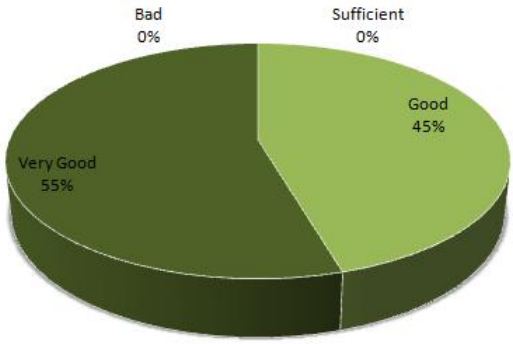


Figure 5.7: Survey Results: Test Times Module evaluation

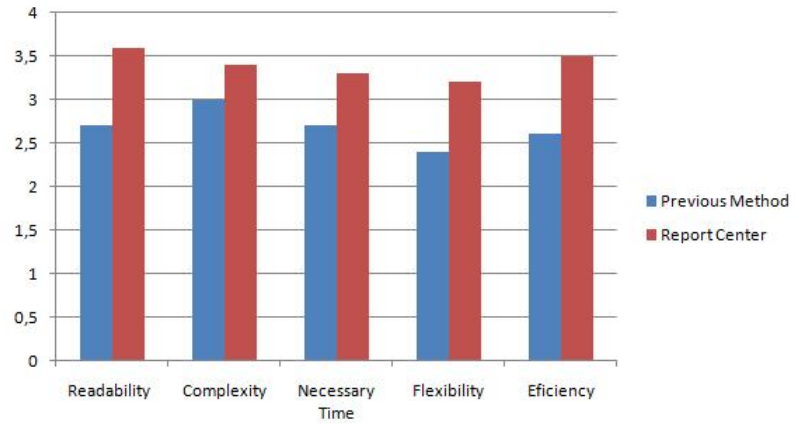


Figure 5.8: Survey Results: Test Time analysis

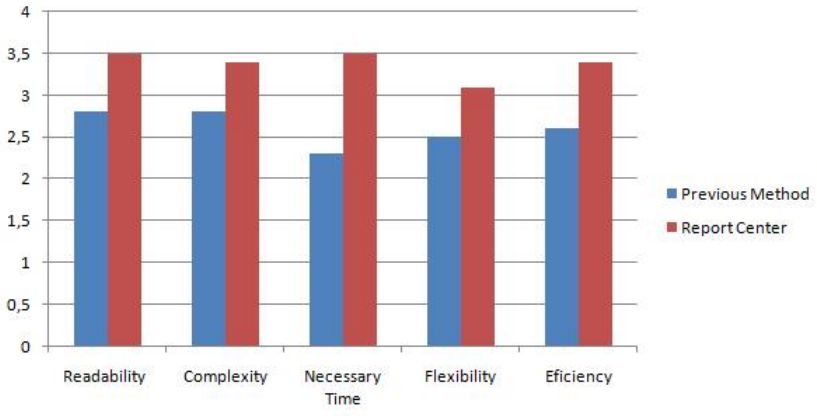


Figure 5.9: Survey Results: Test Time comparison

This module also exceeded all expectations. Not only did all the inquired TPE gave a positive feedback (55% said it was "Very Good" and 45% said it was "Good"), but also this module had a better evaluation in all topics and in both features.

5.3 Conclusion

From the results seen in the previous section, the *Report Center* has an overall positive feedback. Almost all the its features surpassed other equivalent methods.

The *Device Data* module even though it received a good feedback there are certain topics that have be improved. The most critical topics are the "Complexity" and the "Necessary Time" since the other methods also are very simple and fast to use.

The *TPE* that do not compare versions, some have evaluated that feature (*Device Data* Comparison) even though they do not know other methods. This data was not taken in consideration since it would create defective results.

As said in the previous section the *Transfer Letter* module received a good overall evaluation but compared to other methods there is still improvements to be made. This is especially true for its complexity and flexibility.

The Test Time Module was the one that received the best feedback, with an astonishing 55% of the TPE evaluating it as "Very Good" and 45% evaluated as "Good". In all the evaluations topics this module surpass the other equivalent methods, and some of them with a considerable difference.

Chapter 6

Conclusions And Future Work

6.1 Conclusion

After analyzing the data from Chapter 5.2 one can say that the project is going in the right direction. The *TPE* gave a very positive feedback, even though the some improvements must be made so that the Report Center can be a viable replacement to the equivalent methods.

All the features requested by the *TPE* were implemented. All the of these features can be seen in Chapters 4.1, 4.2 and 4.3. Also the concerns in Chapter 1.3 were respected.

The interface of the Report Center was designed and developed to be as intuitive as possible. For instance all links have tooltip that describe their actions, along with a suggestive title. Also with the usage of JavaScript several options became available, such as detect user actions and run a script according to that action. And even though the complexity was the topic that had the worst comparison, it cannot be neglected that its evaluation was above "Good"

In order for all the *TPE* be able to access the *Report Center*, arrangements were made to host the web page in a dedicated server. Due to *PHP*'s portability, the migration from the author's workstation to the server was made with no problems. In order to test the performance of the Report Center and the server, several users were asked to browse the Report Center at the same time. The responsiveness of the Report Center was always in acceptable levels.

To insure the maintainability the *Report Center* was developed in a layer system, as described in Chapter 3.

6.2 Future Work

As concluded in Chapter 5.3 several improvements can still be made to the Report Center. The more critical points are the complexity of the Transfer Letter and the Device Data modules. Also during the survey several TPE gave many insightful suggestions.

A more precise and scientific analysis can be made, to evaluate even further the *Report Center*. This was not done due to time constrains.

A few modules of the Report Center need to be implemented, such as the ST Log, DC Log and TDF Compare.

Also the data will be migrated to a new database with a different schema[21]. Slight adjustments have to be made to the business logic of the *Report Center*.

Anexo A

Example Code

In this Appendix the source code of the feature View Test Time will be presented. The files were organized in execution order.

./test_times.php

```
<?php
    include_once("includes/base.php");
    include_once("db/TestTime.php");

    if(isset($_GET['link_id']) ) //if a version was selected
    {
        $version_id = $_GET['link_id']

        //Gets all the TestTimes of that version
        $test_sums = TestTimes::GetVersionTTS($version_id);

        foreach($test_sums as $tsum)
        { //Checks for other Test Times that
          //can compare with the ones of the current Version
            $can_compare[$tsum['id_ttv']] =
                TestTimes::CanCompareWith($version_id, $tsum['tester'], $tsum['parallelism'] );
        }
        $smarty->assign('test_sums', $test_sums);
        $smarty->assign('can_compare', $can_compare);
    }

    //Version Browser Declarations
    $menu = new Menu("tt_menu_view", $select , "test_times");
    $tipo = "tree_menu";
    $smarty->assign('tipo', $tipo );

    //Display the template with the assigned data
    $smarty->display("test_times/test_times_versions.tpl");
?>
```

./templates/test_times/test_times_versions.tpl

```
{include file="header.tpl" titulo="Test Times"}
{include file="menu_lateral.tpl"}

<script src="scripts/test_times.js" type="text/javascript"></script>


{if isset($test_sums)}
<div class="article">
    <h2>{$test_sums[0].product_size} {$test_sums[0].product_technology} {$test_sums[0].insertion}</h2>
    <h2>Test Sum's</h2>

    <p>
        <table>
            <tr>
                <th>Version</td>
                <th>Tester</td>
                <th>Paralellism</td>
                <th>History</td>
                <th>Compare</td>
            </tr>
            {foreach from=$test_sums item=ttsum}
            <tr>
             <!--Link below will call a JS script that will fetch data from the Database-->
                <td> <a onclick="get_tt_sum({$ttsum.id_ttv})"> {$ttsum.version_name} </a> </td>
                <td> {$ttsum.tester} </td>
                <td> {$ttsum.parallelism} </td>
                <td> <a onClick="getTThistory('{$ttsum.id_ttv}')">Show History</a></td>
                <td>
                    <select id="compare_{$ttsum.id_ttv}"
                        onchange="compare_tt_sum_with($(this).value,{$ttsum.id_ttv},
                        $(this).options[$(this).selectedIndex].text,'{$ttsum.version_name}')"
                        style="width: 8em;" name="compare">
                        <option value="%" selected="selected" > - </option>
                    {foreach from=$can_compare[$ttsum.id_ttv] item=option}
                        <option value="{$option.id_ttv}">{$option.version_name}</option>
                    {/foreach}
                    </select>
                </td>
            </tr>
            {/foreach}
        </table>
    </p>
</div>
    <br/>
<div id="test_time_sum"></div>   <!--Fetched data from JS functions will be imported here-->
{/if}

{include file="footer.tpl"}
```

./scripts/test_times.js excerpt

```
function get_tt_sum(id_ttv_p)
{
    new Ajax.Updater('test_time_sum', //Retrieved data will be inputed
                                      //in the HTML element with this id

    'actions/test_times/action_get_tt_sum.php', //Retrieve the data from
                                                //from this PHP script
    {
        method:'POST',
        parameters: {id_ttv: id_ttv_p},    //Parameters sent to the script
        onFailure: Error_Alert
    });
}

function compare_tt_sum_with(id_ttv_p1, id_ttv_p2, vname1_p, vname2_p)
{
    new Ajax.Updater('test_time_sum',
    'actions/test_times/action_compare_tt_sum.php',
    {
        method:'POST',
        parameters:{id_ttv1: id_ttv_p1,
                    id_ttv2: id_ttv_p2,
                    vname1: vname1_p,
                    vname2: vname2_p},
        onFailure: Error_Alert
    });
}

function Error_Alert()
{
    alert('Error Ocurred. Please retry in a few minutes.');
}

    ...
```

./actions/test_times/action_get_tt_sum.php

```
<?php
    require('../../includes/base.php');
    require('db/TestTime.php');

    if( isset($_POST['id_ttv']) ) //if an id_ttv was sent
    {
        $id_ttv = $_POST['id_ttv']

        //Fetch the all information for that id_ttv
        $tt_sum = TestTimes::Get_TT_Sum($id_ttv);

        $smarty->assign('tt_sum', $tt_sum);
        $smarty->display('test_times/test_time_sum.tpl');
    }
?>
```

./templates/test_times/test_time_sum.tpl

```
<div class="article">
    <h2> Selected Test Sum </h2>
        <table>
            <tr>
                <th>Flow</th>
                <th>Retest</th>
                <th>Sum x4</th>
                <th>Sum x8</th>
                <th>Sum x16</th>
            </tr>
            {foreach from=$tt_sum item=tts} <!-- For each flow -->
            <tr bgcolor="{cycle values="#FFE595,#ffD060" name="flows"}">
                <th><b>{$tts.program_name}</b></th>
                <td>{if $tts.retest == 'f'} No {else} Yes {/if}</td>
                <td>
                    <a title="Show flow details."
                       onclick="get_ttime({$tts.id_ttp4})"> {$tts.sum4} </a>
                </td>
                <td>
                    <a title="Show flow details."
                       onclick="get_ttime({$tts.id_ttp8})"> {$tts.sum8} </a>
                    </td>
                <td>
                    <a title="Show flow details."
                       onclick="get_ttime({$tts.id_ttp16})"> {$tts.sum16} </a>
                </td>
            </tr>
            {/foreach}
        </table>
</div>
<br />
<div id="test_time"></div>
```

Anexo B

Survey

Inquérito de avaliação do *Report Center*

Modulo Device Data

Análise por Flow

Descreva o método que utiliza para analisar os resultados *Device Data*?

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Análise por Versão

Descreva o método que utilizava para analisar os resultados *Device Data* de um versão?

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Comparação de resultados entre diferentes versões

Descreva o método que utilizava para comparar os resultados *Device Data* entre *flows* de diferentes versões?

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Satisfação geral do módulo *Device Data*

	Mau	Suficiente	Bom	Muito Bom
Como avalia o módulo *Device Data* do *Report Center*?	□	□	□	□

Modulo Transfer Letter

Geração de uma *Transfer Letter*

Descreva o método que utilizava para criar uma *Transfer Letters*?

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Satisfação geral do módulo *Transfer Letter*

	Mau	Suficiente	Bom	Muito Bom
Como avalia o módulo *Transfer Letter* do *Report Center*?	□	□	□	□

Modulo Test Times

Consulta dos *Test Times*

Descreva o método que utilizava para consultar os *Test Times*?

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Comparação de *Test Times*

Descreva o método que utilizava para compara *Test Times* de versões diferentes*?*

_____
_____
_____

Avalie o método que descreveu nos seguintes parâmetros:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Avalie o *Report Center* nos seguintes parâmetros, se o utilizasse em vez do método que descreveu:

	Mau	Suficiente	Bom	Muito Bom
Legibilidade dos resultados	□	□	□	□
Complexidade do uso	□	□	□	□
Tempo Necessário	□	□	□	□
Flexibilidade na utilização	□	□	□	□
Eficiência	□	□	□	□

Satisfação geral do módulo *Test Times*

	Mau	Suficiente	Bom	Muito Bom
Como avalia o módulo *Test Times* do *Report Center*?	□	□	□	□

Avaliação geral do Report Center

Já utilizou o *Report Center*? **Sim / Não** (sublinhe a sua resposta)

Se respondeu "**Não**" por favor explicite porque:

- □ Estou a pensar em faze-lo
- □ Não foi suficientemente divulgado
- □ Preciso de mais formação
- □ Não me traz vantagens
- □ O meu *team leader* não me incentiva
- □ Outro:_____

Se respondeu "**Sim**" por favor responda as seguintes questões

	Mau	Suficiente	Bom	Muito Bom
Como avalia o *Report Center*?	□	□	□	□

Vai utilizar o *Report Center* mais vezes? **Sim/ Não** (sublinhe a sua resposta)

O *Report Center* tem tudo o que necessita? **Sim/ Não** (sublinhe a sua resposta)

Acha que deve ser melhorado? **Sim/ Não** (sublinhe a sua resposta)

Se respondeu "**Sim**" na pergunta anterior, por favor explicite porque.

_____
_____
_____

Referências

[1] José Miguel Campos. Fatadesk project - results database. Technical report, Faculdade de Engenharia da Universidade do Porto, December 2007.

[2] Robert H. Gilmore W. Jason; Treat. *Beginning PHP and PostgreSQL 8: From Novice to Professional*. Apress, 2006.

[3] David Flanagan. *JavaScript : The Definitive guide*. O'Reilly, 2002.

[4] A web commerce group case study on postgresql. Technical report, Web Commerce Group. Available in http://www.postgresql.org/files/about/casestudies/wcgcasestudyonpostgresqlv1.2.pdf.

[5] Korry Douglas. *PostgreSQL (2nd Edition)*. Sams, Indianapolis, IN, USA, 2005.

[6] Zend Technologies Inc. An overview on php. White paper, Zend Technologies Inc., 2007. Available in http://static.zend.com/topics/overview_on_php.pdf.

[7] Php usage stats. Available in http://www.php.net/usage.php, 2007. [consulted on June, 2008].

[8] Monte Ohrt and Andrei Zmievski. *Smarty - the compiling PHP template engine*. Lincoln, Inc., 2003.

[9] R&os: Pdf class. Available in http://www.ros.co.nz/pdf/, 2006. [consulted on June, 2008].

[10] Fpdf: Pdf generator. Available in http://www.fpdf.org/, 2004. [consulted on June, 2008].

[11] Igor Henrique Berlitz. Gerador gráfico de relatórios utilizando a classe fpdf. Centro Universitário Feevale Instituto de Ciências Exatas e Tecnológicas, 2007.

[12] ECMA International. *Standard ECMA-262*. 1999.

[13] Joe Marini. *Document Object Model*. McGraw-Hill, Inc., New York, NY, USA, 2002.

[14] Kailash Chandra, Sapana Suhani Chandra, and Shyamal Suhana Chandra. A comparison of vbscript, javascript, and jscript. *J. Comput. Small Coll.*, 19(1):323–335, 2003.

[15] Clinton W. Smullen III and Stephanie A. Smullen. An experimental study of ajax application performance. *JOURNAL OF SOFTWARE (JSW)*, 3, 2008.

[16] Reuven M. Lerner. At the forge: Prototype. *Linux J.*, 2007(153):11, 2007.

[17] jquery documentation. Available in http://docs.jquery.com/Main_Page, 2007. [consulted on June, 2008].

[18] The book of dojo. Available in http://dojotoolkit.org/book/export/html/589, 2007. [consulted on June, 2008].

[19] Google suggest: Frequently asked questions. Available in http://labs.google.com/suggestfaq.html, 2008. [consulted on June, 2008].

[20] David S. Walonick. *Survival Statistics*. StatPac, Inc., 2004.

[21] José Miguel Campos. Report center - transfer letter and test times module. Faculdade de Engenharia da Universidade do Porto, 2007.