Departamento de ENGENHARIA INFORMÁTICA

Instituto Superior de Engenharia do Porto

Erasmusline
Orange Team Project Report

2010 / 2011

1050053 Daniel Lopes
1070462 Pedro Ferreira

isep Instituto Superior de Engenharia do Porto

Erasmusline
Orange Team Project Report

2010 / 2011

1050053 Daniel Lopes
1070462 Pedro Ferreira

Licenciatura em Engenharia Informática

July 2011

Coordinator: **Nuno Escudeiro**

ErasmusLine
The Orange Team

Project Report
June 2011

Students:
Arne Reimer
Arne Lipfert
Stéphane Polet
Nathan Van Assche
Aggeliki Katsiampouri
Mountrakis Stefanos
Ina Ivanova
Zvezdomir Tsvyatkov
Gudmundur Freyr Hallgrimsson
Pedro Ferreira
Daniel Lopes

Erasmusline – Orange Team Project Report

June 2011

Germany

    Students:
Arne Reimer
Arne Lipfert

    Coordinators:
Helmut Dispert

Belgium

    Students:
Stéphane Polet
Nathan Van Assche

    Coordinators:
Kristien Van Assche
Davy De Winne

Greece

    Students:
Aggeliki Katsiampouri
Mountrakis Stefanos

    Coordinators:
Papadourakis Gewrgios

Bulgaria

    Students:
Ina Ivanova
Zvezdomir Tsvyatkov

    Coordinators:
Margarita Todorova

Iceland

    Students:
Gudmundur Freyr Hallgrimsson

Portugal

    Students:
Pedro Ferreira
Daniel Lopes

    Coordinators:
Nuno Escudeiro

Score	
Observations	
Additional Information	

Table of Contents

Illustration Index

Index of Tables

1 Acknowledgements

1.1 P1-CONFIG, P2-INFOX (Germany)

First of all we want to thank Mr. Dispert for giving us the opportunity to take part in this project. It was a great experience to work with other students from all over europe and make contribution to the european idea. We also want to thank all members of the orange team. It was a very nice and funny time to work with you!

Arne Lipfert and Arne Reimer, Germany

1.2 P3-ALERT, P6-EXAM (Bulgaria)

We are very grateful to our teacher - Assoc. Professor PhD Margarita Todorova and entire staff of University of Veliko Tarnovo, our team members and the organization of the event.

1.3 P4-OUT (Belgium)

This project was a very enriching experience. It contained a lot of aspects that we will be able to use in our working life. We didn't only enlarge our technical experience when working on the EramusLine web application, but also our social and cultural experience.

In this acknowledgement we would like to thank various people. First we would like to thank Mrs. Kristien Van Assche and Mr. Davy De Winne for giving us the opportunity to take part in the project. Also, they were always available with advice and guidance, when we ha some questions implementing the web application.

Next we would like to thank the complete Orange Team for this nice experience, not only for the hard work that each team has put in this application, but also for the pleasant weekly online meetings and corporation.

1.4 P5-IN (Greece)

I would like to thank my partner Stefanos Moundrakis for his priceless help in achieving the goal to have the forms prepared for the presentation and also my coordinator Mr Papadourakis Gewrgios for giving me the chance to take part in this project which eventually turned out to be a very beautiful experience. Thanks to them and of course all the partners of that project, the workflow was pleasant enough.

Katsiampouri Aggeliki-Ioanna

I would like to thank mr. George Papadourakis for accepting me at the project, then i would like to thank Stephane from Belgium for his constant help and guidance.

Moundrakis Stefanos

1.5 P8-STATS (Portugal)

I would like to thank my wife for her patience and immense support in troubled times, thank Pedro for keeping being present and sharing his treasured opinions, and for professor Nuno for helping us gain insight and making necessary decisions throughout this project. The team stays in my heart for years to come.

Daniel Lopes

I would like to thank Daniel for his experience which proved invaluable, and Nuno for going forward with this project, from which I learned a lot.

Pedro Ferreira

2 Abstract

MUTW stands for Multinational Undergraduate Team Work. The goal of MUTW is to let students of eleven different universities work together on one common project. The assignment of this year was to implement the ErasmusLine web application. The purpose of this application was to digitalize the complete Erasmus process, so that students who want to go on Erasmus didn't have to fill in all the forms by hand, butt everything would happen digitally.

The eleven universities taken part in this project were separated into two teams, namely team Orange and team Blue. Those two teams would then work for about three months on implementing the application and at the end of those three months, they would have to give a presentation with their final result.

The implementation of the application itself was divided in eight modules, which were distributed among the universities in each team. By dividing this project in different modules, each university could work independently on their own module, but this made the communication part of the project very important. Although each university could work independently on their modules, nevertheless did some modules depend on others.

The application was written in XHTML with PHP as server-side scripting language. Besides XHMTL and PHP we also JavaScript/jQuery as client-side scripting language. For the course-matching module we also used Python. For the complete implementation of this project we used a PHP-library called Plonk, which is very useful to construct a website using the MVC (model-view-controller) model. By using this library the integration of the different modules into one working application was very easy.

This assignment was a very enriching experience for the whole team. During the implementation of the ErasmusLine web application we had some issues and bugs to fix, but eventually we were able to present a working application at our final meeting in Heraklion, Greece.

3 Keywords

Database, data-management, user-management, sql, security-levels, homepage, main-menu, privileges, permissions.

Interoperability, interface, data transfer, file transfer, encryption, security, JSON, 3DES, cURL

Erasmus, forms, server-side form-validation, client-side form-validation, form-persistence, process flow, signatures.

Exams module, database, interface, coordinator, student.

Statistics, analytics, trends, graphs, pivot tables, data warehouse, data refreshment, executive information system, EIS, online analytical processing, OLAP, extract transform load, ETL, data mining, facts, dimensions, measures, posix, daemon, mysql, php, html, jquery.

4 Introduction

4.1 MUTW Program Presentation

The MUTW(Multinational Undergraduate Team Work) Project is created by teachers and students from all over the Europe. Its main purpose is to design all the members together a functional site which is going to help the undergraduate students and coordinators avoid the bureaucracy of filling all the necessary forms in case they want to use the Erasmus Program. The site which is called "Erasmusline" is approachable by the students by using their unique password given by their University and of course by the coordinators who will work as administrators for the site.

Furthermore, this project has deeper goals to be achieved. Students have to work all together for 3 months in order to make the site completely functional. This requires teamwork and communication spirit. As a result, the participants will not only practice their english language skills, but also the skills mentioned above.

The participants are students from 11 different Universities throughout Europe which use the Erasmus Program. Each University offers 2 students for the project. Every university has its own unique part in the project to prepare, and during these 3 months, pieces are getting together in order to make the final result, the site.

4.2 Project Scope and Motivations

Each student has his/her own motivation for participating in this project. Generally, the project is part of the studies for all the students. More specifically, for Belgians is going to be their Internship.

Greeks and Germans are going to use their part in their thesis, which is necessary to be accomplished in order to graduate. For the other members, this project is an additional part in their studies. Apart from the educational stuff, this can be a great opportunity for everyone to work together with people from other countries and test his/her communication skills. Eventually, this project turns out to be a unique and valuable experience which can open new doors to our academic world.

The objective of the project is to create a totally functional website which is going to be used by all those who involve in the Erasmus program, and by that we mean the students and coordinators.

The first ones will be able to fill the forms needed for getting accepted by the University abroad and the coordinators will be able to log in as administrators, so they will be having access in every form in order to make the appropriate moves for communicating with the other University.

The beginning of the project took place in Kiel,Germany where all the participants met for the first time and each team made its first presentation. This presentation included the theoretical timeline until the final presentation and what was the plan for creating the website step by step. There are two teams which take part in the project, the orange and the blue team. Each team consists of 6 Universities from all over the Europe and in the final presentation the coordinators will evaluate each team according to the work which was done these 3 months and the presentation itself. Participants are gonna be also

evaluated separately one by one.

4.3 Project Presentation

As mentioned above, there are two teams which take part in the MUTW project. The blue team and the orange team. The orange team consists of 6 partners and the blue team consists of 5 partners.

The project is separated into 4 areas, The Configuration, the Services, the Business and the Statistics. Each of these areas is also a composition of packages. Every university which participates in the project is responsible for executing one of these packages. In the picture above , it is clearly mentioned what packages include every different area.
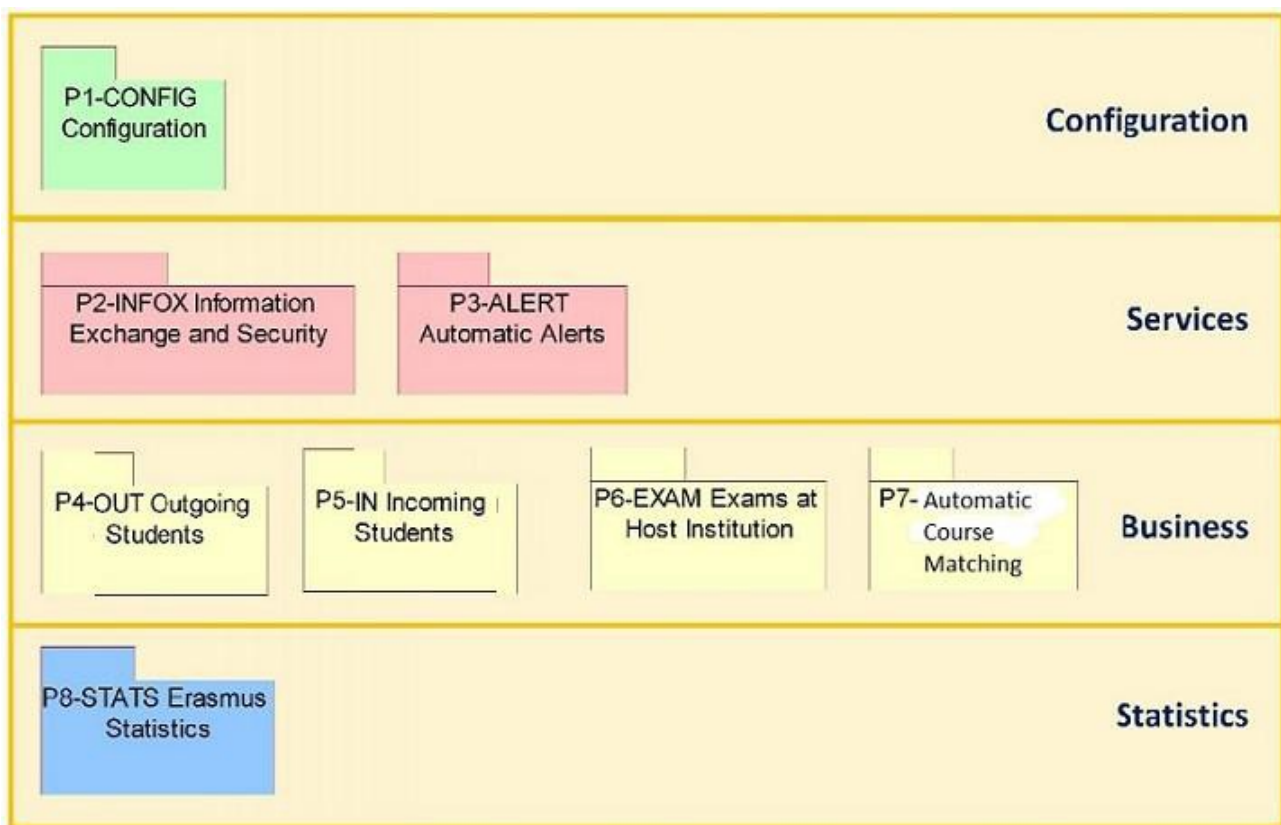
Illustration 1: Package placement

As the blue team consists of 5 partners, the packages P7 and P8 may be dropped out in their project.

In a brief summary of what every package does, the P1-CONFIG is responsible for the general profile of the application, as it manages the accounts and profiles of the users. Every other package is relied on this one. The P2-INFOX holds the servers and database of every institution in order to apply the web application. Next package, P3-ALERT is responsible for the timeout events. Packages P4-OUT and P5-IN are the designers of the forms that must be filled by the users. Package P6-EXAM helps the students to require exams in a lesson, if there is no specific way of evaluation for this lesson. Finally, package P7-MATCH is responsible for matching the lessons from home institution to lessons of the host institution and the last one, P8-STATS collects the operational data and expose it to

the users.

In the table bellow, every institution is matched with each package.

	Orange Team			Blue Team	
Package	Partner	ECTS		Partner	ECTS
P1-CONFIG	Kiel	12-16		Glasgow	10
P2-INFOX	Kiel	12-16		Glasgow	10
P3-ALERT	Veliko Turnovo	15-20		Vigo	12
P4-OUT	Gent	18		Siegen	20
P5-IN	Heraklion	20		Izmir	8
P6-EXAM	Veliko Turnovo	15-20		Vigo	12
P7-MATCH	Reykjavik	6		-	-
P8-STATS	Porto	18		Sofia	20

Table 1: Package Assignment

This application will be used by many different people, so it has to be easily accessible and understandable. It must also provide forms for all types of the Erasmus program, since there are forms for exchanging institution and others for internship in institution or in companies. People who use this application must have the possibility to print any form needed and be also capable of adding,deleting and change their data (only when necessary).

4.4 Technologies Overview

Before beginning analysis and development of the project's application, the Orange team decided from the various technologies listed in the specifications given the necessity of each task that was needed to complete this project. During this project we relied solely on Open Source Technologies as intended by the specifications. These technologies are not abridged by copyleft licenses[1] like the Gnu Public License, but some remain GPL-compatible.

In the next sections the technologies used are separated by task context and specifications wise.

4.4.1 Programming languages

PHP

This programming language is a standard language for quickly writing web applications. It is a dynamically typed, interpreted language with garbage collection and supports Object Oriented Programming, providing a syntax similar to older programming languages like C and Perl.

The project is essentially be written in this programming language, given that is the most easy to setup and learn amongst academics.

Plonk

For basic application workflow, the application uses the Plonk library - a subset of the Spoon library[2] - that automates functionalities like page routing, session handling and page templating.

Python

The Python programing language is an interpreted, dynamically typed programming language with garbage collection, used in the P7-MATCH package for it's use of the Natural Language Toolkit.

NLTK

The Natural Language Toolkit is a suite of libraries designed for natural language processing, convenient for matching students' courses and equivalents between HEI's.

---

1   http://en.wikipedia.org/wiki/Copyleft_licensing
2   http://www.spoon-library.com/

JavaScript

Javascript is an interpreted, dynamic and weakly typed programming language mostly used in client side (Web Browser) scripting. It has some Object Oriented support with garbage collection. This project uses javascript to reduce server-side load by performing tasks like form validation, generating content, Ajax[3] calls to the server, etc.

jQuery

jQuery is a pluggable javascript library that speeds up client-side script development by implementing a Domain Specific Language capable of quickly processing XHTML DOM[4] elements for various aesthetic purposes. It also provides convenient functions prepared for Ajax communication or pre-programmed plugins like calendars, color picker windows, etc.

4.4.2 Other Languages

XHTML

This is a markup language used to design websites in similar fashion to regular HTML but with stricter rules, and similar functionalities to XML, for example, the use of namespaces.

This project uses XHTML version 1.0 of it's Transitional guidelines. Given it's more strict nature a common Web Browser will waste less time processing web page content.

Making a website that will be used by thousands of students brings the concern of usability and accessibility. The ErasmusIne project's Website will be developed under the guidelines of the Web Content Accessibility Guidelines 2.0[5] and strive for a minimum first level of acceptance.

CSS

This stylesheet language changes the presentation aesthetic of websites in a non intrusive way, this way being able to switch style configuration rapidly without touching the markup language files.

4.4.3 Databases

MySQL

A relational database system designed for speed and rapidly building content-oriented websites. This technology serves as the persistent data layer in our application for various tasks such as the workflow of the Erasmus Process, the users and students management, to the Data Warehousing used in the P8-STATS package.

---

3   http://en.wikipedia.org/wiki/Ajax_(programming)
4   http://en.wikipedia.org/wiki/Document_Object_Model
5   http://www.w3.org/TR/WCAG20/

4.4.4 Development Platforms

SourceForge

SourceForge[6] is a project hosting website and development platform that provides infrastructures for the developers to quickly setup a project and use a varied number of features like source code management systems, mailing list systems, forums, project management applications, Wiki applications, etc.

4.4.5 Development Tools

Git

Git is a distributed source code management system used to keep track and monitor source code versions, allowing the developer to determine release versions, track code errors and share application code between other developers distributed around the world.

This system saves the code in containers called repositories. Each time a developer wishes to save a change made to the code, he commits file changes to the local repository. A record of differences is then saved in the repository along with a short description of the author. When the user is ready to share their code changes with the rest of the team, he pushes those changes to the central repository on the SourceForge site, so that other team members can pull those changes into their local repositories.

Git was available to the project via SourceForge, which also provided an online interface[7] to check out repository activity and provided a RSS[8] feed so that users can be notified of recent changes to the code.

During the development of this project the team used the Git repository to keep a record of the project planning, meetings reports, user manuals, the project presentation and this project report, along with the applications code.

A usage example would consist of the following scenario (using a terminal console):

1. In a fresh installation, the developer would start by creating a copy of the SourceForge repository and subsequent directory structure and files, using the following command:

   ```
   git clone git://mutworange.git.sourceforge.net/gitroot/mutworange/mutworange
   ```

2. After making changes to the code, the developer would issue a commit:
   ```
   git commit -a -m "fixed a bug in the X package"
   ```
   Those changes would be saved locally.
3. After making sure the code changes worked, the developer can share the changes with the team:
   ```
   git push origin master
   ```
   Those changes would be merged with the central repository
4. When new code changed are pushed to the central repository, the user can fetch them using the command:

---

6   http://www.sourceforge.net/
7   http://mutworange.git.sourceforge.net/
8   http://en.wikipedia.org/wiki/Rss

```
git pull
```
This fetches the latest changes to the code and merges them with your current local work.

Git also provides code branching functionality, which consists in creating an alternate version or path in the application development. Branches are good for creating new features on pre-existing code, but these features cannot be merged yet with the "master" branch.

Another feature offered by Git is code tagging, enabling to checkout various application versions of a branch in a certain context, for example a tag called *milestone-1* or *release-1.2* would correspond to a point were code development stopped and is ready to be deployed to the servers.

Source code management systems ensure that a team of developers keeps track of their work in an effective and efficient way.

Eclipse / NetBeans

Integrated Development Environments like NetBeans and Eclipse provide an interface for code editing for various programming languages and has a set of plugins to integrate other features in the interface, such as integrated Git repository management without resorting to terminal commands.

4.4.6 Other Tools

Total Validator

Total Validator[9] is a cross-platform application designed to validate Web Sites against established W3C[10] standards. This application not only validates code for XHTML but also validates Web Site accessibility guidelines, according to the WCAG 2.0 specifications. Has a bare minimum we validated the Erasmusline application against WCAG 2.0 level A, but the application also validates on levels AA and AAA where applied.

Delivery

This application tries to make Javascript files smaller by removing comments, spaces and new lines to provide faster downloads by the Web Browsers.

Google Docs

During the course of this project some team members used Google Docs to write technical reports that would later serve to compile into this final report. Google Docs allowed the team members to create and edit text documents concurrently and in real time with no fear of corrupting the documents. Google Docs also supported versioning, which enabled us to see, for example, which team member edited which part of the document.

---

9   http://www.totalvalidator.com/
10 http://en.wikipedia.org/wiki/W3c

Cacoo

Cacoo[11] is an online, realtime collaborative drawing tool which various team members can use to produce UML diagrams, draw software deployment plans, etc. and was used in several points of this project.

MySQL Workbench[12]

This tool helps the user design and project databases in a visual way by creating Entity Relation models of the tables, specifying relational constraints, create and edit indices, creating user roles and permissions, default values for tables, etc. and later export those properties to a SQL file to be immediately imported to a MySQL server.

4.4.7 Conventions

For the sake of usability this projects web-site and database content will be using the UTF8 character set encoding. This was a major concern since some Erasmus partners use different alphabets – for example Greek and Turkish - and the team had to use a universal character set that accepted the various alphabets existent in the European Continent.

Other conventions were merely determined for development comprehension of the team. While editing code the team member would have to follow a series of guidelines to ensure that everyone could understand his code when reading it. Some of these conventions were:

- using UpperCamelCase on object class names
- using lowerCamelCase on class methods
- documenting classes and methods using PHPDoc
- keep the code lines to a 80 character minimum, the excess code would be put in the following lines
- opening brackets in the first line of the code statement

These conventions ensured the whole team understood each other and understood what was being developed.

While developing the Application Interface it was very important when designing the stylesheets not to interfere with the default behavior of the XHTML interface elements. It was agreed that default behavior would be left intact and set additional behavior by conventionally extending these said behaviors. This way we ensured that when the team members developed in isolation, no necessary code rewrite would be needed.

---

11 http://www.cacoo.com/
12 http://wb.mysql.com/

4.4.8 Observations

During the course of this project, the P8-STATS team assisted the development of this project on deciding the proper conventions - listed in the previous section – and which tools to use, for example, accessibility validation.

The P8-STATS team was also in charge of setting up the SourceForge infrastructure for the rest of the team to use. This consisted in:

- Setting up user members to access the hosted project

- Creating mailing lists for regular email conversations

- Creating Forums to use in Polls, communication with the Blue Team, etc.

- Setting up the Git repository directory structure

- Instructing the team on how to use Git and which tools to use

Repository Structure
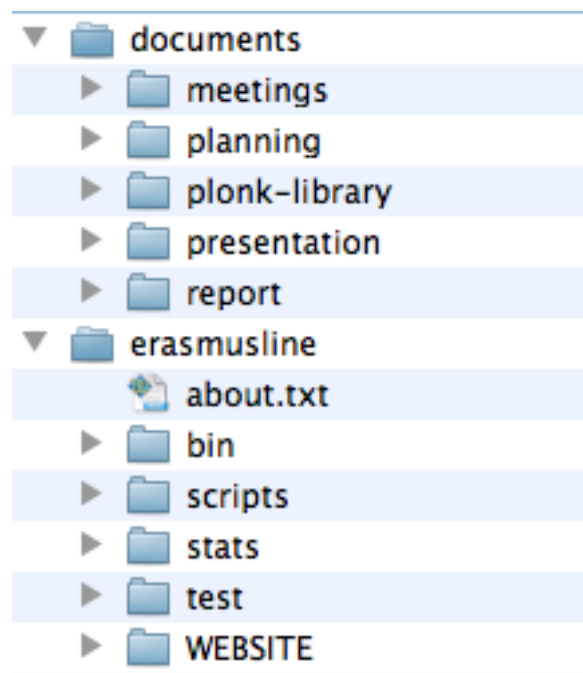
Illustration 2: Git Repository Structure

The Git repository was divided in two main directories:

- Documents – this directory holds early documentation, meeting reports and logs, general and package specific project planning, the final presentation and this report.

- Erasmusline – this directory contains startup and setup automation scripts, database scripts, unit test scripts, and the ErasmusLine application itself.

4.5 Report Structure

This report was divided into several logical chapters and sections for the reader to better understand what will be discussed. The following list explains this report's structure:

1. Introduction

   • Brief MUTW Program Presentation - a general presentation about the Multinational Undergraduate Team Work Program.

   • Motivations and Project Scope - the reasons for students applying for this project and the projects targets and objectives.

   • Brief Project Presentation - a brief presentation about 'ErasmusLine'

   • Technologies- reference to the technologies used in 'ErasmusLine'

   • Report Structure - the structure this report follows

2. Project Management

   • Project Management Plan

   • Meetings Plan

   • Team Work Development Methodology

3. Application Overview - overview of what each package does, how it interacts with each other, and basic erasmus process workflow.

4. Technical Description - analysis of all 'ErasmusLine' packages

   • ErasmusLine Architecture

   • P1 - CONFIG - the Configuration package

   • P2 – INFOX - the Information Exchange and Security package

   • P3 - ALERT - the Automatic Alerts package

   • P4 - IN - the Incoming Students package)

   • P5 - OUT- the Outgoing Students package

   • P6 - EXAM- the Exams at Host Institution package

   • P8 - STATS- the Erasmus Statistics package

   • P9 - MATCH- the Equivalent Courses Matching package

5. Conclusion

   • Project outcome - 'ErasmusLine' sum-up

   • Accomplished Goals - targets accomplished by Orange Team

   • Other Accomplished Works - supplementary achievements

- Limitations and Future Work - prospects for 'ErasmusLine' in the future
- Final Thoughts - each team members' point of view on how the project went

5 Project Management

5.1 Project Management Plan

The Project development was controlled using the software tool OpenProj[13] and editing the existing Gantt Chart, accessible by the team's members in the SourceForge Git repository[14].

Every package member was responsible for filling out their package Work Breakdown Structure, assign each deliverable time boxes and regularly filling the percentage of their work done.

		Name	Duration	Start	Finish	Predeces...	Percent C...	
1		ErasmusLine	65 days?	3/14/11 8:00 AM	6/10/11 5:00 PM		25%	
2		⊟P8-STATS	60.062 d...	3/14/11 8:00 AM	6/6/11 8:30 AM		72%	
3		⊟Analysis and Requirements gathering	12 days	3/14/11 8:00 AM	3/29/11 5:00 PM		86%	
4		Business Rules	5 days	3/14/11 8:00 AM	3/18/11 5:00 PM		80%	
5		⊟EIS	5 days	3/21/11 8:00 AM	3/25/11 5:00 PM	4	90%	
6		Measures	5 days	3/21/11 8:00 AM	3/25/11 5:00 PM		90%	
7		Dimensions	5 days	3/21/11 8:00 AM	3/25/11 5:00 PM		90%	
8		Report	2 days	3/28/11 8:00 AM	3/29/11 5:00 PM	4;5	80%	
9		⊟Data Warehouse	36.062 d...	3/30/11 8:00 AM	5/19/11 8:30 AM	3	90%	
10		Database schema	4.031 days	3/30/11 8:00 AM	4/5/11 8:15 AM		90%	
11		Define data source(s)	3 days	4/5/11 8:15 AM	4/8/11 8:15 AM	10	90%	
12		⊟Architecture	6 days	4/8/11 8:15 AM	4/18/11 8:15 AM	10	95%	
13		DW refreshment	2 days	4/8/11 8:15 AM	4/12/11 8:15 AM	11	90%	
14		ODS	4 days	4/8/11 8:15 AM	4/14/11 8:15 AM	11	90%	
15	✔	Indexing Solutions	6 days	4/8/11 8:15 AM	4/18/11 8:15 AM	11	100%	
16		⊟Implementation	15 days	4/18/11 8:15 AM	5/9/11 8:15 AM	12	93%	
17		DW refreshment	15 days	4/18/11 8:15 AM	5/9/11 8:15 AM		90%	
18		ODS	15 days	4/18/11 8:15 AM	5/9/11 8:15 AM		90%	
19	✔	Indexing	15 days	4/18/11 8:15 AM	5/9/11 8:15 AM		100%	
20		⊟Deployment	5.031 days	5/9/11 8:15 AM	5/16/11 8:30 AM	16	87%	
21		DW refreshment	5 days	5/9/11 8:15 AM	5/16/11 8:15 AM		80%	
22		ODS	5.031 days	5/9/11 8:15 AM	5/16/11 8:30 AM		90%	
23		Data Marts	5 days	5/9/11 8:15 AM	5/16/11 8:15 AM		90%	
24		Report	3 days	5/16/11 8:30 AM	5/19/11 8:30 AM	10;12;1...	50%	
25		⊟EIS	12 days	5/19/11 8:30 AM	6/6/11 8:30 AM	3;9	35%	
26		Architecture	12 days	5/19/11 8:30 AM	6/6/11 8:30 AM	24	40%	
27		Implementation	12 days	5/19/11 8:30 AM	6/6/11 8:30 AM	24	50%	
28		Deployment	12 days	5/19/11 8:30 AM	6/6/11 8:30 AM	24	30%	
29		Report	12 days	5/19/11 8:30 AM	6/6/11 8:30 AM	24	20%	
30		⊟P4-OUT	57 days?	3/14/11 8:00 AM	5/31/11 5:00 PM		6%	
31		Collect Different forms	15 days?	3/14/11 8:00 AM	4/1/11 5:00 PM		20%	
32		Setting up chart with the form flow	3 days?	3/25/11 8:00 AM	3/29/11 5:00 PM		0%	
33		Agree with P-IN on layout forms	5 days?	3/28/11 7:00 AM	4/1/11 5:00 PM		0%	
34		Determ which forms can be common	5 days?	4/4/11 7:00 AM	4/8/11 5:00 PM		0%	

Illustration 3: The Project's Gantt Chart in OpenProj

Apart from checking our project's and each individual package's performance, the team acted on democratic decision - when selecting the tools, when resolving open ended questions – recurring to presential meetings through the software tool Adobe Connect[15], IRC channels, direct email and the SourceForge Mailing List system.

Given the democratic nature of the team, each package team would also self-manage

---

13 http://openproj.org/
14 http://mutworange.git.sourceforge.net/
15 http://www.adobe.com/products/adobeconnect.html

their own package planning, using their own tools but always reporting to a general Gantt Chart.

5.1.1 Project Plan Description

This project was developed during the second semester of 2011, between the 21st of March and the 17th of June.

The following tables show each packages main tasks and milestones, each with their corresponding time box.

P1-CONFIG (Germany)

Milestones / Tasks

- **Database   21/03 – 29/04**
  - Design
  - SQL-Script
  - Installation Script
- **Design   25/03 – 15/04**
  - Design Mock Templates
  - Integration with Plonk Templating library
- **User Account   18/04 – 29/04**
  - Login functionality
  - Account page

P2-INFOX (Germany)

- Discussion about communication Protocol   28/03 – 1/04
- Define Security System   04/04 – 29/04
- Programming Outgoing data   02/05 – 13/05
- Programming Incoming data   02/05 – 13/05

P3-ALERT (Bulgaria)

- Research   22/03 – 06/04
- Making the daemon   07/04 – 20/04
- Synchronization with the others part of the project   21/04 – 27/04
- Email sending system   28/04 – 5/04
- Pop-ups   28/04 – 5/04
- GUI   05/05 – 09/05
- Testing   10/05 – 23/05

P4-OUT (Belgium)

- Collect Different forms   14/03 – 01/04
- Setting up chart with the form flow   25/03 – 29/03
- Agree with P-IN on layout forms   28/03 – 01/04
- Determine which forms can be common   04/04 – 08/04
- Implement non/common forms   11/04 – 29/04
- Creating ECTS forms   28/03 – 01/04
- Implement flow of forms   11/04 – 10/05
- Integrate information exchange module   18/04 – 22/04
- Intensive testing of flow   10/05 – 31/05

P5-IN (Greece)

- Research
- Develop the IN student forms
- Testing
- Report

P6-EXAM (Bulgaria)

- Research
- Making the student module
- Making the home coordinator module
- Making the host coordinator module
- Synchronization with other parts of the project
- Testing
- Report

P7-MATCH (Iceland)

- Research
- Develop data scrapper to match dummy data
- Develop production code
- Integration tests
- Testing
- Report

P8-STATS (Portugal)

- **Analysis and Requirements gathering**   **14/03 – 29/03**
  - Business Rules
  - **Key Performance Indicators**
    - Measures
    - Dimensions
  - Report
- **Data Warehouse**   **30/03 – 19/05**

- - Database schema
  - Define data sources
  - **Architecture**
    - DW ETL / Data refreshment
    - ODS and Metadata structure
    - Indexing Solutions
  - **Implementation**
    - DW ETL / Data refreshment
    - ODS and Metadata structure
    - OLAP module development

  - **Deployment**
    - DW ETL / Data refreshment
    - ODS and Metadata tables
    - Data Marts – Local and Remote DW
    - Report
- **EIS Interface   19/05 – 06/06**
  - Architecture
  - User Stories
  - Implementation
  - Deployment
  - Report

ERASMUSLINE (Orange Team)

- **Integration   16/05 - 31/05**
  - Integration Tests
  - Debugging
- **Final Report   01/06 – 17/06**
  - User Manual
- **Presentation   08/06 – 17/06**
  - Presentation Notes

The complete Gantt Chart can be found in the fourth Annex of this report.

5.2 Meetings Plan

As agreed on the first presential meetings, the team met initially remotely each week on mondays on 18 o'clock according to the UTC-0[16] time zone. Each meeting would last from fifty minutes to two hours if necessary.

During the requirements gathering and development stages of the project, the topics in each meeting discussed by each package members, were the in the following list:

- Report last week's activity

- Plan next week's activity

- Express difficulties and problems encountered, but not solve them right away

---

16 http://en.wikipedia.org/wiki/Coordinated_Universal_Time

Each package team would prepare in advance what to express in this first part of the meeting to keep this part short and quickly address issues in the third topic. This first part of the meeting lasts twenty minutes and helps the team stay updated and motivated with the evolution of the project.

After each team reported these topics, the whole team would follow-up with assigning team members to help solve the expressed problems in the third topic. If any other meetings were needed to solve pending issues, the affected members would arrange extra meetings during the week.

5.2.1 Observations

Initially each package team would prepare and assume control of the meetings in a rotational fashion each week, at least once. This proved to be non-productive given that some team members didn't prepare nor had the skills to conduct the meetings.

After a democratic vote from the team, it was decided unanimously that the P8-STATS team should prepare and manage the meetings as well as the whole team.

5.2.2 Communication Tools

The next sections describe the communication tools used to interact between team members.

Adobe Connect

This proprietary platform allowed team members to communicate with each other through live text, video and voice chat on modern web browsers. Most meetings where held in this platform.

The application also allowed the team members to share a computer screen to better explain an issue or expose a better alternative during the decision making.

The interoperability of this platform was enough to communicate through various systems.

Mailing Lists

When an urgent decision came up and the whole team needed to be present, a team member would broadcast an email through the SourceForge mailing list system and the rest of the team would reply as soon as possible, broadcasting their response to the rest of the team through the same mailing list topic.

In terms of interoperability, using email messages were enough if the team couldn't use live text chat and using the mailing list provided a way to use the emails like a forum.

IRC

If by some reason Adobe Connect wasn't available to some team members, the team would use an IRC channel created by the team to hold extra meetings – mostly integration meetings. It provided an online interface[17] or the team member could use an IRC client.

The interoperability was good if the team only needed text chat and the internet

---

17 http://www.mibbit.com/

connection was slow.

Forum

The SourceForge online platform also provided a private and a public forum, if for example, the team wanted to make a public conversation with the Blue Team or make a private poll about a certain topic.

Skype

This proprietary software[18] was seldom used between two team members for private or more hands-on conversations.

5.2.3 Observations

Most of the project's success was due to some early discussion of important decisions, specially by forum or mailing list, mainly by the P8-STATS team, Annex 2 as an email example to Belgium.

As another example, the successful choice of how to distribute the information parted from a document sent by mail early in development which can be checked as Annex 1.

5.3 Team Work Development and Methodology

Apart from the communication tools available to us there are other factors that affected the team members that directly shaped the outcome of this project:

- Different spoken/written language

- Different Multi-Cultural / Multi-Religious backgrounds

- Different perceptions

- Different communication styles

- Different life, academic or professional experience

- Different time zones

- Different countries

These aspects are part of a set of barriers with constant presence during the project:

1. Physical Barriers – the differences between team members' locations, time zones, failure in the communication channels (slow internet connection, sudden network drops, poor visual quality on video conferences, etc.)

2. Personal Barriers – the teams' different cultural backgrounds, the members physical and mental condition during communication, weak communication skills, etc.

3. Semantic Barriers – Different interpretations of a sentence or situation according to the context.

---

18 http://www.skype.com/

Given the small time frame to develop this project, it was crucial to establish a communication pattern that ensured objectivity in dealing with important decisions:

- Maintain an assertive style of communication
- Avoid parallel conversations during meetings – this could be achieved with some socializing chat before and after the meetings' set schedule
- Use proper, universal conventions during discussions – for example, begin sentences with predetermined words during the weekly report
- Avoid aggressive behavior by maintaining objectivity - it is normal for deadlines to often provoke stressful situations.
- Instill a spirit of camaraderie – working for the same goal and motivational discussions brings down some of the discussed barriers.

5.3.1 Observations

Any person working in a team which members he hardly knows will witness the above experiences. It was important not only to manage this project and deliver a working demonstration, but it was crucial to manage the team's emotional intelligence and keep a positive outlook.

Although developing this project in the actual presence of the team members – during the final team meeting - proved to be more productive, the initial obstacles in the beginning of the project weren't present at that time.

5.3.2 Development Methodology

After specifications were met and development tools decided upon, the team designed each package's interface as a point of communication between each other's packages. This technically translated in each package objects' method names and arguments to pass to each other, for example, the methods the P4-OUT object calls in the P8-STATS object to trigger the data gathering – for statistical purposes - after a Erasmus process phase has been created, approved or rejected in the application.

Other interfaces were more complex, for example for inter-server communication. The P2-INFOX package would need to establish a universal means of communication between other servers. That meant defining a message structure for the other packages to abide, and a response message structure providing the success or error of the acknowledged message. After those structures for communication were set, the team could begin development.

Taking advantage of modern source code management systems it is possible for a team member to develop their package code in isolation but still have at their disposal the rest of the packages and application to test their correct integration. Git[19] is a distributed source code management system provided in the SourceForge development platform that enabled the team to view the progress of each package, continue developing and run the Erasmusline application in each computer.

To test each packages functionality, the team use Test Driven Development[20] techniques when ever necessary. This was fundamentally important when business/workflow rules change and code rewrites might introduce errors. Test Driven Development can also serve as documentation to explain the Business rules and workflow of a particular functionality in the application.

To speed up development, installation and Continuous Integration[21] scripts were implemented to allow the developer to:

- quickly setup the databases and apply configuration settings to them
- run tests to check for malfunctioning code
- populate the databases with preliminary data
- activate background services

The Orange Team was developing the application on several Operating Systems – MacOSX, Gnu/Linux, Windows, etc. Using Continuous Integration helped discover cross-platform issues in our application's implementation that otherwise would be discovered in later times.

By using these automated scripts for each platform, the team ensured it would be possible to deploy the application on a wide variety of platforms without getting in the way of developing the application.

___

19 http://git-scm.com/
20 http://en.wikipedia.org/wiki/Test-driven_development
21 http://en.wikipedia.org/wiki/Continuous_integration

6 Application Overview

6.1 P1-CONFIG

Configuration – this package is responsible for:

- Validation of the other packages;
- User accounts and profiles;
- ErasmusLine home page and main menu.

6.2 P2-INFOX

Information Exchange and Security - This package allows exchanging documents and students' processes between instances of the ErasmusLine application. ErasmusLine server and database will be installed in every institution. This package provides a mechanism for securely exchange of the information.

6.3 P3-ALERT

Automatic Alerts - Automatic alerts system which is used to send messages for actions that are going to run out of date.

6.4 P4-OUT

Outgoing Students – this package helps OUT students, home coordinators and stuff to manage documents related to the entire process, assists in conversion from ESTC credits to a local scale.

P5-IN

Incoming Students – helps IN students, host coordinators and stuff to manage documents related to an entire exchange process, assists processing of transcript of records.

6.5 P7-MATCH

Automatic Course Matching – assists student to choose courses with requirements close to home institution`s courses. This package is based on comparing textual descriptions given from host institutions and student.

6.6 P8-STATS

Erasmus Statistics – this package aims to provide information that will help to improve Erasmus processes and to evaluate Erasmus impact and progress in a separate institutions.

6.7 Package Interaction

The application is build by using the Plonk PHP library. The software architecture used by this library is Model – View – Controller. That means that the logic of the application is separated into three parts:

- Model - manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state.

- View - renders the model into a form suitable for interaction, typically a user interface element.

- Controller - receives user input and initiates a response by making calls on model objects.

Every Package controller could load another desired package controller and communicate by the controller's interfaces. Remote interaction between packages would have to be made by the P2-INFOX package controller.

7 Technical Description

This chapter exposes the technical aspects of each package, providing insight on the independent motivations behind the packages, the analysis each team made according to the specifications, the architecture design made for project, the evolution in the analysis and development during the project time box and the final package structure.

8 P1-CONFIG

In this module the main configuration is set. It can actually split up to three parts: Data management, User management and page-layout.

Data Management

In the data management part we first made a short overview of the data that will be needed for the normal user functionality such as login, selecting institutions and selecting courses.

Next we arranged a meeting to discuss with the other team-members their requirements for the database system. Additionally we came to the conclusion that we will use a MySQL database with UFT8 encoding. So it is made clear that all European partner institutions will be able to store data in their own language characters like it is in Greece, Germany or Turkey.

During the project some groups came up with some adjustments to the database. Actually we decided to make no changes because of side effects to other packages but finally we had no other choice. With getting more and more in touch with the project and the workflow we allowed to make adjustments to the database with the condition to report changes immediately to all team members and acknowledge the changes not until every group has given the ok for their package.

Next figure shows the final database:

Illustration 4: Main Database Schema EER Diagram

As it is shown in the draft the most important table is the 'erasmusstudent' table. It is the

heart of the database and connects most of the other tables. It is necessary for the whole erasmus process, from the start to the beginning. Another important table is called 'erasmuslevel'. Here are the levels defined which a erasmus student will go through. Every level has e next level except the last one.

User Management

The data management plays a big role for the user management. All user data is stored in the database. We have a certain table with all necessary information about the user no matter what its role in the application is because every user has to login first (students or administrative staff). Another table defines the role of the user. Next figure shows all roles:

Role	*Description*
Student	An erasmus student who want to participate in the erasmus program
Teaching Staff	Same level as International Relations Office Staff
Erasmus Coordinator	Same level as International Relations Office Staff
Higher Education Institution	Same level as International Relations Office Staff
Industrial Institution	Not used yet in our application
International Relations Office Staff	These users coordinate the whole erasmus process and handle the students events.
Administrator	Special user who can organize the International Relations Office Staff.

Table 2: User Roles

We also decided to make a register mask public so that everybody can register and do not have to wait until an administrative will set up the account. Also we want to avoid any mistakes with names etc.

Page Layout

After the team discussed the framework they want to use and started with their package, we tried to set up to different layouts. Both layouts were good enough for the final page so we started to make a vote to see the favorite layout. We decided to give the more discreet but professional layout the chance to present our work. Next figure shows a part of the header a team member created on his own.
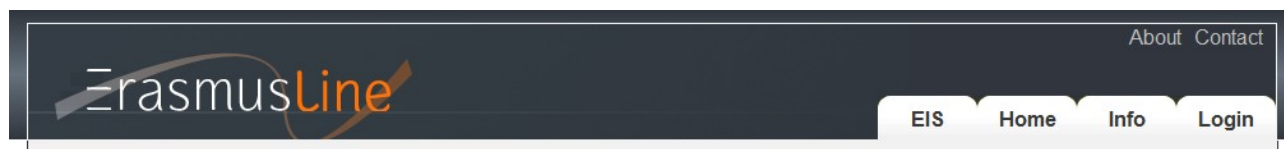
Illustration 5: Erasmusline Site Header

We created an Erasmusline banner on the top left of the site which includes the typical

colors of the Erasmus Project. On the right site are a few tabs for the main menu. These tabs have also a hover effect to make a small visual effect.

The main part of the application is in a box with 980 pixel with. This was very important for us that the content will not become wider because a lot of students use small netbooks with a maximum width of 1024 pixel. So the screen will always have a small buffer for the scrollbars e.g.

9 P2-INFOX

The INFOX-Module is the interface for transferring text-data and files. Its major function is to make sure all data that has been send is given to the specified method on the host server. To do so the team decided to use the cURL library. The data is furthermore send inside a JSON-string which is encrypted with 3DES algorithm.

cURL

The cURL library was first released in 1997. It is a computer software project providing a library and command-line tool for transferring data using various protocols.

cURL includes a command line interface and the libcurl-library which supports several protocols including HTTP / HTTPS and file transfer. It offers also username and password authentication and is IPv6 compatible.

In the application we only use a small amount of the libcurl library. The table shows the options that are set and their meaning:

CURLOPT_URL	Sets the destination URL
CURLOPT_POST	If true it defines that post data is send
CURLOPT_RETURNTRANSFER	If true the transfer returns a string
CURLOPT_SSL_VERIFYPEER	If false cURL stops to verify the peers certificate
CURLOPT_POSTFIELDS	The full data to post in a HTTP post operation

JSON

The JavaScript Object Notation is a text based file format for communication between different application / server. We decided to use JSON instead of Xml because of its slim but powerful usage. All data which is transferred between two server is packed in a JSON string which can also handle multi arrays.

3DES encryption

The Data Encryption Standard is a symmetric encryption algorithm. Even thou the DES is less save, encrypting it three times (Triple DES) makes it very safe and it is still the most used encryption method.

Implementation

There are a few ways the start a transfer. First there is the data transfer. It needs the methods, the tables and the data for each method.

public function dataTransfer($method, $table, $data, $idInst) { }

A JSON string is constructed with the tables and the data. In the next step another JSON string is constructed with the method to be called on the host server and the first JSON string as a parameter.

Example of the JSON String:

```
[
        {
                "method":"Method 1",
                "params":{
                        "table":"Table 1",
                        "data":{
                                "Data 1.1":"Value 1.1",
                                "Data 1.2":"Value 1.2"
                        }
                }
        },
        {
                "method":"Method 2",
                "params":{
                        "table":"Table 2",
                        "data":{
                                "Data 2.1":"Value 2.1",
                                "Data 2.2":"Value 2.2"
                        }
                }
        }
]
```

In the next step the string is encoded with 3DES encryption so that no plain text will be send over the WWW.

On the host server, the encrypted String is decoded into Plain text and the JSON string is decoded into an object. The method will be loaded (if it exists) and finally called.

The other way to make a transfer is for files. This method requires the filename, the specified host institution, the user-id and finally the method to load on the host server.

public function fileTransfer($method, $file, $idInst, $userid) { }

Again a JSON string is constructed with the method and the user-id as folder-parameter. This string is also encoded with 3DES algorithm to avoid sending plain text.

On the host server the encrypted string is decoded to plain text and the JSON string is again decoded into an object and the given method from the corresponding module is called.

10 P3-ALERT

The ALERTS – This package generates automatic alerts for pending tasks. These alerts are triggered by timeouts associated to requests that are not answered in due time.

10.1 Module logic

At a scheduled time once a day the daemon checks the database for deadlines for the forms that erasmus student should fill in. If there are three days and less till the expiration of terms for completing of a document, it sends an email to the student. The daemon is running on the server.

10.2 Achieved Goals

The logic of the daemon is implemented

Daily checks the database for deadlines

Finds the students that are close to the deadline for fill in documents

10.3 Further Improvements

Send an email to remind the students

Improving the performance of the demon

Testing the alert system

10.4 Technical Decisions

PHP

The application is created via PHP programming language. PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge. A competitor to Microsoft's Active Server Pages (ASP) server-side script engine.

MySQL

For the management of the database queries is used MySQL. MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

11 P4-OUT

11.1 Register

The first thing students need to do when they arrive at the application is sign up to be able the login and fill in their pre-candidate form.

Illustration 6: Registration Form

This registration form is just for students. They can register themselves by filling in this form. Afterwards they get an email with a link, so we can verify that the filled in a correct email address.

Since not only students have to use this application, but also lecturers, Erasmus coordinators and so on, there's a different register form available for the administrator of the application, so he can add these people to the database, with their own user level.

11.2 Login Functionality

Since Plonk didn't have a login-method on board and we didn't want to have to rewrite the login functions on each page, we made a little adjustment to Plonk so we only had to write the login functions one time en we are able to use them over the complete web application.

11.3 Collecting the different forms

The first step in the implementation of the P4-out module was collecting all the forms that need to be filled when participating in Erasmus. We got most forms from our university from our International Office, but we still needed to get all the forms from the other partners.

There are 3 different types of forms:

• ECTS-forms: These are official forms which are used by all the institutions.

• Common-forms: These forms are the same in all institutions.

• Non-Common forms: These forms are not the same in all the institutions, so we had to try to make a common form for all these forms, or make a form per institute.

Once we had all the forms, which took a bit longer than expected, we could start implementing all the forms.

11.4 Implementing the forms

11.4.1 Introduction

While we were waiting for the other forms, we already started with implementing the ECTS- and common forms for the application. And afterwards we tried to build common forms of the non-common ones, so we could one application which would be the same at all the partners.

11.4.2 JavaScript form-checking

For the client side form-checking of our forms, we used the jQuery Validation Engine. The big advantage of this library unlike the other libraries we have investigated is that this library uses instant form checking instead of checking the whole form when it is submitted. So the user is immediately informed if he inserts a fault value.

Another advantage of this library is that it's very easy to implement it in your code. Also there's the possibility to add extra rules to the library.

```
<input type="text" class="validate[required,custom[email]]" name="email" />
```

Illustration 7: Form Validation

11.4.3   PHP form-checking

For the server side form-checking, we used the benjaminkeen.com PHP validation library.
We chose this library because it's easy to implement and extend it with our own rules. Also this library provides form persistence when the server marks a fault in the filled in form.

$rules[]   =   "required,email,   Please   enter   your   email   address.";
$rules[]   =   "valid_email,email,Please   enter   a   valid   email   address.";

11.4.4   Store the forms in the database

To store the different forms in the database, we had several solutions.

One table for each form

The first solution was to create a table for each form in the Erasmus process. In that case, we could create a table with all the important fields of the form and the student ID and it would be easy to find a form of a student. On the other hand, this solution would have made our database design a lot more extended.

One table for all the forms

Illustration 8: Forms table

The second solution, and the one we have used in our application, was to make one table for all the forms. As you can see in the image we store all the content of the forms in a string (JSON) and with the type of the form and the student ID we can easily find each form of each student back.

11.4.5 Signatures

Digital signatures

The first option we had in implementing the signatures for the application was to completely do the signing of the forms digitally. In the spirit of this project, namely digitalizing the Erasmus process, this would be the ideal solution, because almost no forms would have to be printed out.

The first option we had with this solution was add a signature field to our forms, and the user would then be able to sign this form by either dragging an image of their signature in that box, confirming this with his password. Or we would let the user upload his signature at registration, so when he has to sign a form, the image would already be available at the signature field and he would just have to insert his password to confirm the signature.

The second option we examined was whether it was possible to let the users sign (and even register) with their Electronic ID. But since not all participating countries already use Electronic ID, we didn't implement this solution.

Eventually, after doing some research with the other partners, we noticed that completely digitally signing the forms wasn't feasible, because not all partners agreed with this solution.

Printing and scanning

The solution we finally used in the web application, which was actually against the objective of the whole project, was that the first person we needs to sign a form would print out the form from the application, and manually hand it over to each person who needs to sign that form in his own institution. Afterwards, if every one of the home institution has signed, the last person would then scan the signed form into the application and send it to the host institution.

Using this solution, there's still some paperwork involved in the application, but we tried to reduce it to the minimum.

11.4.6 Flow

Register: Pre-candidate

The first step in starting the Erasmus process is to register on the web application. Here a student (all others users, for example coordinators, are added by an admin) has to fill in some general information. Afterwards the student gets an email send to the address that he has filled in. The email contains a link to confirm his email address.

After confirming his email address, the student is now able to login to the website and fill in his pre-candidate form. Here the student has to choose three countries and he has to fill in his motivation to go on Erasmus. Also the student is able upload his CV, transcript of Record or Certificate of Foreign Language here.

When he has filled in his pre-candidate, the student has to wait for the confirmation of the Erasmus Coordinator before he can continue on the website.

Student-Application Form: Learning Agreement

After the confirmation of the Pre-candidate, the student is now able to fill in his Student-Application Form and Learning Agreement. This was the hardest part to implement for our module, because once the student has filled in both forms, there are several possibilities, it is not as straight forward as approving or disapproving the pre-candidate form. These forms have to be send at the same time to the host institution, but each form can be accepted/denied separately.

For this reason, we have solved this problem by using different codes for each possible state in this process.

30) Student-Application filled in, Learning Agreement not filled in yet.

00) Student-Application rejected, Learning Agreement rejected
01) Student-Application rejected, Learning Agreement approved

02) Student-Application rejected, Learning Agreement pending

10) Student-Application approved, Learning Agreement rejected
11) Student-Application approved, Learning Agreement approved
12) Student-Application approved, Learning Agreement pending

20) Student-Application pending, Learning Agreement rejected
21) Student-Application pending, Learning Agreement approved
22) Student-Application pending, Learning Agreement pending

These codes are used to show/hide the forms on the home page or to be able to decide what the next step in the Erasmus process is for this student.

11.4.7   Coordinators

If a coordinator logs in to the application he is redirected to a different home page than if a student logs in. At this home page, the coordinator can view all the forms that are currently pending for his conformation.

If he for example clicks on "show pre-candidate forms" (for a home coordinator) he can view all the pre-candidate forms that still need to be approved/denied. From here on he is able to read the forms, give his motivation about the form and approve/deny it.

Show Precandidate Forms
Show Student Application Forms
Show Learning Agreements
Show Change of Learning Agreements
Students
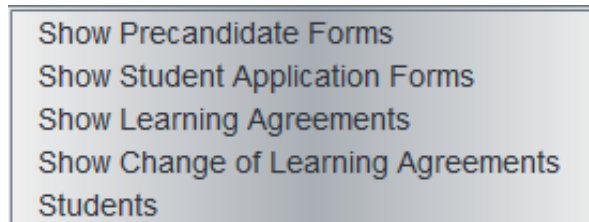
Illustration 9: Options Side Bar

11.4.8   Residences

Once a student is registered on the application and he has filled in his pre-candidate form with his preferred countries, he can start looking for a residence at these countries. Off course, since he isn't sure at which country he will be going, that's only certain after the confirmation of the pre-candidate, he isn't able to sign up for the residence, but we thought that it was a good idea to make the option available so he can already start searching the database of the residences.

11.4.9   Information Exchange

The INFOX module provided by the German team is very important for our module. They have provided us with some sort of an API which we can use to transfer database data between the different institutions. There are two mayor groups of information that need to be synchronized. The first one is info about the institutions and the second about users.

Institutions: This information is being fully synchronized between the institutions which are partners. The information that is being synchronized consists of information about the institutions itself, educations and courses. This is done every time information is changed in a particular institute.

Users: Information about students and the different coordinators is only copied when it's certain that a student goes on Erasmus. In that way it is possible to minimize traffic and comply with the privacy policies of the different partners. So info is sent in JSON format every time a student fills in a form that requires approval of the host institution. Together with this JSON string the form is also sent in PDF. This is necessary because the forms need to be printed, signed and scanned in order to cope with the signature problem.

12 P5-IN

This package refers to the forms that have to be completed by the students that are planning to use the Erasmus program and by their coordinators. In the report below, we will explain all the themes and problems that had to be solved from the beginning of the project and how did they finally come to a solution.

12.1 Forms to be filled

The forms that this package includes are the Learning Agreement Change, the Registration the Certificates of stay / arrival / departure and the Transcripts of Records. The first one must be completed by the student but not necessarily. It is used only when the student wants to add, remove or make any change to the learning agreement form they have already filled in the beginning. The registration form is the last form to be filled by the student, and it includes the information about whether the student wants to rent a room through the Institution or not. The other forms are programmed to pull data from the database, so the coordinators have to fill only a few gaps.

12.2 Process / Workflow

The Learning agreement change form is send to the local coordinator , who approves it or not. Whether it is approved, it is sent to the host Institution, where they also have to approve it. The student also fills the Accommodation Registration form where he/she applies for accommodation via the partner Institution. After the student arrives to the partner University, the coordinator of that University has to fill the first certificate of arrival. This form (as all the certificates plus the Transcript of Records) are programmed to pull all the necessary data (eg student's and coordinator's data) from the tables in the database. So, the coordinator has actually only to complete the date when the student arrived. At the end of the accommodation period, the coordinator also completes and sends the certificate of departure and the Transcripts of Records to the home institution. The certificate of stay is automatically filled by the database when the other two certificates are send.

12.3 Design of the forms

The first thing we had to do was to make easily understandable but in the same time appealing forms to the user of the website. The Belgian partners of the project have created their own framework on which we are all based to work , and the Netbeans software were the first two tools we had in our hands for designing the forms. We used the HTML programming language and the CSS style sheet language to succeed a harmonic combination. The final style of the forms of course had to be harmonized with the general style of the website, so we knew from the start that the CSS would probably face some changes.

We collected all the corresponding forms that are now used for the Erasmus program from our University because we had to make sure that all the necessary data will be

enclosed in our electronic forms.

12.4 Functionality of the forms

The second step was to make the forms functional, in other words working correctly. That turned out to be the hardest part, since a lot of code was needed and many other programming languages. Apart from that, the framework we used is totally new, so we had no tutorials or any other help from internet to walk through.

The problems we had to face were much enough. First of all, the variables which constitute the tables of the database had to change all the time, because there were always some new data to be implemented. So, others were added, others were removed and as a result, the code was changing all the time. In addition, we had to cooperate with the belgian partners in order to make similar forms and functionalities. Of course, communicating via internet is not the easiest way to solve a problem. Time was also our enemy since the project had to finish throughout 3 months.

Despite that facts, we managed to make the forms functional and appealing. Continuing, we present the languages we used and the results of our work.
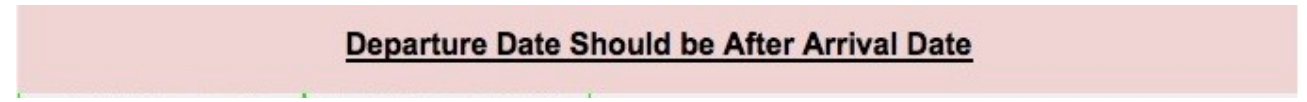
Javascript – PHP for checking

We used javascript for checking the validity of the forms. Javascript is a programming language which is used in the client -side and it makes the website valid. For instance, if someone forgets to fill one field or write numbers in a field for letters, then the code which includes javascript is running and usually messages are thrown on the screen pinpointing the wrong which deterred the website to work correctly. One example is the picture below. We set javascript on the Accommodation Registration Form. When the student is going to fill this form and he/she forgets one field, then the message " This field is required" is thrown on the screen. There is also one more message which is for the departure and arrival date, and it shows to the student the correct format for writing down the date.

Illustration 10: Accommodation and Payment option

We also used PHP programming language for checking the forms in the server-side. These two languages have of course different way of coding. The picture shows the message which is thrown on the top of the website when someone is going to fill the departure certificate before the arrival one.

Departure Date Should be After Arrival Date

PHP for validation

PHP is written and integrated into HTML code and it is useful for the server side by a PHP processor module which generates the web page document. It is compatible for every framework, and it is necessary for the proper functionality of a web page. So, we used php for including the functions in the codes.

Store – Pull Data

All the data that are filled in every field must be stored somewhere in order to keep a database and make the correct process for executing the Erasmus program. To succeed that, we had to use database tools which create tables and store our data with the appropriate commands. We first had created a database in the beginning of the project and we were planning to follow that one, but there were some themes that occurred afterwards so we changed our plans. The database tool we used was MySQL that runs as a server providing multi user access to a number of databases. There were many options about how to create the database, such as creating a separate table for every form, but that would extend too much the general database. So, we concluded in creating one table for all the forms.

As we mentioned before, the half of the forms that had to be created in this package must be programmed to pull the stored data from the database. That means in the HTML code, we had to write the appropriate code in order to pull the data and show it in he correct side. In the picture below, we present the Transcript of Records form which is a very good example of pulling data , but still there are some fields to be filled by the coordinator.

Illustration 11: Transcript of Records

We also decided that it would be better to have first a general table which will show all the matriculation numbers, names and surnames of the students which use the Erasmus program, so it would make things easier for the coordinator to check the certificates that must be sent and those which have already be sent. The table has also the choice to search every student by name or the matriculation number.

Illustration 12: Send Certificate

When the coordinator finds the correct student, there is a button next to the name which takes you to the certificates.

Illustration 13: Student Certificate

And when both are filled and sent, the certificate of stay is also automatically completed, as it pulls the data from the other two certifications.

Illustration 14: Certificate Information

Finally, for the Accommodation Registration form, we decided that it should be better to have the choice of wanting to book a room or not. In case someone wants to book, then after he/she picks that choice on the form, goes to the next page where other fields have to be filled. We must point here that this form also pulls data from the database.

Illustration 15: Accommodation Form

Illustration 16: Accommodation Availability

12.5 Exchanging Information

The last part of the package is the exchanging information part. The data that are filled in any institution which uses this website must be accessible from the other Universities. That was a mixture of the package P2 and the database we used.

Technically, the forms of this package are ready for use.

13 P6-EXAMS (Bulgaria)

13.1 Introduction

The EXAMS – Module is supposed to be used whenever Erasmus students have courses in arrears, for which there are no interchange options in the host institution, they can request to take the exam on that course at the host institution.

13.2 Technical Decisions

PHP

The application is created via PHP programming language. PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge. A competitor to Microsoft's Active Server Pages (ASP) server-side script engine. [22]

MySQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.[23]

Plonk library

For basic application workflow, the application uses the Plonk library - a subset of the Spoon library.[24] - that automates functionalities like page routing, session handling and page templating.
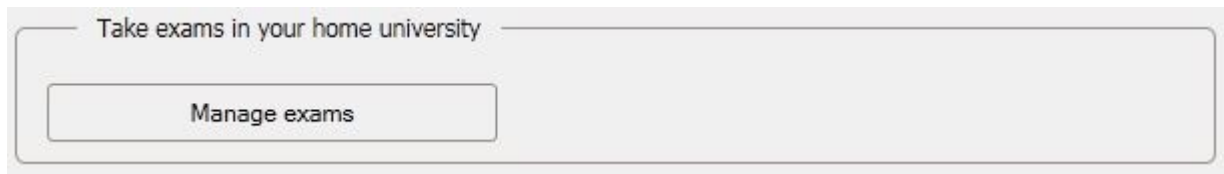
---

22 http://en.wikipedia.org/wiki/PHP
23   http://en.wikipedia.org/wiki/MySQL
24   http://www.spoon-library.com/

13.3 Interfaces

There are the interfaces list on the screens below:

Student interface:
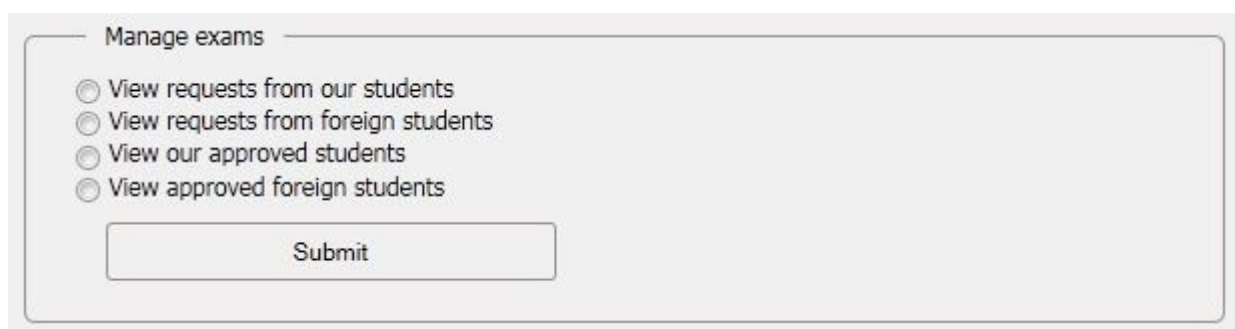
Illustration 17: Student Exams Interface

Coordinators view:

Illustration 18: Coordinator Exams Interface

Students have possibility to manage their exams. Here are the steps that one student must pass to declare that wants to take an exam:

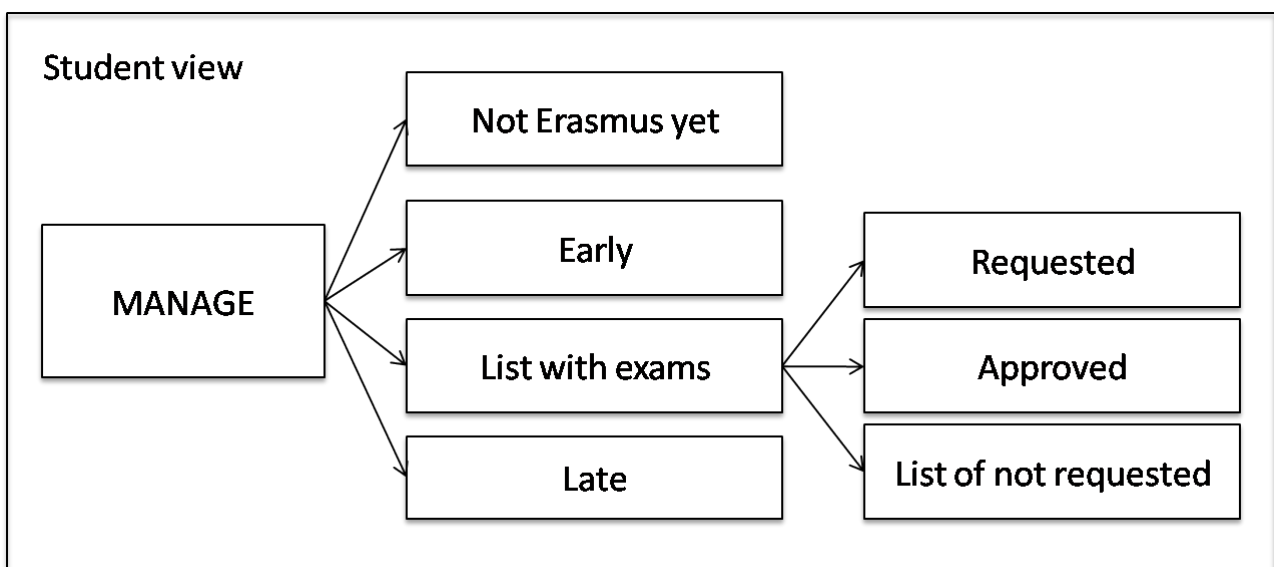*Illustration 19: Exams Validation Workflow*

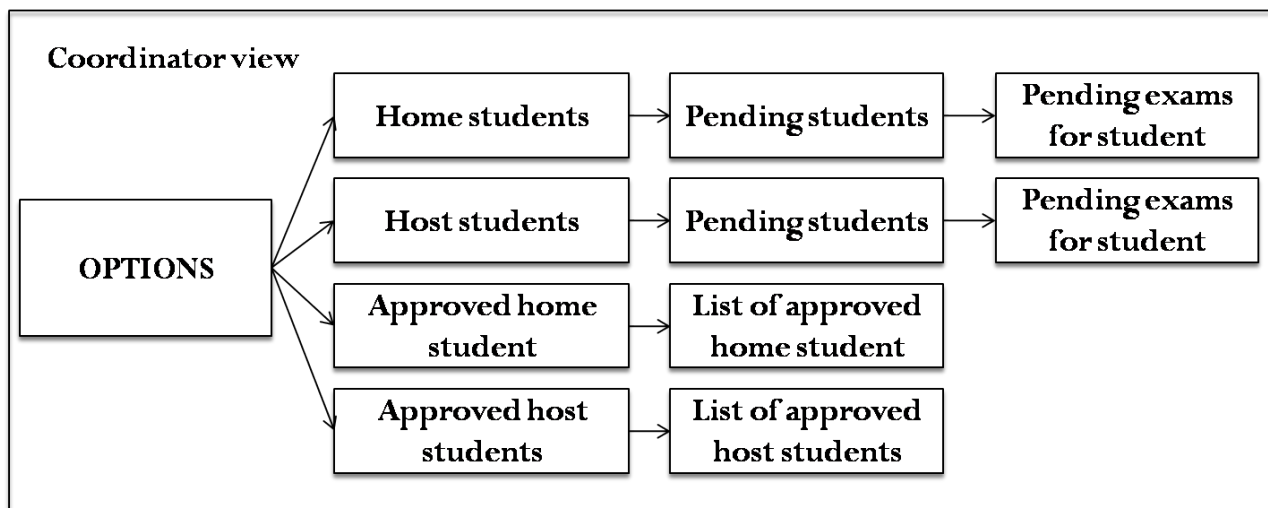And the steps for approving of the exam by the coordinator:

Illustration 20: Exams Validation Workflow by the Coordinator

14 P7-MATCH

The P7-MATCH module was developed but couldn't be integrated with the Erasmusline application given to technical issues.

Although for future development, the P7-MATCH module documentation is included in the Annexes.

15 P8-STATS (Portugal)

The next sections explain the objectives of this package, demonstrates the requirements gathering and Business Rules definition for the package as well as the more technical aspects of the package such as deployment, communications and decisions that shaped it's development.

15.1 Executive Information Systems

Executive Information Systems (EIS) are Information Systems that access large amounts of business information from various sources - internal and external to the organization - and provides decision support mediums to the organization's senior executives.

These mediums permit the executive managers to highlight patterns in the business process by analyzing, comparing and determining trends in the provided mediums - spreadsheets, graphic charts, pivot tables, reports, etc.

As an example of EIS use, consider the following scenario:

Erasmus administrators can do a weekly check on student enrollment information. Using a web browser, they can access that information in a personalized fashion by presenting subsets of the information - views. These views can be "drilled-down" to minute levels of information or "rolled-up" to display a broader view.

Since executive users are not necessarily data analysts, the EIS user interface must be as simple as possible, but without sacrificing presentation dynamics.

With these criteria in mind, the following key requirements for an EIS where identified:

- Cross Platform
- Ease of Use
- Limited Training
- Quick Response
- Process Large Volumes of Data
- Deployment Through the Web
- Easy Graphical Presentation Options
- Ability to Access Subsets of Data (Drill Down)

15.2 Architecture

An EIS can be divided into three main levels of architecture, that covers the technology used to extract, analyze and visualize information from other data sources with a friendly and flexible user interface. The levels discussed in the next sections are the following:

- Data Management
- Model Management
- Data Visualization

The architectural levels will be explained in the following sections.

15.3 Data Management Level

The Data Management Level deals with the extraction, analysis and processing of the internal and external data sources and passes that data through an ETL process - Extract, Transform and Load - that organizes and aggregates the data into the Data Warehouse. The use of ETL and Data Warehouses can be more efficient to the EIS due to it's data treatment and Warehouse data model schema.

15.3.1   ETL

The ETL process parses the data and performs these possible operations:

- Translation into database compatible rows / columns
- Translate code values
- Remove / Add / Translate code values
- Change data encoding
- Sorting
- Data validation
- Summarize data
- Join data from various sources
- Generating surrogate keys

Also, ETL process can make use of temporary database tables for intermediate processing, called Operational Data Stores (ODS). These tables hold historical data that can me mined for statistical information to be included in the Data Warehouse.

For the P8STATS package, the ETL process will be executed either by an event trigger - saving data in the ODS's when a certain workflow phase is reached - and periodically - part of the data refreshment process to aggregate new information to the Data Warehouse.

15.3.2　Data Refreshment

This phase of the ETL process is comprised of design choices, like data structures, techniques to update and optimize the flow of information from the data sources to the Data Warehouse and update cycle.

Based on the statistical overview document of the Erasmus Program of 2010 – in the Annexes – it was verified that with the large volume of students mobility, having a completely centralized database wouldn't be viable in terms of database synchronization and efficiency - 198523 students got mobilized by 2747 Institutions in the academic year 2008/2009.

The process begins with the loading of information from the Operational Data Stores.

Loading

The information sent to the ETL component will be fetched from the application data sources while other subsets of information are already stored in the ODS's or Metadata tables - these hold information previously gathered by the business requisites - which is going to be processed by the ETL component and integrated into the Data Warehouse.

To aggregate and integrate data to be viewed in the EIS, we must determine an update cycle for the Data Warehouse. The information models that need to be gathered over time are noted in the following table:

Model	Update cycle
Erasmus Process Efficiency	Every Semester
Erasmus Process Efficacy	Every Semester (after Efficiency process)

Table 3: EIS update cycle for the Data Refreshment phase

After the Process Efficiency data is integrated into the Institution's local Data Mart - a local database, part of the Data Warehouse - the Efficacy process gathers the data from the various Institution's Data Marts and additional data to aggregate and integrate new data into the central Data Mart.

More on this discussion in the Deployment sections of this report.

Aggregation

The aggregation process of Student Application data fetches the information from the

ODS database and integrates those values in the Data Warehouse database.

Some technical aspects have to be considered in the loading and aggregation of data sources:
- Keep an historical record of information already loaded, so as not to fetch the same data twice. This is a achieved with a caching mechanism, to prevent repetitive DB queries for example.
- The Data Warehouse Database Schema must be optimized for query performance, that includes
  - Which indices are needed
  - Index type vs Data type
  - Database/Table engine

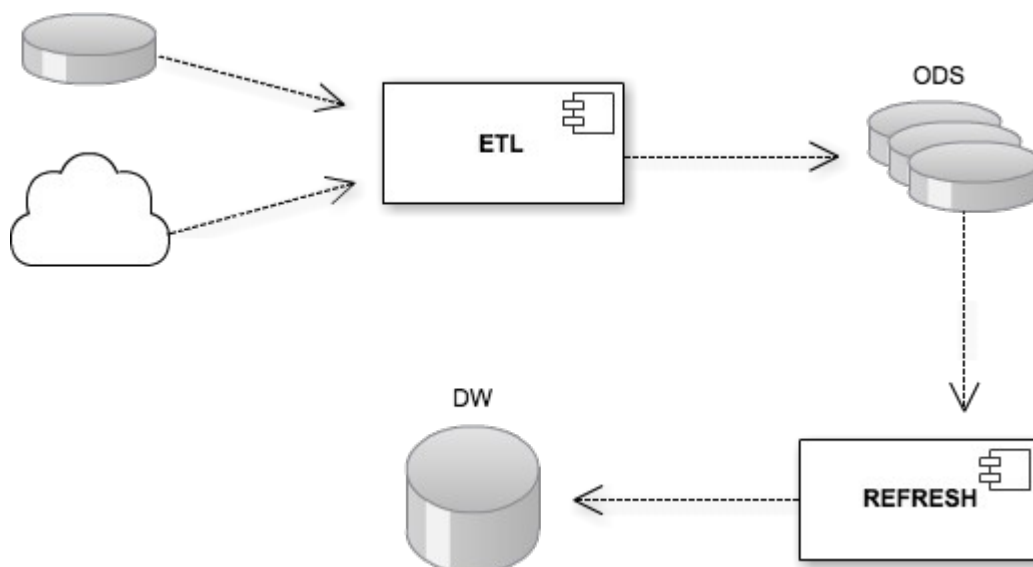The technical considerations and decision are explained in detail the next sections.

Illustration 21: ETL Process Workflow

Illustration 22: ODS and Metadata DB Schema

15.3.3   Data Warehousing

A Data Warehouse holds the processed information into a database designed properly for the retrieval of large amounts of data. The database structure is composed of fact tables and dimension tables, connected by their keys in a Star or Snowflake schema.

A fact table holds facts - an aggregation of contextualized fields and numerical values for measurement.

A Dimension table holds the various combinations of a given dimension, often identified by a part of the composite key in the fact table and serves as a means to restrict and summarize the information contained in the fact table ("roll-up" and "drill-down" operations).

The following analysis breakdown describes which data can be mined given the project Specifications and the Erasmusline business information, and what Key Performance Indicators can be obtained, measured and stored in the Data Warehouse.

Data Warehouse Statistical Specifications

Objectives:

- Efficacy
- Efficiency

Key Performance Indicators (KPI):

- Efficiency:
    - Response time between Process phases
    - Student participation
    - Lodging availability

- Efficacy:
    - Applications
    - Student credits
    - Student grades

Measure Table

Ref / Measure	Description	Example
M1: Real Value	The current value	# Applications current month; # Declined applications current month
M2: Accumulated	Accumulated value	# Applications in a year; # Applications in a semester
M3: Homologous	Current value comparison with previous value	# Applicants in the previous year
M4: Relative	Real value compared to accumulated value %	Percentage of applicants in January
M5: Average	Average of the accumulated value	Average of applicants in January
M6: Maximum	Maximum pick of the accumulated value	Maximum # of applicants in January
M7: Minimum	Minimum pick of the accumulated value	Minimum # of applicants in January
M8: Relative HEI	Relative percentage against the total number of students in the HEI	Percentage of applicants against the total number of students

Table 4: Defined Measures for the Data Warehouse

KPI vs Measure Table

Ref / Indicator	Ref / Measure
I1: Response time	M5 / M6 / M7
I2: Participation	M2 / M4 / M8
I3: Lodging Availability	M2 / M4
I4: Applications	M1 / M2 / M3 / M4 / ...
I5: Student credits	M5 / M6 / M7 / M3,5
I6: Student grades	M5 / M6 / M7 / M3,5

Table 5: Cross referencing KPI's with Measures

Dimensions Table

Ref / Dimension	Ref / Hierarchy	Ref / Example
D1: Gender	H1: Gender	E1: Male | Female
D2: Lodging	H2: Residence Type	E2: Campus
D3: Mobility	H3: Mobility Type	E3: Internship | Exchange | Both
D4: Academic Date	H4: Year / Semester	E4: 2011; 2nd
D5: Institution	H5: Country / Institution	E5: Portugal; ISEP
D6: Study	H6: Area / Degree / Course	E6: Engineering; Informatics; Software Engineering
D7: Phase	H7: Process Phase	E7: Pre-Candidacy

Table 6: The Dimension Classification, their levels of detail and examples

KPI vs Dimensions Table

Ref / Indicator	Ref / Dimension
I1: Response time	D3 / D4 / D5 / D6 / D7
I2: Participation	D1 / D2 / D3 / D4 / D5 / D6
I3: Lodging Availability	D1 / D2 / D4 / D5
I4: Applications	D1 / D2 / D3 / D4 / D5 / D6
I5: Student credits	D1 / D3 / D4 / D5 / D6
I6: Student grades	D1 / D3 / D4 / D5 / D6

Table 7: Cross reference between KPI's and Dimensions
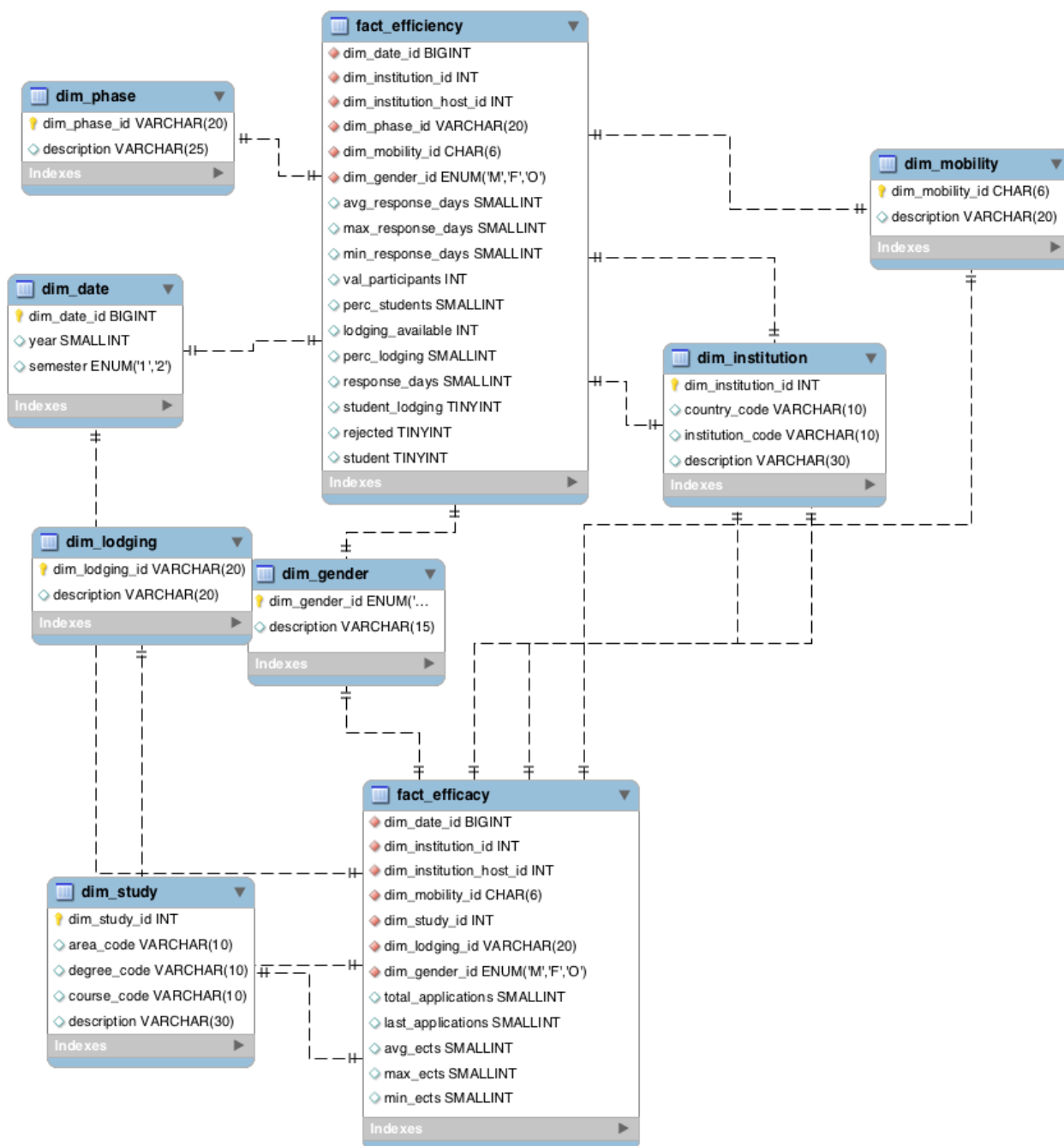
15.3.4   Database Schema

Illustration 23: Data Warehouse Database Schema

15.4 Model Management

This level manages and transforms data according to business guidelines and stored metadata by executing several operations explained next.

15.4.1 OLAP

Online Analytical Processing is a set of techniques applied to the Data Warehouse to summarize and visualize business information in a multidimensional perspective across multiple dimensions. Since the Data Warehouse is designed to use a multidimensional data model, information is organized in Data Cubes and thus enabling the EIS users to gain more insight into the data by performing these operations:

- Facts aggregation
- Slice and Dice across multiple Dimensions
- Drill-Down and Roll-Up the data from one hierarchy to another

Here are some examples of OLAP in action in an Accounting context:

- Comparison of sales (fact) of a product (dimension) over years (dimension) in the same region (dimension).
- How may members (fact) have opened a savings account (dimension), in USA branch (dimension), over a period (dimension)?
- How many mortgage loans (fact) have been approved in fixed mortgage (dimension) or Adjustable Rate Mortgage (dimension) in New York City (dimension), over a period (dimension)?
- What is the total sales value (fact) of a particular product (dimension) in a particular grocery store (dimension), over a period (dimension)?
- What is the amount spent (fact) for a particular product promotion (dimension) in a particular branch (dimension) or in a particular city (dimension), over a period (dimension)?

15.4.2 Data Mining

Data mining tools are especially appropriate for large and complex sets of data. Through statistical or modeling techniques, data mining tools make it possible to discover hidden trends or rules that are implicit in a large database. Data mining tools can be applied to data from data warehouses or relational databases. Data discovered by these tools must be validated and verified to become operational data that can be used in the decision process.

15.5 Data Visualization

15.5.1   The EIS Web Interface

The interface should be flexible and intuitive enough for users without previous skills to select the indicators they want and operate filters by several dimensions and metrics - this operation can be called a "scenario". Theses "scenarios" can be saved and edited for later used by the user.

Illustration 24: EIS Interface Mockup

15.5.2   Interface Features

As mentioned earlier, each scenario configuration can include the following features:

- Select OLAP cubes - the Objectives - for performance monitoring
- Generate pivot tables
    - Selecting Dimensions for columns and rows
    - Selecting measures for columns and rows
    - "**Drill-Down**" data by displaying aggregated information for a member of a Dimension
    - "**Roll-Up**" data by displaying aggregated information from 'All' members in a Dimension
    - Swap Columns / Rows
    - Filter Dimensions by Value Type
    - Highlight Dimensions with a different color, depending on Column / Row location
    - Settings Management
        - Create a new scenario
        - Save a scenario
        - Load a previous scenario
    - Export results
        - Spreadsheet
        - Text file (CSV format)

The after-development version of the interface can be seen in the next illustration.

For instructions on how to use the Interface, please read the EIS Installation and User Manual, enclosed in the Annexes of this report.

Illustration 25: The EIS interface after development

15.6 Development Process

The P8-STATS team began by analyzing the Objectives and Key Performance Indicators based on the business rules imposed by the Erasmusline application workflow and the working version of the application's database. This implied studying the basic functionality of a Data Warehouse and it's implementation in the MySQL RDBMS.

The team began to prototype a database schema in a star fashion – as proposed in the analyzed Data Warehousing guidelines – and make a few test queries on high volumes of fact records with dummy data. Analyzing the limitations of MySQL and it's table engines, BTREE type indices was the only type available when using MyISAM table engine.

Another specification the team assumed is that the user will normally consult sub-sets of data, usually by academic date. We analyzed the advantages of implementing the fact tables using the Merge Table Engine available in MySQL and concluded that partitioning the data by year or semester would provide better performance not only in queries – the SQL optimizer would only fetch a certain partition, given that the user will usually analyze recent data - but also in the ETL and Data refreshment process, allowing to avoid locking and rebuilding a fact table's indices while aggregating new data. Also in the consulted literature, implementing index cache size alterations and creating a separate index cache pool for the Dimension tables would provide additional performance. By loading the indices beforehand to hot cache these would stay in resident in memory to provide a path to the corresponding Merge Table partitions.

Given that the Efficacy data is to accessed by students in every institution and assuming that the data itself is an aggregation of all institution's data, the team decided to house Efficacy data in a central server, and leave the Efficiency data local. That way each institution accesses their own Efficiency data.

To manage these tables and conduct the activities of OLAP processing, ETL and Data Refreshment and managing a central and local Data Warehouse, the team decided that a background process would be conveniently needed. The team decided to implement Unix Daemons, being a common implementation in Open Source Operating Systems based on Unix, like Linux and FreeBSD. Theses Daemons act as Servers and gateways to access the Data Warehouse. The implementation consists of an OLAP module, an ETL module, a JSON-RPC communication module, a caching module, a performance profiler with logging and a Scheduling module, to trigger events such as Data Refreshment.

While analyzing the EIS, the team desired to create an interface that would implement client-side processing to reduce load on the servers and only rely on the Daemons for result delivery. Client-side processing was achieved using the jQuery Javascript library by implementing client-side templating, via the jQuery Tmpl plug-in, allowing us to generate HTML content from the Browser. Communication would be made by JSON-RPC strings and background HTTP requests like Ajax, with the specified Scenario management and query processing being made by the local Daemon. Communication with the central Daemon and it's database would be relayed by the local Daemon, directly delivering the result of the central database.

Being concerned with the accessibility guidelines, validation tools would be used to enforce such guidelines. For example, the use of ajax is accepted but the user must be notified at all times of the shift of attention in the interface.

15.7 Technical Decisions

15.7.1   MySQL 5.1

MySQL is a RDBMS server that provides storage of information with relational traits, that supports multiple databases and database engines, multiple users and threaded access. It is an open source solution and currently the most popular relational database for small and medium sized projects. For the P8STATS package, we will use the stable release MySQL 5.1 because it is a generally available release and can be installed and found in any mainstream Operating System today.

MyISAM / MERGE Engine

The P8STATS package requires fast access and processing times with low overhead, in order to serve the client as fast as possible given that the amount of data in the Data Warehouse will grow exponentially every semester (in a good scenario). The Data Refreshment phase needs to be executed quickly and in the background so as to allow users to continue using the EIS without any downtime.

For the Dimension tables we use the MyISAM database engine, since up to the 5.1 release offers a fair relation between data consistency and throughput, although consistency could require some configuration on the underlying OS. The kind of data we are housing is not extremely critical that needs to be kept history logs or support for transactions so MyISAM satisfies our objectives.

The Fact tables use the MERGE_MyISAM database engine, allowing us in the Data Refreshment phase to partition our data by refreshment period (Year or Semester) and keep it in separate tables. The main MERGE table houses the union of all the partition tables and their indices thus achieving better performance in some scenarios, for example, querying data from a single semester, which will be a common task in the EIS. Table partitioning also allows the databases' physical files to be moved to separate disks, allowing better performance by, for example, having the first semester table file in a SSD disk with a btrfs file system partition.

The tables' indices can also be preloaded to cache on demand, after the Refreshment process is done, to avoid loading indices on database query time.

15.7.2   PHP 5.3

PHP is a popular programming language for the rapid development of Web Sites. It's fundamentally a dynamically typed language with Object Oriented Programming support and very tightly coupled with server-side scripting of applications.

It was decided to implement the P8STATS package in this programming language to allow academics of any technical level to easily comprehend and extend the application.

PHP permits developing open source applications - one of the project's requirements. Based on that premise, some components where designed and optimised for POSIX compatible operating systems like GNU/Linux, which are fundamentally open source

systems. These components are System Daemons that are active agents in the OLAP and ETL processing.

15.7.3 Daemons

Unix Daemons are processes that run the background, performing all sorts of tasks and intercommunicating with other applications if needed.

The Daemons designed for the P8STATS package generally do the following tasks:

- Periodically executes the ETL process, according to determined refreshment cycles.
- Communicate with the ErasmusLine application - for example saving ODS information
- Save user's EIS scenarios
- Load EIS scenarios and perform OLAP operations on the Data Warehouse
- Cache EIS query results, product of OLAP operations
- Request data from other daemons - for example retrieve Efficacy data from the central Data Mart

Every Erasmusline installation has a Slave Daemon to perform these tasks. A Master Daemon is deployed with the central Data Mart - that could be on the Slave Daemon's machine - and can be reached by configuring a simple settings file in JSON format and setting the Master's IP address. All daemons support BSD-style and TCP-IP socket communication, supported by the PHP socket API. BSD-style sockets allow less overhead in local inter-process communications since it doesn't rely on the TCP-IP layers stack. Remote communications rely only on TCP-IP sockets.

15.7.4 POSIX Compatible Operating Systems

Operating Systems like GNU/Linux, MacOSX, Solaris or BSD follow standards that emulate the functionality of a Unix machine, meaning the OSs' internal API and system utilities follow the same convention that allowed the P8-STATS package to run in a much wider array of systems.

Given the Open Source orientation of this project, this package was optimized to run in Linux and BSD systems like Ubuntu or FreeBSD, given that it enables us to use a more straightforward API for special tasks like concurrent programming and socket communication.

Modern POSIX compliant operating systems like Ubuntu enable a quicker installation and setup of the software needed by the Erasmusline application and subsequently this package, by using pre-configured terminal task commands to group-install a basic LAMP software stack – **L**inux, **A**pache, **M**ySQL and **P**HP – and quickly deploy the application by pulling the latest Erasmusline application release from the SourceForge Git repository.

15.8 Deployment Scenario

The architectural deployment of the EIS application can be explained by the following figure:
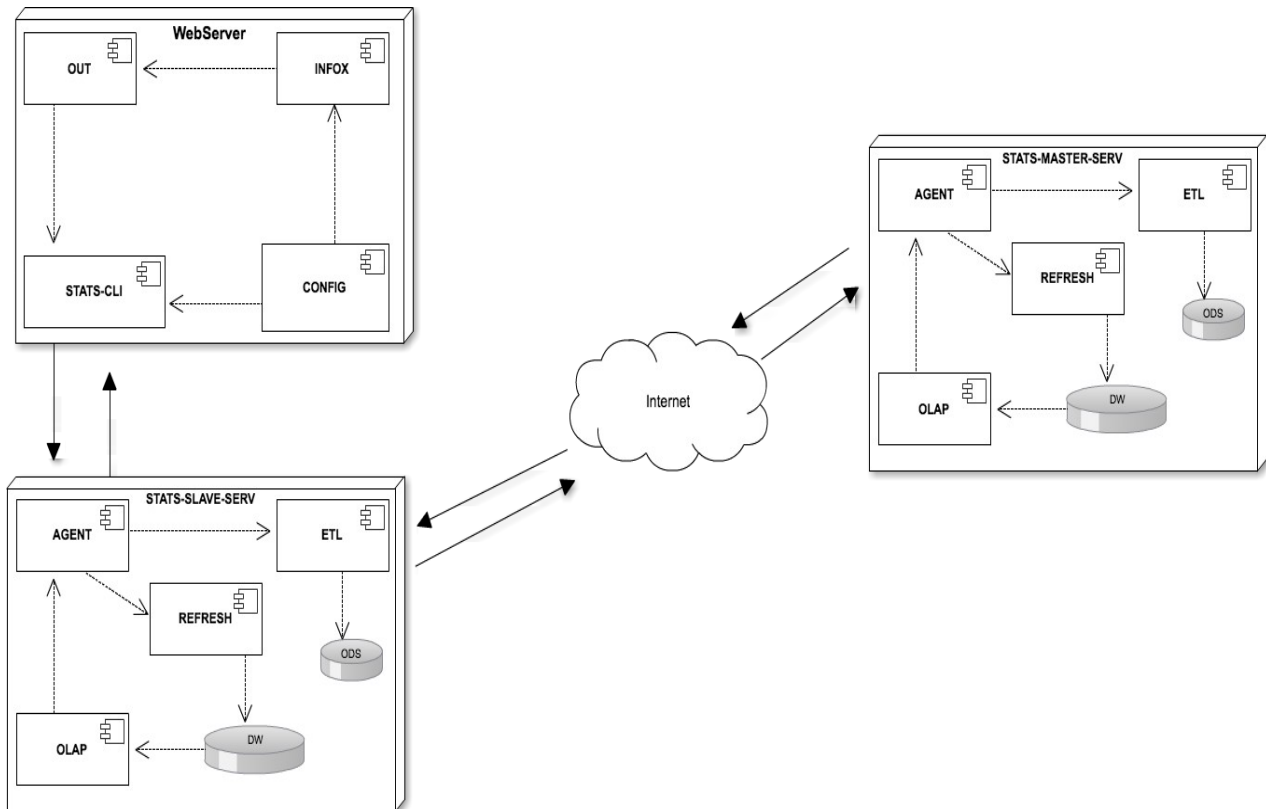

Illustration 26: Data Flow between Erasmusline, the slave and master STATS daemon

The deployment components can be separated in three major entities:

- A Master Server
    - Contains the Data Warehouse with the data referent to the Erasmus Process Efficacy (explained in the following sections)
    - Gathers information from the Slave Servers to summarize Efficacy indicators data
    - Communicates with the Slave Servers to deliver summarized Efficacy data
- A Slave Server for each Institution
    - Communicates with the other Erasmusline packages in order to gather Efficiency indicators data - mainly the P4OUT and P1CONFIG packages
    - Communicates with the Master Server to provide the EIS interface with Efficacy data.
    - Delivers the EIS interface the Efficiency data

- EIS Interface and Connector
    - User interface to manage scenarios and display statistical information
    - Connector component that receives event data referent to the Erasmus workflow process - sending that data to the Slave Server

15.8.1 Server Communication

The Slave and Master EIS servers accept text based communications by following the guidelines of the JSON-RPC 2.0 Specifications[25]. This Remote Procedure Call specification is very light weight since it doesn't rely on the verbosity of SOAP RPC[26] messages and allows API extension since it is not a strict API.

Here is an example communication scenario.

The EIS interface requests that a certain scenario may be executed, a JSON message structure is sent to the Slave Server:

```
{"jsonrpc":"2.0","id":1,
"method":"runScenario",
"params":{"cube":"fact_efficacy",
    "columns":["dim_gender.dim_gender_id","dim_institution.institution_code"],
    "rows":["dim_mobility.dim_mobility_id","measure.M1"],"filters":{}}}
```

The Slave Server realizes the analysis is to be made on Efficacy Performance Indicators so it relays this message to the Master Server who holds the Efficacy Data Mart.

The result of the scenario execution will be delivered to the EIS interface in this format:

```
{"jsonrpc":"2.0","result":[{"Mobility Type":"both","M | fkl | Applications":1130,"M | gent | Applications":1280,"M | gun | Applications":1075,"M | isep | Applications":943,"F | fkl | Applications":1006,"F | gent | Applications":964,"F | gun | Applications":1092,"F | isep | Applications":1020},
{"Mobility Type":"intern","M | fkl | Applications":1262,"M | gent | Applications":1030,"M | gun | Applications":1356,"M | isep | Applications":967,"F | fkl | Applications":884,"F | gent | Applications":1170,"F | gun | Applications":1035,"F | isep | Applications":997},
{"Mobility Type":"study","M | fkl | Applications":976,"M | gent | Applications":966,"M | gun | Applications":1101,"M | isep | Applications":1144,"F | fkl | Applications":784,"F | gent | Applications":1087,"F | gun | Applications":851,"F | isep | Applications":1121}],"id":1}
```

If any exception or error occurred we would receive a different response format:

```
{"jsonrpc": "2.0", "error": {"code": 101, "message": "INVALID_QUERY"}, "id": 1}
```

---

25 http://groups.google.com/group/json-rpc/web/json-rpc-2-0
26 http://en.wikipedia.org/wiki/SOAP

Other Key Performance Indicators – Efficiency – are locally processed by the server's OLAP module.

15.8.2　ETL / Data Refreshment Communication

Each server has a scheduling module that triggers server tasks to be executed from time to time or at a specific date.

At the end of each semester the Slave Servers trigger the data refreshment task which in turn runs in the background and gathers new data from the ODS tables to be integrated into the Data Warehouse and it's corresponding Data Marts.

The Master Server however needs to communicate with every running Slave Server to check for new aggregated data to be integrated to it's Data Mart. The Slave Servers periodically check for the Master Server's location and gives it their location. The Master Server in turn saves their locations in the database. When the time comes to refresh the Master Server's Data Mart, it calls every saved Slave Server and asks them for the aggregated data with extra information relevant to other Key Performance Indicators. After saving that data in the local ODS tables, the Master Server triggers it's own data refreshment task and integrates the new data into the Data Warehouse.

15.9 Performance Analysis

Given the European Commission's Statistical Overview, to achieve a million student records in a single database it would be necessary six semester's worth of student applications from every Institution with Erasmus Protocol – more or less 196.000 records each semester.

As a performance test, we altered our database population script to insert 960.000 Efficacy records spread through three years, including every semester.

Since a single Efficacy record is an aggregation of several application's – from five to fifty random test values - by a set of dimensions, this example will serve as a worst case scenario were you have roughly 225.000 applications per semester.

Given the logic separation between merging tables, each table – one for each year/semester – was populated in parallel and joined into the main Efficacy Merge table.

Bellow is list describing the performance measures:

PC Specifications: Intel i5 Processor 2core / 4threads ; 4GB Ram; GNU/Linux

- 960.000 records / 6 tables
- **Elapsed time: 12.34min**
- **Used memory: 694.90kB**

Next, we performed a regular query generated by our OLAP module, without any cache warmup to test the elapsed time of execution:

```
SELECT  dim_institution.institution_code as `Institution`,

        dim_institution_host.institution_code as `Host Institution`,

        dim_date.year as `Year`,dim_date.semester as `Semester`,sum((fact_efficacy.total_applications)) as
`Applications`
  FROM fact_efficacy,

                dim_institution as dim_institution,

                dim_institution as dim_institution_host,

                dim_date as dim_date
  WHERE (fact_efficacy.dim_institution_id = dim_institution.dim_institution_id)

        AND (fact_efficacy.dim_institution_host_id = dim_institution_host.dim_institution_id)

        AND (fact_efficacy.dim_date_id = dim_date.dim_date_id)

        AND dim_date.year=2010 AND dim_date.semester=2
  GROUP BY dim_institution.institution_code,dim_institution_host.institution_code,dim_date.year,dim_date.semester
```

Time elapsed: 1.67sec

For comparison purposes we created a similar Efficacy fact table, but without underlying merge tables and performed the same query, under the same circumstances:

```
SELECT  dim_institution.institution_code as `Institution`,

        dim_institution_host.institution_code as `Host Institution`,

        dim_date.year as `Year`,dim_date.semester as `Semester`,sum((fact_efficacy.total_applications)) as
`Applications`
  FROM fact_efficacy_solid as fact_efficacy,

                dim_institution as dim_institution,

                dim_institution as dim_institution_host,

                dim_date as dim_date
  WHERE (fact_efficacy.dim_institution_id = dim_institution.dim_institution_id)

        AND (fact_efficacy.dim_institution_host_id = dim_institution_host.dim_institution_id)

        AND (fact_efficacy.dim_date_id = dim_date.dim_date_id)

        AND dim_date.year=2010 AND dim_date.semester=2
  GROUP BY dim_institution.institution_code,dim_institution_host.institution_code,dim_date.year,dim_date.semester
```

Time Elapsed: 2.21sec

Performing these and similar queries on the database we could determine that there was a performance improvement of 25%~30%.

16 Conclusion

16.1 Project Outcome

After three months of work the Orange Team crossed the finish line of the 2011 MUTW Project. In the last 103 days we achieved several goals of our project and had a productive time working together in a multinational european team.

The aim of our work was to develop a web-application which covers all necessary needs for applying for the Erasmus Exchange Process. In more than 30 meetings we discussed the building of the application, fixed problems and coordinated our further work. Beside our weekly monday meetings we arranged several special meetings to discuss and fix problems between the affected teams.

After this hard work we are convinced that the orange team-developed ErasmusLine application will fulfill the expectations and is a strong symbol for multinational european teamwork.

16.2 Accomplished Works

P1 – Config (Germany)	• Database design • Home page and main menu • Website design • Data management • User management • Access permissions
P2 – Infox (Germany)	• Information exchange with JSON and cURL • File exchange with cURL • Encryption
P3 – Alert (Bulgaria)	• Sending emails according to death lines by a daemon
P4 – Out (Belgium)	• Outgoing workflow design • Form design • Pre-candidate phase • Setup phase • Prepare stay phase • Stay phase • Tear down phase
P5 – In (Greece)	• Form Design • Functionality in PHP • Validation in Javascript, PHP, jQuery • Store – pull data from database using MySQL • Email forms to coordinators/institutions
P6 – Exam (Bulgaria)	• Students request to take exams at host university • Home/host coordinator accepts students request
P7 – Match (Iceland)	• Searching the website of an institution for course information • matching corresponding courses • listing all matched courses
P8 – Stats (Portugal)	Portugal • Data Warehouse database design and integration • ETL and Data Refreshment design and implementation • EIS web-interface design and integration

16.3 Other Accomplishments

Given the project schedule and deadlines, the several package teams decided to help other teams with development and documentation. These phases of inter-package help resulted in the following functionalities:

Developed by the **P8-STATS** team:

- Institutions Partnership

- Institution Management

- Education / Courses Management

- Residence / Owner Management

- Synchronization of Institution, Educations, Courses, Residence and Owners data between Application instances

- P2-INFOX draft message protocol and message encryption code

Developed by the **P8-STATS** and **P4-OUT** teams:

- P2-INFOX integration for various modules

- P5-IN package integration and workflow testing

- Overall testing and validation

16.4 Future Work

a) **Accessibility standards**
   To comply to European Union standards the website has to match the W3C Web Accessibility Initiative (WAI) standards, mainly the second version of the Web Content Accessibility Guidelines (WCAG v2). To achieve this goal the website has to be improved to match all requirements that handicapped persons have unlimited access to ErasmusLine.

b) **Internship process**
   It turned out that the internship process is a complicated process. Because of the higher priority of the university exchange process it couldn't be finished in time.

c) **Integration of the Official Erasmus Contract**
   The Official Erasmus Contract is a document which is specific for each institution. There has to be a standardization process to include a form into the application.

d) **Data mining integration**
   Better integration with P1-Config and P4-Out to improve the workflow of information to the ODS.

e) **General bug fixing**
   Software is never free of bugs. General bug fixing has to be done in future work.

16.5 Final Thoughts

"It was an unique experience, for which I am grateful to be a part of, with some great benefits for my future career."

Pedro Ferreira – Instituto Politécnico do Porto, Portugal

"An extraordinary trip where we faced a great deal of tough decisions, but we emerged more grown up, more cooperative and we take home the fond memories of this amazing experience."

Daniel Lopes – Instituto Politécnico do Porto, Portugal

"Legen...Wait for it...Milk!"

Stéphane Polet – Katholieke Hogeschool Sint-Lieven, Belgium

"A great project, which improved not only my technical skills, but also gave me a lot of cultural and social experience."

Nathan van Assche – Katholieke Hogeschool Sint-Lieven, Belgium

"*It was a great experience to meet and work with people from other countries, I improved my communication skills and my team spirit.*"

Aggeliki Katsiampouri – Technical Univercity of Crete, Greece

"Great experience, I was lucky for join this project I hope to have a similar experience in the future."

Stefanos Moundrakis – Technical Univercity of Crete, Greece

"I am happy that i was part of this wonderful project, I met great professionals and smiling faces, improved my programming skills."

Ina Ivanova – University of Veliko Tarnovo, Bulgaria

"I am glad to participate in this project and I gained a lot of experience which I am sure will be useful in the future."

Zvezdomir Tsvyatkov – University of Veliko Tarnovo, Bulgaria

"Being a part of this international team I really learned a lot and I think it will be a big advantage for my future career."

Arne Lipfert – Fachhochschule Kiel – University of Applied Sciences, Germany

"It has been a great experience to work with other students from all over europe to improve my technical and social skills."

Arne Reimer – Fachhochschule Kiel – University of Applied Sciences, Germany

17 References

Books / Academic Articles

The Data Warehouse Toolkit : the complete guide to dimensional modeling / Ralph Kimball, Margy Ross. - 2nd ed.

Executive Information Systems: Development Lifecycle and Building by using the Business Inteligence Tools / Lungu Ion, Vatuiu Teodora.

Building Information out of Data: Executive Information System at Penn State University / The Pennsylvania State University Executive Information System Coordinating Committee

Building and Optimizing Data Warehouse "Star Schemas" with MySQL / Bert Scalzo

Practices of an Agile Programmer / Venkat Subramaniam, Andy Hunt

Leading with Emotional Intelligence / Reldan S. Nadler

Other Articles

The Erasmus Programme December 2010 – A Statistical Overview / European Commission, Education and Culture DG

Web Sites

Saiku – Next Generation Open Source Analytics / http://www.analytical-labs.com

Data Warehouse – GeekInterview / http://www.learn.geekinterview.com/data-warehouse/

LearnDataModeling.com / http://www.learndatamodeling.com/

Open Source Business Intelligence / http://www.squidoo.com/osbi

Google Chart Tools / http://code.google.com/apis/chart/

jQuery Form Validator / http://www.position-absolute.com/articles/jquery-form-validator-because-form-validation-is-a-mess/

jQuery UI Library / https://github.com/jquery/jquery-ui

jQuery Progress Bar / http://plugins.jquery.com/project/jQueryProgressBar

jQuery Date Picker / http://www.eyecon.ro/datepicker/

jQuery Masked Input / http://plugins.jquery.com/project/maskedinput

jQuery Password Plugin / http://plugins.jquery.com/project/password_strength

PHP Mailer / http://phpmailer.worxware.com/

PHP Validation / http://www.benjaminkeen.com/software/php_validation/

PHP mcrypt module / http://php.net/manual/en/book.mcrypt.php

Kevin van Zonneveld - Create daemons in PHP /
http://kevin.vanzonneveld.net/techblog/article/create_daemons_in_php/

Gonzalo Ayuso - Pivot tables in PHP / http://gonzalo123.wordpress.com/2010/01/24/pivot-tables-in-php/

jQuery Template Plug-in / http://api.jquery.com/jquery.tmpl/

InfoQ - The State of Accessibility with Ajax / http://www.infoq.com/news/ajax-accessibility

Web Content Accessibility Guidelines (WCAG) 2.0 / http://www.w3.org/TR/WCAG20/

18 Glossary

MySQL – An open source database system. It is very safe and is modified continuously.

UTF8 – A multibyte character encoding for unicode. It has a wide range of characters in which all european characters are included.

Plonk – An open source PHP Framework developed by Bramus Vandamme.

MVC (Model View Controller) – A software architecture to separate the logical code from the layout (view).

INFOX – A short version of **INFO**rmation e**X**change

cURL – A short version for "**C**lient for **URL**". Developed 1997 by Daniel Stenberg and open source under the MIT – Licence

JSON (JavaScript Object Notation) – A text based standard design for data interchange

3DES (Triple Data Encryption Standard) – An encryption algorithm which is very

powerful and very hard to hack.

HTTPS – A secure way to perform a HTTP connection

IPv6 – The next generation of IP addresses. This offers more addresses and will be standard in the near future.

WWW – A short version of World Wide Web, the Internet

ODS - An operational data store is a database designed to integrate data from multiple sources for additional operations on the data.

DW - A data warehouse is a database used for statistical and reporting purposes.

OLAP (Online Analytical Processing) - Interactive analysis of data that has been transformed from raw (operational) data into understandable enterprise-wide data.

Objective - Tangent goals the EIS is trying to demonstrate.

Key Performance Indicator - Statistical value that indicates progress or predicts future progress of a business process.

Dimension - A dimension is a structural attribute acting as an index for identifying measures within a multidimensional data model. A dimension is basically a domain, which may be possibly partitioned into an hierarchy of levels. For example, in the context of selling goods, possible dimensions are product, time, and geography; chosen dimension levels may be Product category, Month, and District.

Measure - A measure is a point into the multidimensional space. A measure is identified if for each dimension a single value is selected. For example, a "sales volume" measure is identified by giving a specific product, a specific sale time, and a specific location.

Drill-Down - The navigation among levels of data raging from higher level summary (up) to lower level summary or detailed data (down). The drilling paths may be defined by the hierarchies within dimensions or other relationships that may be dynamic within or between dimensions. An example query is: for a particular product category, find detailed sales data for each office by date.

Roll-Up - The querying for summarized data. Aggregation involves computing the data relationships (according to attribute hierarchy within dimensions or to cross-dimensional formulas) for one or more dimensions. For example, sales offices can be rolled-up to districts and districts rolled-up to regions; the user may be interested in total sales or percent-to-total.

Slice and Dice - The process employed by users to explore and query multidimensional information within a hypercube interactively.

ETL - Techniques used to integrate data from heterogeneous data source into a new data store.

Data Refreshment Plan - Rules and guidelines to integrate information into the Data Warehouse.

SSD - Solid State Drives are disk drives that can be accessed like a conventional Hard Drive but instead of electro-magnetized metal plates, SSD's use microchips to store information, like a conventional USB disk pen.

btrfs - B-Tree File System is a GPL licensed file system for GNU/Linux Operating Systems. It supports crash recovery, snapshots, transactions, defragmentation, etc.

19 Annexes

20 Annex 1: Early Database Deployment Draft

Written by P8-STATS team.

Decentralized Idea:

-The application is self autonomous and each institution is responsible to install and host it.

(Example: brain cells implementation?)

What this implies?

-Every Institution will have their own webpage (that at the beginning it's the one designed by us but we could give the incentive to each one make their own interface) and Database.

This way the login problems of the accounts not being centralized are nullified.

How it will work:
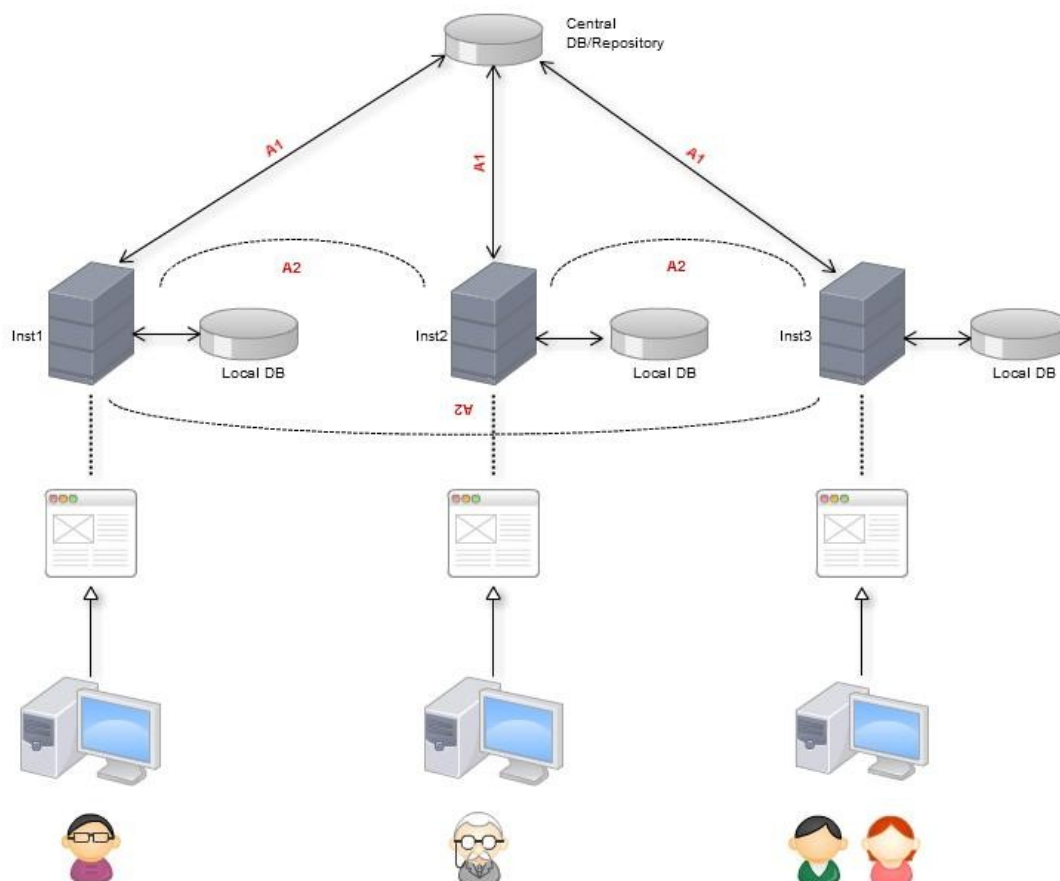
A - Given this implementation, information between Institutions has to be shared and we have to decide how we will accomplish this.

A1 -

    a) We can use a repository that controls and keeps track of all Institutions and the new ones, therefore the application when installed accesses this repository to 'register itself' and get the list of all the running ones, after that, an alert is generated to all the actives ones so they can update themselves with the "new born".

    b) Taking into consideration that Institution will need to become partners first before exchanging students, if there's no other reason for a central repository we can avoid it and say that they have to agree on partnership first by their own communications and if accepted they add each other on the application.

A2 - Database mirroring is completely out of question given the huge amount of Databases that can exist so the information will be separated, therefore we will have o create some kind of communication protocol between Institutions for when they need to get info that's stored on the other side and so on.

Pros:

    -Account management and 'power attribution' simplified since every Institution is responsible for them self

    -Law problems with info also probably solved

    -Pre candidate forms being different can also be solved by this (everyone is responsible for their own)

Cons:

    -Information about the host coordinator, the student that comes from far away, stuff like that, needs to be accessed

    -Database design might be altered because some extra info from receiving students might be stored and stuff like that depending on the implementation, infox will have to support all the communications about this info

    -Harder to update/patch

Problems:

    -The info that isn't stored and needs to be exchanged, how to work around that? Are there any law implications on how they are shared? Can they be temporarily stored or they can just be accessed to 'read only'?

    -Some kind of central control will be required if the central repository is to be implemented (extra staff/administrators needed).

Possible Solutions:

    -After a process is accepted, a ticket is created between the receiving and giving Institution, so when they need info about each other they can use that ticket that will block their view only to what's allowed and as soon the out process passes, the ticket expires.

    -Going with A1 b avoids that, probably the best way to go but still with some implications.

21 Annex 2: Early Email Example to Belgium

Written by P8-STATS team.

-

```
Heya,


We have a site with all or almost all the info for ERASMUS apparently but
 it's all in portuguese ( http://www.ipp.pt/index.php?a=25&id=104&sub=250
).


We can translate the forms if you like, but the idea of reaching an
uniformization seems unlikely(up to your investigation). In that case we
suggest looking into this idea/concept:


-Creating each and every form and storing them in file format( ala pdf, etc )


How?


-Given their location, presenting the correct form in lets say pdf and
 they have to print, fill, sign, scan it and upload it back(ie just one
example, web forms seem impractical because you have to sign)


Why?


-The idea is to keep this expandable, we produce a product that, let's
say, is ready to support 5 universities but the design makes it able to
expand even while already deployed they just need to add the new pdf to
the page
giving us the responsibility of making the core encapsulated and well
documented in order for further improvements that really should be their
problem(ala adding the respective universities forms).



I would also like to ask and therefore suggest if your making this
planning with the Greeks since i believe you should both use the same
methods etc.



As a last reminder don't forget communication, ask ask ask ask ask and ask
any doubts or suggestions (ala the meetings post has 0 comments!!).
```

Any further doubt feel free to mail or add me in google talk -
ped.j.ferreira@gmail.com

Regards,

Pedro Ferreira

ISEP, Portugal

‑

22 Annex 3: ErasmusLine User Manual

Written by P4-OUT team.

1. INTRODUCTION

This document acts as a guide to install the ErasmusLine web application at your own university.

In the dvd included in this document you will find all the necessary files to install the application at your university's web server and database server.

2. INSTALL APPLICATION

To install the application at your university, the first step is the copy the whole MUTW-folder included on the dvd to your web server. This folder contains all the functionality needed to run the application.

3. INSTALL DATABASE

On the dvd, in the folder database, you will find an mysql-dump called erasmusline.sql which can be easily imported on your local database server.

4. ADAPT CONFIG.PHP

4.1 Summary

In order for the web application to work flawlessly together with your database, you may have to adapt a few settings in the config.php file. This file can be found in the folder mutw/core/includes.

4.2 Database name

Standard this name is set to "ErasmusLine" but if you are using a different name for the database just set the right name here.

4.3 Database user

With the variables DB_USER en DB_PASS you can chance the username and password for the database.

4.4 Name of the University

For this parameter it is necessary to fill in the correct name of your university as it is written in the database!

4.5 Mail-Information

Since some information for the application is sent by mail, you will have to adapt some settings to work at your university. The SMTP-server is the most important parameter to change!

5. ADD PARTNERS

5.1 Start Database

22.1.1 In the database included on the dvd are all the partners which are for the moment a part of ErasmusLine included. Also all the educations and courses of these institutes are included.

5.2 New Partners

If a new partner starts using ErasmusLine, for the moment each institute will have to add this new partner to its own database. Also each education and course of this new partner will have to be added to the database by the administrator. We are still looking for a way to make this update go automatically.

6. CHANCE RIGHTS

For the application to be able to save some important pdf's and pictures of the users, you have to ensure that the application has the right to create subfolders and save files in the FILES-folder!

23 Annex 4: EIS Installation and User Manual

Written by P8-STATS team.

24 Introduction

This manual contains basic instructions on how the Systems Administrator can install the Erasmusline P8-STATS package application on a Linux system and to setup the Executive Information System.

This manual will also demonstrate how a regular user can consult the Executive Information System by using it's online User Interface.

25 Installation

The Erasmusline application may be distributed as an archive download or the System's Administrator can simply clone the SourceForge Git repository, as explained in the Erasmusline Project Report.

Here are the instructions to make an installation in a Linux system using the Git repository – for this example we will use the Ubuntu Linux distribution.

1. First it is necessary to have a basic LAMP stack installed – Linux, Apache, MySQL and PHP. Enter the following command in an opened terminal window:

```
sudo apt-get install lamp-server^
```

This command will install all the servers needed for the Erasmusline application to run.

2. If you don't have Git, curl or php related libraries installed in the system use the following command:

```
sudo apt-get install git curl php-pear php5-mcrypt phpunit
```

3. Next clone the SourceForge Git repository to your machine with this command:

```
git clone git://mutworange.git.sourceforge.net/gitroot/mutworange/mutworange
```

You will notice a newly created directory called **mutworange/.** This directory contains all the documentation and the Erasmusline application itself.

The example form now on will be executed from the user's home directory.
If there is a particular release version to use, you can issue the following commands:

```
cd $HOME/mutworange/
git tag -l       # lists all tags
git checkout release-1.0          # example release version
```

Before executing the automatic install script, the Administrator needs to setup the database user. This can be done by executing the following command:

```
cd $HOME/mutworange/erasmusline/scripts/
mysql -uroot -p******* < create_user.sql # creates the default db user for the #
application. Needs root user permissions.
```

After this step the Administrator can use the automated installation to create databases and run the ErasmusLine P8-STATS master and slave servers:

```
cd $HOME/mutworange/erasmusline/bin/
./install.sh
```

Now the administrator need to run the application on the Apache Server. The Administrator create a symbolic link to the default Apache Server *www documents* directory:

```
cd /var/www
sudo ln -s $HOME/mutworange/erasmusline/WEBSITE/ erasmusline
```

The ErasmusLine application can now be accessed through the url:

http://localhost/erasmusline

After logging in as Administrator you can access the EIS through the interface or directly through this url:

http://localhost/erasmusline/index.php?module=stats

25.1 Setting up communication with the Master Server

If the machine's installation is not meant for the Master server, the administrator will need to remove the master daemon installation and configure the slave server to communicate with a remote master server.

1. Let's start by removing the master server in the machine by typing these commands in the terminal window:

```
rm -rf $HOME/mutworange/erasmusline/stats/daemons/statsd_master
```

2. To configure the slave daemon to communicate with the remote master, the Administrator needs to edit the slave configuration file:

```
cd $HOME/mutworange/erasmusline/stats/daemons/statsd_slave

cp config.json config.inc.json      # do not alter the main config file for backup
                                    # purposes

vim config.inc.json                 # use vim, nano or emacs to edit the file
```

The relevant part of the configuration file will look like the following:

```
"master_serverconfig" : {
    "sockType" : "unix",
    "sockFile" : "/tmp/erasmusline/statsd_master/statsd_master.sock",
     "serverIP" : "0.0.0.0",
     "serverPort" : "8118"
   },
```

Alter the socket type and server ip to look like the following example:

```
"master_serverconfig" : {
    "sockType" : "tcp",
    "serverIP" : "95.136.101.157",
    "serverPort" : "8118"
},
```

Save the file and restart the slave daemon:

```
cd $HOME/mutworange/erasmusline/bin
./restart_stats_daemons
```

If the server address is correct, the local EIS installation will communicate with the proper Master Server.

25.2 Daemons Monitoring

25.2.1   Application Logs

The Master and Slave server logs are organized in the /tmp directory can be changed to another directory by editing each daemon's configuration files (see previous example).

To consult the logs, the Administrator can use the following commands in the terminal window:

```
tail -f /tmp/erasmusline/statsd_master/statsd_master.log
tail -f /tmp/erasmusline/statsd_slave/statsd_slave.log
```

To alter the verbosity of the logs, the Administrator can edit each daemon's configuration file and for example, reduce the log verbosity level:

```
"daemonconfig" : {
        "appName" : "statsd_slave",
        "logVerbosity" : 7
    },
```

25.2.2   Crash control

To assure the daemons are constantly running, the Administrator can install and configure a *process supervision tool* like Monit[27]:

```
sudo apt-get install monit
sudo vim /etc/monit/monitrc
```

---

27 http://mmonit.com/monit/

26 User Manual

After the installing and setting up the Erasmusline application, the users need to login by registering a new user or logging in as an Administrator for example. After logging in, select the EIS tab in the Erasmusline Interface.
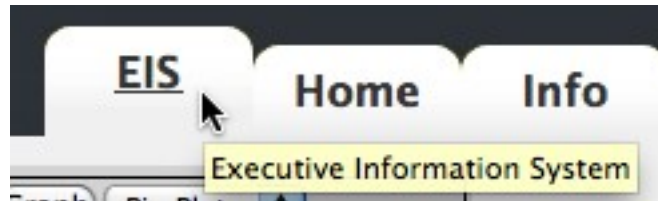
The EIS tab on the top-right of the screen

The EIS interface will start by loading the application's libraries, taking a few seconds to be ready.
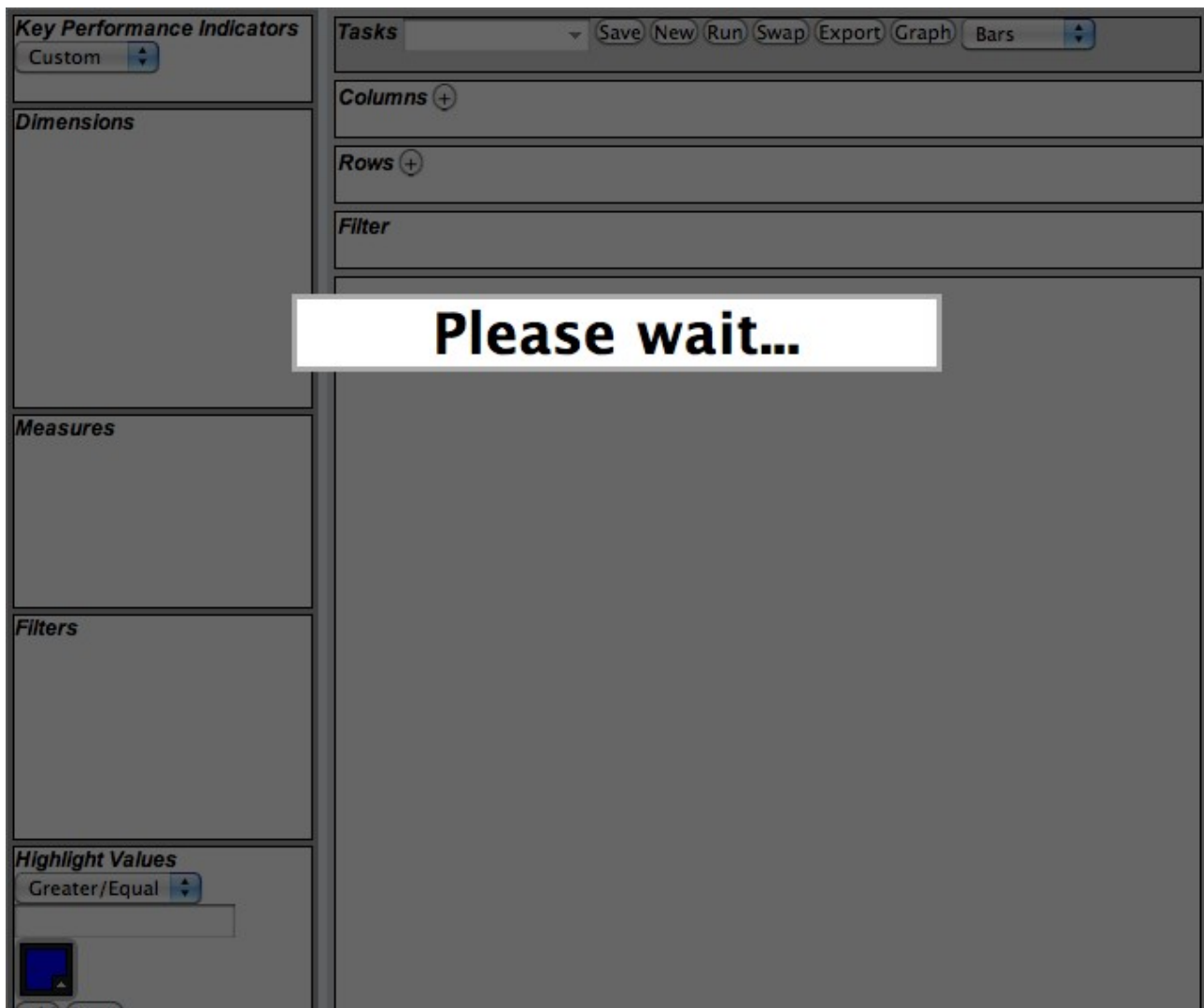
The EIS interface loading screen

When the loading is completed, the users have access to the EIS interface and can begin creating new analysis scenarios right away.

The EIS interface in action

26.1 The Sidebar

The sidebar on the left of the screen enables the users to select which **Key Performance Indicators** to measure and analyze. To begin, select the KPI to load the available Dimensions and Measures.
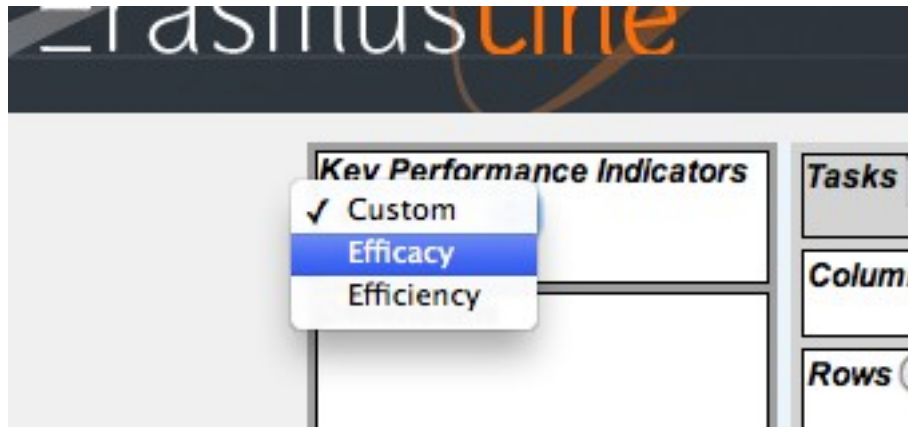
Selecting the preferred Efficacy
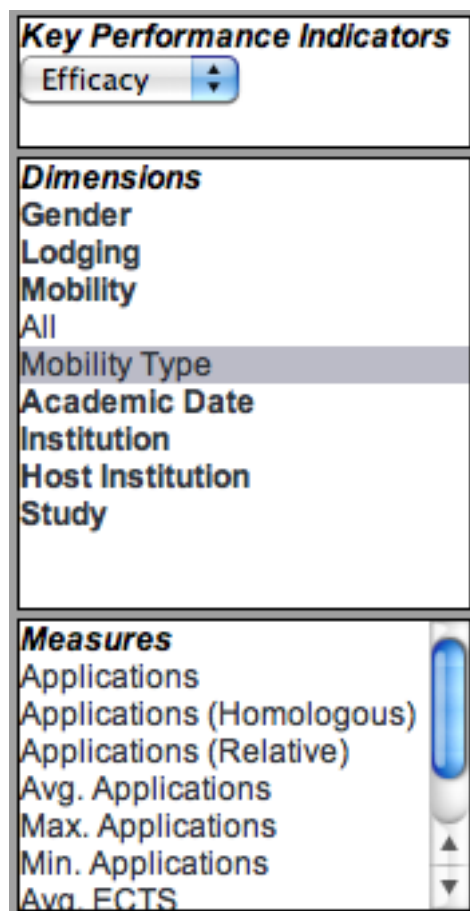
*Sidebar displaying KPI's,
Dimensions and Measures*

To generate an output Pivot Table[28], the users must select the **Dimensions** and **Measures** they wish to add to the **Columns** and **Rows** in the middle Toolbar. If a Measure is not selected, the default is always the first item in the Measure list.
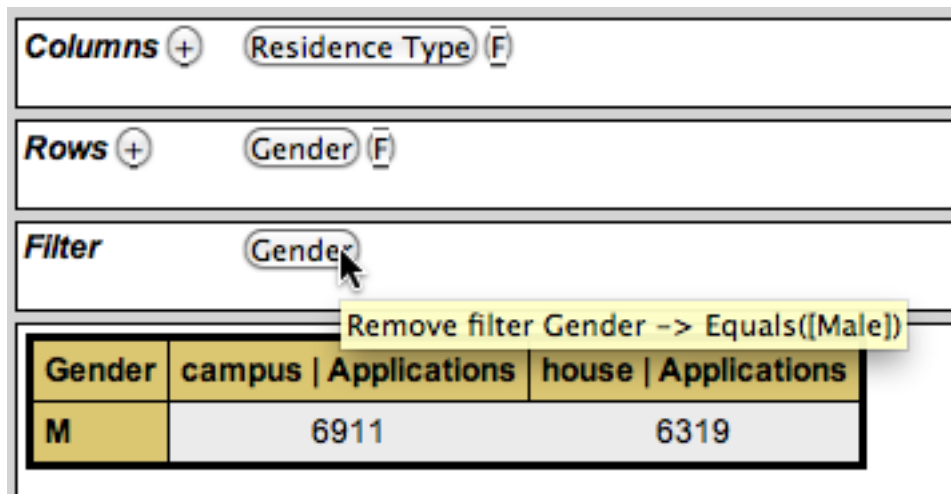
The middle Toolbar, displaying Columns and Rows with the selected attributes

If the **Dimensions** have an available filter option, press the **'F'** button to display the filter options in the left sidebar:
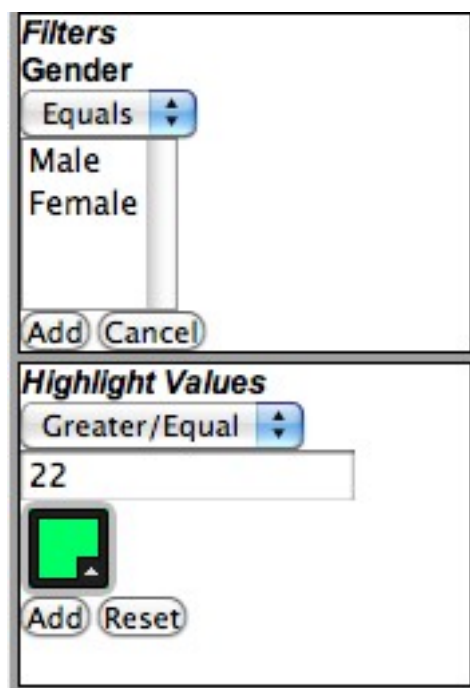
The filter options for the Gender Dimension

Another option in the sidebar is the possibility to **highlight values** from the result Pivot Table. The users select the numeric values, the color and the condition in which to paint

---

28 http://en.wikipedia.org/wiki/Pivot_tables

the table cells' background. This allows the users a better comprehension of large Pivot Tables.

26.2 The Toolbar
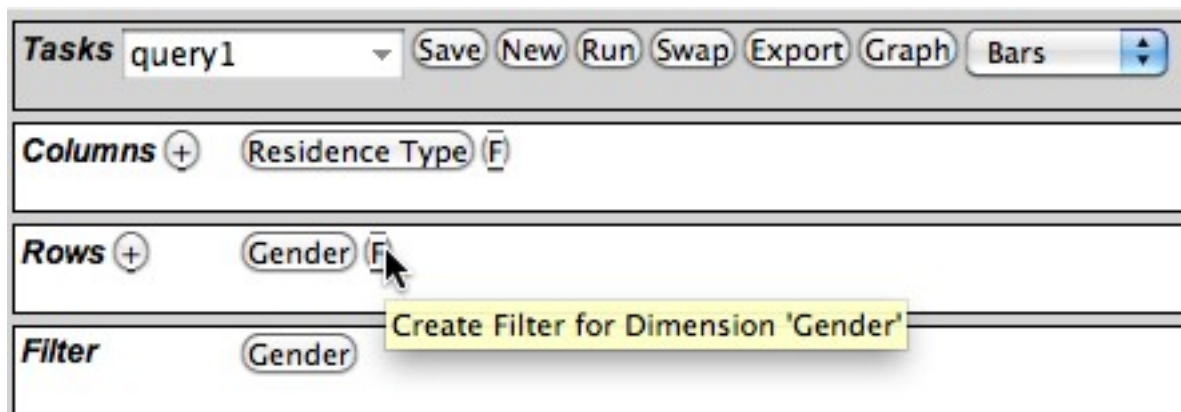
The middle Toolbar running a previously saved Scenario called 'query1'

The toolbar consists of four functional elements: the Tasks bar, the Columns bar, the Rows and Filter bar.

In the previous example, the users select Dimensions and Measures and distributes them between Columns and Rows. Pressing the **'Run'** button in the Tasks bar generates a Pivot Table bellow the main Toolbar. The user can swap Column elements by Rows using the **'Swap'** button, a new table is then generated.

The user can alter the table's Columns and Rows by pressing the selected Dimension and Row buttons, thus removing them from the toolbars – "Drill-down and Roll-up" operations - or create a new filter as previously shown in the previous section.

To load this Scenario next time the users' login, the users can enter a description on the select box in the Tasks bar and press the **'Save'** button. The scenario will be available to be loaded – the Columns, the Rows, the Filters and the Highlighted values – by choosing it from the select box.

Gender	Host Institution	campus | fkl | Applications	house | fkl | Applications
M	isep	655	601
F	isep	372	478

Pivot Table with Highlighted values superior to 500

26.3 Other Outputs

In the Tasks bar, the users can press the **'Export'** button to generate a downloadable version of the pivot table in CSV[29] file format. In this format the users can open these files in any modern spreadsheet application available in every Office software suite.

	A	B	C	D	
1	Gender	Host Institution	campus | fkl | Applications	house | fkl | Applications	
2	M	isep	655	601	
3	F	isep	372	478	

CSV file displayed in a spreadsheet tool

26.4 Charts

The Google Chart Tools API[30] provides a series of tools to generate various types of Charts and Graphs suited for the EIS interface.

The EIS interface can generate **Bars**, **Lines** and **Pie Charts** based on the loaded scenario settings and can extend to other types of Chart using the Chart Tools API. Users can choose the preferred graph from the select box on the Tasks bar and press the **'Graph'** button to display a screen with the generated.

---

29 http://en.wikipedia.org/wiki/Comma-separated_values
30 http://code.google.com/apis/chart/
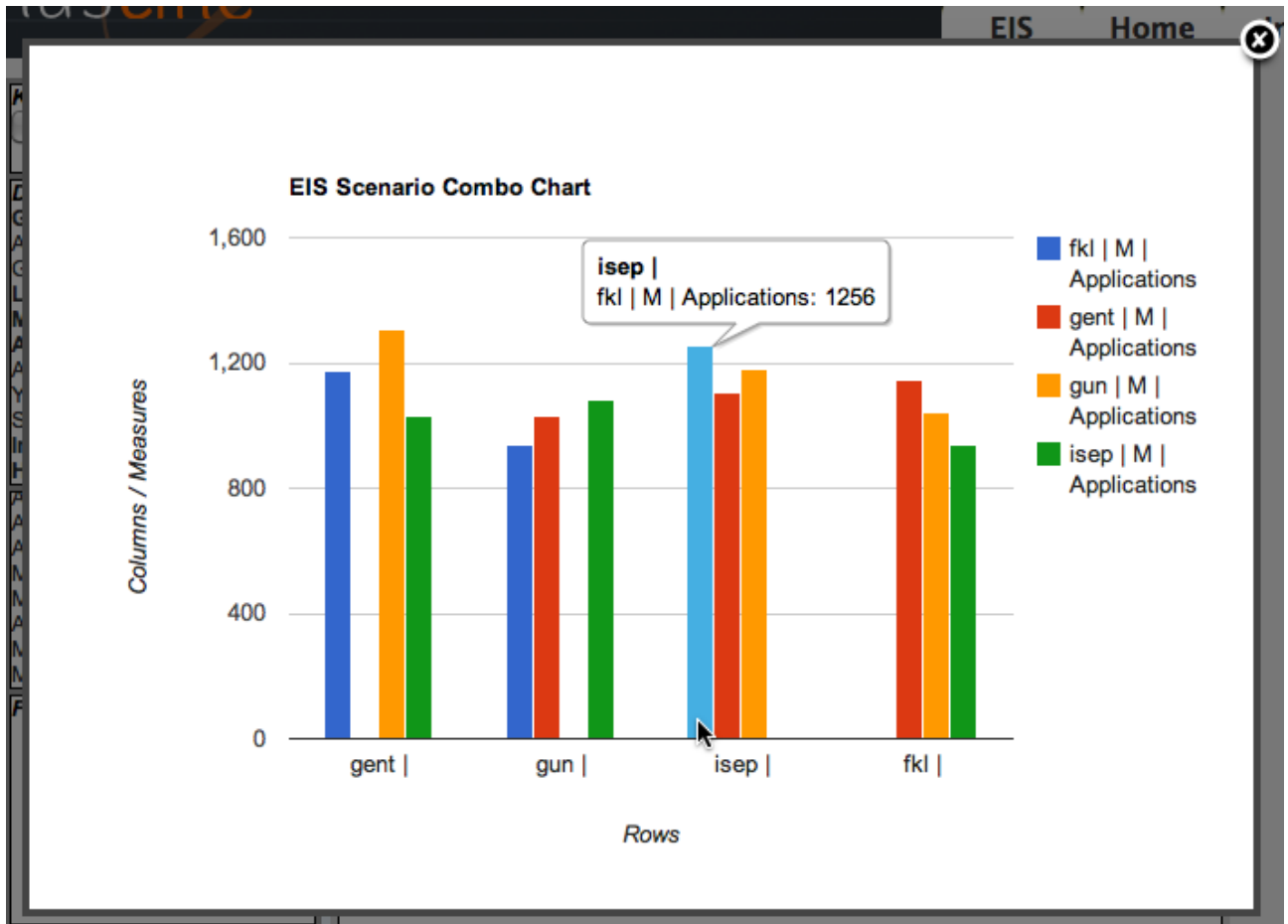
Generated scenario Bar Chart output
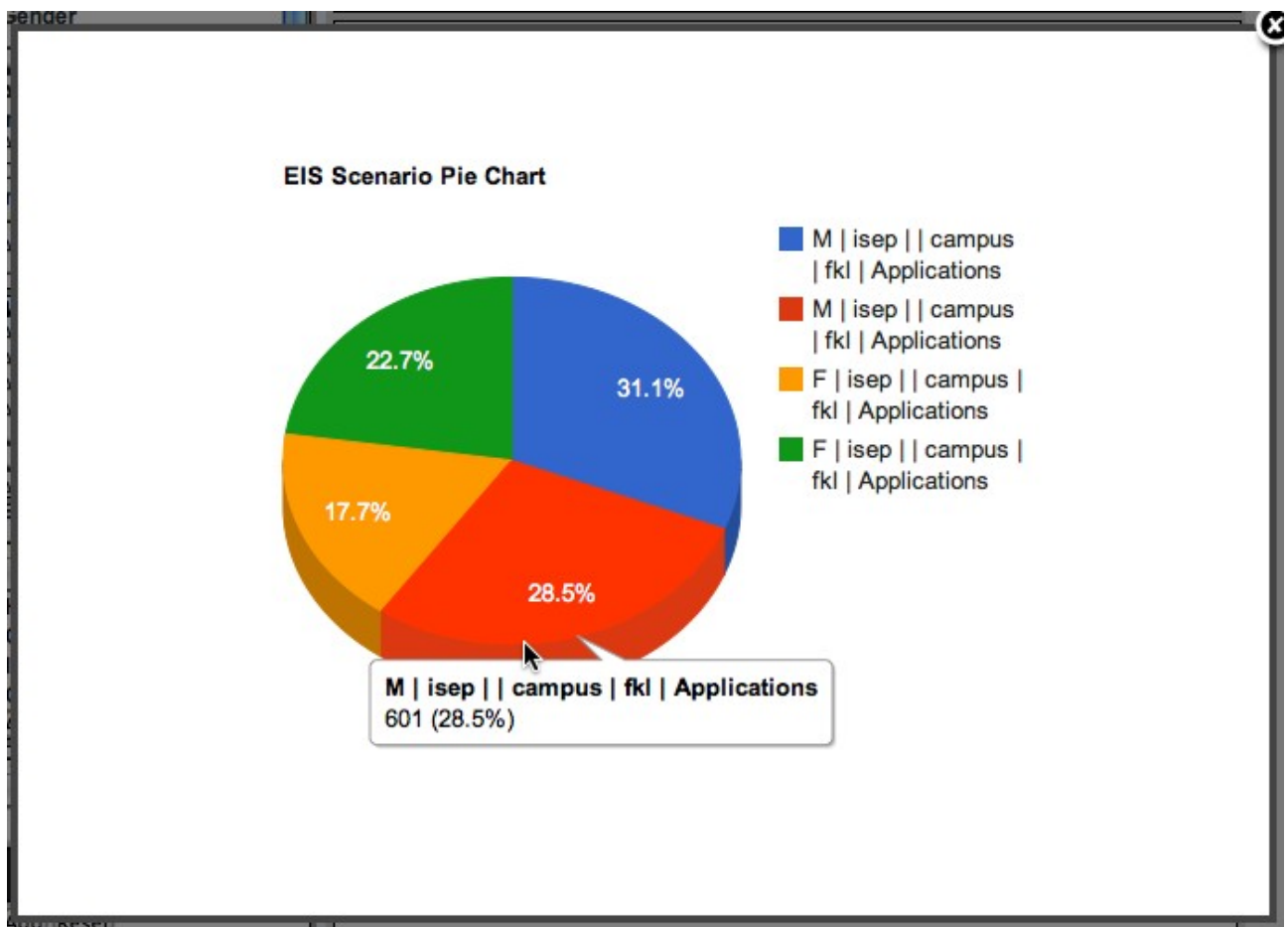
Pie Chart generated graphic

27 Annex 4: Gantt Chart

	ⓘ	Name	Duration	Start
1		⊟ErasmusLine	65 days?	3/14/11 8:00 AM
2		⊟P8-STATS	60.062 d...	3/14/11 8:00 AM
3	▦	⊟Analysis and Requirements gathering	12 days	3/14/11 8:00 AM
4		Business Rules	5 days	3/14/11 8:00 AM
5		⊟Performance Indicators	5 days	3/21/11 8:00 AM
6		Measures	5 days	3/21/11 8:00 AM
7		Dimensions	5 days	3/21/11 8:00 AM
8		Report	2 days	3/28/11 8:00 AM
9		⊟Data Warehouse	36.062 d...	3/30/11 8:00 AM
10		Database schema	4.031 days	3/30/11 8:00 AM
11		Define data source(s)	3 days	4/5/11 8:15 AM
12		⊟Architecture	6 days	4/8/11 8:15 AM
13		DW refreshment	2 days	4/8/11 8:15 AM
14		ODS	4 days	4/8/11 8:15 AM
15	✔	Indexing Solutions	6 days	4/8/11 8:15 AM
16		⊟Implementation	15 days	4/18/11 8:15 AM
17		DW refreshment	15 days	4/18/11 8:15 AM
18		ODS	15 days	4/18/11 8:15 AM
19	✔	Indexing	15 days	4/18/11 8:15 AM
20		⊟Deployment	5.031 days	5/9/11 8:15 AM
21		DW refreshment	5 days	5/9/11 8:15 AM
22		ODS	5.031 days	5/9/11 8:15 AM
23		Data Marts	5 days	5/9/11 8:15 AM
24		Report	3 days	5/16/11 8:30 AM
25		⊟EIS	12 days	5/19/11 8:30 AM
26		Architecture	12 days	5/19/11 8:30 AM
27		Implementation	12 days	5/19/11 8:30 AM
28		Deployment	12 days	5/19/11 8:30 AM
29		Report	12 days	5/19/11 8:30 AM
30		⊟P4-OUT	57 days?	3/14/11 8:00 AM
31		Collect Different forms	15 days?	3/14/11 8:00 AM
32	▦	Setting up chart with the form flow	3 days?	3/25/11 8:00 AM
33	▣	Agree with P-IN on layout forms	5 days?	3/28/11 7:00 AM
34	▦	Determ which forms can be common	5 days?	4/4/11 7:00 AM
35	▦	Implement non/common forms	15 days	4/11/11 7:00 AM
36	▣	Creating ECTS forms	5 days?	3/28/11 8:00 AM
37	▣	Implement flow of forms	22 days?	4/11/11 7:00 AM
38	▣	Integrate information exchange module	5 days?	4/18/11 7:00 AM
39	▣	Intensive testing of flow	16 days?	5/10/11 7:00 AM
40		⊟P1-Config	30 days?	3/21/11 8:00 AM
41		⊟Database	30 days?	3/21/11 8:00 AM
42	▣	Design	5 days?	3/21/11 8:00 AM
43	▣	SQL-Script	3 days?	3/25/11 8:00 AM
44	▣	Corrections	4 days?	3/29/11 7:00 AM
45	▣	Final_Database	3 days?	4/4/11 7:00 AM
46	▣	Installation-Script	18 days?	4/6/11 7:00 AM
47		⊟Design	16 days?	3/25/11 8:00 AM
48	▣	3_Templates	6 days?	3/25/11 8:00 AM
49	▣	including_into_plonk	10 days?	4/4/11 7:00 AM
50		⊟User_Account	10 days?	4/18/11 7:00 AM

Finish	Predeces...	Percent C...
6/10/11 5:00 PM		25%
6/6/11 8:30 AM		72%
3/29/11 5:00 PM		86%
3/18/11 5:00 PM		80%
3/25/11 5:00 PM	4	90%
3/25/11 5:00 PM		90%
3/25/11 5:00 PM		90%
3/29/11 5:00 PM	4;5	80%
5/19/11 8:30 AM	3	90%
4/5/11 8:15 AM		90%
4/8/11 8:15 AM	10	90%
4/18/11 8:15 AM	10	95%
4/12/11 8:15 AM	11	90%
4/14/11 8:15 AM	11	90%
4/18/11 8:15 AM	11	100%
5/9/11 8:15 AM	12	93%
5/9/11 8:15 AM		90%
5/9/11 8:15 AM		90%
5/9/11 8:15 AM		100%
5/16/11 8:30 AM	16	87%
5/16/11 8:15 AM		80%
5/16/11 8:30 AM		90%
5/16/11 8:15 AM		90%
5/19/11 8:30 AM	10;12;1...	50%
6/6/11 8:30 AM	3;9	35%
6/6/11 8:30 AM	24	40%
6/6/11 8:30 AM	24	50%
6/6/11 8:30 AM	24	30%
6/6/11 8:30 AM	24	20%
5/31/11 5:00 PM		6%
4/1/11 5:00 PM		20%
3/29/11 5:00 PM		0%
4/1/11 5:00 PM		0%
4/8/11 5:00 PM		0%
4/29/11 5:00 PM		0%
4/1/11 5:00 PM		50%
5/10/11 5:00 PM		0%
4/22/11 5:00 PM		0%
5/31/11 5:00 PM		0%
4/29/11 5:00 PM		0%
4/29/11 5:00 PM		0%
3/25/11 5:00 PM		0%
3/29/11 5:00 PM		0%
4/1/11 5:00 PM		0%
4/6/11 5:00 PM		0%
4/29/11 5:00 PM		0%
4/15/11 5:00 PM		0%
4/1/11 5:00 PM		0%
4/15/11 5:00 PM		0%
4/29/11 5:00 PM		0%

Wed 16 Mar | Thu 17 Mar | Fri 18 Mar | Sat 19 Mar | Sun 20 Mar | Mon 21 Mar | Tue 22 Mar | Wed 23 Mar | Thu 24 M

0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12 | 18 | 0 | 6 | 12

p8_stats - page3

p8_stats – page5

105 / 126

p8_stats – page12

		Name	Duration	Start
51		Login_Functionality	5 days?	4/18/11 7:00 AM
52		Account-Page	5 days?	4/25/11 7:00 AM
53		⊟P2-INFOX	35 days?	3/28/11 7:00 AM
54		Discussion_about_Protocol	5 days?	3/28/11 7:00 AM
55		Programming_outgoing_Data	20 days?	4/4/11 7:00 AM
56		Programming_incoming_Data	20 days?	4/4/11 7:00 AM
57		Installing_security_system	10 days?	5/2/11 7:00 AM
58		⊟P3-Alerts	45 days?	3/22/11 8:00 AM
59		Research	12 days?	3/22/11 8:00 AM
60		Making the daemon	10 days?	4/7/11 7:00 AM
61		Synchronization with the others part of the p...	5 days?	4/21/11 7:00 AM
62		Email sending system	5 days?	4/28/11 7:00 AM
63		Pop-ups	5 days?	4/28/11 7:00 AM
64		GUI	3 days?	5/5/11 7:00 AM
65		Testing	10 days?	5/10/11 7:00 AM
66		⊟P6-Exams	27 days?	5/5/11 7:00 AM
67		Research	4 days?	5/5/11 7:00 AM
68		Making the student module	10 days?	5/11/11 7:00 AM
69		Making the home coordinator module	10 days?	5/11/11 7:00 AM
70		Making the host coordinator module	10 days?	5/11/11 7:00 AM
71		Synchronization with the others part of the p...	5 days?	5/25/11 7:00 AM
72		Testing	8 days?	6/1/11 7:00 AM

Finish	Predeces...	Percent C...	Fri 11 Mar				Sat 12 Mar				Sun 13 Mar				Mon 14 Mar				Tue 15 Mar		
			0	6	12	18	0	6	12	18	0	6	12	18	0	6	12	18	0	6	12	18
4/22/11 5:00 PM		0%																		
4/29/11 5:00 PM		0%																		
5/13/11 5:00 PM		**0%**																		
4/1/11 5:00 PM		0%																		
4/29/11 5:00 PM		0%																		
4/29/11 5:00 PM		0%																		
5/13/11 5:00 PM		0%																		
5/23/11 5:00 PM		**0%**																		
4/6/11 5:00 PM		0%																		
4/20/11 5:00 PM		0%																		
4/27/11 5:00 PM		0%																		
5/4/11 5:00 PM		0%																		
5/4/11 5:00 PM		0%																		
5/9/11 5:00 PM		0%																		
5/23/11 5:00 PM		0%																		
6/10/11 5:00 PM		**0%**																		
5/10/11 5:00 PM		0%																		
5/24/11 5:00 PM		0%																		
5/24/11 5:00 PM		0%																		
5/24/11 5:00 PM		0%																		
5/31/11 5:00 PM		0%																		
6/10/11 5:00 PM		0%																		

n 11 Apr | Tue 12 Apr | Wed 13 Apr | Thu 14 Apr | Fri 15 Apr | Sat 16 Apr | Sun 17 Apr | Mon 18 Apr | Tue 19 Apr

6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18

n 16 May | Tue 17 May | Wed 18 May | Thu 19 May | Fri 20 May | Sat 21 May | Sun 22 May | Mon 23 May | Tue 24 May

6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18 0 | 6 | 12 18

p8_stats – page26

p8_stats – page28