Project Management Component

- for Flygprestanda AB's web system

LTH School of Engineering at Campus Helsingborg

Bachelor thesis:
Andreas Lundgren

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2009

Abstract

Flygprestanda AB make software and databases for hundreds of airline companies all over the world. Their services include both software development and technical-engineering calculations with focus on delivering all information needed to perform a commercial flight from one place to another.
The company is currently developing a web system with Google Web Toolkit which will be used within the company. There is also a possibility that the system will be sold to other companies in the future.
An important part of the system is a component which allows the user to plan, schedule and review tasks contained in Flygprestanda AB's daily operations.

During the thesis describe in this report the best way to realise this component has been researched. An initial investigation has concentrated on finding a suitable project management technique on which to base the component, and how to practically implement it.
The project management techniques have been compared based on what information they can present and the possibility to realise them with software. Flyprestanda AB's current way of managing projects has also been taken into account. The chosen technique is the Gantt chart, which satisfies the demands.
The investigation has also compared the advantages and disadvantages of developing the project management component from the ground up or using an existing solution.
For the purpose of finding a programming language to use if the component was to be developed from scratch, a variety of web programming languages such as Adobe Flash and JavaScript have been compared.
Different existing solutions, such as FusionWidgets and EJS TreeGrid, have been investigated with focus on their presentation- and interaction capabilities, their ways to communicate with a database, and to some extent their available licenses and offered support.

The decision made during the investigation was not to develop the component from the ground up, but to integrate an existing component, namely the JavaScript solution EJS TreeGrid.
The integration of EJS TreeGrid was realised by adapting the component and placing it in the web system's interface trough JSNI, and writing code that made it able to communicate with the system's MySQL database.
A component intended to provide a way of managing milestones in a project, the GWT-YUI-Carousel, was added later during the work. This component was also adapted to the system and its database.

Keywords: web system, project management, scheduling of tasks, integration

Sammanfattning

Flygprestanda AB konstruerar mjukvaror och databaser för hundratals flygbolag världen över. Deras tjänster innefattar både mjukvaruutveckling och tekniskt ingenjörsarbete med fokus på att tillhandahålla all information som behövs för att genomföra en kommersiell flygning från en plats till en annan.
Företaget utvecklar för närvarande ett webbsystem i Google Web Toolkit som kommer att användas inom företaget, med möjlighet till försäljning till andra företag i framtiden.
En viktig del av applikationen är en komponent som tillåter användaren att planera, schemalägga och granska aktiviteter inom Flygprestanda AB's dagliga arbete.

Under examensarbetet beskrivet i denna rapport har det bästa tillvägagångssättet för att konstruera denna komponent undersökts. En inledande undersökning var koncentrerad på att hitta en passande projekthanteringsteknik på vilken komponenten skulle baseras. Den utredde också hur komponenten skulle implementeras rent praktiskt.
Projekthanteringsteknikerna har jämförts baserat på vilken information de kan presentera och möjligheterna att realisera dem med mjukvara. Hänsyn har också tagits till Flygprestanda AB's nuvarande sätt att hantera projekt. Den valda tekniken är Gantt-schemat, vilken uppfyller kraven väl.
Utredningen har också jämfört för- och nackdelarna mellan att utveckla projekthanteringskomponenten från grunden eller använda en existerande lösning.
Under utredningen har olika programmeringsspråk jämförts, såsom Adobe Flash och JavaScript. Olika existerande lösningar, exempelvis FusionWidgets och EJS TreeGrid, har undersökts med fokus på deras presentations- och interaktionsmöjligheter. Även deras möjligheter att kommunicera med en databas har jämförts, samt deras tillgängliga licenser och support.

Under utredningen fattades beslutet att inte utveckla komponenten från grunden, utan istället integrera en existerande lösning; JavaScript-lösningen EJS TreeGrid. Integrationen av EJS TreeGrid utfördes genom att anpassa komponenten och placera den i webbsystemets gränssnitt med JSNI, samt att utveckla kod som gjorde det möjligt för den att kommunicera med systemets MySQL-databas.
En ytterligare komponent, GWT-YUI-Carousel, ämnad att ge ett sätt att hantera milstolpar i ett projekt, lades till senare under arbetet. Även denna komponent integrerades med systemet och dess databas.

Nyckelord: webbsystem, projektledning, schemaläggning av aktiviteter, integrering

Foreword

First of all, I would like to give a significant thank you to my mentor at Flygprestanda, Vadim Feldman who helped and listened to me during the entire thesis. An equally considerable appreciation goes to Christin Lindholm, my examinator at LTH. Thank you for guiding me through the different parts of the thesis.
During my weeks at Flygprestanda I received a great deal of help from Martin Lindström, David Altengård, Philiphe Netterby and Andreas Holstenson for which I am very grateful. Without their help, I would never have completed my work.

List of contents

1 Background **2**
1.1 Flygprestanda AB **2**
1.2 Web system **2**
2 Problem description **3**
3 Goals **3**
4 Work method **4**
4.1 Project model **5**
4.2 Time plan **6**
5 Current system **8**
5.1 Google Web Toolkit **8**
5.2 Database **9**
6 Investigation **10**
6.1 Project management **10**
6.1.1 Techniques 10
6.1.1.1 Critical Path Method *10*
6.1.1.2 PERT *12*
6.1.1.3 RAM *13*
6.1.1.4 Gantt *14*
6.1.2 Choice of technique 15
6.2 Implementation **16**
6.2.1 Programming languages 16
6.2.1.1 Flash *16*
6.2.1.2 JavaScript *17*
6.2.1.3 GWT widget *18*
6.2.2 Possible existing solutions to integrate 18
6.2.2.1 FusionWidgets *19*
6.2.2.2 AnyGantt *19*
6.2.2.3 jsGantt *20*
6.2.2.4 EJS TreeGrid *20*
6.2.3 Own or existing solution? 21
6.2.4 Choice of solution 21
7 Integration **22**
7.1 EJS TreeGrid **22**
7.1.1 Adding to the interface 22
7.1.2 Removing and adding functions 24
7.1.3 Database communication 26
7.1.3.1 Reading from database *26*
7.1.3.2 Updating to database *27*
7.1.4 Functions outside EJS TreeGrid 28

7.2 GWT-YUI-Carousel 29
7.2.1 Adapt to system 29
7.2.2 Database communication 30
7.2.3 Functions outside GWT-YUI-Carousel 30
8 Problems 31
9 Conclusions 32
10 Future development 34
11 Terminology 35
12 References 36
Appendix 39
Screenshots 39

Introduction

This report intends to give an explanation to the thesis as a whole. It describes the goals and the work conducted to fulfil them.
The report contains an introduction to the existing web system at Flygprestanda AB, and an investigation of different project management techniques. The implementation and integration conducted is explained, as well as the problems encountered. There are also chapters containing conclusions and possible future developments.

The report does not contain any code from the existing system or the finished implementation, except for simple code examples.
The solutions discussed are only on such a level that no sensitive business information is revealed, this to respect the company interests of Flygprestanda AB.

1 Background

The first step towards conducting this bachelor thesis was taken when Vadim Feldman, who works as manager for the Software Project Department at Flygprestanda AB contacted the school with an offer for a student to perform a thesis at the company. The teachers at school told us students about the opportunity and it was seized by e-mailing Flygprestanda AB in December 2008. After a short e-mail conversation it was decided that we should meet at Flygprestanda AB's head office in Malmö the same month. At this first meeting an introduction to the company was made and a first insight into the system to work with. It felt like there was quite a lot of information to take aboard, and the real essence of the job felt far from clear.
Shortly after, Christin Lindholm was contacted with an enquiry about acting as the examinator for the thesis. The response was positive and in January we sat down at school to discuss how to start. A few days later the next visit to Flygprestanda occurred. This time a more thorough explanation was made about what it was they wanted to be done. That same day the initial research for the thesis started, and from then on the nature of the job became more and more clear.

1.1 Flygprestanda AB

Flygprestanda AB, hereafter Flygprestanda, make software and databases for hundreds of airline companies all over the world. Their services include both software development and technical-engineering calculations with focus on delivering all information needed to perform a commercial flight from one place to another. These services consist of, among others; airport analysis, mass and balance calculations and drift-down calculations. [1]
Flygprestanda have 32 employees and the head office is situated in central Malmö. [2]

1.2 Web system

Flygprestanda have during the last year constructed a system for use within the company by the employees, and in particular the managers. The system is web based, which means that it consists of web pages that the user views in a web browser.
The system aims to provide a platform for managing many of the internal processes and resources; such as services, customers, employees and projects. One of the most important functions of the system is the ability to manage, i.e. schedule and edit, the projects which are carried out at the company. It is within this part of the system that this thesis resides.

Because of Flygprestanda's business interests and the fact that this report is available to the public, only the very surface of the implementation of the existing system is mentioned and explained. See chapter *5 Current system* for this short insight into the system.

2 Problem description

As mentioned, a vital part of Flygprestanda's web system is the one which supports management of in-house projects.
Flygprestanda was in need of a component which would extend the existing abilities concerning this part of the system.
At the start of the thesis, the system lacked a suitable way to:

- View
- Create
- Edit
- Schedule

…the tasks contained in the projects at the company.
The system specifically lacked a way to present the projects and their tasks in some kind of time-oriented way, for example along a time-axis or in some sort of diagram. Without this function, the job to schedule projects and their tasks is significantly more cumbersome and therefore time-consuming.

The first problem was to find a suitable project management technique on which to base the presentation of projects and tasks. After a technique had been found, a way of realising it had to be researched. Depending on this research, a solution would be either developed from scratch or consist of an already existing implementation. Regardless of this choice, the solution then had to be integrated into the existing system.

Not part of the original problem description, but added later during the thesis, was a *milestone carousel.* The function of the milestone carousel is to show and present a way to select milestones when managing a project. Milestones are a way of grouping together activities within a project, and among other things, set an end date to them.

3 Goals

The main goal of this thesis is to solve the problems described in chapter *2 Problem description* in a suitable way.

An important goal, not mentioned in the problem description, is that the component should be seamlessly integrated with the current system, and should also be easy to reuse, as a whole or with certain selected portions, in other parts of the system.

Another important aspect of the component is that its user interface and functions need to be self-explanatory to the user. There should be a small risk of the user misinterpreting the functions in the component. Despite the importance of this, there have been no specific requirements set concerning the user interface, instead all parts of it have been continuously discussed with and tested by Flygprestanda, represented by Vadim Feldman, who is one of the most prominent future users of the system.
No other user tests have been performed since the system is going to be used inside the company. The inside use means that receiving feedback from the users is easier and fixes and enhancements of the system reach them quicker.

4 Work method

The work with the thesis was divided in three separate steps:

1. Investigation
    a. Project management techniques
    b. Programming languages
    c. Existing solutions
    d. Own or existing solution?
2. Implementation or adaptation of component
3. Integration

As mentioned in chapter *2 Problem description*, the initial investigation would answer the question about which project management technique that would be used as the basis for the function of the component. The investigation would also examine the best suitable way to implement the component, either from scratch or by adapting an existing component. The investigation would be carried out by looking up information on the internet, both from sites recommended by Flygprestanda and from new sites found via search engines and links.
During the second step the work of implementing or adapting the component would be carried out, to finally, in the third step be integrated into the system.

Apart from these steps a time plan is also available which describes the process in more detail, see chapter *4.2 Time plan*.

4.1 Project model

On the department at Flygprestanda where the thesis was conducted, they follow a project model called *RAD*. RAD stands for *Rapid Application Development* and is a development life cycle constructed specifically for software development. One of the main characteristics of RAD is the prioritisation of prototypes over planning, actually most of the planning takes place during the development of the prototypes. This allows for the software prototypes to be developed faster and the requirements to be changed easier.
The life cycle starts with the developer building preliminary models for data and processes based on requirements.
After this, usually several prototypes are created one after another, sometimes each one for a different part of a whole system. With each completed prototype, the solution is verified against the requirements. If the solution is approved, the work with next prototype is continued, if not, the process starts over again for that particular part of the system.

While working after RAD means that the software is developed quicker, it can also affect the functionality and performance of the final software in a negative way. It is therefore important that the company working after this model takes future maintenance, such as correcting faults and missing features, into the account.
[3]

The work with this thesis was initially supposed to follow the RAD model, in the sense that prototypes where to be delivered, specifically two prototypes; divided into design and function, or stand-alone application and integration. This was not the case, instead the work flow became more linear with different parts developed one after another, with no real distinction between the steps and slightly more informal acceptance tests. The work was not conducted based on a particular requirement specification, instead requirements were set during the development in close contact with Flygprestanda.
See chapter *9 Conclusions* for further analysis of the project model.

4.2 Time plan

The work with this thesis was performed during 17 weeks in early February to late May 2009. The investigation and development was conducted during the first 14 weeks at the Flygprestanda head office in central Malmö with the use of the company's own equipment. The work schedule consisted of 8 hours a day (fulltime), Mondays to Thursdays. The Fridays were spent at home compiling the job done during the week, and with a short amount of time spent on writing the report.
This writing was intensified during the last three weeks at home.

The time plan set in the beginning of the thesis is depicted in *Diagram 1*, and the time plan eventually followed is in *Diagram 2*.

Week number		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Calendar week		5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Activity	**Dates** (approx)															
Introduction		■														
Investigate and prepare	2/2 - 12/2		▒	▒												
Step 1: Integrate component	16/2 - 12/3				■	■	■	■								
Test and validate	12/3 - 19/3							▒	▒							
Step 2: Extract and feed data to component	19/3 - 15/4								■	■	■	■	■			
Test and validate	15/4 - 23/4												▒	▒		
Write report	Fridays + 27/4 - 10/5														■	■

Diagram 1: Initial time plan

Week number		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Calendar week		5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Activity	Dates																	
Introduction		▓																
Investigate and prepare	2/2 - 5/2		▒															
Integrate and adapt component	9/2 - 24/2			▓	▓	▓												
Read data from database and present in component	24/2 - 2/3					▒	▒											
Upload changed data to database from component	3/3 - 16/3						▓	▓	▓									
Style the interface	17/3 - 18/3, 22/4 - 27/4								▒					▒	▒			
Implement support for milestones	19/3 - 23/3								▓	▓								
Investigate milestone carousel	1/4 - 2/4										▒							
Integrate milestone carousel	2/4 - 8/4										▓	▓						
Test, fix problems and validate	7/4 - 28/4											▒	▒	▒	▒			
Implement templates	20/4 - 23/4													▓				
Write report	4/5 - 24/5															▒	▒	▒

Diagram 2: Followed time plan

As seen in *Diagram 1*, the thesis was initially planned to be performed during 15 weeks, but because of different factors, such as last minute problems, it demanded two more weeks. See chapter *9 Conclusions* for a further discussion of this and other differences between the plan and reality.

5 Current system

The system into which the project management component will be integrated is a web based system currently under development at Flygprestanda. The system is supposed to work as a place where the employees in the different departments, and in particular the managers at the company, can plan and follow their work and projects. The system hosts, among other things, a time-track function which monitors the employee's arrival and departure dates. It also has support for storing, presenting and editing information about the company's customers, services and software.
To get access to the system a username and password is required, which every employee at the company has.
Since the system is not completed yet, only certain parts of it are in use at this point. The system will, before it is launched, undergo a redesign of the interface and improvements of certain features will be carried out.
If the outcome of the use within Flygprestanda goes well, there is a future possibility that the system will be sold in some form to other companies.

The web system uses Google Web Toolkit as a framework and a MySQL database for storing the data contained in the system. These are explained in chapter *5.1 Google Web Toolkit* and *5.2 Database* respectively.

As mentioned earlier, with regards to Flygprestanda's business interests, only small parts of the system are explained in this report.

5.1 Google Web Toolkit

Flygprestanda's web system uses *Google Web Toolkit* as its software framework. It is therefore also the framework used to build the component in this thesis.
Google Web Toolkit, abbreviated GWT, is an open source development framework constructed by Google to ease the development of interactive web applications. The first version of GWT was released May 16, 2006.
The framework makes it possible for developers to create and maintain advanced AJAX applications without the need to write any JavaScript. Instead all coding is done in the Java programming language. GWT compiles the written Java code into working JavaScript, which can be viewed in a regular web browser. The resulting JavaScript code is also optimised to load and execute faster than code written by hand in JavaScript. Redundant parts of the code are removed, which means the result does not contain any unnecessary code that makes its size larger.

Because of the code being written in Java, the developer is able to step through and debug it. This would not be possible if the code was constructed directly in JavaScript.

GWT is a multi-platform framework, much thanks to the use of Java, which means it can be used on a variety of different platforms, in this case Linux, Windows and Mac OS X.
[4]

5.2 Database

Flygprestanda's web system uses a *MySQL* database. In this chapter, only the parts of this database which are important for the thesis are explained.

The projects contained in Flygprestanda's operations are represented by project instances in the database. Each of these instances can be one of the types; software project, service project, sales project or customer project.

Each project contains tasks, which in turn are represented by task instances. These task instances contain data such as name, description and dates. They can also have attributes which work like dependencies and link the tasks together.

Within a project, the tasks can be grouped together by milestone instances, which decide when a collection of tasks are to be conducted, i.e. start and end date.

A more straight-forward way to explain the hierarchy is by giving an example of how the different instances can be viewed in the system.
First the user selects the project he or she wishes to view. All the milestones and tasks contained in the selected project are shown. By selecting a certain milestone, the view of the tasks is updated to only show the ones belonging to the selected milestone.

6 Investigation

This chapter describes the initial investigation carried out. The purpose of the investigation was to find a suitable project management technique and an appropriate way to implement the component.

6.1 Project management

The first part of the initial investigation was to determine the project management technique on which to base the function and layout of the component. In this chapter the most prominent of the techniques found during the investigation are presented. There also follows an explanation to why the chosen technique was preferred above the others.

6.1.1 Techniques

A project management technique is a way of managing a project by visualising its work flow. The work flow often consists of the activities within the project, their start- and end dates, resources and the people responsible of carrying out the activities. As will be explained in this chapter, the contents differ between the various techniques.

6.1.1.1 Critical Path Method

The *Critical Path Method*, or *CPM* for short, is a project management technique which relies heavily on mathematical calculations. It is used in projects where the duration of the different activities and the dead-line for the entire project is important. CPM aims to provide a way to find which certain activities are critical and how much time and effort is needed to complete them. The steps towards achieving this are described in order below:

1. The first step is to make a **list of** which **activities** that are contained within the project to manage.
2. With these activities specified, the next step is to determine the **sequence of** these **activities**, i.e. which activity needs to be completed before another and so on. These are called **dependencies**.
3. One of the most important steps when working with CPM is the step where a **network diagram** is drawn. In this diagram, each activity is represented by a node, and the dependencies between them are depicted as lines with arrows.
4. After the network diagram is drawn the **estimated completion time** for each activity is added to the corresponding nodes. These estimated times are based on past experiences and factors such as personnel and other resources.
5. With the network diagram completed, the work with identifying the so called ***critical path*** can begin. When jumping between the activity nodes via the dependency lines and adding the time of each visited node, the path with the highest sum is the critical path. The significance of the critical path is described below.

6. It is important to **update** the network diagram during the duration of the managed project. As activities are finished or delayed this new data needs to be entered into the diagram. It is possible that a new critical path is created, and this needs to be taken into account when continuing the work with the project.

The critical path found in the network diagram is the central part of CPM. It tells the project manager how long it will take for the project to be completed and which activities that, if delayed, will increase this time. The critical path therefore also gives the manager an insight into which activities that need to be prioritised in order to complete the project in time.
There are different methods to prioritise the activities in the critical path. Among these is the so called "fast tracking" approach which means that more activities are performed in parallel with each other. Another method is "crashing the critical path" which means giving more resources to the critical activities and thereby reducing their durations.

As a conclusion CPM has the following benefits:

- Gives a graphical overview of the entire project
- Shows the sequence of the activities
- Helps calculate the total duration of the project
- Determines the critical activities
- Shows when and which activities that can be performed parallel to each other
- Reveals where resources are needed

Although CPM is an efficient way to manage projects it also has its limitations, for example when the duration of each activity is hard, if not impossible, to estimate. In these cases CPM does more harm than good since it most certainly will give a false view on the duration of the project.
[9, 10]

6.1.1.2 PERT

A method very similar to the Critical Path Method is *PERT* which stands for *Program (or Project) Evaluation and Review Technique*.
The difference between the two is that PERT takes another view on the estimated time of each activity in the project. It does not, as CPM, only rely on one number to decide the duration of each activity, but instead it can use three; the shortest possible time, the most likely time and the longest time. These times are used to calculate the expected times for every activity.
By doing this, PERT makes it easier to schedule a project and its activities without knowing beforehand the exact durations of them.
An example of a PERT diagram is shown in *Image 1*.
[9, 11]

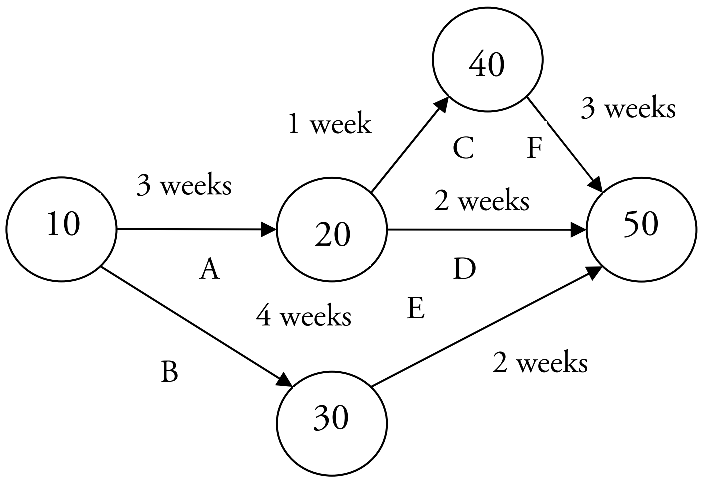


Image 1: A PERT network diagram for a project stretching over seven weeks with five milestones (10 through 50) and six activities (A through F). The critical path is A>C>F. Inspired by: [12]

6.1.1.3 RAM

RAM, which is short for *Responsibility Assignment Matrix*, is a way of mapping activities to resources, most commonly people. It is used to clarify the different roles within a project and what responsibilities each role has. The responsibilities can consist of tasks or deliverables, such as documents or program code. *Image 2* shows an example of a RAM.

RAM sometimes goes under the name *RACI* which stands for *Responsible Accountable Consulted Informed*. These four words represent four responsibilities within the model, and are explained below.

- **Responsible:** Persons who do the actual work.
- **Accountable:** Persons that make sure the work gets done. Approve the work done by the Responsible.
- **Consulted:** Persons who in some way provide input to the work, for example by being consulted.
- **Informed:** Persons who are informed about the progress of the work, often only when the work is finished.

RAM is vital when the people within a project are uncertain about what needs to be done and by whom. If not dealt with, people may think someone else is responsible for something which they themselves are responsible for. By using the RAM technique the responsibilities become clear for the ones involved.
[15]

	Person A	Person B	Person C	Person D	Person E
Activity 1	A	R		I	
Activity 2	C	A			
Activity 3	C		I	R	A
Activity 4		I	C		R

Image 2: An example of a RAM, with the responsibilities represented by the letters R = Responsible, A = Accountable, C = Consulted, I = Informed.

6.1.1.4 Gantt

The *Gantt chart* takes its name from Henry Gantt (1861–1919), who designed and distributed a version of the chart around 1910–1915.
A Gantt chart is easiest described as a system with time floating along a horizontal axis. This axis represents the entire duration of the managed project which is broken down into smaller time constants, for example days, weeks or months. Along a vertical axis, the different activities within the project are listed. These activities are represented in time by bars along the horizontal axis, which show when an activity is planned to take place and the duration of it. For an example of this type of Gantt chart, see *Image 3*.
While this simple version of a Gantt chart offers a good way to both manage and present a projects flow, it can be extended to show more information. For example a vertical line that show today's date can be added, different colouring of the bars can be used to show completion and which resources are needed for which activities. It is often also suitable to show dependencies between activities by linking them together with arrows. Milestones represented by some sort of indicator can also be added.

The main strength of the Gantt chart is that it gives the ability to present tasks in a way that is understandable to both managers and persons not familiar with project management.
[13, 14]

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
Activity 1	■	■						
Activity 2		▨	▨					
Activity 3				■	■	■	■	
Activity 4						▨	▨	▨

Image 3: An example of a simple Gantt chart without any extra information. It consists solely of a list of the activities and a time axis in months.

6.1.2 Choice of technique

A variety of factors had to be taken into account when deciding which project management technique to base the component on.
Some of these essential factors were:

- The company's current way of managing projects
- The user-friendliness of the techniques
- The possibility to realise the technique with software

On all three of these factors only one of the candidates stood out; the Gantt chart. First of all, Gantt charts were already in use at Flygprestanda at the start of the thesis. A solution based on the Gantt chart would therefore be a less heavy transition for the users.

Looking specifically at RAM, it is clear that it lacks information about the time and resources needed for a specific activity. Since this type of information is important for this thesis, RAM is not suitable.

When it comes to the user-friendliness of the techniques, the Gantt chart stands out even here. It is probably one of the most used, and therefore most recognisable, of the project management techniques. It is a technique which most people have come across at least once. As mentioned in chapter *6.1.1.4 Gantt* a Gantt chart is easy to understand even for a person not familiar with project management. It does not contain as much advanced calculations as for example CPM or PERT but can still contain a great deal of information.
Compared to the other techniques, the Gantt chart has a clearer time line, which makes it easier, with just a glance, to see where the project is at a certain point in time, as well as what was and is to come.
From a developer's point of view, the Gantt chart is likely to be the technique, compared to CPM and PERT, that takes less time to implement. RAM might be less time-consuming, but because of the reason discussed above it can be ruled out.

As the text above indicates, the Gantt chart was the technique chosen for this thesis.

6.2 Implementation

This chapter discusses the different ways to implement the project management component. It explains different programming languages that can be used if the component is developed from scratch, and different existing solutions that can be used if it is not.
The chapter is concluded by a discussion about whether the component should be developed from the ground up or not.

6.2.1 Programming languages

If the project management component should be developed from scratch, a suitable programming language needs to be used.
There are a variety of programming languages used in web development. Some work better than others for certain tasks and some are more difficult to master than others. This chapter gives a short explanation of the most common and best suited languages for this thesis, together with their advantages and disadvantages.

6.2.1.1 Flash

Flash is one of the most common multimedia platforms used on the Internet. It is a popular way of adding different kinds of animations and interactivity on a web site, namely advertisement, video players and lately even complete web applications. One good example of a successful site using Flash is *YouTube* (http://www.youtube.com) which uses Flash for its playback of videos among other things.
Most web browser, even on mobile devices such as mobile phones, are able to display Flash content by having a Flash player installed.

Flash is owned and maintained by the company *Adobe Systems*, who acquired the technology after the purchase of the developer Macromedia.

Flash uses so called *vector graphics*, which makes Flash content scalable to any size without losing the quality. This is because of vector graphics not using individual pixels to represent an image, instead it uses different types of lines, curves and shapes.

For the purpose of creating interactive applications, Flash contains a scripting language called *ActionScript*, which has a similar syntax to JavaScript (explained in chapter *6.2.1.2 JavaScript*). It is used to create buttons, text fields and other interactive components. In newer versions of Flash, version 9 and up, ActionScript 3.0 is included. This version of ActionScript is object oriented and makes it easier to write complex Flash applications, which has made complete web applications in Flash more common.

If using the right tools and developing simple applications, Flash does not require any particular programming skills. For advanced implementations however, there is a need for the developer to be able to write and understand Flash code.

Compared to other techniques for allowing interactive content on a web page, such as Java, Flash has some advantages. Most of the time Flash is both faster to load and initialise.

Among the downsides with using Flash, the closed nature of the technology can be mentioned. Flash is a proprietary format which means only the owner, in this case Adobe Systems, has the right to view the source code. This goes against many people's view of the Internet as a shared place where everyone has equal rights to view and edit information.
Another downside with Flash is its sometimes high demands on computer performance, compared to other techniques. If a computer that is to view Flash content lacks sufficient performance, the content will most certainly take up system resources and the content will play slowly.
[16, 17, 18]

6.2.1.2 JavaScript

JavaScript is a scripting language which can add interactive functionality to an otherwise static web page. In the simplest manner, JavaScript is often included in HTML pages to enable functions such as dynamic text, events listeners and validation of data. Lately, more advanced web functions have been constructed with the use of JavaScript. An interactive web application such as *Gmail* (http://www.gmail.com) is totally dependent upon JavaScript to function.
More specifically, JavaScript offers ways to manipulate the content on a web page, often based on input from the visitor. If the visitor for example clicks a button a panel can float down revealing the requested information.

JavaScript gets its name from the similarity in syntax to the Java programming language. Apart from this there is no other connection between the two, meaning JavaScript is not a scripting version of Java.

Since JavaScript code can be executed on the client side instead of the server side, the servers serving the web page do not need to perform any extra work. This makes JavaScript more responsive and improves the user experience. JavaScript code does not need to be compiled before it is executed, like for example Java code.

One of the most important things about JavaScript is that it is a free language and there is no need to purchase a license before implementing it.

A drawback with JavaScript is the potential of executing malicious code. Through security holes in the web browser possibilities of running JavaScript code, with

the user unaware of it, can arise. The code can collect password or similar private information or make it possible for other malicious code to install and run. These security concerns are addressed by only letting JavaScript code perform web-related actions, and only gather data concerning the same website from which the script is executed.
[19, 20]

6.2.1.3 GWT widget

A *GWT widget* is a type of user-interface object written in Java specifically to be used in a system that uses Google Web Toolkit as its framework.
When used in the system it is displayed as some sort of graphical element which displays information arranged in a certain way, which is often in some way editable for the user. In other words, a widget is used to give a single place where a set of given data can be viewed and/or edited.
Applications built with GWT often consist of several different widgets which together build up the entire interface.

There are a few official widgets available from Google which can be included in a system, either extended or in their simplest form. On the other hand, the largest amount of widgets available for GWT are developed by people not working for Google, and are almost exclusively distributed as free software. These widgets add all sorts of functionalities, from pie charts to weather forecasts.

A GWT widget is easy to include in a GWT system, much thanks to the object oriented nature of the Java programming language. But because of GWT being a new platform, it can be difficult to find a widget that suits ones needs.

6.2.2 Possible existing solutions to integrate

As mentioned earlier, the possibility of integrating an existing solution instead of developing one from the ground up was investigated. This chapter looks at the different solutions found. The solutions discussed where possible candidates to become the central part of the project management component.
The solutions are explained based on their presentation- and interaction capabilities, their ways to communicate with a database, their available licenses and the offered support.
There was no particular budget set during the thesis, the prices were instead discussed after a suitable solution had been found. There was however the requirement that the number of users should not be limited.

For pictures and demos of the solutions, please visit their respective websites. Because of their images being copyright protected, they are not included in this report.

6.2.2.1 FusionWidgets

FusionWidgets is a collection of Flash components developed by InfoSoft Global. The collection contains real-time gauges and self-updating charts for monitoring of data, as well as financial charts.
What made FusionWidgets interesting for this thesis is that it has the ability to create Gantt charts. These charts can have a variety of different layouts and are therefore easily adapted to the look of the web system. They can also be animated to please the eye of the user.
The FusionWidgets can retrieve data via an XML interface, which makes the database communication easier to implement.
Because of FusionWidgets being Flash based it also brings with it the advantages of the Flash technology, for example the short load times.

There are a variety of FusionWidget licenses available. All of them, except the OEM license, do not allow an application containing a FusionWidget to be sold to a third part. Since it is possible that the finished system can be sold to another company, only the OEM license is suitable. The OEM license contains free support and upgrades.

FusionWidgets lacks a vital part of the component, to be precise interaction capabilities that change the data presented, for example the ability to click and drag the bars in the Gantt chart to change the dates. Because of this, it was deemed inappropriate for the component.
[21]

6.2.2.2 AnyGantt

AnyGantt is very similar to FusionWidgets, particularly in the sense that it also is a Flash based solution. Like FusionWidgets, or any Flash component for that matter, it is easy to integrate with a variety of web languages, like ASP, PHP, JSP and HTML.
AnyGantt is developed by a company called *AnyChart*, which also develops maps and charts for other uses.
AnyGantt has similar licences to FusionWidgets, an OEM license is required to get the right to sell an application containing AnyGantt to a third part.
Regarding support AnyChart promises answer to e-mails within 24 hours on weekdays and 48 hours on weekends.

The main drawback with AnyGantt is the same as for FusionWidgets, it completely lacks interaction which changes the presented information.
[22]

6.2.2.3 jsGantt

The js in *jsGantt* stands for JavaScript which means that it is a JavaScript solution. Moreover it uses CSS to present the chart and XML to receive the data. The developers, Shlomy Gantz and Brian Twidt, are proud to announce that it does not require any images to be displayed.
Among the features provided by jsGantt are dependencies, task completion, milestones, resources and changing of the time axis.

jsGantt is completely free, but like FusionWidgets and AnyGantt it does not have the possibility to edit the data presented in the chart. This makes it unsuitable for the component.
Because of it being free it also lacks guaranteed support from the developers. [23]

6.2.2.4 EJS TreeGrid

EJS TreeGrid stands for *Editable JavaScript TreeGrid*, and has an exceptionally descriptive name. EJS TreeGrid, hereafter *TreeGrid*, is namely a component purely written in JavaScript which is able to create and display editable tables, grids and trees on a web page.
The component, which is developed by a company called *COQsoft*, has a large variety of functions, among which are server communication, copy and paste, exporting, searching and filters. One particularly interesting feature for this thesis is TreeGrid's capability to create interactive Gantt charts, i.e. Gantt charts which can be edited and saved to server.
TreeGrid comes with a rich API which makes it easier to communicate with the component both ways and to control its various functions. The appearance of the component can be changed via CSS code and by replacing images.
Since TreeGrid is JavaScript based it does not execute on the server side, but on the client side. This minimises the risk of overload on the server.
TreeGrid's presentation capabilities are similar to the ones offered by jsGantt. It can, apart from the obvious objects, represent dependencies, milestones and flags. It has built in functions for adding, editing and removing activities as well as different zoom levels for the time axis.
TreeGrid is added to a web page by a few lines of code, with which also the data and layout can be loaded.

There is a selection of licenses for TreeGrid available. The one interesting in this case was the one called *TreeGrid Grand + Extended API*, since it offers the rights to include TreeGrid in an unlimited amount of applications and distributions. It also includes the Extended API, which needed to be used to properly integrate the component into the web system. With the license comes support via e-mail. [24, 25]

6.2.3 Own or existing solution?

With the different programming languages and possible solutions in hand, the initial investigation during the thesis had to lead to a conclusion about which way to continue. Should the component be developed from scratch or should an existing component be used?

Building the component from the ground up means that the result will meet the functionality requirements better since you have complete control over what functionality to implement and how to implement it. The obvious downside with developing the component from scratch is that it takes significantly more time. There is also a risk that the finished code will contain errors, for which there lacks sufficient time to fix.

An existing solution might not fit the requirements perfectly, but has on the other hand been developed and tested during a longer time period, and by more people.

For the component to be usable to the people at Flygprestanda, it had to contain a certain amount of interaction capabilities, i.e. ways to edit the information presented by the component. This could be done by letting the user manually edit the information in fields, but it would not be particularly user-friendly. Instead, the desired way to interact with the component was to click, drag and move a representation of the data inside the component itself.

These types of interaction capabilities can be time consuming to develop, particularly since a great deal of testing, both of functionality and stability, is required.

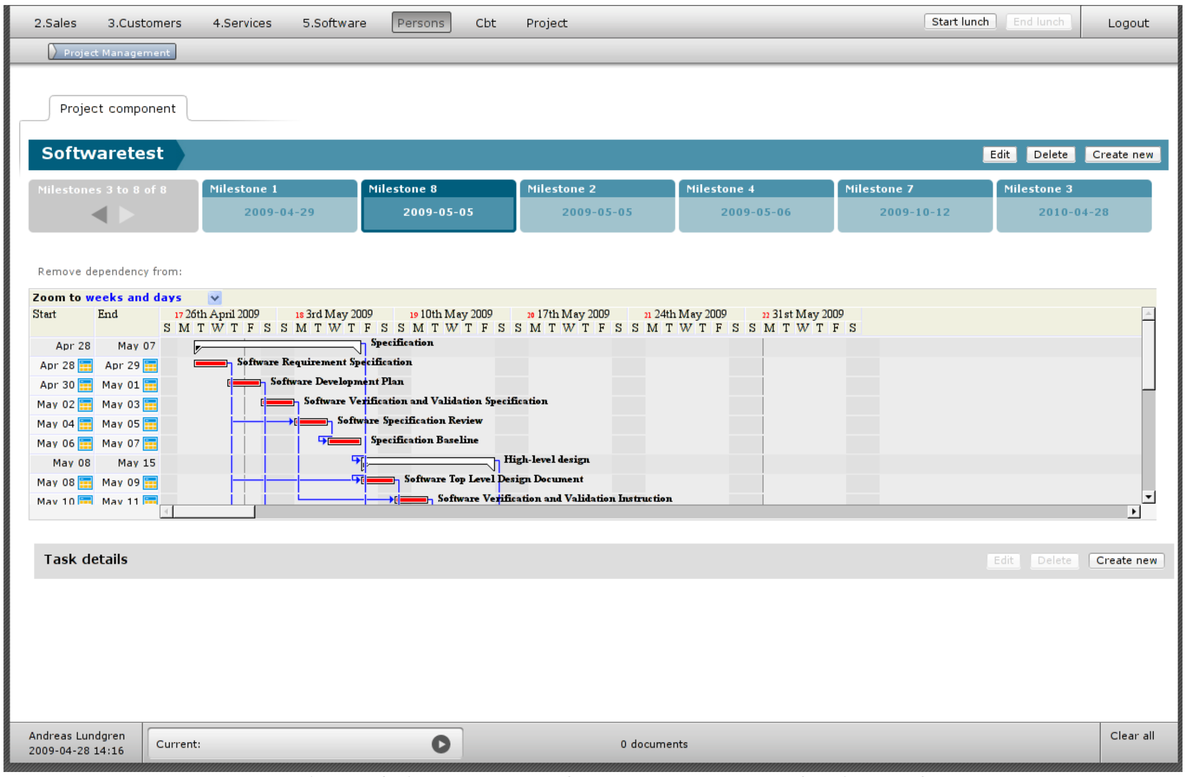
Because of this and the limited time available to conduct the thesis the decision fell upon integrating an existing solution instead of developing one from scratch. See next chapter, *7 Integration*, for a write-up of how this integration was carried out.

6.2.4 Choice of solution

TreeGrid distinguished itself from the other solutions by being the only one to offer interaction (ways to edit and save data presented), an extended API and good documentation. Because of this, TreeGrid was the solution chosen to integrate into the web system.

7 Integration

This chapter describes the work with adapting and integrating the different parts of the project management component with the web system. It only contains methods and code examples, in respect to Flygprestanda's company secrets.



Start	End	
Apr 28	May 07	Specification
Apr 28	Apr 29	Software Requirement Specification
Apr 30	May 01	Software Development Plan
May 02	May 03	Software Verification and Validation Specification
May 04	May 05	Software Specification Review
May 06	May 07	Specification Baseline
May 08	May 15	High-level design
May 08	May 09	Software Top Level Design Document
May 10	May 11	Software Verification and Validation Instruction

Image 4: Screenshot of the TreeGrid component inside the web systems GUI. Above the TreeGrid component is the milestone carousel, explained in chapter 7.2 GWT-YUI Carousel.

7.1 EJS TreeGrid

Integrating TreeGrid consisted of a number of separate steps described in order below. *Image 4* shows the finished integration of TreeGrid into the web system.

7.1.1 Adding to the interface

The first thing that needed doing to integrate TreeGrid with Flygprestanda's web system was to find a way to get it to show up in the graphical interface. Since TreeGrid is a JavaScript based component, and the GWT framework is written in Java, there was a demand for means to include JavaScript code into the Java code. GWT contains precisely this ability. By writing the JavaScript code to be integrated in a form of comments, the compiler knows that this code should be interpreted as native JavaScript code. The method used is called *JSNI*, which stands for *JavaScript Native Interface*.
This example shows how the JavaScript code is inserted into the Java code:

```
private native void overrideOnSaveFunction()
/*-{
   var thisGantt = this;

   $wnd.Grids.OnSave = function(grid, row, bool)
   {
      if (grid.HasChanges())
      {
         var changes = grid.GetChanges(row.ARow);
         grid.AcceptChanges(row.ARow);

         thisGantt.@se.flygp.websystem.project.client.GanttComponent::
             updateChanges(Ljava/lang/String;)(changes);
      }
   };
}-*/;
```

Notice the "/*-" before and the "-*/" after, as well as the word "native" in the method declaration. This tells the compiler that the code contained within is pure JavaScript. When calling the method "overrideOnSaveFunction" from anywhere in the outside Java code, the JavaScript within the method gets executed. This is a simple way of using handwritten JavaScript code, and has been crucial for this thesis.
There are a few possible drawbacks with using JSNI, which demands that the developer is extra careful. JSNI is only recommended by Google if there is no other way to achieve the same results. This since the code within the comments does not get optimised by the compiler, and a risk of memory leakage can arise.
What is also worth noticing is the code line which begins with "thisGantt.@". One this line a Java method is called by the JavaScript. "thisGantt" is a reference to the instance of the current class, and the code before the "::" is the path to the Java method called. The remaining code contains the name of the method, the parameter types and the actual parameter. This provides a way of communicating the other way, i.e. having JavaScript calling Java code.
JSNI has been used a great deal during this thesis, not only to add TreeGrid to the interface, but also to realise the communication between TreeGrid and the system. This is explained more in chapter *7.1.3 Database communication.*
[29]

The adding of the TreeGrid component to the web system is done using the following code:

```
private native void createChart(String data)
/*-{
   var DataIO = new $wnd.TDataIO();
   DataIO.Layout.Url = "GanttDef.xml";
   DataIO.Data.Data = data;
   var GanttGrid = $wnd.TreeGrid(DataIO, "gantt_container");
}-*/;
```

The "createChart" method is just like the example before a method with pure JavaScript within it. This code, according to TreeGrid principles, creates a variable "DataIO" which acts as the input and output object for the chart. This object receives a reference to the XML file which contains code that determines the layout of the chart, and gets loaded with the data sent as a parameter to the method. These steps are described in more detail below.
The chart itself is then created on the last line. The "$wnd" part is a reference to the JavaScript window object, and "gantt_container" tells in which CSS div tag the chart should be placed.
With this code the TreeGrid component can be placed virtually anywhere within the existing system.

To create a TreeGrid chart, it needs to be loaded with an XML formatted file or text which tells which functions the chart should contain and another which contains the actual data to be presented. This last one containing the data can be set anew at a later point, for example when the data is changed by the user outside the chart. This is used in the solution since it provides a good way of updating the chart so that it corresponds to the data stored in the database.

7.1.2 Removing and adding functions

By editing the layout XML discussed in the above chapter, you can remove, add or adapt functions to the resulting Gantt chart. This was done during this thesis. As a base for the layout XML a combination of two different XML documents provided by the developer COQsoft was used; one which contained a suitable way of interpreting the data XML and one which provided the layout itself.
The resulting XML document was in turn almost completely rewritten to only contain the absolutely necessary functions.

TreeGrid offers a great deal of functions for the Gantt chart. Most of these were not needed for this implementation. They therefore needed to be removed or edited. This could in some cases be done as easy as the example below shows:

```
<Toolbar Visible="0"/>
```

This line of code removes the toolbar of the Gantt chart. This toolbar contains various buttons for functions such as printing, adding tasks and display options. None of these functions were needed for this implementation and the entire toolbar was removed.

In other cases the removing of functions was not as straight-forward. Many times one change in the XML document affected something else in the Gantt chart, it therefore became a balance act to remove certain parts. One example of this was the instructions which told the chart how to interpret the data sent to it. They had to perfectly fit the structure of the data XML.

The layout XML document also contains code which tells what actions performed by the user in the Gantt chart that should be listened to, for example what should be executed when the user clicks on a certain task.
The following example shows the instruction which controls just this:

```
<Actions ClickCell="Focus AND StartEdit + taskClicked(this)"/>
```

The action “ClickCell” is predefined in the TreeGrid API and is performed every time the user clicks a cell within the chart. The instructions belonging to “ClickCell” tells what is supposed to be executed when this happens. The first two instructions, “Focus” and “StartEdit”, are also predefined in the API and make sure that the cell gets visually selected and editable. The added instruction “taskClicked(this)” calls a handwritten JavaScript function which in turn tells the rest of the system that a task has been clicked. This makes it possible to choose a task in the chart and have its information collected from the database and presented in a view outside of the chart.

7.1.3 Database communication

In order for the TreeGrid component to work together with the system, it needs to be able to both read and update data to the systems database.
This chapter only explains the principles of these processes, in an effort to make them sound less complicated than they actually are.

7.1.3.1 Reading from database

The Flygprestanda web system's database is a *MySQL* database. To communicate with this database an interface has been constructed by Flygprestanda as a way to read and update the database through Java code. Via this interface, calls can be made to receive object representations of the data stored in the database. These objects can be handled just like any other objects in Java, and if needed edited and updated back to the database via the interface.

When retrieving data to present in the Gantt chart, a call is made to a method in the database interface. This call contains parameters which decide what is to be fetch. The returned object can for example be a list of tasks for a specific project or a single milestone.
We concentrate specifically on the case when we want to retrieve the tasks belonging to a certain project, to display them in the Gantt chart.
When having received this list of tasks it can not be sent straight to the TreeGrid component, since the only way to feed data to it is via XML. We therefore need to somehow create an XML representation of the list. This is done with a solution called *JDOM*. JDOM provides a way of accessing, manipulating and outputting XML from Java code. In this case we are focusing on its outputting capabilities. With the help of JDOM we can for each objects in the list create an XML representation. All of these representations can be added to a single XML document, according to the layout of data decided by TreeGrid (exemplified below). The result is an XML document which follows given XML standards, and contains all of the tasks retrieved earlier from the database.
The created document can now be sent as a parameter to a JSNI method which updates the Gantt chart with the data.

The parts of the project management component which are outside of the Gantt chart, do not need to use this way of communicating with the database. Since they are written in Java, they can receive object representations of the data.

The layout of the XML document sent to the Gantt chart needs to be formatted in a way so that the chart can understand it. Below is an example of three tasks represented in XML format:

```
<I id="1820" dbid="1820" T="Preliminary Design Review" S="5/12/2009"
E="5/13/2009" C="75" D="1817$1821"/>
```

The "id" and "dbid" attributes uniquely identifies each task. The "T" attribute contains the name of the task, while "S" and "E" gives the start and end dates. The "C" attribute holds the completion of each task in percent and "D" contains the dependencies connected.
[30]

7.1.3.2 Updating to database

When the user makes a change of the data presented in the Gantt chart, this change needs to be updated to the database.
The updates are performed in basically the same way as when reading from the database, except it is conducted the other way round.

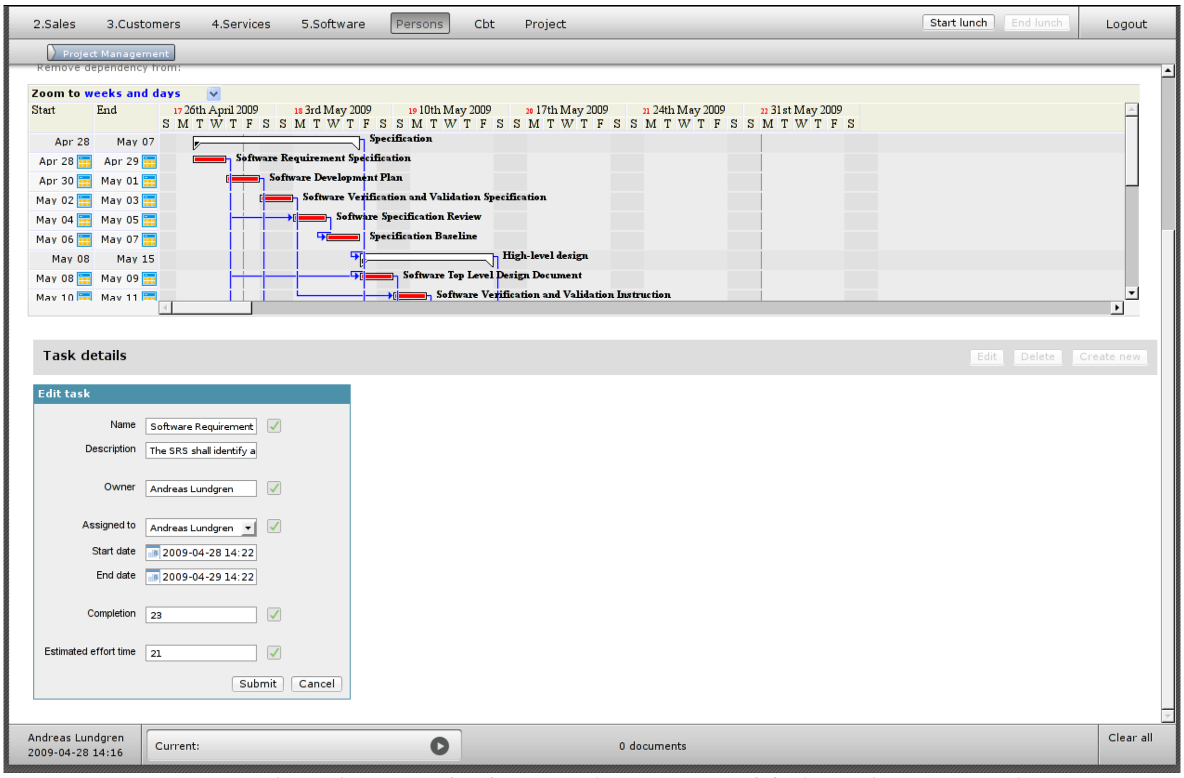
The changes done in TreeGrid are caught with the help of actions similar to the one concerning clicking cells, as described in chapter *7.1.3.1 Reading from database*. As soon as a change is made, for example if a user moves the end date of a bar representing a task, TreeGrid fires an event which consist of a call containing the change made in XML format, like below:

```
<I id="1811$1816" Changed='1'><U N='E' V="1241733600000"/></I>
```

The XML above contains information about which task was changed; "id", the value changed; "N" and the new value; "V".
This event is caught by a handwritten JavaScript function which in turn calls a Java method with the XML as a parameter. The XML received by the Java method gets interpreted by code using JDOM. The code translates the XML to an object which is analysed. The result of this analyse becomes the change that is to be uploaded to the database via the database interface. This is true for all changes made within the Gantt chart, i.e. changing of dates and dependencies.

7.1.4 Functions outside EJS TreeGrid

The Gantt chart provides ways to schedule tasks and add dependencies between them. Other functions, such as viewing all information about a task and editing the name and other attributes, are placed outside the TreeGrid component. This part can be called the show and edit panel and is graphically situated below the Gantt chart, as shown in *Image 5*. Every action performed in the Gantt chart is notified to this panel and the information within it is updated. It also works the other way round when editing a task in the panel; when the changes are saved, the Gantt chart is update to reflect these changes.



Start	End	
Apr 28	May 07	Specification
Apr 28	Apr 29	Software Requirement Specification
Apr 30	May 01	Software Development Plan
May 02	May 03	Software Verification and Validation Specification
May 04	May 05	Software Specification Review
May 06	May 07	Specification Baseline
May 08	May 15	High-level design
May 08	May 09	Software Top Level Design Document
May 10	May 11	Software Verification and Validation Instruction

Name	Software Requirement
Description	The SRS shall identify a
Owner	Andreas Lundgren
Assigned to	Andreas Lundgren
Start date	2009-04-28 14:22
End date	2009-04-29 14:22
Completion	23
Estimated effort time	21

Andreas Lundgren
2009-04-28 14:16

Image 5: *The "show and edit panel" is situated below the Gantt chart.*

7.2 GWT-YUI-Carousel

The work with the TreeGrid component was finished ahead of schedule, and therefore the thesis became extended to cover a milestone implementation.
As described in chapter *5.2 Database* using a milestone is a way of grouping tasks together within a project. The system needed a way of presenting the milestone for a project and let the user choose which milestone to view.
It was decided that this should be implemented by using a carousel placed above the Gantt chart. This carousel should contain representations of all the milestones and give the ability to scroll between them and select a milestone.

After a short investigation into the possibilities of creating this component, the *GWT-YUI-Carousel*, hereafter *Carousel*, was found. The component is a GWT widget and the *YUI* part of the name stands for *Yahoo User Interface*, which means that the component was originally developed for use in the *Yahoo User Interface library*. This library consists of different utilities and controls, written in JavaScript and is used when building interactive web applications.
The person responsible for turning it into a GWT widget is Nicolas Hoppenot, but the original Carousel component was developed by Bill Scott.
By using Carousel you can create interactive carousels which can contain such objects as pictures or just plain HTML code. These Carousels can then be added to a web page.
[31, 32, 33]

7.2.1 Adapt to system

The adding of Carousel to the interface of the web system was easy, since it already was a GWT widget. It was just a matter of including its API and creating an object of it.
The step that posed a bigger challenge was to make it behave the desired way. There were times when the carousel could scroll outside the boundaries of the number of milestones. Making the milestones selectable also took time since Carousel did not have any initial support for this.

The selection of a milestone works in almost the same way as when selecting a task in the Gantt chart. When the user clicks a milestone an action listener notifies a JavaScript function which executes. This function in turn calls a Java method and the remaining system now knows a milestone has been clicked. The JavaScript function also changes the appearance of the selected milestone by replacing its CSS class name.

7.2.2 Database communication
Loading the carousel with data from the database was only a matter of using the database interface to retrieve the milestones, and then creating representations of these. Each milestone is represented by a snippet of HTML code which contains div tags that are easy to style with CSS.
The snippets are added to the carousel by available methods in the Carousel API.

7.2.3 Functions outside GWT-YUI-Carousel
The carousel only presents and makes milestones selectable. To be able to edit the properties of a milestone an edit panel is implemented.
This panel provides a way for the user to create new milestones and edit existing ones. The panel is placed above the carousel and is only visible when the user chooses to create or edit a milestone.

A function added at the very end of the thesis is the possibility to choose a project template when creating a new milestone. By selecting for example the software template, tasks contained in a software project are automatically created in the database and added to the new milestone. These standard tasks are for instance of the type "Software Requirement Specification". The standard dependencies between the tasks are also added.
By selecting a template, there is no need to manually add all the standard tasks to a milestone. Without a template selected, the created milestone contains no tasks.

8 Problems

There were many problems, most of them small, encountered during the implementation of the project management component. This chapter discusses the most significant ones.
The minor issues faced consisted of everything between conflicting date standards to delayed asynchronous database queries. These issues are not discussed here but are documented in a work log for future reference.

A great deal of the problems that cropped up during the thesis concerned the TreeGrid component. The problems were mostly of graphical nature or had something to do with the communication with the TreeGrid component.
The first major problem encountered was that TreeGrid, while inside the graphical interface of the system, showed errors such as broken lines and missing elements. As can be seen in the *Appendix*, some of these errors reappeared towards the end, when there was no time left to fix them.
The graphical errors claimed a great deal of time in the beginning of the thesis, despite the help from the graphic designers at Flygprestanda. The problem arose because of conflicts in the CSS code for the system and the component. Even with the problem found, the solution was not obvious.
Similar problems regarding TreeGrid's and the web systems JavaScript code also emerged, but were fixed by a programmer at Flygprestanda by rewriting small parts of the web system. Instead, what took time was to find the reason to the malfunctions.

The part that posed the biggest challenge during the integration was to get the TreeGrid component to communicate with the database, mainly updating changes. TreeGrid offers some ways to achieve this, but none of them were suitable, largely because they required a response from the database in return of an update.
Instead the communication had to be realised by listening to and catching events fired by TreeGrid when an edit took place, and updating to the database based on these. This is explained in more detail in chapter *7.1.3.2 Updating to database*.

9 Conclusions

The final component developed and integrated fulfils the specified goals. It gives the users of Flygprestanda's web system a suitable way to view, edit, delete and schedule tasks within projects conducted at the company. It has been approved by Flygprestanda, and is, with future modifications (see chapter *10 Future development*), going to be a part of the web system in the near future.
The project management technique chosen during the initial investigation, the Gantt chart, turned out to be a good technique on which to base the component. The choice of the Gantt chart also led to the finding of the suitable TreeGrid component, which has been both a pleasure and a nuisance to work with.
The Carousel incorporated towards the end of the thesis also posed a few problems along the way, but the final implementation works very well.

Working with GWT has been a delight, because of Java being a familiar programming language. It was also enlightening to discover and utilise the possibilities to include existing JavaScript libraries and my own written JavaScript code.
The existing system developed at Flygprestanda was hard to understand in the beginning, but with much work and a great deal of help from programmers at Flygprestanda, the code soon became comprehensible. Many clever functions and interfaces made it easy to communicate with other parts of the system, in particular the database.

The project model initially intended to be followed during the thesis, became neglected. No prototypes as such were developed, instead Flygprestanda represented by Vadim Feldman continuously followed the progress. As mentioned before, no specific requirement specification was used. This could have turned out to be a mistake since the risk of misunderstanding increases. Instead it proved to be a good way to develop things quickly, much thanks to the regular communication with Flygprestanda.

When comparing the time plan constructed at the start of the thesis with the actual work conducted, you can see several differences. First of all the plan of the actual work is a great deal more detailed. This is due to the difficulty of identifying the different activities at the start of the thesis. The initial plan was based on speculations on what needed doing, how it should be divided and how much time it would take.
The most obvious difference between the initial and final plan is that the work with the Gantt component took about half the time. It was originally thought that it would take up the entire thesis. Thanks to a successful investigation, the work with the component could be started earlier than planned. This work was then also carried out quicker than expected. Because of this, the contents of the thesis needed to be extended to fill the designated time. As mentioned, the work

with the carousel component begun, divided in the same way as the work with the Gantt chart, i.e. first investigation then implementation.
Another clear difference is that the original plan states that the work is divided into two steps, with a prototype delivered for each of these steps, followed by testing and validation. Since no specific prototypes were delivered, this did not become the case.

When it comes to the writing of this report, it was meant to be undertaken during the Fridays of each week. Because of self-discipline not living up to this expectation, most of the work with the report was schedule for three weeks at home after the weeks at Flygprestanda's office. These weeks, for the same above reason, soon became two weeks. Eventually the work with the report was intensified and it was finished in time.

All in all, both the student and Flygprestanda are satisfied with the conducted work and the outcome of this thesis.

10 Future development

As with almost any implementation of a software component, the work conducted during this thesis leaves some room for future improvements, both concerning functions and efficiency. These improvements exist mostly due to insufficient time to implement them.

A major addition to the developed component would be to add support for calculations. These calculations would concern for example effort times for each task and would be calculated by the application. The current implementation relies on the user calculating the time by him or herself. The addition of automatic calculations would increase the usability and effectiveness of the component.

Certain parts of the component are static, that is to say they use fixed value to identify data. This is true in the case of the chosen project, it is hard coded in the current implementation. The reason to this is that the part where selecting the project takes place will be placed in another part of the system. The static values also made it easier to test the component during development.
Making the whole component dynamic is an improvement which needs to be carried out to make the project management part of the system fully usable.

Another possible improvement is the use of different levels of information for different users. When for example a manager uses the component he or she will be able to view all the employees' charts and edit them. When a single employee logs in, he or she can only see the tasks concerning him or her, and perhaps only edit some of these values. This function can be realised by identifying users with the help of the account they logged in with. This ability exists within the system as a whole, and it is only a matter of using this for the project management component.

The look and feel of a system is vital for the usability, and at the moment all parts except the Gantt chart itself is designed to fit the final system. Even the Gantt chart was originally meant to be styled accordingly, but because of other commitments for the designers at Flygprestanda, there was no time to do this.

When it comes to the implementation of the relation between milestones, project and tasks, a possible improvement would be to use some kind of root for all the tasks belonging to a certain milestone. The current implementation uses an attribute on each task which tells which milestone it belongs to. This means, to get the tasks for a certain milestone, all the tasks for the project in which the milestone resides needs to be checked. This implementation can be inefficient if there are many tasks within a project.

11 Terminology

Term	Explanation
AJAX	Short for asynchronous JavaScript and XML. A collection of different techniques with which a developer can build interactive web applications. [5]
API	An API, Application Programming Interface, is a library of routines, data structures and classes used to support the building of applications. [28]
CSS	Cascading Style Sheet is a language used to describe the presentation of a document written in a markup language, for example HTML. [27]
HTML	HTML stands for HyperText Markup Language and is the most common language used to construct web pages. [26]
Java	An object oriented and platform independent programming language developed by Sun Microsystems. [6]
Open source	A type of license which gives permission to others, apart from the developers, to view and in some cases edit a software products source code. [7]
XML	Stands for Extensible Markup Language and is a specification for creating custom markup languages. XML sets a standard for sharing structured data between different systems, especially over the Internet. [8]

12 References

[1] Flygprestanda AB: (2009). *Flygprestanda AB – Performance Engineering*. Retrieved May 7, 2009 from: http://www.flygp.se/index.shtml, under menu "Products"

[2] Hoover's, Inc: (2009). *Flygprestanda Ab – Company Overview – Hoover's.* Retrieved May 7, 2009 from: http://www.hoovers.com/free/co/dnb_factsheet.xhtml?HD=yhcyfysrs&src=global&TYPE=InDepth

[3] Best Price Computers: *Rapid Application Development Defined and Exlained.* Retrieved May 14, 2009 from: http://www.bestpricecomputers.co.uk/glossary/rapid-application-development.htm

[4] Google Code: (2009). *Product Overview – Google Web Toolkit.* Retrieved May 15, 2009 from: http://code.google.com/intl/sv-SE/webtoolkit/overview.html

[5] w3schools.com: (1999 - 2009). *AJAX Introduction*. Retrieved May 27, 2009 from: http://www.w3schools.com/Ajax/ajax_intro.asp

[6] About.com: (2009). *What is Java?*. Retrieved May 27, 2009 from: http://java.about.com/od/gettingstarted/a/whatisjava.htm

[7] Opensource.org: (July 7, 2006). *The Open Source Definition.* Retrieved May 27, 2009 from: http://www.opensource.org/docs/osd

[8] w3schools.com: (1999 - 2009). *XML Introduction – What is XML?*. Retrieved May 27, 2009 from: http://www.w3schools.com/xml/xml_whatis.asp

[9] Mind Tools: (1995-2009). *Critical Path Analysis and PERT*. Retrieved May 20, 2009 from: http://www.mindtools.com/critpath.html

[10] NetMBA: (2002-2007). *CPM – Critical Path Method*. Retrieved May 20, 2009 from: http://www.netmba.com/operations/project/cpm/

[11] NetMBA: (2002-2007). *PERT Chart*. Retrieved May 20, 2009 from: http://www.netmba.com/operations/project/pert/

[12] NetMBA: (2002-2007). *Image: pert.gif.* Retrieved June 3, 2009 from: http://www.netmba.com/images/operations/project/pert/pert.gif

[13] Ganttcharts: *Gantt History*. Retrieved May 27, 2009 from: http://www.ganttchart.com/history.html

[14] NetMBA: (2002-2007). *Gantt Chart*. Retrieved May 20, 2009 from: http://www.netmba.com/operations/project/gantt/

[15] Mind Tools: (1995-2009). *The Responsibility Assignment Matrix (RAM)*. Retrieved May 20, 2009 from: http://www.mindtools.com/pages/article/newPPM_RAM.htm

[16] Adobe Systems Incorporated: (2009). *ActionScript 3.0 overview*. Retrieved May 27, 2009 from: http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html

[17] w3schools.com: (1999-2009). *Flash Introduction*. Retrieved May 21, 2009 from: http://www.w3schools.com/Flash/flash_intro.asp

[18] Adobe Systems Incorporated: (2009). *Bitmap vs. Vector-based graphics*. Retrieved May 27, 2009 from: http://www.adobe.com/education/webtech/CS2/unit_graphics1/gb_bitmap_id.htm

[19] w3schools.com: (1999-2009). *JavaScript Introduction*. Retrieved May 21, 2009 from: http://www.w3schools.com/JS/js_intro.asp

[20] Developer Shed: (October 4, 2004). *JavaScript Security*. Retrieved May 27, 2009 from: http://www.devarticles.com/c/a/JavaScript/JavaScript-Security/

[21] InfoSoft Global: *FusionWidgets v3 – Interactive flash charts and gauges for dashboards and real-time monitors*. Retrieved May 21, 2009 from: http://www.fusioncharts.com/widgets/Default.asp

[22] AnyChart: (2009). *AnyChart – Cross-Platform Flash Charting Solutions For Your Project*. Retrieved May 21, 2009 from: http://anychart.com/products/anygantt/overview/

[23] jsGantt: *FREE JavaScript gantt – JSGantt HTML and CSS only*. Retrieved May 21, 2009 from: http://www.jsgantt.com/

[24] COQSoft: *EJS TreeGrid Description*. Retrieved May 22, 2009 from: http://treegrid.com/treegrid/www/#Description.html

[25] COQSoft: *EJS TreeGrid Prices*. Retrieved May 22, 2009 from: http://treegrid.com/treegrid/www/#Prices.html

[26] w3schools.com: (1999 - 2009). *Introduction to HTML*. Retrieved May 27, 2009 from: http://www.w3schools.com/htmL/html_intro.asp

[27] w3schools.com: (1999 - 2009). *CSS Introduction*. Retrieved May 27, 2009 from: http://www.w3schools.com/css/css_intro.asp

[28] PCMAG.com (1996 - 2009). *API Definition from PC Magazine Encyclopedia*. Retrieved May 27, 2009 from: http://www.pcmag.com/encyclopedia_term/0,2542,t=API&i=37856,00.asp

[29] Google Code: (2009). *Coding Basics – Google Web Toolkit*. Retrieved May 24, 2009 from: http://code.google.com/intl/sv-SE/webtoolkit/doc/1.6/DevGuideCodingBasics.html#DevGuideJavaScriptNativeInterface

[30] JDOM: *JDOM.* Retrieved May 24, 2009 from: http://www.jdom.org/

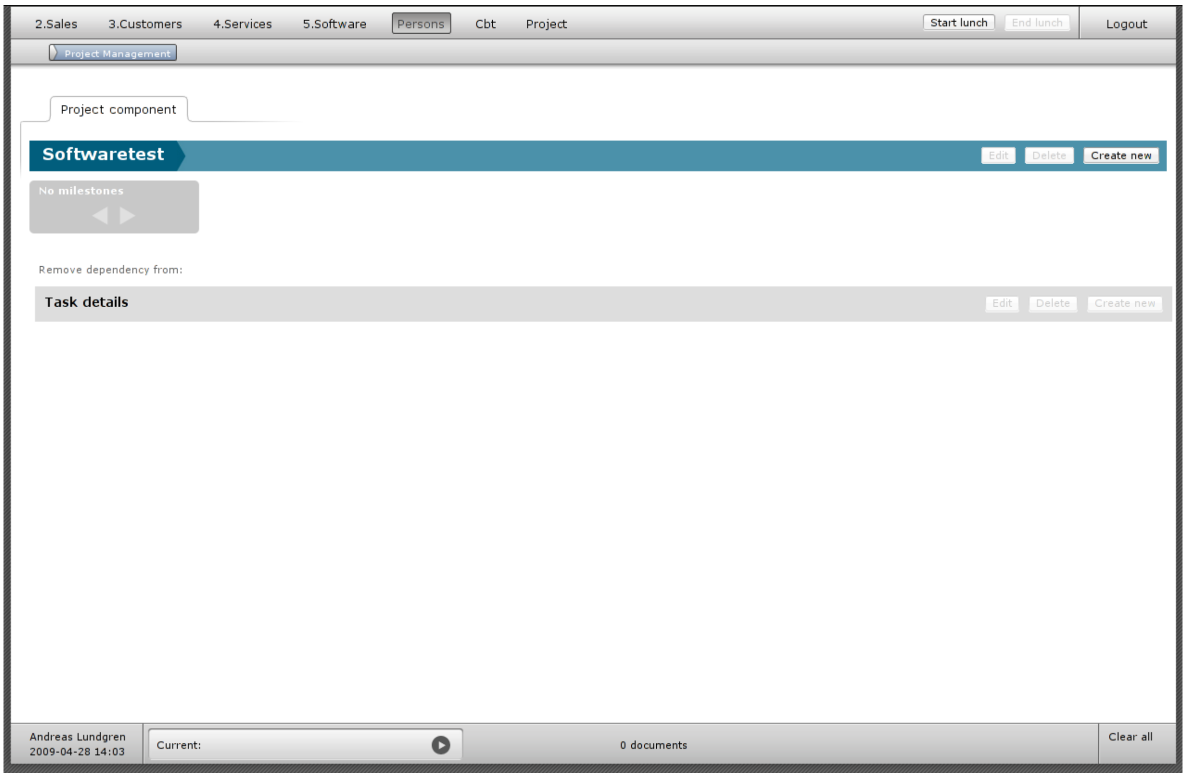
[31] Yahoo Developer Network: (2009). *The Yahoo! User Interface Library (YUI).* Retrieved May 24, 2009 from: http://developer.yahoo.com/yui/

[32] Google Code: (2009). *gwt-yui-carousel.* Retrieved May 24, 2009 from: http://code.google.com/p/gwt-yui-carousel/

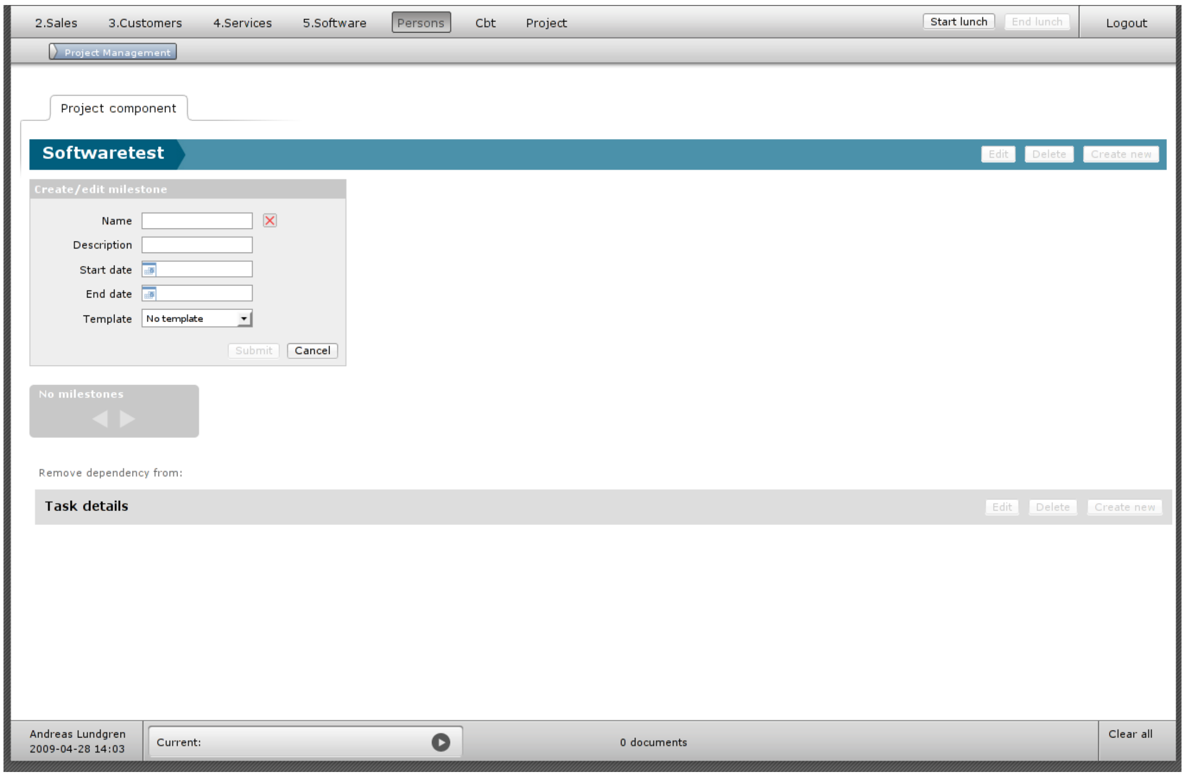
[33] Carousel Component: (October 21, 2008). *Carousel Component 1.0 – Documentation*. Retrieved May 24, 2009 from: http://billwscott.com/carousel/

Appendix

Screenshots



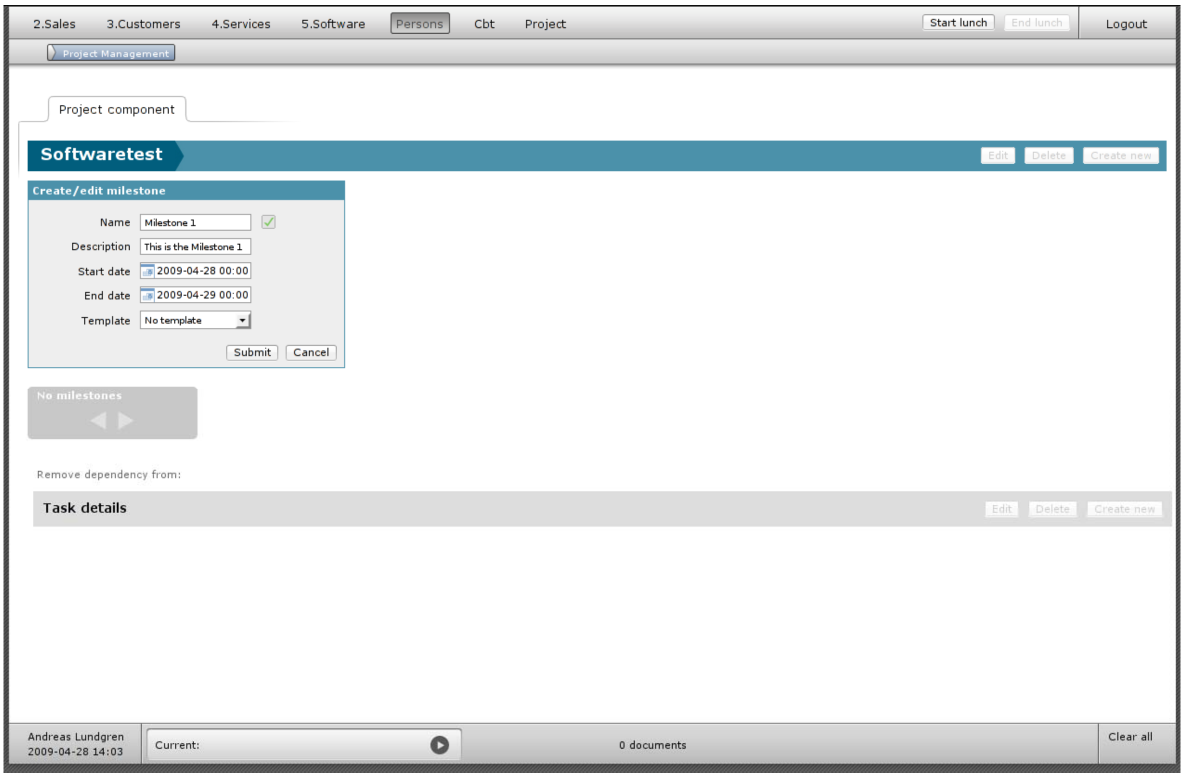
Screenshot 1: The standard view, with no milestones created. Viewing the project “Softwaretest”.



Name	
Description	
Start date	
End date	
Template	No template

Andreas Lundgren
2009-04-28 14:03
Current:
0 documents

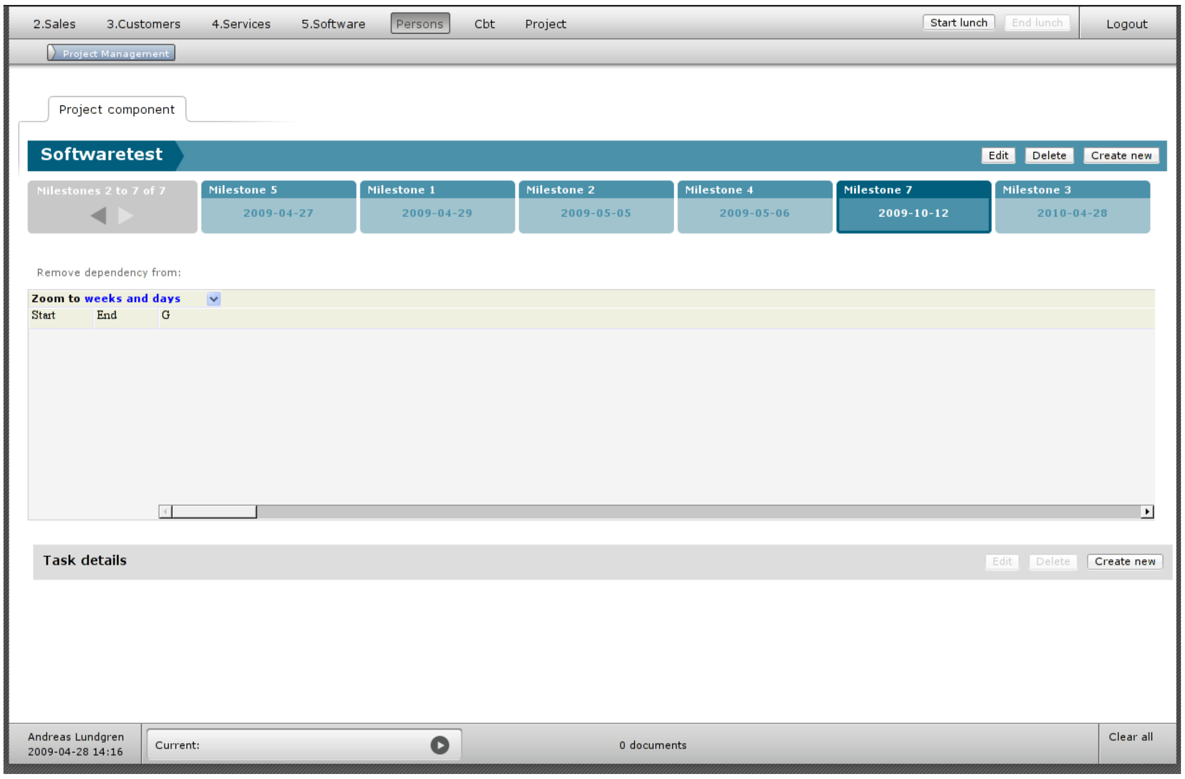
Screenshot 2: Creating new milestone. All fields are empty, and therefore not correctly validated.



Name	Milestone 1
Description	This is the Milestone 1
Start date	2009-04-28 00:00
End date	2009-04-29 00:00
Template	No template

Andreas Lundgren
2009-04-28 14:03
Current:
0 documents

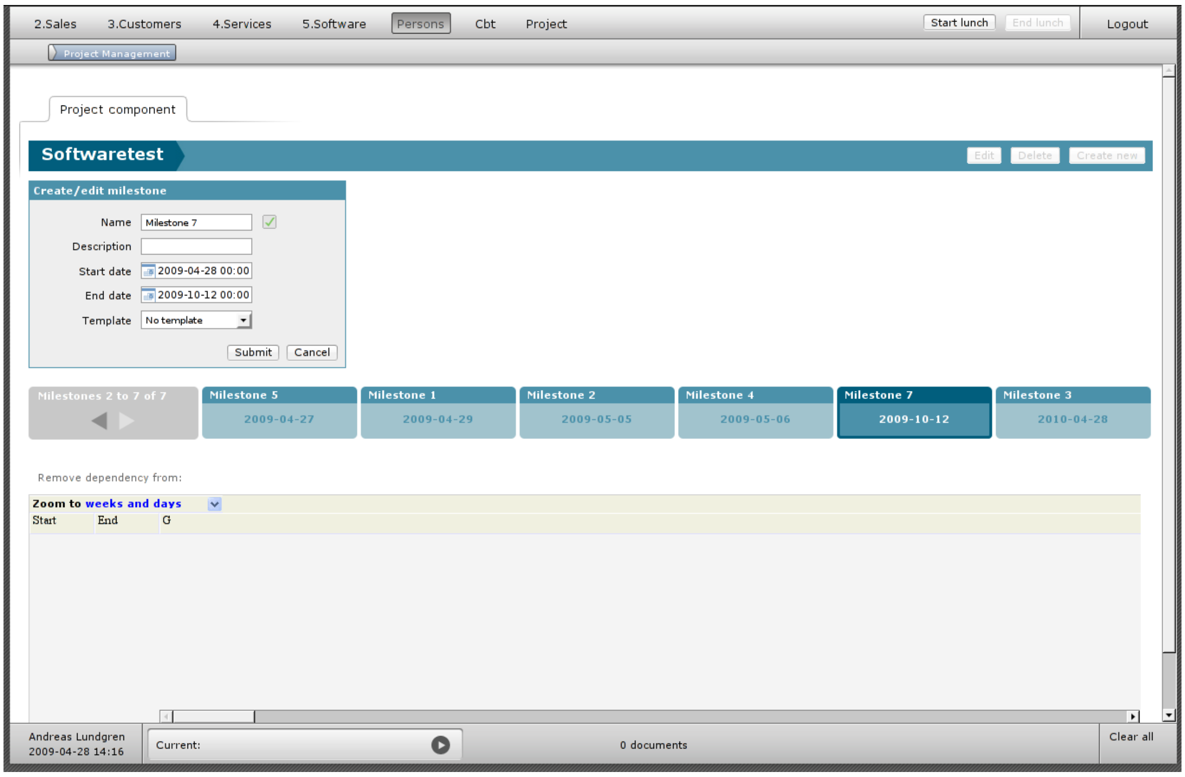
Screenshot 3: Creating new milestone. Fields are correctly validated.



Milestone 5	Milestone 1	Milestone 2	Milestone 4	Milestone 7	Milestone 3
2009-04-27	2009-04-29	2009-05-05	2009-05-06	2009-10-12	2010-04-28

Andreas Lundgren
2009-04-28 14:16
Current:
0 documents

Screenshot 4: Milestone carousel showing six of seven created milestones, sorted by end date. Selected Milestone 7 contains no tasks.

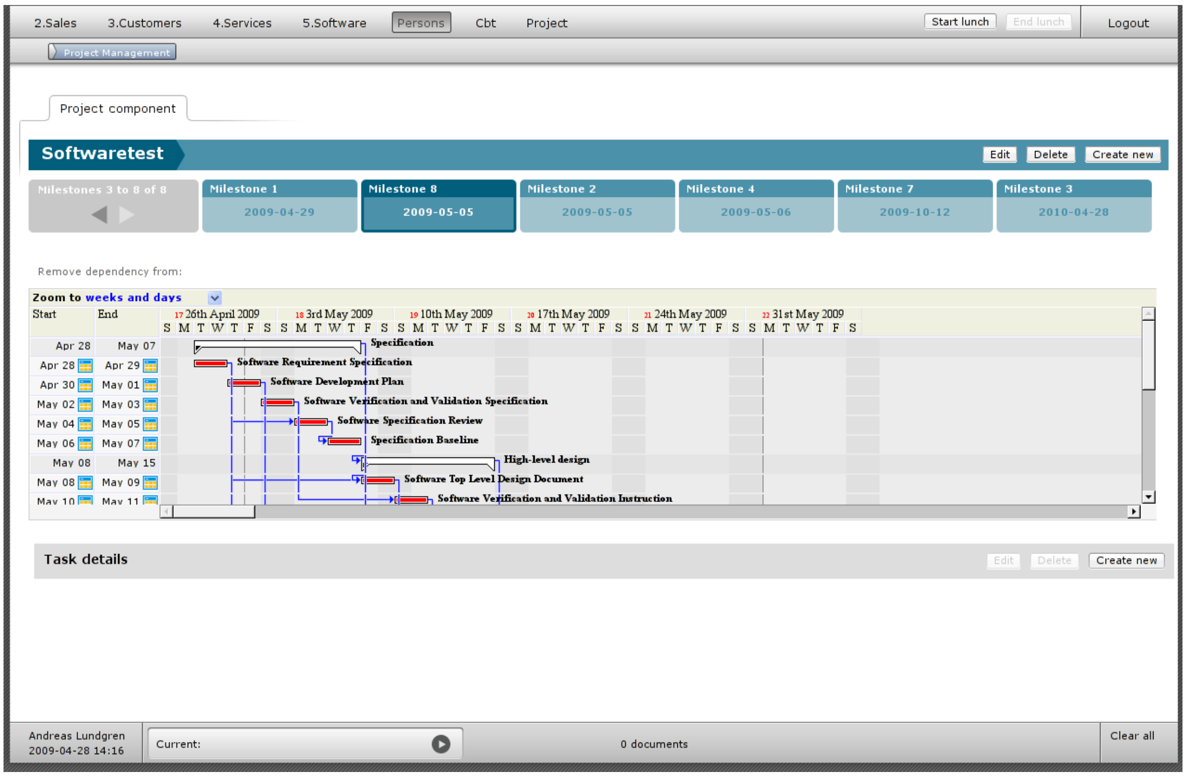


Name	Milestone 7
Description	
Start date	2009-04-28 00:00
End date	2009-10-12 00:00
Template	No template

Milestone 5	Milestone 1	Milestone 2	Milestone 4	Milestone 7	Milestone 3
2009-04-27	2009-04-29	2009-05-05	2009-05-06	2009-10-12	2010-04-28

Andreas Lundgren
2009-04-28 14:16
Current:
0 documents

Screenshot 5: Editing Milestone 7.


2.Sales 3.Customers 4.Services 5.Software Persons Cbt Project Start lunch End lunch Logout

Project Management

Project component

Softwaretest Edit Delete Create new

Milestones 3 to 8 of 8	Milestone 1	Milestone 8	Milestone 2	Milestone 4	Milestone 7	Milestone 3
	2009-04-29	2009-05-05	2009-05-05	2009-05-06	2009-10-12	2010-04-28

Remove dependency from:

Zoom to weeks and days

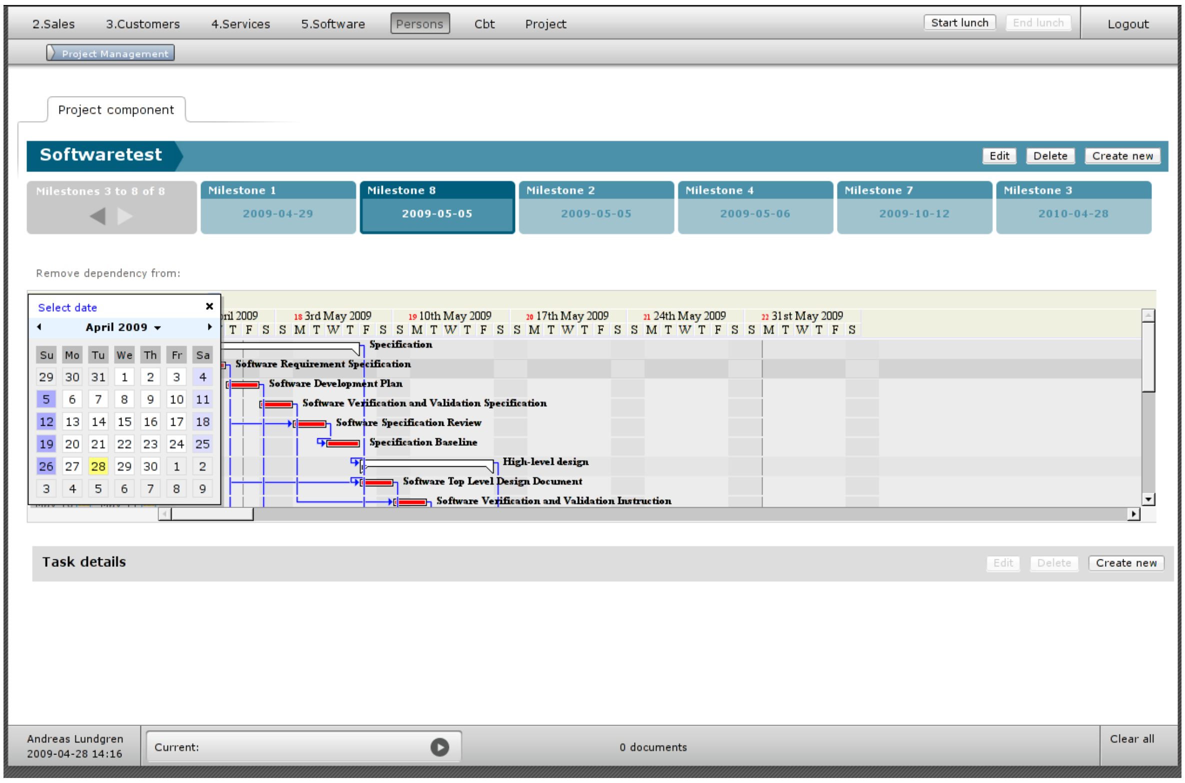
Start	End	Task
Apr 28	May 07	Specification
Apr 28	Apr 29	Software Requirement Specification
Apr 30	May 01	Software Development Plan
May 02	May 03	Software Verification and Validation Specification
May 04	May 05	Software Specification Review
May 06	May 07	Specification Baseline
May 08	May 15	High-level design
May 08	May 09	Software Top Level Design Document
May 10	May 11	Software Verification and Validation Instruction

17 26th April 2009 18 3rd May 2009 19 10th May 2009 20 17th May 2009 21 24th May 2009 22 31st May 2009

Task details Edit Delete Create new

Andreas Lundgren 2009-04-28 14:16 Current: 0 documents Clear all

Screenshot 6: Milestone 8 selected containing several tasks.


2.Sales 3.Customers 4.Services 5.Software Persons Cbt Project Start lunch End lunch Logout

Project Management

Project component

Softwaretest Edit Delete Create new

Milestones 3 to 8 of 8	Milestone 1	Milestone 8	Milestone 2	Milestone 4	Milestone 7	Milestone 3
	2009-04-29	2009-05-05	2009-05-05	2009-05-06	2009-10-12	2010-04-28

Remove dependency from:

Select date ×

April 2009

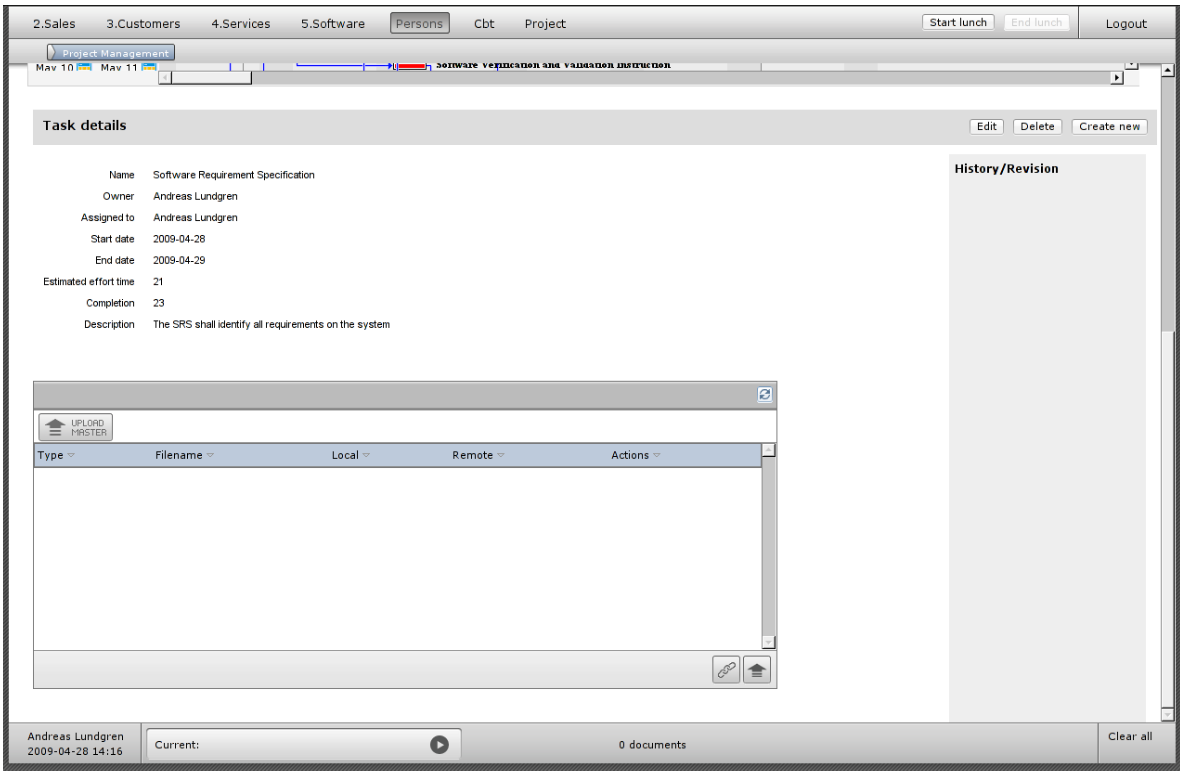
Su	Mo	Tu	We	Th	Fr	Sa
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Specification
Software Requirement Specification
Software Development Plan
Software Verification and Validation Specification
Software Specification Review
Specification Baseline
High-level design
Software Top Level Design Document
Software Verification and Validation Instruction

Task details Edit Delete Create new

Andreas Lundgren 2009-04-28 14:16 Current: 0 documents Clear all

Screenshot 7: Showing built-in calendar, where dates can be selected.

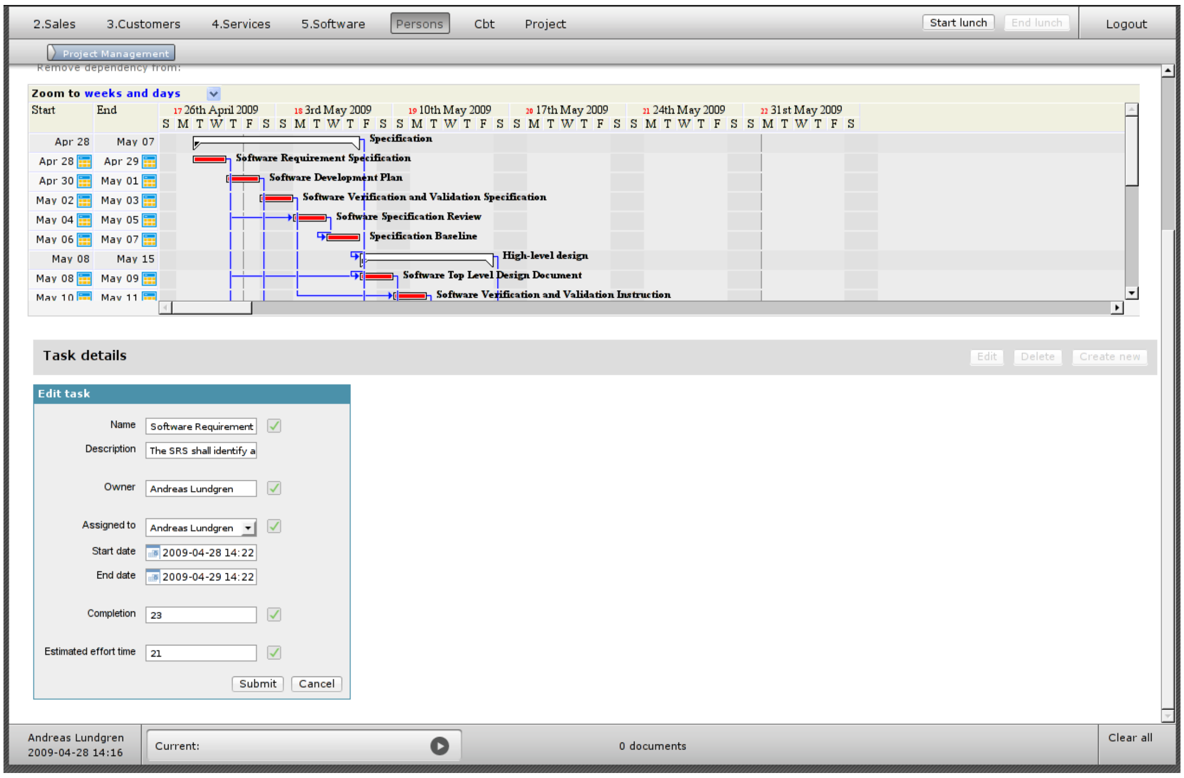


Name	Software Requirement Specification
Owner	Andreas Lundgren
Assigned to	Andreas Lundgren
Start date	2009-04-28
End date	2009-04-29
Estimated effort time	21
Completion	23
Description	The SRS shall identify all requirements on the system

Type	Filename	Local	Remote	Actions

Andreas Lundgren
2009-04-28 14:16
Current:
0 documents

Screenshot 8: Viewing details for a selected task.

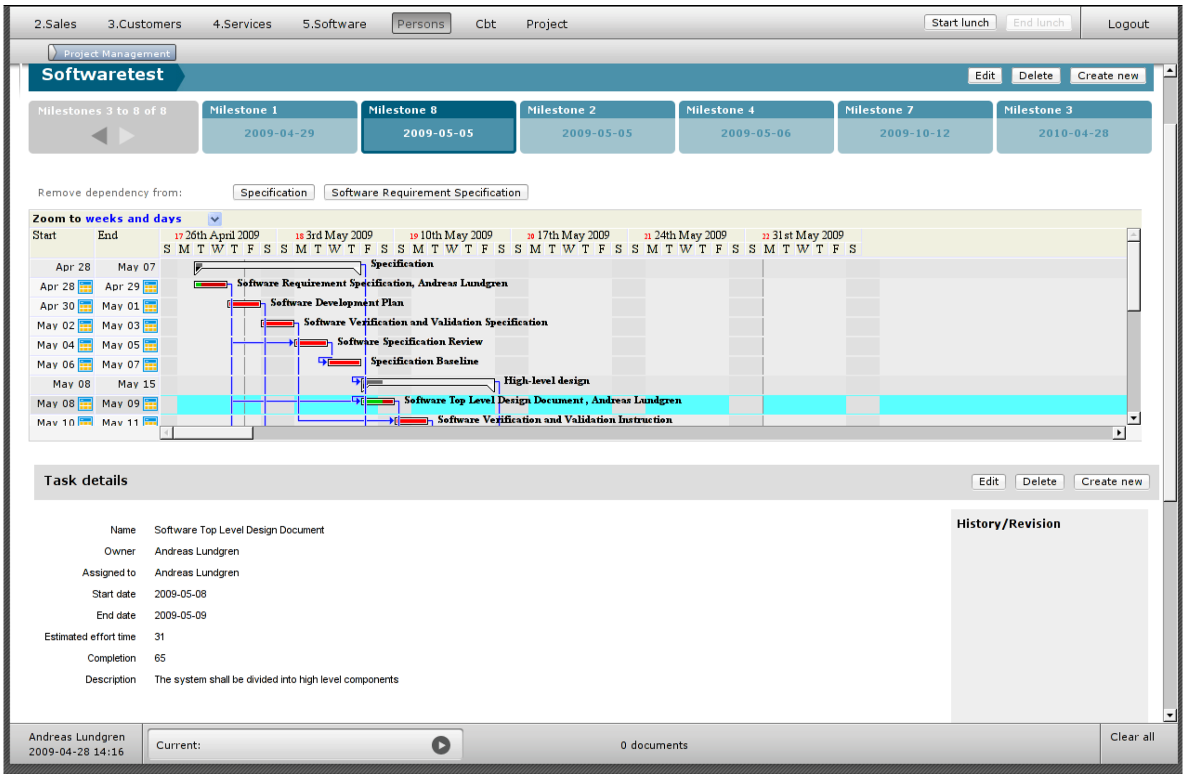


Start	End	
Apr 28	May 07	Specification
Apr 28	Apr 29	Software Requirement Specification
Apr 30	May 01	Software Development Plan
May 02	May 03	Software Verification and Validation Specification
May 04	May 05	Software Specification Review
May 06	May 07	Specification Baseline
May 08	May 15	High-level design
May 08	May 09	Software Top Level Design Document
May 10	May 11	Software Verification and Validation Instruction

Name	Software Requirement
Description	The SRS shall identify a
Owner	Andreas Lundgren
Assigned to	Andreas Lundgren
Start date	2009-04-28 14:22
End date	2009-04-29 14:22
Completion	23
Estimated effort time	21

Andreas Lundgren
2009-04-28 14:16
Current:
0 documents

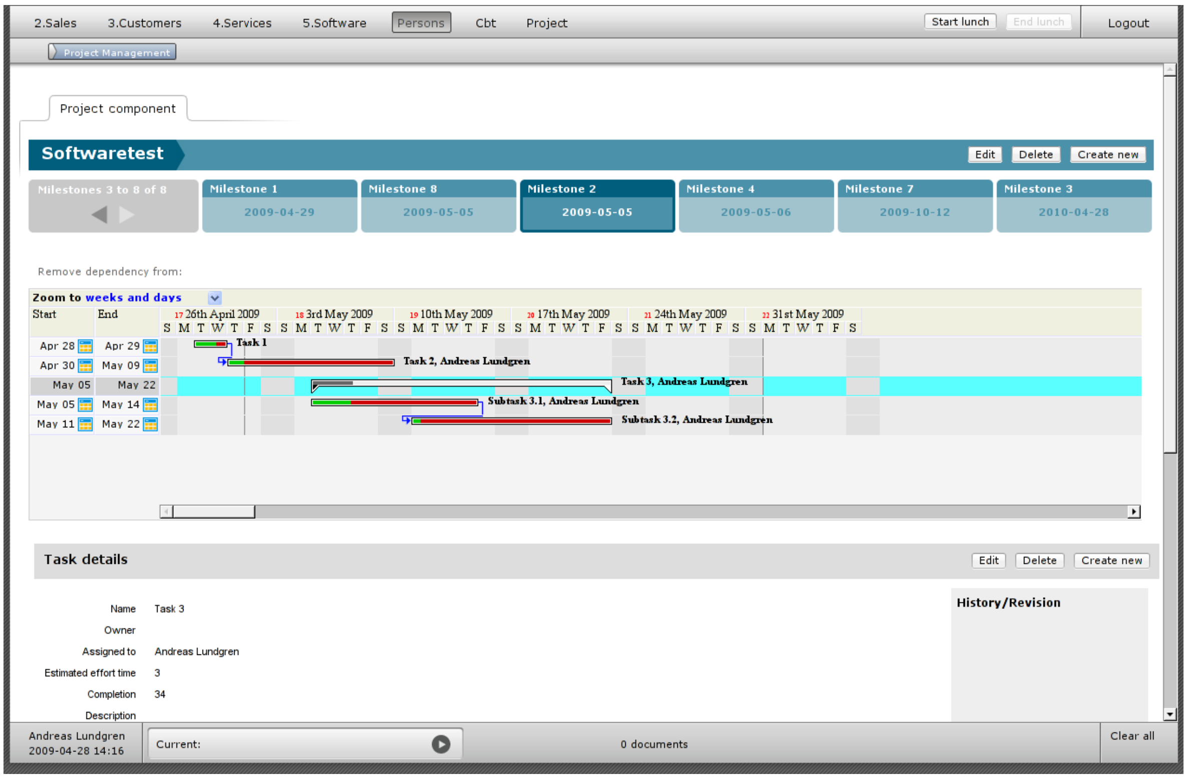
Screenshot 8: Editing a selected task. Fields are correctly validated.



Start	End
Apr 28	May 07
Apr 28	Apr 29
Apr 30	May 01
May 02	May 03
May 04	May 05
May 06	May 07
May 08	May 15
May 08	May 09
May 10	May 11

Name	Software Top Level Design Document
Owner	Andreas Lundgren
Assigned to	Andreas Lundgren
Start date	2009-05-08
End date	2009-05-09
Estimated effort time	31
Completion	65
Description	The system shall be divided into high level components

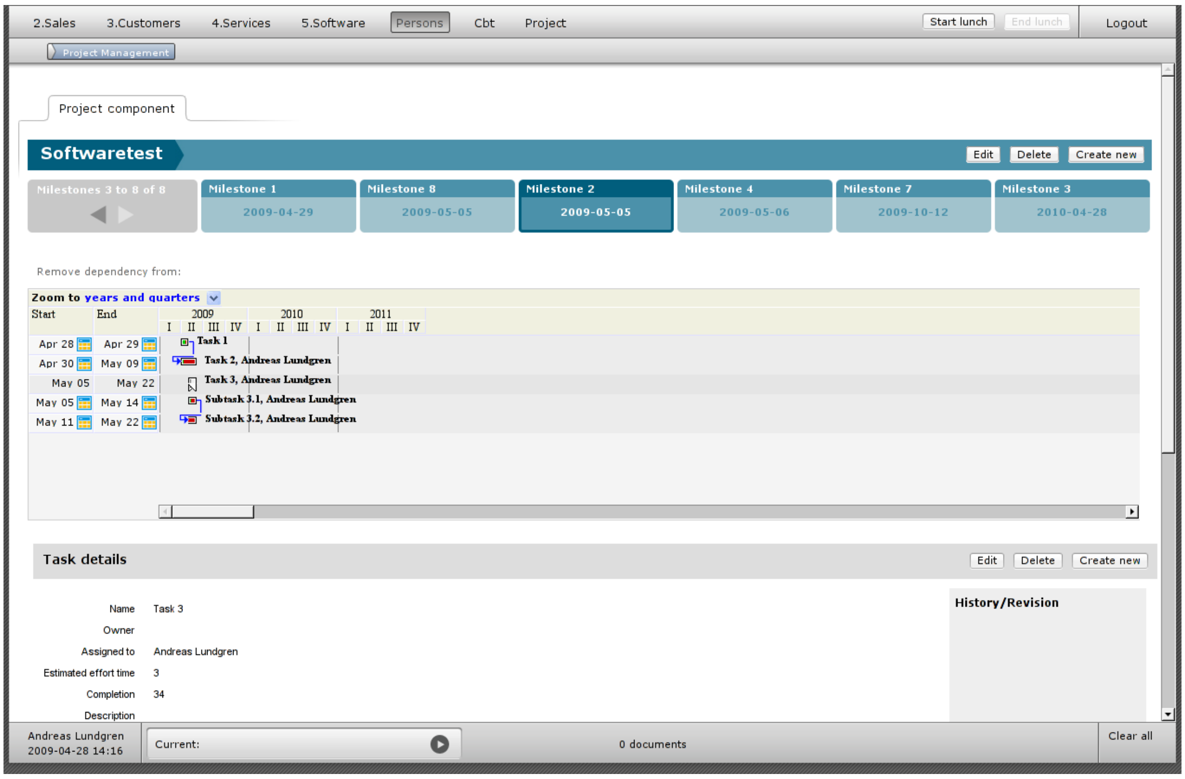
Screenshot 10: Showing completion with green bar on tasks. Presenting removable dependencies to selected task, above Gantt chart.



Start	End
Apr 28	Apr 29
Apr 30	May 09
May 05	May 22
May 05	May 14
May 11	May 22

Name	Task 3
Owner	
Assigned to	Andreas Lundgren
Estimated effort time	3
Completion	34
Description	

Screenshot 11: Showcasing representation of tasks and subtasks.



Start	End	
Apr 28	Apr 29	Task 1
Apr 30	May 09	Task 2, Andreas Lundgren
May 05	May 22	Task 3, Andreas Lundgren
May 05	May 14	Subtask 3.1, Andreas Lundgren
May 11	May 22	Subtask 3.2, Andreas Lundgren

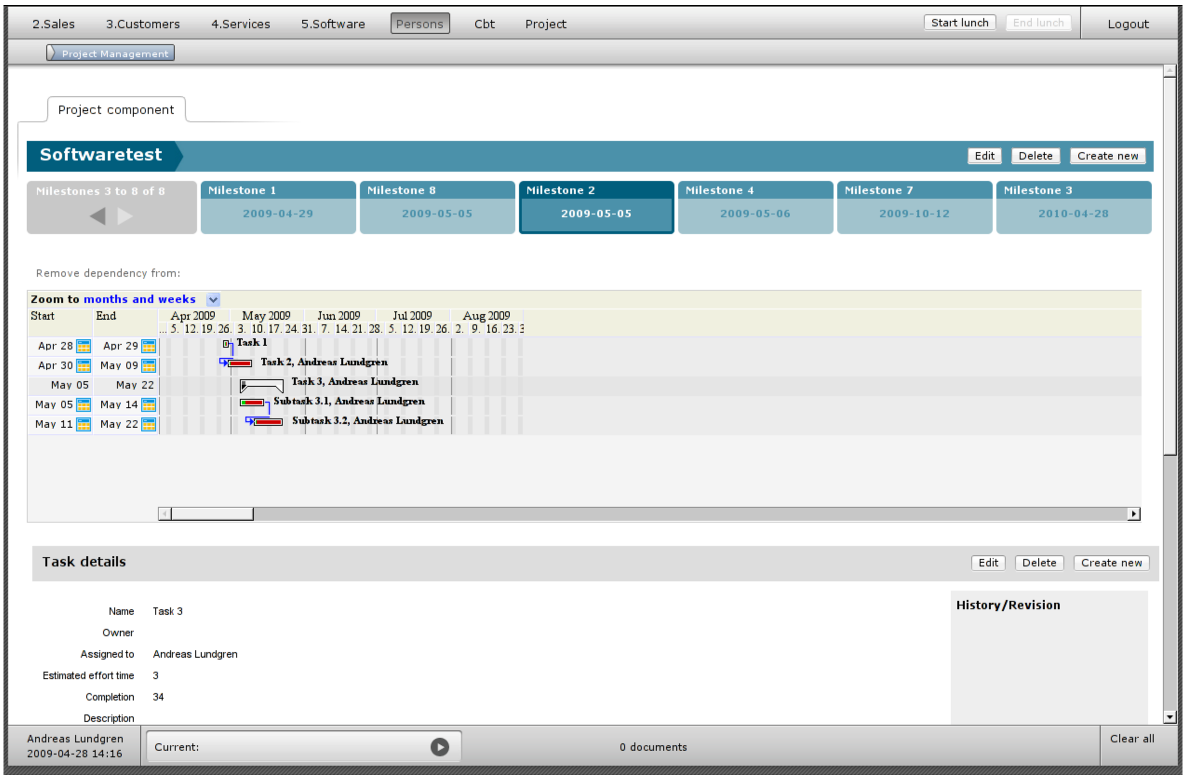
Task details

Name	Task 3
Owner	
Assigned to	Andreas Lundgren
Estimated effort time	3
Completion	34
Description	

History/Revision

Andreas Lundgren 2009-04-28 14:16 Current: 0 documents Clear all

Screenshot 12: Zoom level “years and quarters”.



Start	End	
Apr 28	Apr 29	Task 1
Apr 30	May 09	Task 2, Andreas Lundgren
May 05	May 22	Task 3, Andreas Lundgren
May 05	May 14	Subtask 3.1, Andreas Lundgren
May 11	May 22	Subtask 3.2, Andreas Lundgren

Task details

Name	Task 3
Owner	
Assigned to	Andreas Lundgren
Estimated effort time	3
Completion	34
Description	

History/Revision

Andreas Lundgren 2009-04-28 14:16 Current: 0 documents Clear all

Screenshot 13: Zoom level “months and weeks”.