Final Degree Project

Redesign and Improvement of Knowledge Management Software

Studies: Telecommunication Engineering
Author: Diana Cantó Estany
Tutor: Professor LIU Fuqiang
Supervisor: Professor Marcel Fernandez
Date: November 2013
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Basic Workflow elements</td>
<td>30</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Nodes</td>
<td>30</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Transactions</td>
<td>31</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Actions</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Forms</td>
<td>31</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Form elements</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Tasks</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Mail</td>
<td>36</td>
</tr>
<tr>
<td>5.5</td>
<td>Join and Fork</td>
<td>36</td>
</tr>
<tr>
<td>5.6</td>
<td>Advanced simulation</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>Task management desktop</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>Novelties</td>
<td>41</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Structure</td>
<td>42</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Not workflow related tasks</td>
<td>43</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Subtasks</td>
<td>44</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Other details</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>Future Work</td>
<td>47</td>
</tr>
<tr>
<td>A</td>
<td>OpenKM Specifications</td>
<td>48</td>
</tr>
<tr>
<td>B</td>
<td>Courses</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>Platform final design</td>
<td>51</td>
</tr>
<tr>
<td>D</td>
<td>CSS files</td>
<td>53</td>
</tr>
<tr>
<td>E</td>
<td>JSP files</td>
<td>83</td>
</tr>
<tr>
<td>F</td>
<td>Invoice Example - Figures</td>
<td>89</td>
</tr>
<tr>
<td>G</td>
<td>Modified Files</td>
<td>92</td>
</tr>
</tbody>
</table>
# List of Figures

4.1 Old colour design .............................................. 19  
4.2 New colour design ............................................. 20  
4.3 Hovered and non-hovered button ............................ 20  
4.4 Login page ..................................................... 21  
4.5 Code for the Select All button .............................. 22  
4.6 Profile edit menu with "select all" buttons ............... 23  
4.7 Code for sorting tables ....................................... 24  
4.8 User list menu listed alphabetically A-Z and Z-A ...... 24  
4.9 Pop-up menu to create a new user .......................... 25  
4.10 Code to display the pop-up menu ........................... 26  
4.11 Ajax functions for the pop-up menu ........................ 27  

5.1 Workflow form example source code ...................... 31  
5.2 Workflow form example render ............................... 32  
5.3 Use of workflow process instance objects ............... 32  
5.4 Form for the reception of a document .................... 34  
5.5 Task properties setting ...................................... 35  
5.6 Task properties code ......................................... 35  
5.7 Example of the use of a MAIL node ...................... 36  
5.8 Workflow using FORK and JOIN nodes .................... 37  
5.9 Image file ...................................................... 38  

6.1 Eisenhower matrix ............................................. 42  
6.2 Create task ..................................................... 43
Redesign and Improvement of KM Software

6.3 View Task ........................................... 44

C.1 Main Desktop - Old Design ................................ 51
C.2 Main Desktop - New Design ................................ 52

F.1 Form .................................................. 89
F.2 Source code ........................................... 90
F.3 Decision handler ....................................... 91
F.4 Action handler ........................................ 91
Redesign and Improvement of KM Software

RESUM

La gestió del coneixement ha esdevingut una qüestió de gran importància per a les empreses i entitats aquests últims 20 anys. Així doncs, adaptacions en la tecnologia, com bases de dades, sistemes experts, repositoris de coneixement, sistemes de suport per la presa de decisions en grup, intranets i sistemes digitals de suport pel treball en equip han sigut introduïdes per tal de sumar esforços en la tasca de la gestió de coneixement.

L’OpenKM és una plataforma basada en software lliure que ens ajuda a gestionar de manera efectiva la intel·ligència disponible als nostres recursos. També inclou una eina de gestió de tasques que permet als administradors fer un seguiment exhaustiu de les tasques assignades als diferents usuaris. Així doncs, en aquests projectes el codi font de l’OpenKM és modificat per tal d’afegir noves capacitat i renovar la imatge de la plataforma. A més a més, es proposa un nou escriptori pel gestor de tasques amb la finalitat de permetre noves funcionalitats i ajudar als usuaris a millorar les seves habilitats en la gestió del temps.
RESUMEN

La gestión del conocimiento ha llegado a ser una cuestión de gran importancia para empresas y entidades durante estos últimos 20 años. En consecuencia, adaptaciones en la tecnología, como bases de datos, sistemas expertos, repositorios de conocimiento, sistemas de apoyo en la toma de decisiones en grupo, intranets y sistemas digitales de apoyo para el trabajo en equipo han sido introducidas con el objetivo de sumar esfuerzos en la tarea de la gestión del conocimiento.

OpenKM es una plataforma basada en software libre que nos ayuda a gestionar de manera efectiva la inteligencia disponible en nuestros recursos. También incluye una herramienta de gestión de tareas que permite a los administradores hacer un seguimiento exhaustivo de las tareas asignadas a los usuarios. Así pues, en este proyecto el código fuente de OpenKM es modificado con tal de conseguir nuevas capacidades y de renovar la imagen de la plataforma. Además, se propone un nuevo escritorio para el gestor de tareas con el fin de permitir nuevas funcionalidades y ayudar a los usuarios a mejorar sus habilidades en la gestión del tiempo.
ABSTRACT

Knowledge management has become a main issue for companies and institutions in the last twenty years. Thus, specific adaptations of technologies such as knowledge bases, expert systems, knowledge repositories, group decision support systems, intranets, and computer-supported cooperative work have been introduced to further enhance knowledge management efforts.

OpenKM is a free software based platform which helps to efficiently manage the intelligence of the available resources. Moreover, it includes a task management system which allows administrators to keep track of the tasks assigned to any users. However, the platform does not include all the functionalities needed to performance all desirable tasks from a user point of view. Thus, in this project the OpenKM source code is modified in order to add new capabilities and renew the image of the platform. Furthermore, a new task management desktop is proposed in order to allow new functionalities and help the users improve their time management skills.
Chapter 1

Introduction

Knowledge management efforts have a long history, to include on-the-job discussions, formal apprenticeship, discussion forums, corporate libraries, professional training and mentoring programs. More recently, with increased use of computers in the second half of the 20th century, new technologies development has made that companies and institutions focus their efforts on handling a huge amount of information. Few years ago, the focus was on obtaining this information. Over time, the focus shifted to digitalizing and building the infrastructure to share this documentation around the world. Nowadays, the focus is on finding the best way of managing and classifying all the documents, video and other sources of knowledge to make them easily accessible.

But knowledge management is not the only problem that concerns institutions and companies. The concept of ”soft skills” has appeared as the future personal attributes that will make the difference among hundreds of well-prepared professionals. Unlike hard skills, which are about a person’s skill set and ability to perform a certain type of task or activity, soft skills relate to a person’s ability to interact effectively with co-workers and customers and are broadly applicable both in and outside the workplace. Thus, it has appeared to be a key element on making the difference among rival companies, as well.

Soft skills enhance an individual’s interactions, job performance and career prospects and are an important part of someone’s individual contribution to the success of an organization. In consequence, new software and tools are being developed and further improved with the aim of strengthening these attributes. Nevertheless, the concept of soft skills is hugely broad including proficiencies such as communication skills, conflict solving, personal effectiveness, strategic thinking and team building, to name a few.

Directly related to personal effectiveness, concepts like task management and time management can be found. These two attributes can be easily tracked in a digital way providing the user with tools that will help him analyse their amount of work, their available time and the way it is being spent. Task management may form part of project management and process management and can serve as the foundation for efficient workflow in
Redesign and Improvement of KM Software

an organisation. Therefore, we can have a positive impact on an individual’s soft skills and, by extension, to its performance in its workplace.

In this project the use of specific software that combines both knowledge and task management tools is proposed. OpenKM consists of 4 desktops: the main one, where all the documentation is accessible; the searcher, where different kinds of criteria can be used to find the desired information; the dashboard, where the user can keep track of its tasks and the new documentation uploaded to the system; and the administration desktop, where the administration can create users and profiles, start and assign tasks and define other configuration properties.

The focus of this project is to improve the platform in order to make it more comfortable, dynamic and intuitive. Therefore, the source code has been modified allowing new functionalities such as sorting lists or having pop-up menus and the image of the platform has been renewed. Furthermore, a new task management desktop has been proposed in order to allow the specification of tasks’ priority and deadline, as well as including in the system tasks not directly assigned by the administrator.
Chapter 2

Objectives

In this project, the use of a particular knowledge management platform is proposed. The main objective of this project is then to improve this platform and provide its users with better tools. Thus, the open source platform is modified to implement and evaluate those changes and new functionalities.

This project is specifically focused on a knowledge management platform which includes, as well, task management tools. Besides being a platform to storage and find documents in an easy, fast and tidy way, it is also a vehicle of communication between administrators and normal users. Administrators are able to perform a great amount of actions such as assigning tasks to the rest of users and keeping track of the evolution of those tasks.

To achieve this objectives (improve the platform and provide its users with better tools), the open source code will be modified in order to make the platform more comfortable, dynamic and intuitive. Therefore, in this project the open source code of OpenKM will be modified in order to add some new capabilities to the OpenKM platform and make possible a new set of actions both for administrators and normal users.

Thus, one of the objectives of this project is to modify the OpenKM platform in order to achieve the following:

- Propose a new and fresh design for the whole platform
- Increase the efficiency and efficacy in the users’ performance of tasks by adding tools to both sort and re-use the available information
- Increase the interactivity user-platform both through the design and the new functionalities

Once the code is modified, the platform will be installed in a server in conjunction with another platform called OpenMeetings to provide the customer with a full service.
Redesign and Improvement of KM Software

Furthermore, the possibilities provided but the workflows (assignment of tasks) and the whole task management system will be analysed and tested, setting the bases to understand what can be done and how and identify its weaknesses and the points to be improved.

Finally, a new desktop for the OpenKM platform will be proposed to be added, having a higher impact in task and time management and providing the users with enough tools to manage their tasks according to its importance and its deadline, as well as including in the system tasks without requiring the intervention of an administrator.
Chapter 3

OpenKM

With the pass of the years different tools to facilitate knowledge management have appeared in the market. However, it is difficult to find free software technologies able to fulfil the user expectations. OpenKM is focused on creating an Open Source Electronic Document Management System, that due to its characteristics can be used by big companies as well as by small ones as a useful tool in processing knowledge management, providing a more flexible and cost effective alternative to other proprietary applications.

The use of a Knowledge Management System in a company or institution helps to manage, more efficiently, the intelligence of the available resources of this company; this implies an increase of productivity in the short term. Through these systems the information, as well as the knowledge generated within the organization, can be both globally accessed.

Technologically speaking, a document management system has been created to provide the knowledge management that will use free software standards and technologies. OpenKM provides the solution to:

- Control the growing volume of information to be managed
- End the diversification of information storage sources
- Prevent the duplication of information
- Obtain valid information through the control of document versions
- Reduce time and resources used in locating information
- Avoid knowledge leaks or wrongful access by users
- Restrict non-authorized users access to the information

Through the OpenKM system, relevant and useful information can be accessed and found in the least possible time to resolve problems and make decisions. OpenKM helps to
Redesign and Improvement of KM Software

improve corporation processes, integrating common tasks (and interrelated tasks) into one single tool and permitting the different departments of the organization controlled and supervised parameterized access to the information.

Given that this is a project based on free software, all the tools and results emerging from the project are available to the general public, with usage rights free of charge under a GNU GPL version 2 licence. The OpenKM software can be downloaded quite easily. OpenKM is based on J2EE technology and the Apache Tomcat so it can be installed and run on various platforms.

Therefore, the aim of this project strives on improving the available software by adding features and flexibility and trying to fulfil the user necessities as much as possible.

3.1 Developer Guide

The very first step of this project focused on both installing OpenKM and downloading the source code in order to proceed with the planned improvements and modifications. The developer environment can be set in any operating system (Linux, Windows, etc.) since it is a multi-platform system, but Linux is recommended, because it is what it is being used.

In the OpenKM wiki [1] there is information about how to install OpenKM for developers. The requirements for OpenKM 4.0 and 5.0 are the following:

<table>
<thead>
<tr>
<th>Software</th>
<th>Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maven</td>
<td>maven.apache.org/</td>
</tr>
<tr>
<td>Eclipse Juno</td>
<td>eclipse.org/juno/</td>
</tr>
<tr>
<td>Subversion eclipse plugin</td>
<td>subclipse.tigris.org</td>
</tr>
<tr>
<td>Maven to eclipse plugin</td>
<td>m2eclipse.sonatype.org</td>
</tr>
<tr>
<td>Java JDK 1.6</td>
<td>oracle.com/technetwork/java/javase/downloads/</td>
</tr>
</tbody>
</table>

The complete list of OpenKM specifications is available in the appendix A.

On one hand, if our objective is to modify the source code of OpenKM we need to import this code. The first thing to do is installing Eclipse and Java. Then, Eclipse needs to be executed and using Eclipse Market Place, Maven support can be installed as well. Once this is finished, support for Subversion should be installed and then we can proceed with the OpenKM checkout.

This process will take some time because it will download all the libraries needed by OpenKM. After downloading the source code, some project properties will have to be changed in order to remove the errors got during the process.
The built will have to be configured selecting the project, writing the goals, setting the virtual machine memory parameters and configuring Java. Finally, the first compilation can be launched and an OpenKM folder will be automatically created into the Target folder in our project.

On the other hand, OpenKM has to be installed in our computer so we can use it and check the changes we implement. Once we have OpenKM Community (which comes with Tomcat and OpenKM application) installed, we need to remove OpenKM.war folder and make a link from the OpenKM Target folder to the Tomcat folder. With this action we will be able to modify the source code in our project in Eclipse and see the changes directly in the OpenKM installed in our computer.

Last but not least, Tomcat server has to be configured in Eclipse so we are finally able to access our local OpenKM installation.

Besides the purely knowledge management tools that OpenKM is offering, there is another tool which will be also a focus of this project. This tool is more related to task management and we will refer to it as workflow. Workflow allows us to define tasks, assign them to our OpenKM users and have a complete follow-up of the process, starting when the task is assigned and finishing when the task has been executed.

To be able to use and develop workflow, jBPM3 should be installed as well. The information available in the OpenKM wiki [1] urges to install JBoss Developer Studio as the base developing platform and then download, install and configure jBPM3 in JBoss Developer Studio. However, jBPM3 can be also installed and configured easily in Eclipse and this is how this project has been done.

Once having both the OpenKM source code and jBPM3 we have the possibility to develop workflow using some OpenKM classes to make API calls. This aspect will be further detailed in section 5.

### 3.2 OpenKM source code

When going through the OpenKM source code 4 clearly distinguishable sections are found:

- **Admin**: that includes the part of the platform that is only accessible for admin users.
- **Fronted**: that encompasses the main desktop as well as the search and the dashboard panels.
- **Login page**: that includes the login desktop and the error pages in case there is an error in the login information.
Redesign and Improvement of KM Software

- Web-Inf: were all the OpenKM classes and libraries needed are placed.

As it is explained in the following sections, the source code uses different languages (Java, HTML, CSS, jQuery). Therefore, to be able to understand and modify those files I took some online courses to increase my knowledge and have a better impact in the platform. The list of the aforementioned courses can be find in the annex B.

3.2.1 Admin pages

The administration desktop is quite broad and offers a wide range of possibilities:

- Activity log: any operation done by OpenKM users are automatically audited in the database. This is a complete audit tracking log for any operation done in OpenKM.

- Automation rules: for adding some rules related to the actions that are performed, such as specifying a new path where documents will be created.

- Configuration: where the administrator can see the parameters that have been loaded during OpenKM start up and ensure that changes in OpenKM.cfg have been loaded correctly.

- Crontab: with this feature you can create scheduled tasks in a easy way. In this page you can see the already registered tasks. The last start and last stop columns show the last time a task was executed. If you want to see the output of the script, you can set an email and you will be notified every time the task is executed.

- Database query: using database query, administrators can make queries on OpenKM databases. The query type can be JDBC or Hibernate.

- Languages: offers the possibility of adding new languages to the platform. Nowadays it is only available in English and Spanish.

- Metadata: where the administrator can have an overview of the registered group properties. New metadata properties can be added as well.

- Mime types: for creating, editing and removing mime types.

- Profiles: each user must be assigned a profile. A profile is a definition of general configuration parameters of the UI interface, available extensions, and other OpenKM features that allows for customization of the end user interface.

- Reports: for creating and using own Jasper Reports.

- Repository export: to export all the folder structures and documents to a server disk.
Redesign and Improvement of KM Software

- Repository import: to import folder structure to OpenKM server.
- Scripting: enables administrators to execute some BeanShell scripts in folders, fired on every notified event.
- Statistics: to have an overview about the repository size and its distribution: documents by context, documents size by context, folders by context, memory usage and disk usage.
- Thesaurus: where the administrator can start thesaurus folder tree creation.
- Users: creating, editing and deleting the accounts for the users of the platform.
- Utilities: from this panel some advanced administrator utilities can be accessed.
- Workflow: listing of the workflows available and its state if they have been assigned to any user.

These parts of the platform are mainly formed by several JSP files and some CSS and JS files which provide the desired format and functions.

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP allows Java code and certain pre-defined actions to be interleaved with static web mark-up content, with the resulting page being compiled and executed on the server to deliver a document.

CSS files combined to the XHTML tags in the JSP files define the format of the platform, thus those will be the target of section 4.1.

3.2.2 Fronted

The index.jsp file together with some other HTML and CSS files defines the skeleton of the platform. This includes the main desktop, the search panel and the dashboard panel.

The main desktop shows the different folders and documents available to the user. On this desktop documents can be previewed, downloaded, uploaded, deleted, etc. Users can subscribe to the different documents so every time a modification is made, these users will be notified by e-mail. OpenKM also allows users to synchronize their e-mail account with the platform, so they can have access to their inbox directly from OpenKM, that storages mailing in folders according to date of reception.

On the search panel users can look for documents. The platform allows several ways of performing this task: by the document’s name, key words assigned to the different documents, folder, type of document, date, users, among other. Enquiries can be saved in order to perform future searches in a faster way.
On the dashboard panel a user can have an overview of the updates on the information available in the system (last documents uploaded or modified) and also of the tasks (workflows) he is involved in. It is also the place where users will be actually able to accomplish their tasks and access their e-mail inbox.

3.2.3 Login pages

This section of the platform is formed by some JSP files which configure the login desktop of the system as well as the error messages in case of wrong login information.

3.2.4 Web-Inf

Web-Inf counts with more than 2000 files being mainly OpenKM classes. These classes are files containing Java code that can be used in workflows or in extensions which enable to add features at OpenKM Core level.
Chapter 4

New Design and Tools

As was previously mentioned, the main objective of this project is to propose the use of OpenKM as a knowledge management platform. That means providing the future users with a platform that fulfils their needs in terms of design, tools and possibilities.

Therefore, a new design for the whole platform has been created and new tools to increase the efficiency and efficacy in the users’ performance of tasks have been added to the system. All these aspects will be analysed and described in detail in this chapter.

4.1 Design

The layout of a webpage or a program is the very first thing a user sees when using it. How things look, where they are placed, how they are organised are definitely aspects which will either make our user feel comfortable and willing to benefit from the tools we are offering or just disliking our platform and using it only when it is due to.

Knowledge management is a practice which requires an easy but constant maintenance. If we want our company to have a good document management system there is a need to facilitate the use of the platform, as everyone in the company, no matter which position holds, should be both using and contributing to the system. Taking into account all these facts, some changes in the platform’s design have been made. The aforementioned changes include:

- Colour
- Interactivity
- Size
- Shape
Redesign and Improvement of KM Software

To be able to implement these changes some code has been added to the CSS files. Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a mark-up language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document.

The final version of these files can be found in the section D.

4.1.1 Colour

Colour plays a vitally important role in the world in which we live. Colour can sway thinking, change actions and cause reactions. It can irritate or soothe your eyes, raise your blood pressure or suppress your appetite. Colour is a powerful form of communication. For instance red means ”stop” and green means ”go”, and this is something not just to keep in mind but to take advantage of when designing our platform.

One more thing to take into account is that colour associations and meanings vary from culture to culture. Mixing, combining certain colours can boost specific culture, religion or gender associations, while avoiding certain colours and combinations of colours can neutralise inherent negative cultural connotations.

OpenKM platform was basically grey as you can see in figure 4.1. Different tones of grey were used in its elements and only green and red were present when a message was displayed after human interaction.

In Europe and America, grey is the colour most associated with boredom, solitude and emptiness. Grey is the colour of conformity- not having any personality of its own. Then, if we intend users to usually contribute to the system proactively, we should avoid this kind of associations.

As it is shown in figure 4.2, the primary colour for this new version of OpenKM is blue. Blue is the favourite colour of more than half of the world’s people - it is the colour least disliked by most cultures and, last but not least, OpenKM logo is also blue, so everything stays as a whole.

Figure 4.1: Old colour design
4.1.2 Interactivity

Some basic changes in the CSS and HTML code have been made to allow some interaction with the user:

- Submitting buttons change their colour when the user hovers over them.
- The text in the headers of the columns which can be sorted is underlined when a user hovers over it.
- The cursor changes to pointer every time that hovers an element containing a link or allowing an action.

You can see one of these examples in the figure bellow.

Figure 4.3: Hovered and non-hovered button

4.1.3 Size

Blue is sharply refracted by the eyes. This causes the lens to flatten and to push the blue image back. We perceive that blue areas are receding and smaller. In addition, the size of the text in the platform was rather small. Therefore, the font size was slightly increased in most of menus and desktops.

4.1.4 Shape

With the aim of having a little bit more organic design, some buttons, tables and other elements have had their corners slightly rounded.
Redesign and Improvement of KM Software

The font style has also been changed achieving a less typewriter style.

(a) Old login page   (b) New login page

Figure 4.4: Login page

4.2 Tools

As it was mentioned in the section 3.2.1, the administration desktop is one of the main parts of the OpenKM software, being actually the one containing more pages. That is why the aim of this section will be to improve the admin pages adding some tools to them.

4.2.1 Usability

While getting to know OpenKM and using its software, some basic needed improvements in terms of usability could be noticed. Usability is the ease of use and learnability of a human-made object. Usability evaluation allows assessing whether our product is:

- **Efficient enough**: are users able to carry out their task while expending reasonable resources such as time, cognitive or physical demand?

- **Effective enough**: can users complete the tasks they are supposed to perform with the tool? Is their performance complete and accurate?

- **Sufficiently satisfactory for the users**: what is the users attitude towards the system? Do they experience discomfort?

Although there are a variety of usability evaluation methods which include all stages of design and development, from product definition to final design modifications, this project
Redesign and Improvement of KM Software

is based in the lack of resources or features someone who used the software for the first time noticed. I focused in three of them which will be detailed below.

4.2.2 Select all

When the administrator wanted to create a new profile for a new user, the menu for creating this new profile offered 19 sub-menus with a total of 126 options and there was no way to select more than one option at the same time.

With the aim of allowing the administrator to accomplish its intended tasks at its desired speed, the possibility of selecting all the fields related to each sub-menu has been added. This way we strive on giving the administrator as much flexibility as possible. In the figure 4.5 you can see the code related to this function. The code provided by OpenKM worked differently for two possible situations:

1. Creating a new profile: none of the checkboxes is checked.
2. Editing an existing profile: some of the checkboxes are checked.

That is why the added function foresees these two cases and acts accordingly, either selecting all the checkboxes in a submenu or deselecting them.

```javascript
$(document).ready(function(){
    $('select-db').click(function(event) {
        // Iterate each checkbox
        if(this.checked){  // <-- si es un click de "select" -->
            $('db').each(function() {
                this.checked = true;
            });
        } else{  // <-- si es un click de "unselect" -->
            $('db').each(function() {
                this.checked = false;
            });
        }
    });
});
```

Figure 4.5: Code for the Select All button

As all the sub-menus are loaded in the same page, the only detail to take into consideration is that every sub-menu needs to have its own id, so they don’t interfere between each other.

This block of code and the corresponding modifications to the source code (such as modifications in the HTML/CSS documents) has been added to the following pages:
4.2.3 Sort

When having a table with several columns displaying a great amount of information it is useful for the user to have the possibility of sorting the mentioned table by its different columns. This feature has been added to the different tables in the administration menu using a jQuery function called \texttt{tablesorter()}.

Although this is an existing function, some adaptation to our specific tables had to be done. As it can be seen in the figure 4.7, there are three blocks of code inside \texttt{tablesorter()}.

The first one manages the images, like in the tables were there is a column indicating which users are active or inactive. In this case, when an image is detected, the code goes through \texttt{textExtraction} and uses the \texttt{src} attribute of the image as the reference to sort the column.

The second block, which corresponds to \texttt{header}, manages the columns that should not offer the possibility to be sorted, like columns having the same information in every row. \texttt{header} also allows us to have links in the header of our table. That way when the user clicks on this specific header, instead of sorting the column any other action can be performed.
Redesign and Improvement of KM Software

(for instance, displaying a menu).

Finally, the third block uses the widget *zebra* to maintain the bicolour design of our table. Without this widget, the colour would stay stuck at its row. That way when a column would be sorted, each row would move together with its colour and we could have several rows with the same colour displayed one after each other.

```
$("#tablesorter").tablesorter({
  textExtraction: function(s){
    var $el = $(s),
        $img = $el.find('img');
    return $img.length ? $img[0].src : $el.text();
  },
  headers: {
    0: {
      sorter: false
    },
    8: {
      sorter: false
    }
  },
  widgets: ['zebra'],
});
```

Figure 4.7: Code for sorting tables

The following image shows the *profile_list* page both listed alphabetically from A to Z and from Z to A.

![User list menu listed alphabetically A-Z and Z-A](image)

Figure 4.8: User list menu listed alphabetically A-Z and Z-A

This block of code and the corresponding modifications to the source code (such as modifications in the HTML/CSS documents) has been added to the following pages:
Redesign and Improvement of KM Software

activity_log.jsp  administration_rule_list.jsp  crontab_list.jsp
language_list.jsp  logged_users.jsp  message_list.jsp
process_definition_list.jsp  profile_list.jsp  report_list.jsp
role_list.jsp  user_list.jsp

4.2.4 Pop-up menu

Every time an administrator wanted to edit or create a user or modify any mime type or mail configurations, the edit/create menu was displayed in a new page. That way the already existing information that is available in the "list" pages (user_list.jsp and similar ones) remained hidden but sometimes it is useful to have an example of how things have been done previously. Moreover, each user action requires the page to be reloaded from the server (or a new page loaded) which makes the process inefficient.

These are the reasons to have changed the edit/create/delete menus that appeared in new pages to pop-up menus that are displayed in the same page were the administrator is working so it is able to check any information it may need.

To do so, I have used the jQuery widget `dialog()`. The basic dialog window is an overlay positioned within the viewport and is protected from page content (like select elements) shining through with an iframe. It has a title bar and a content area, and can be moved, resized and closed with the 'x' icon by default.

After modifying the CSS and the correspondent JSP files, the result is the following.

Figure 4.9: Pop-up menu to create a new user
Redesign and Improvement of KM Software

The code supporting the pop-up menu is showed in figure 4.10. It is formed by two blocks: the first one distinguishes the three cases we can have (creating, editing or deleting a user) and the second one opens the dialog and defines some of its attributes (the rest can be found at the JSP file in the annex E).

```javascript
$(".task").click(function()
    if($(this).is("#create")){
        var html = '<input class="required only_on_blur'
    ajax:Auth?action=validateUser name="usr_id" value=""/>';
    $("#usr_id").empty().html(html);
    $("#dialog-form").dialog('option', 'title', "CREATE USER");
    ajaxCreate();
} else if($(this).is("#edit")){
    var editid = $(this).attr("name");
    editid = editid.split("-");
    var editid = editid[1];
    var id = $("#td-userid-"+editid).html();
    $("#dialog-form").dialog('option', 'title', "EDIT USER");
    ajaxEdit(id);
} else if($(this).is("#delete")){
    var deleteid = $(this).attr("name");
    deleteid = deleteid.split("-");
    var deleteid = deleteid[1];
    var id = $("#td-userid-"+deleteid).html();
    $("#dialog-form").dialog('option', 'title', "DELETE USER");
    $("#dialog-form").html(html);
    ajaxDelete(id);
}

$("#dialog-form").dialog("open");
});

$("#dialog-form").dialog({
    autoOpen: false,
    height: 390,
    width: 400,
    modal: true,
});
```

Figure 4.10: Code to display the pop-up menu

The main problem we face when including this pop-up menu is that we need to communicate with the server asynchronously. Before changing anything, when we wanted to create, edit or delete a user and we clicked on the icon a new page was loaded. In this process, a request to the server was made and then the answer from the server was displayed through this new page. But now, we want to stay in the same page but at the same time communicate with the server. To solve this, some Ajax functions have been
Redesign and Improvement of KM Software

-added. The corresponding code is displayed in figure 4.11.

Ajax (an acronym for Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behaviour of the existing page.

```javascript
function ajaxEdit(id)
    
$.ajax({
    type : "GET",
    url: "Auth",
    data: {action : "userEdit", user_id : id},
    success : function(data){
        $('#dialog-form').empty().html(data);
        $('#form-cancel').click(function(){
            $('#dialog-form').dialog("close");
        });
    },
});

function ajaxCreate{}

$.ajax({
    type : "GET",
    url: "Auth",
    data: {action : "userCreate"},
    success : function(data){
        $('#dialog-form').empty().html(data);
        $('#form-cancel').click(function(){
            $('#dialog-form').dialog("close");
        });
    },
});

function ajaxDelete(id){
    
$.ajax({
    type : "GET",
    url: "Auth",
    data: {action : "userDelete", user_id : id},
    success : function(data){
        $('#dialog-form').empty().html(data);
        $('#form-cancel').click(function(){
            $('#dialog-form').dialog("close");
        });
    },
});
}
```

Figure 4.11: Ajax functions for the pop-up menu
Redesign and Improvement of KM Software

The three functions are quite similar. Based on the id which indicates the user we want to edit/delete and using a get method, in case the communication with the server is successful the information that the server is giving us will be displayed on the pop-up menu.

This block of code and the corresponding modifications to the source code (such as modifications in the HTML/CSS documents) has been added to the following pages:

- automation_rule_list.jsp
- config_list.jsp
- crontab_list.jpg
- language_list.jsp
- mime_list.jsp
- user_list.jpg
Chapter 5

Workflow

JBoss jBPM is a flexible, extensible framework to process languages. jPDL is one process language that is built on top of that common framework. It is an intuitive process language to express business processes graphically in terms of tasks, wait states for asynchronous communication, timers, automated actions... To bind these operations together, jPDL has the most powerful and extensible control flow mechanism.

jPDL has minimal dependencies and can be used as easy as using a java library. But it can also be used in environments where extreme throughput is crucial by deploying it on a J2EE clustered application server. jPDL can be configured with any database and it can be deployed on any application server.

In this way, tasks can easily be defined and assigned to any OpenKM user in a graphical environment, providing most of the code automatically.

Each of our workflow process usually has three files:

- Process Info File: that contains the source code.
- Graphical Info File: that contains the code related to the form, position and size of the nodes that appear in the Image File.
- Image File: that includes the image of our process that is later on displayed in OpenKM.

Then, depending on the action performed by our workflow process we can also have:

- Additional Files: such as forms that need to be included in the deployment.
- Classes and Resources: usually javascripts that need to be included in the deployment.
Workflows can be either deployed in Eclipse or directly uploaded to the OpenKM system using a .par file. Once having them in the system, the administrator can define which users have the right to start which workflows modifying their profile settings. However, the administrator can assign a workflow to every user no matter if the user has the right to start the workflow or not.

5.1 Basic Workflow elements

5.1.1 Nodes

Nodes are represented as boxes. The most important ones are:

START: every single workflow has to start with a START node and no other START node can be present. Once a user executes a workflow, the flow will go immediately across the transition which connects the START node with the next one.

STATE: when the flow arrives to a STATE node it gets stuck there until a user (usually the administrator) fires a signal.

TASK: when some human interaction is needed, a TASK node is required. The flow will be stuck at this node waiting for human intervention.

DECISION: a DECISION node is used to guide the process running path. This means you can tell the workflow which way should go. Decision criteria can be specified as follows:

- Adding conditions to transactions or BeanShell script which returns a boolean. Branches go across its transitions examining the conditions until found the first one which match a criteria.

- Creating a class which implements the DecisionHandler interface. The decide method returns the transition to follow.

NODE: nodes are usually employed to execute complex logic evaluations or actions (see section 5.1.3). When the flow arrives to a NODE node, it is executed immediately and flows across the output transaction to the next node. A workflow gets never stuck at a NODE node.

END: one END node must be present in a workflow, although several are also allowed. When a workflow arrives to an END node, it is automatically finished.
5.1.2 Transactions

A transaction is a connection between two nodes and is represented by an arrow. During a transaction execution some actions can be executed. Workflow does not stop during a transaction execution.

The most important thing to take into account is that different transactions going out from the same node must have different names.

5.1.3 Actions

Actions are parts of the logic implemented as programs. These programs can do several actions like connecting with third-party applications to retrieve or store data, send mails, evaluate workflow variables and take some decision, etc. Actions can be defined in two ways:

- Using a BeanShell script.
- Using an action handler, this is a Java class which implements the ActionHandler interface.

5.2 Forms

When you want to retrieve data from a user in the workflow, you have to define a way to do so. OpenKM uses the `forms.xml` file to draw the components in the form. See the example below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.4//EN" "http://www.openkm.com/dtd/workflow-forms-2.4.dtd">

<workflow-forms>
  <workflow-form task="get-data">
    <input label="Number to guess" name="number"/>
    <button name="submit" label="Submit"/>
  </workflow-form>
</workflow-forms>
```

Figure 5.1: Workflow form example source code

Here we can see an XML which describes the components related to a task called "get-data". This means that when the workflow is executing the "get-data" task, it needs the
Redesign and Improvement of KM Software

user who is assigned to this task to enter some information to continue. In the description, we can see an INPUT where the user should type a number and a button to submit the information.

This form definition is rendered as this:

![Workflow form example render](image)

Forms are also the way to retrieve data from a user and use this data afterwards. This means creating a workflow process instance object, which is an object that will be present during the whole workflow life cycle. These objects store the user data in memory and this user data can be used in each node or action.

When a user submits a workflow form, the objects are always FormElements type. Basically any form submitted by a user will store in memory FormElements type. OpenKM workflows are thought to use FormElements to initialize data values.

The following example shows two forms: in the first one the user gives a number as input and the second form shows this number to the user. This is easily done using the keyword data.

```xml
<workflow-forms>
  <workflow-form task="get-data">
    <input name="number" label="Number to guess" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="out-data">
    <input name="number" label="Number to guess", data="number" />
    <button name="submit" label="Submit" />
  </workflow-form>
</workflow-forms>
```

![Use of workflow process instance objects](image)

Figure 5.3: Use of workflow process instance objects
5.2.1 Form elements

Forms accept different elements which will help us to perform our task in the easiest way:

**INPUT**: used to enter free text. *label* and *name* attributes are required.

**SUGGESTBOX**: this is an implementation of the typical AJAX suggest box component. *label*, *name*, *filterQuery*, *valueQuery* and *dialogTitle* attributes are required.

**CHECKBOX**: best used to represent boolean data. *label* and *name* attributes are required.

**TEXTAREA**: similar to INPUT but for longer text. *label* and *name* attributes are required.

**SELECT**: in this case, the value can be selected from a predefined list. Every select component can contain n arbitrary number of options bigger than one. *label* and *name* attributes are required.

**BUTTON**: used to submit the information entered in the form. *label* and *name* attributes are required.

**SEPARATOR**: represents a way to draw a separator to group different elements. *label* and *value* attributes are required.

**TEXT**: prints a text which can contain HTML tags. *label* and *name* attributes are required.

**VALIDATOR**: used to validate the information entered by the user. More than one validator can be used in every form element.

**DOWNLOAD**: used to download documents. *label* and *name* attributes are required. It should also contain one or more node elements which represent the document to download. This node element should have at least the *label* attribute.

**UPLOAD**: used to upload documents. *label* and *name* attributes are always required. If willing to upload a new document, *folderPath* or *folderUuid* should be specified as well. If willing to update an existing document, then *documentUuid* should be set and *type* should have ”update” as its value.

In the figure 5.4 shows a form that simulates the reception of a document.
5.3 Tasks

One of the main utilities of creating a workflow is the possibility to assign tasks to any user. Tasks can be created easily with a node task that represents one or many tasks executed by a human person. When the process execution arrives to a task node, a task instance is created in the workflow member list. This node will enter in a wait state until the user notifies the conclusion of the task.

There are different ways to assign a task:

- **Actor**: this is a String which identify the actor who will perform the task. When the process flow enters in the task node, the task is created and is added to the list of this actor’s pending task.

- **Pooled Actor**: a sequence of actors. When this task is created is not assigned to any actor but it appears in the list of pooled tasks of all of them. Once one of the actors self-assigns the task, it is added to his own list of pending task and disappears from the pooled tasks list.

- **Swimlane**: a swimlane defines an assignment which is the same for several tasks in a process. This option allows us to assign any task we might have in the workflow to the user who started the flow, no matter who actually this person was.
Redesign and Improvement of KM Software

- **Expression**: this is an assignment expression evaluated by the jBPM identity component. It can make an assignment by user or role. For example, using the "group/ROLE_USER" expression we create a collection of pooled actors who have the role "USER".

- **Handler**: here you can specify a class which implements the AssignmentHandler interface. This class assigns a TaskInstance or a SwimlaneInstance to a swimlane-ActorId or a set of PooledActors.

jBPM allows us to define this task properties in a menu (see figure 5.5) and get the code automatically (see figure 5.6).

```
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="taskWorkflow">
  <start-state name="start-state1">
    <transition to="task-node1"/></transition>
  </start-state>

  <task-node name="task-node1">
    <task name="get-data">
      <assignment actor-id="okmAdmin"/></assignment>
    </task>
    <transition to="task-node2"></transition>
  </task-node>

  <task-node name="task-node2">
    <task name="out-data">
      <assignment actor-id="okmAdmin"/></assignment>
    </task>
    <transition to="end-state1"></transition>
  </task-node>

  <end-state name="end-state1"></end-state>
</process-definition>
```

Figure 5.5: Task properties setting

Figure 5.6: Task properties code
5.4 Mail

A notification email can be send when a task gets assigned to an actor just using the `notify="yes"` attribute on a task. Setting `notify` to "yes", "true" or "on" will cause jBPM to send an email to the actor that will be assigned to the task. The email is based on a template (see JBPM3 mail configuration) and contains a link to the related document in the web application.

Similarly to the assignments case, emails can be sent as a task reminder. The reminder element in jPDL is based upon the timer. The most common attributes will be `duedate` and `repeat`. The only difference is that no action has to be specified.

If the email is not related to the accomplishment of a task but the result of it (for instance if we want to receive a notification every time a workflow gets to certain state) there is the possibility to use a MAIL node.

The example show below is a simple workflow which notifies the supervisor that a project has been finished. Any user simply starting the workflow making reference to the desired project will be able to send an email notifying the end of the project.

![Diagram](image_url)

```xml
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="mailWorkflow">
  <start-state name="start-state1">
    <transition to="Send-mail"/>
  </start-state>
  <mail-node name="Send-mail" to="openkm-admin@openkm.com">
    <subject>
      Project Finished
    </subject>
    <text>
      The project has been finished. See the link below.
    </text>
    <transition to="end-state1"/>
  </mail-node>
  <end-state name="end-state1"/>
</process-definition>
```

Figure 5.7: Example of the use of a MAIL node

5.5 Join and Fork

Until now we have seen how to assign a task either to a single user or to a pool of users. However, we only needed one of those users to execute the assigned tasks. With JOIN and FORK nodes we have the possibility to give the same task to more than one user and the process will get stuck at the JOIN node until each of the users finishes the task.
Redesign and Improvement of KM Software

The same way, different tasks can be also assigned to different users and the process will wait until all of them finish, thus allowing the simulation of a real process where different people have different roles and different tasks.

The most basic example is shown in figure 5.8

![Workflow diagram](image)

Figure 5.8: Workflow using FORK and JOIN nodes

5.6 Advanced simulation

In this section one complete workflow example will be detailed. It has a great variety of nodes and include both decision and action handlers.

This example simulates the situation in which some user wants to ask for a refund. It includes a decision handler that will check the amount to be refunded and depending on it will directly go to the "approver" (the person who will finish the workflow after actually having refund the correspondent amount) or go through the "reviser" (the person who will check the request details and ask for further information if needed).

Something to point out in this process is that in the case the reviser needs more information from the user (a receipt, a clarification, etc.) the "user" node is automatically assigned to the user who started the workflow, so it will work for anyone.

In addition, some "just-in-case" transactions have been added. This means that although the normal flow from the "approver" should go to "end" through the "approved" transaction, the approver has still the possibility of going back to the "reviser" or even to cancel the request if something goes wrong. This way the process has enough flexibility to handle many different situations.
All the files related to this example (action handler, decision handler, form and source code) are available in the F.
Chapter 6

Task management desktop

Task management is the process of managing tasks through its life cycle. It involves planning, testing, tracking and reporting. Task management can help either individuals achieve goals, or groups of individuals collaborate and share knowledge for the accomplishment of collective goals. Tasks are also differentiated by complexity, from low to high.

Effective task management requires managing all aspects of a task, including its status, priority, time, human and financial resources assignments, recurrence, notifications and so on. These can be lumped together broadly into the basic activities of task management.

Managing multiple individual or team tasks may require specialised task management software. Specific software dimensions support common task management activities. These dimensions exist across software products and services and fit different task management initiatives in a number of ways.

Task management may form part of project management and process management and can serve as the foundation for efficient workflow in an organisation.

Software dimensions criss-crossing nearly all lines of task management products include task creation, task visualization, notifications, assign resources, compatibility, configurability, scalability and reporting.

- **Task creation** encompasses collaborative capabilities for turning ideas into actions (tasks). Includes activities involved before setting tasks, particularly patterns of collaboration involving planning.

- **Task visualization** encompasses presentation of tasks, most often through time and list forms. Priority visualization encompasses classification (e.g., budget, time, stakeholders) and mechanism (e.g., colour code or text). Calendaring covers scheduling (e.g., availability, meetings, appointments and other potential conflicts) and notifications.
Redesign and Improvement of KM Software

- **Notifications** encompasses configurable settings for informing past, present and pending deadlines.

- **Assigning resources** encompasses the ability to delegate tasks and tools to single or multiple people.

- **Compatibility** encompasses the ability of a task management environment to connect to other systems, software and environments. It includes setting a structure and restrictions on communication going from the task management environment to other software, systems and environments.

- **Configurability** encompasses ability to add, remove and manage functionality and usability in task management environments.

- **Scalability** encompasses ability to perform a task properly when a change in the quantity of users is done to meet the specific task requirements.

- **Reporting** encompasses presentation of information by displaying either in tabular or graphical display.

However, OpenKM is just offering a few of these options because despite of the fact that the administrator has a detailed overview of the state of the task during the whole life cycle, the user has only access to two lists: tasks that have been assigned to him and tasks that he might choose to complete. No further information/flexibility in the task performance is provided.

In addition, task management is a concept strongly related to time management. If we are good at managing our tasks we will probably be more efficient and, the same way, a good time management encompasses a good task management.

Time management is the act or process of planning and exercising conscious control over the amount of time spent on specific activities, especially to increase effectiveness, efficiency or productivity. Time management may be aided by a range of skills, tools, and techniques used to manage time when accomplishing specific tasks, projects and goals complying with a due date. This set encompasses a wide scope of activities, and these include planning, allocating, setting goals, delegation, analysis of time spent, monitoring, organizing, scheduling and prioritizing. Initially, time management referred to just business or work activities, but eventually the term broadened to include personal activities as well.

A time management system is a designed combination of processes, tools, techniques and methods. Usually time management is a necessity in any project development as it determines the project completion time and scope.

The major themes arising from the literature on time management include the following:

- Creating an environment conducive to effectiveness.
Redesign and Improvement of KM Software

Writers on creating an environment for effectiveness refer to issues such as the benefit of a tidy office or home to unleashing creativity, and the need to protect ”prime time”. Literature also focuses on overcoming chronic psychological issues such as procrastination.

- Setting of priorities

Which mainly focus on working in priority order and setting gravitational goals.

- Carrying out activity around those priorities.
- The related process of reduction of time spent on non-priorities.

These all four themes take us to the point of considering the possibility of adding a tool in our system to cover most of them. That way both users and administrators can benefit from a good time management which will definitely have an impact in terms of efficiency, effectiveness and productivity.

What OpenKM is currently offering consists in:

- Assignment of tasks to a single user/ more than one user.
- Graphical notification for the user when a task has been assigned to him (flicking icon).
- Distribution in either the ”Pendent tasks” list or the ”Unassigned pending tasks” list in the workflow dashboard.
- Possibility for the admin to keep track of the process (when the workflow is started, when its different steps are accomplished and when it is ended).

Nevertheless, the user has no overview about the priority or deadline of the tasks he has been assigned to. In addition, only the workflow tasks can be triggered, leaving any other duty or request out of the system which will probably lead our user to employ other task management tools or, in the worst case, to forget about those other duties.

Those will be the main issues to be covered by the new proposed desktop.

6.1 Novelties

As it was already mentioned in the previous section, the necessity of helping users to manage their time and tasks and the limitations of the available software brings us to think about a new dashboard and some new features which will be described in this section.
6.1.1 Structure

One of the most important things for having a good time management consists in setting priorities. This will help the user to distinguish those more important tasks from the ones which are unimportant and, therefore, it will be easier to focus on them.

Nowadays, there are many different techniques to set priorities which are usually based in the urgency and the importance of our tasks. Our structure is based on the Eisenhower Method which evaluates all tasks using the criteria important/unimportant and urgent/not urgent and puts them in according quadrants as shown in figure 6.1.

I have chosen this method because of its simplicity and its graphical distribution, which makes information easier to process.

This way the user can easily have an overview of its tasks and act accordingly:

- **Important and Urgent**: this will be the first tasks to accomplish. Having them highlighted will also push the user to finish them as soon as possible.

- **Important but not Urgent**: this should be the principal quadrant in the matrix. Tasks assigned to this quadrant allow programming and preparation so they can be accomplished with no hurry or stress. A deadline must be specified and are done personally.

- **Urgent but not Important**: these are the tasks that should be delegated. If not having such a possibility our user can take them into account and find the proper way/moment to finish them.
Redesign and Improvement of KM Software

- **Not Urgent, not Important**: this quadrant should be empty as its tasks should be dropped.

In addition, having all the tasks visible can be a motivational factor for the user to finish as many as possible. Although unimportant and not urgent tasks should be automatically dropped, keeping them in the matrix can also motivate us to complete our duties, as they might be the ones more related to our hobbies.

### 6.1.2 Not workflow related tasks

Right now every OpenKM user has access to a list of ”Pending tasks” that an administrator has already assigned to him and a list of ”Unassigned pending tasks” that he can choose to perform. However, all those tasks must be created and introduced to the system by an administrator. The result is that the user will not have a realistic task list because in our daily life at work we can be asked to do something during a meeting, having a conversation, etc.

Then, the possibility to include in the list non-workflow related tasks will be added to the system, thus preventing the user to have different task management tools which will bring him to a less efficient and less effective task and time management. The user will be able to define both priority and deadline for the aforementioned tasks as it is showed in figure 6.2. Moreover, in case these attributes are not specified, the tasks would stay at the pendent list instead of being directly assigned to the matrix.

![Create task](image)

**Figure 6.2: Create task**
6.1.3 Subtasks

At the moment, the workflows that are implemented might have several steps to accomplish or different small tasks for the user to fulfil. The same way, the tasks added directly by a user may need the description of one or several subtasks. By clicking on the task, the user should be able to have an overview of the task itself (date it was started, deadline, description) and also the possibility of adding subtasks as well as editing or deleting them.

The next figure shows an example of the view task.

View task: REPORT FROM CLIENT MEETING

This task is only there to demonstrate the features of the Task Manager. Here the user should find the detailed description of the task.

Deadline: 22.12.2013

Priority: High

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Deadline</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect all the data for the report</td>
<td>22.11.2013</td>
<td>High</td>
</tr>
<tr>
<td>Define report template</td>
<td>15.12.2013</td>
<td>Medium</td>
</tr>
<tr>
<td>Create the graphics for the report</td>
<td>27.11.2013</td>
<td>High</td>
</tr>
<tr>
<td>Uploading report to the system</td>
<td>01.12.2013</td>
<td>Medium</td>
</tr>
<tr>
<td>Printing report</td>
<td>01.12.2013</td>
<td>Low</td>
</tr>
</tbody>
</table>

Figure 6.3: View Task

6.1.4 Other details

To keep following the guidelines established in the design of the whole platform earlier in this project, pop-up panels for editing, deleting or adding tasks should be implemented.

Also, taking advantage of the draggable() and droppable() functions would allow users to move tasks from one place to the other in a really easy way.

Having the possibility not just to delete tasks when they are done but also to cross them out will be a positive feature for those used to have their task lists on paper or perceiving the fact of finishing many tasks as a motivation to accomplish even more.
Chapter 7
Conclusions

The introduction of Knowledge Management strategies as part of the practises of companies and other institutions shall improve the way they perform in a near future.

Both public and private organisations are showing their efforts on developing, maintaining and improving technology to support these strategies in order to reinforce themselves toward the society. Thus, open source platforms such as OpenKM are being further developed in order to fulfil the customers’ needs and avoid huge expenses on non-free software. Nonetheless, other currently hot topics can be found just beside knowledge management. The concept of soft skills has appeared as the future personal attributes that will make the difference among hundreds of well-prepared professionals. In consequence, new software design is also taking into consideration how these attributes can be improved and strengthened in order to help institutions achieve further objectives.

In this project, the use of a specific open source software that combines both knowledge and task management tools is proposed. Having access to the source code has allowed the inclusion of new features which comprise the possibility of listing information by fields, adding select-all buttons and creating pop-up menus as well as the implementation of a new, more interactive design. At the same time, the task management tool of this software has been evaluated both from the administrator and the basic-user side, and some changes have been proposed in order to enhance the impact of the platform on the users’ soft skills related to task and time management and, indirectly, have a positive impact on the organisation itself.

The first conclusion that can be drawn from this project is that the inclusion of new capabilities to the platform (listing information or select-all buttons) improves the quality of the platform itself. Thus, users can perform their tasks in a fastest and more efficient way. Allowing users to access existing information which can be taken as an example for the task they intend to accomplish will definitely help them to succeed and be more confident on the use of the software. In addition, these features reinforce one of the basics of knowledge management: making information easily accessible (including the
Redesign and Improvement of KM Software

information of the system itself). Furthermore, ajax allows retrieving data from the server asynchronously which implies saving time and resources small-scale.

Other of the objectives of this project consisted in giving a new design to the platform and it can be concluded that this objective has definitely been accomplished. This new design includes a new blue based colour palette which erases the associations with boredom and emptiness related to grey, more organic shapes and fonts and an increase of the interactivity user-platform achieved by the addition of distinguishing attributes to elements that may perform a task: hand shaped pointer when hovering over a button; colour of buttons being altered when the user hovers over them; background colour of the selected item being altered; text leading into a link underlined when hovering over it, etc.

The conclusion drawn after getting to know how workflow works and trying out its elements in the different simulations is that workflow is a very powerful tool with lots of possibilities and applications. However, being able to create mid or high complexity workflows will require programming knowledge on XML (for forms) and java (for actions) which could lead to an investment in complementary training by the company side. Additionally, the use of external software is needed to create the .par file that contains the workflow. This means that OpenKM cannot actually carry out all the functions it is meant for independently.

The last of the objectives of this project consisted in evaluating the current task management tool, basically composed by the workflow view panel, and proposing a new task management desktop. In order to fulfil this objective, novelties regarding both structure and functionalities of the current system were taken into consideration. The proposed structure is based on the Eisenhower Method in order to have a clear understanding of the status of the tasks in a visual way. New functionalities include being able to set priority and deadline to a workflow when it started. With the aim of building a totally independent platform which prevents users to work with similar software, in this new version also tasks not related to workflow can be added to the system.

Thus, it can be concluded that the proposed task manager is an improvement in comparison with the original one, being more flexible and offering more possibilities to OpenKM users.
Chapter 8

Future Work

Future lines in the design and the improvement of features of OpenKM keep open based on this project.

In the side of the design, the way it evolves and gets modified is totally open and related to the evolution of the platform itself. What users need, what OpenKM offers.

In the side of the task manager, this document includes the necessary information to modify the current system. It is described how the structure could be implemented and which functions could help users managing their tasks in an easier way. Hence, OpenKM source code can be easily modified to provide users with this new feature.

In the side of the workflow, it can be used not only to keep track of tasks but also to improve users’ quality of performance. Thus, a field that could be studied is how workflows can be developed in an easier way. At the moment, designing a workflow is a quite easy process regarding the structure and the creation of node tasks. However, the implementation of actions still needs some advanced knowledge. At the same time, workflows are thought to be accomplished by one single person (or a few of them) but there could also be place for tasks assigned to a group or team. This means moving towards a more transversal system in the sense that one user could be working in different teams at the same time, or hold a position in the company, and workflow could be assigned to that user, several users, a whole team or maybe the users holding the same position in different temps (e.g. all team leaders). Then, not only the way of creating a workflow but also the way a workflow is started and assigned should be studied and possibly modified.

As seen, future lines keep open after this project. Nevertheless, future projects involving OpenKM or other platforms with similar proposes could use this project as base and make further implementations.
Appendix A

OpenKM Specifications

OpenKM architecture is based on

- Apache Tomcat
- Java J2EE (JDK 1.6)
- GWT (Google Web Toolkit - Ajax)
- Hibernate
- Spring
- Lucene Text Search Engine
- POI File Format Conversion
- OpenOffice
- jBPM
- KEA
- Weka
- Aperture
- Rdf2go
- Openrdf

OpenKM supported interfaces

- Webdav
Redesign and Improvement of KM Software

- Webservices

Operating system support

- Unix
- GNU / Linux
- MacOS

Backend storage repository
Either a local filesystem or DBMS can be used to store information:

- Oracle
- PostgreSQL
- MySQL
- MS SQL Server
- etc.
Appendix B

Courses

Codecademy is an online interactive platform that offers free coding classes in programming languages like Python, PHP, JavaScript, and Ruby, as well as mark-up languages including HTML and CSS. The site offers feedback, badges for completing exercises, as well as a function that keeps track of a user’s total score and displays it to others.

Code Year is a free Codecademy program for anyone who is interested to learn how to program. The program intends to help people follow through on a New Year’s Resolution to learn how to program, by introducing a new course for every week. The different sections that I completed are the following:

- Web Fundamentals (160 exercises) - Introduction to HTML and CSS
- Web (Original) (260 exercises) - HTML, CSS and JavaScript on the web
- JavaScript (216 exercises) - Making websites interactive and building browser based games
- jQuery (68 exercises) - Introduction to jQuery
- jQuery (Original) (162 exercises) - Advanced jQuery
- PHP (86 exercises) - Introduction to PHP
- Python (296 exercises) - Introduction to Python
Appendix C

Platform final design

Figure C.1: Main Desktop - Old Design
Redesign and Improvement of KM Software

Figure C.2: Main Desktop - New Design
Appendix D

CSS files

This appendix includes the two most important CSS files of the platform:

- **styles-new.css**: CSS for the main desktop, the dashboard panel and the search panel.
- **styles-nou.css**: CSS for the administration desktop.
background-repeat: repeat;
background-position: left;
background-color: #0071B9;
padding: 0px 0px 0px 0px;
cursor: default;
font-size: 12px;
border: 1px solid #989696;
}

.okm-MainMenuItem {
  padding: 0px 20px 0px 0px;
cursor: default;
font-size: 12px;
height: 20px;
white-space: nowrap;
}

.okm-TopMenuBar .gwt-MenuItem-selected {
  cursor: pointer;
  font-size: 12px;
  /*border-right: 1px solid #08246d;
   border-left: 1px solid #08246d;
   border-top: 1px solid #08246d;*/
  background-color: #C3D9FF;
  white-space: nowrap;
}

.okm-SubMenuBar .gwt-MenuItem-selected {
  background-color: #E9F4FB;
  color: #0071B9;
  font-size: 12px;
  cursor: pointer;
  /*border: 1px solid #08246d;*/
  white-space: nowrap;
}

.okm-MainMenuItem-Base-HorizontalSeparator {
  background-image: url(img/menu_horizontal_separator.gif);
  background-position: center right;
  background-repeat: repeat-x;
}

.okm-MainMenuItem-Base-Childs {
  background-image: url(img/menu_arrow.gif);
  background-repeat: no-repeat;
  background-position: center right;
}

/* menu bar */
.okm-MenuBar {
  background-image: url(img/back_menuitem_new.gif);
  cursor: default;
  font-size: 12px;
}

.okm-MenuItem {
  font-size: 12px;
  cursor: default;
padding: 0px 20px 0px 0px;
}

.okm-MenuItem-strike {
  cursor: default;
  font-size: 12px;
  color: #8D8D7A;
}

.okm-MenuBar .gwt-MenuItem-selected {
  background-color: #E9F4FB;
  color: #0071B9;
  font-size: 12px;
  cursor: pointer;
  /*border: 1px solid #08246d;*/
  white-space: nowrap;
}

/* general for all menus it seems to be necessary to have correct font size */
.gwt-MenuItem {
  font-size: 12px;
}
/* tree */
.okm-Tree {
    white-space: nowrap;
    font-size: 12px;
    cursor: default;
}

.okm-Tree .okm-TreeItem {
    white-space: nowrap;
}

.okm-Tree .gwt-TreeItem-selected {
    white-space: nowrap;
    background-color: #C3D9FF;
}

.okm-Tree-MenuPopup {
    background-color: #ECEBEB;
    border: 1px solid #AAAAAA;
    padding: 1px;
}

.okm-Tree-TextBox-selected {
    background-color: #E9F4FB;
    border: 1px solid #A5A596;
    font-size: 10px;
    font-family: tahoma, verdana, arial, sans-serif
}

.okm-Tree-TextBox {
    background-color: #FFFFFF;
    border: 1px solid #A5A596;
    font-size: 10px;
    font-family: tahoma, verdana, arial, sans-serif
}

gwt-TreeItem {
    white-space: nowrap;
    cursor: default;
}

/* Popup */
.okm-Popup, .okm-Popup-Error {
    font-size: 12px;
    background-color: #E9F4FB;
    border: 1px solid #A5A596;
}

.okm-Popup-label, .okm-Popup-Error-label {
    font-size: 12px;
    font-weight: bold;
    text-align: center;
    padding-top: 5px;
    height: 20px;
    background-image: url(img/popup_label_back.gif);
    background-repeat: repeat;
}

.okm-Popup-text {
    background: #FEFCF8;
    border: 1px solid #A5A596;
}

.okm-Popup .Caption {
    font-size: 12px;
    font-weight: bold;
    text-align: center;
    padding-top: 5px;
    height: 20px;
    background-image: url(img/popup_label_back.gif);
    background-repeat: repeat;
    cursor: pointer;
    cursor: hand;
}

.okm-Popup-Error .Caption {
    font-size: 12px;
    font-weight: bold;
}
/* VerticalBar */
.okm-VerticalBar {
  background-image: url(img/separator.gif);
  background-repeat: repeat;
  cursor: col-resize;
}

.okm-VerticalBar-Point {
  background-image: url(img/separator_points.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: col-resize;
  font-family: "Small Fonts";
  font-size: 4px;
}

/* HorizontalBar */
.okm-HorizontalBar {
  background-image: url(img/horizontal_separator.gif);
  background-repeat: repeat;
  cursor: row-resize;
}

.okm-HorizontalBar-Point {
  background-image: url(img/horizontal_separator_points.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: row-resize;
  font-family: "Small Fonts";
  font-size: 4px;
}

/* Solution to IE transparent popup <iframe> problem */
iframe {
  filter: progid : DXImageTransform . Microsoft . Alpha(opacity = 0);
}

.okm-Popup-HorizontalBar {
  background-image: url(img/horizontal_separator_blue.gif);
  background-repeat: repeat;
  cursor: row-resize;
}

.okm-Popup-HorizontalBar-Point {
  background-image: url(img/horizontal_separator_points_blue.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: row-resize;
  font-family: "Small Fonts";
  font-size: 4px;
}

/* File browser */
.okm-FileBrowser {
  cursor: default;
}
.okm-ToolBar-selected {
    background-image: url(img/bottom_selected.gif);
    background-repeat: no-repeat;
}

.okm-ToolBar-Big-selected {
    background-color: #C3D9FF;
    cursor: pointer;
    cursor: hand;
}

.okm-ToolBar-Big-selected .okm-NoWrap {
    color: #000000;
}

.okm-ToolBar-BigTMP-selected {
    background-color: #C3D9FF;
    color: #0071B9;
    cursor: pointer;
    cursor: hand;
}

.okm-ToolBar-BigTMP-selected .okm-NoWrap {
    color: #000000;
}

.okm-ToolBar-Border {
    border-top: 1px solid #A5A596;
    border-left: 1px solid #A5A596;
    border-right: 1px solid #A5A596;
}

/* File upload */
.okm-FileUploadPopup {
    font-size: 12px;
    background-color: #EFEBE7;
    border: 1px solid #110011;
}

.okm-FileUploadPopup-label {
    font-size: 12px;
    text-align: center;
    background-image: url(img/errorpopup_label_back.gif);
    background-repeat: repeat;
}

.okm-FileUploadPopup-prog-border {
    height: 15px;
    width: 205px;
    background: #fff;
    border: 1px solid #000;
    margin: 0;
    padding: 0;
}

.okm-FileUploadPopup-prog-bar {
    height: 11px;
    margin: 2px;
    padding: 0px;
    background: #C3D9FF;
}

/* Tab panel */
.gwt-TabPanel {
    white-space: nowrap;
}

.gwt-TabPanelBottom {
    /* border-left: 1px solid #87B3FF; */
    white-space: nowrap;
    /*padding-bottom: 5px;
    border-bottom: 1px solid #bbb;
    border-left: 1px solid #bbb;
    border-right: 1px solid #bbb; */
}

.gwt-TabBar {
    white-space: nowrap;
    background-color: #E9F4FB;
}
background-color: #FFFFFF;
}

.okm-PanelUnselected {
  background-color: #F6F6EE;
}

/* Logout */
.okm-LogoutPopup {
  font-size: 12px;
  background-color: #EFEBE7;
  border: 1px solid #110011;
}

.okm-LogoutPopup-label {
  font-size: 12px;
  text-align: center;
  background-image: url(img/logoutpopup_label_back.gif);
  background-repeat: repeat;
}

/* Tab Version */
.okm-Version-SelectedRow {
  background-image: url(img/version_selected.gif);
  background-repeat: repeat-x;
  cursor: default;
}

.okm-Version-Title {
  background-image: url(img/table_title.gif);
  background-repeat: repeat;
  color: #6D6D5F;
  font-size: 10px;
  text-align: left;
  padding: 4px 4px 4px 4px;
  height: 22px;
  cursor: default;
}

.okm-Version-Title-Margin {
  padding: 4px 4px 4px 4px;
}

.okm-Version-Title-RightBorder {
  border-right: 1px solid #A5A596;
}

.okm-Version-Title-LeftBorder {
  border-left: 1px solid #FFFFFF;
}

.okm-Version-Title-RightBottomBorder {
  border-bottom: 1px solid #A5A596;
  border-right: 1px solid #A5A596;
}

/* Tab Security */
.okm-Security-Title {
  background-image: url(img/table_title.gif);
  background-repeat: repeat;
  color: #6D6D5F;
  font-size: 10px;
  text-align: left;
  padding: 4px 4px 4px 4px;
  height: 22px;
  cursor: default;
}

.okm-Security-Title-Margin {
  padding: 4px 4px 4px 4px;
}

.okm-Security-Title-RightBorder {
  border-right: 1px solid #A5A596;
}

.okm-Security-Title-LeftBorder {
  border-left: 1px solid #FFFFFF;
}

.okm-Security-Title-RightBottomBorder {
  border-bottom: 1px solid #A5A596;
  border-right: 1px solid #A5A596;
}
padding: 5px;
}

/* Draggable */
.okm-Draggable {
  background-color: #C3D9FF;
}

/* Disable text selection */
.okm-DisableSelect {
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: -moz-none;
  -o-user-select: none;
  user-select: none;
}

/* Enable text selection */
.okm-EnableSelect {
  -webkit-user-select: text !important;
  -khtml-user-select: text !important;
  -moz-user-select: text !important;
  -o-user-select: text !important;
  user-select: text !important;
}

/* Bookmark */
.okm-Bookmark-Panel {
  border: 1px solid #A5A596;
  background: white;
}

/* ScrollTable */
.gwt-ScrollTable {
  background: white;
}
.gwt-ScrollTable .headerWrapper {
  background-color: #C3D9FF;
  /*background-image: url(img/table_title.gif);
  background-repeat: repeat;*/
}
.gwt-ScrollTable .headerTable tr {
  /*background-image: url(img/table_title.gif);
  background-repeat: repeat;*/
  background-color: #C3D9FF;
  border: 1px solid #E9F4FB;
  color: #000000;
  font-size: 10px;
  text-align: center;
  height: 22px;
  cursor: default;
}
.gwt-ScrollTable .headerTable tr td {
  border-left: 1px solid white;
  border-right: 1px solid #A5A596;
  white-space: nowrap;
  margin: 0px;
  overflow: hidden;
  height: 22px;
}
.gwt-ScrollTable .headerTable .gwt-controller tr td {
  border-left: 0px;
  border-right: 0px;
}
.gwt-controller .separator {
  border-right: 1px solid white;
  border-left: 1px solid #A5A596;
}
.gwt-ScrollTable .dataTable tr {
  background: white;
}
.gwt-ScrollTable .dataTable tr:hover {
  background-color: #e0ecff;
}
.gwt-ScrollTable .dataTable tr:hover td {
  border-left: 1px solid #e0ecff;
  border-right: 1px solid #e0ecff;
}

.gwt-ScrollTable .dataTable tr.selected {
  background-color: #E9F4FB;
  height: 22px;
}

.gwt-ScrollTable .dataTable tr.selected td {
  border-left: 1px solid #E9F4FB;
  border-right: 1px solid #E9F4FB;
}

.gwt-ScrollTable .dataTable tr td {
  border-left: 1px solid white;
  border-right: 1px solid white;
  height: 20px;
  cursor: default;
  white-space: nowrap;
  margin: 0px;
  overflow: hidden;
}

/**
 * CalendarWidget
 */
.CalendarWidget {
  border: 1px solid #0071B9;
  background-color: #E9F4FB;
  padding: 2px;
  font-family: Tahoma, Geneva, sans-serif;
}

.CalendarWidget .navbar {
  width: 100%;
  background-color: #E9F4FB;
  vertical-align: middle;
  border-bottom: 1px solid #ACA899;
}

.CalendarWidget .navbar .gwt-Button {
  padding-left: 5px;
  padding-right: 5px;
}

.CalendarWidget .table {
  font: 10pt sans-serif;
  text-align: center;
  background-color: #E9F4FB;
}

.CalendarWidget .weekheader {
  background-color: #C3D9FF;
}

.CalendarWidget .weekheader .days {
  width: 3em;
}

.CalendarWidget .cell {
  cursor: pointer;
}

.CalendarWidget .cell .gwt-HTML {
  border: 1px solid #ACA899;
  background-color: #EEE;
}

.CalendarWidget .cell .gwt-HTML span {
  width: 100%;
  height: 100%;
  line-height: 2em;
}

.CalendarWidget .today .gwt-HTML {
  background-color: #C3D9FF;
}
/* DashBoard Widget */
.okm-DashboardWidget .topLeft {
    background-image: url(img/topLeftHeader-new.gif);
    background-repeat: no-repeat;
    cursor: pointer;
}

.okm-DashboardWidget .topCenter {
    background-image: url(img/topCenterHeader-new.gif);
    background-repeat: repeat-x;
    text-align: center;
    cursor: pointer;
}

.okm-DashboardWidget .topRight {
    background-image: url(img/topRightHeader-new.gif);
    background-repeat: no-repeat;
    cursor: pointer;
}

.okm-DashboardWidget .data {
    background-color: white;
    border-left: 1px solid #A5A596;
    border-bottom: 1px solid #A5A596;
    border-right: 1px solid #A5A596;
}

.okm-DashboardWidget .data tr:hover {
    background-color: #e0ecff;
}

.okm-DashboardWidget .data td {
    height: 20;
    padding: 0px 4px;
}

/* Hyperlink */
.okm-Hyperlink {
    text-decoration: underline;
    cursor: pointer;
    cursor: hand;
}

.okm-Hyperlink a:hover {
    text-decoration: underline;
    color: blue;
}

.okm-Hyperlink a {
    text-decoration: underline;
    color: black;
}

/* News */
.okm-NotViewed {
    font-weight: bold;
}

/* Workflow form panel */
.okm-WorkflowFormPanel {
    background-color: #E9F4FB;
    border-left: 1px solid #A5A596;
}

.okm-WorkflowFormPanel-Title {
    white-space: nowrap;
    padding: 2px 7px 2px 7px;
}
filter: alpha(opacity=40); zoom: 1;

*/

**ToggleButton**

gwt-ToggleButton-up,
gwt-ToggleButton-up-hovering,
gwt-ToggleButton-up-disabled,
gwt-ToggleButton-down,
gwt-ToggleButton-down-hovering,
gwt-ToggleButton-down-disabled {
  margin: 0;
text-decoration: none;
background: url("img/hborder.png") repeat-x 0px -27px;
}

gwt-ToggleButton-up,
gwt-ToggleButton-up-hovering,
gwt-ToggleButton-up-disabled {
padding: 3px 5px 0px 5px;
height: 20px;
}

gwt-ToggleButton-up {
border: 1px outset #ccc;
cursor: pointer,
cursor: hand;
}

gwt-ToggleButton-up-hovering {
border: 1px outset;
border-color: #9cf #69e #69e #7af;
cursor: pointer;
cursor: hand;
}

gwt-ToggleButton-up-disabled {
border: 1px outset #ccc;
opacity: .5;
zoom: 1;
filter: alpha(opacity=40);
}

gwt-ToggleButton-down,
gwt-ToggleButton-down-hovering,
gwt-ToggleButton-down-disabled {
padding: 4px 4px 2px 6px;
}

gwt-ToggleButton-down {
background-position: 0 -513px;
border: 1px inset #ccc;
cursor: pointer;
cursor: hand;
}

gwt-ToggleButton-down-hovering {
background-position: 0 -513px;
border: 1px inset;
border-color: #9cf #69e #69e #7af;
cursor: pointer;
cursor: hand;
}

gwt-ToggleButton-down-disabled {
background-position: 0 -513px;
border: 1px inset #ccc;
cursor: default;
opacity: .5;
zoom: 1;
filter: alpha(opacity=40);
}

/* RichTextArea */

RichTextToolbar {
  background: #E9F4FB;
border: 1px #0071B9;
}
/* Mail */
.okm-Mail {
  background-color: #E9F4FB;
}

.okm-Mail-White {
  background-color: #FFFFFF;
}

.okm-Mail-Link {
  cursor: pointer;
  cursor: hand;
}

.okm-Mail-MenuPopup {
  background-color: #E9F4FB;
  border: 1px solid #AAAAAA;
  padding: 1px;
}

/* Horizontal split panel */
.hsplitter {
  background-image: url(img/separator_points.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: col-resize;
  font-family: "Small Fonts";
  font-size: 4px;
  background-color: #E9F4FB;
  horizontal-align: 50%;
  border-left: 1px solid #A5A596;
  border-right: 1px solid #A5A596;
  width: 10px;
}

.okm-HorizontalSplitPanel .hsplitter {
  background-image: url(img/separator_points.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: col-resize;
  font-family: "Small Fonts";
  font-size: 4px;
  background-color: #E9F4FB;
  horizontal-align: 50%;
  border: 0px;
  width: 10px;
}

/* Vertical split panel */
.vsplitter {
  background-image: url(img/horizontal_separator_points.gif);
  background-repeat: no-repeat;
  background-position: 50% 50%;
  cursor: row-resize;
  font-family: "Small Fonts";
  font-size: 4px;
  background-color: #E9F4FB;
  height: 10px;
  vertical-align: 50%;
}

/* KeyMapControl panel */
.okm-KeyMapControl {
  background-color: #C3D9FF;
  border-bottom: 1px solid #A5A596;
}

.okm-PaginationControl {
  background-color: #C3D9FF;
  border-top: 1px solid #A5A596;
}

.okm-KeyMap-Selected {
  background-image: url(img/filebrowser_selected.gif);
  background-repeat: repeat-x;
}
.whitePanel {
  background-color: white;
}

/* Chrome fix for button images */
@media screen and (-webkit-min-device-pixel-ratio:0) {
  .okm-NoButton, .okm-YesButton, .okm-AddButton, .okm-DeleteButton, .okm-DownloadZipButton,
  .okm-HomeButton, .okm-DownloadButton, .okm-ViewButton, .okm-ChangeButton, .okm-CompactButton,
  .okm-CleanButton, .okm-SearchButton, .okm-SaveButton, .okm-CommentButton {
    padding-left: 20px;
  }
}

/* Explorer fix for button images */
<!--[if IE]>
  .okm-NoButton, .okm-YesButton, .okm-AddButton, .okm-DeleteButton, .okm-DownloadZipButton,
  .okm-HomeButton, .okm-DownloadButton, .okm-ViewButton, .okm-ChangeButton, .okm-CompactButton,
  .okm-CleanButton, .okm-SearchButton, .okm-SaveButton, .okm-CommentButton {
    padding-left: 12px;
  }
<![endif]-->
body {
    background-color: #FFFFFF;
    margin: 0;
    padding: 0;
    width: 100%;
}

body, td, a, div, p {
    font-size: 12px;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}

a:link, .w, .w a:link {
    color: #0071B9;
    text-decoration: none;
}

a:visited {
    color: #551a8b;
    text-decoration: none;
}

a:active {
    color: #FF0000;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

a.button {
    background-color: #E9F4FB;
    border: 1px solid #0071B9;
    border-radius: 5%;
    padding: 3px 5px 3px 5px;
    font-size: 12px;
    font-weight: bold;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}

h1 {
    text-align: center;
    color: #0071B9;
}

h2 {
    text-align: left;
    padding-left: 10px;
    color: #0071B9;
}

img {
    border: 0;
}

ul {
    margin: 0px;
    padding-left: 30px;
}

textarea, input, select {
    border: 1px solid #A5A596;
    font-size: 12px;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}

input.submit {
    background-color: #E5E5E1;
    border: 1px solid #A5A596;
    font-size: 12px;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}

input.but {
    background-color: #0071B9;
    border: 1px solid #3079ED;
    border-radius: 5%;
}
color: #FFFFFF;
font-weight: bold;
cursor: pointer;
min-width: 55px;
height: 22px;
}

input.but:hover {
  background-color: #FFFFFF;
  color: #0071B9;
  border: 2px solid #0071B9;
}

input#fumi:hover {
  cursor: pointer;
  font-weight: bold;
}

task:hover{
  cursor: pointer;
}
.ui-dialog{
  font-size: 14px;
  font-family: Tahoma, sans-serif;
  background: #E9F4FB;
  border: #0071B9;
}
.ui-dialog .ui-dialog-titlebar {
  background: #0071B9;
  border: #0071B9;
  color: #FFFFFF;
}
.ui-dialog.ui-dialog-titlebar-close{
  background: #FFFFFF;
  color: #0071B9;
}
.ui-widget-content {
  border: 1px solid #0071B9;
}
.table.form {
  margin-left: auto;
  margin-right: auto;
  border: 1px solid #0071B9;
  background-color: #E9F4FB;
}
.table.form#home {
  border-radius: 5%;
  text-align: center;
  width: 250px;
  height: 200px;
}
.table.form td.imp {
  color: #0071B9;
}
.table.form tr.fuzzy {
  background-color: #C3D9FF;
}
.table.form th {
  font-weight: bold;
  font-size: 12px;
  height: 16px;
  background-image: url("../img/table_title.gif");
  border: 1px solid #A5A596;
}
.table.results {
  margin-left: auto;
  margin-right: auto;
  margin-bottom: 15px;
  border: 1px solid #789DB3;
  border-collapse: collapse;
}
Appendix E

JSP files

This appendix includes the user_list.jsp file, one of the 87 JSP files of the administration desktop which have been modified in this project.
var $el = $(s),
  $img = $el.find('img');
return $img.length ? $img[0].src : $el.text();
}
headers: {
  0: {
    sorter : false
  },
  8: {
    sorter : false
  }
},
widgets: ['zebra'],
});

function ajaxEdit(id){
  $.ajax({
    type: "GET",
    url: "Auth",
    data: {action : "userEdit", usr_id : id},
    success: function(data){
      $('#dialog-form').empty().html(data);
      $('form-cancel').click(function (){
        $('#dialog-form').dialog("close");
      });
    },
  });
}

function ajaxCreate(){
  $.ajax({
    type: "GET",
    url: "Auth",
    data: {action : "userCreate"},
    success: function(data){
      $('#dialog-form').empty().html(data);
      $('form-cancel').click(function (){
        $('#dialog-form').dialog("close");
      });
    },
  });
}

function ajaxDelete(id){
  $.ajax({
    type: "GET",
    url: "Auth",
    data: {action : "userDelete", usr_id : id},
    success: function(data){
      $('#dialog-form').empty().html(data);
      $('form-cancel').click(function (){
        $('#dialog-form').dialog("close");
      });
    },
  });
}
</script>
<title>User List</title>
<body>
<c:set var="isAdmin" ><%=BaseServlet.isAdmin (request)%></c:set>
<u:constantsMap className="com.openkm.core.Config" var="Config" />
<c:choose>
  <c:when test= "$(isAdmin)" >
    <c:url var="messageList" value="LoggedUsers" >
      <c:param name="action" value="messageList" ></c:param>
    </c:url>
  </c:when>
  <c:choose>
    <c:when>
      <ul id="breadcrumb">
        <li class="path">
          <a href= "Auth">User list</a>
        </li>
        <li class="action">
          <a href= "Auth?action=roleList" >Role list</a>
        </li>
        <li class="action">
          <a href= "LoggedUsers" >Logged users</a>
        </li>
      </ul>
    </c:when>
  </c:choose>
</c:choose>
<table class="form">
  <tr>
    <td><strong>Role</strong></td>
    <td>
      <select name="roleFilter">
        <option value=""></option>
        <c:forEach var="role" items="${roles}">
          <c:choose>
            <c:when test="${role.id == roleFilter}"
              option value="${role.id}" selected="selected">
              ${role.id}
            </c:when>
            <c:otherwise>
              <option value="${role.id}">
                ${role.id}
              </option>
            </c:otherwise>
          </c:choose>
        </c:forEach>
      </select>
    </td>
  </tr>
  <tr>
    <td colspan="2" align="right">
      <input type="submit" value="Search" class="but" />
    </td>
  </tr>
</table>

<br/>
<div style="text-align: center; id=dest">
  <input type="button" id="fumi" value="Force user mail import" />
</div>
<br/>
<table id="tablesorter" class="results" width="80%">
  <thead>
    <tr>
      <th>Id</th>
      <th class="sort">Id</th>
      <th class="sort">Name</th>
      <th class="sort">Roles</th>
      <th class="sort">Profile</th>
      <th class="sort" width="80px">Active</th>
      <th class="sort" width="80px">Chat</th>
      <th width="150px">
        <c:url value="Auth" var="urlCreate">
          <c:param name="action" value="userCreate" />
        </c:url>
        <c:if test="${db}"
          a href="${urlCreate}">
          <img id="create" class="task" src="img/action/new.png" alt="New user" title="New user" />
        </c:if>
      </th>
    </tr>
  </thead>
  <tbody>
    <c:forEach var="user" items="${users}" varStatus="row">
      <c:url value="Auth" var="urlEdit">
        <c:param name="action" value="userEdit" />
        <c:param name="usr_id" value="${user.id}" />
      </c:url>
      <c:url value="Auth" var="urlDelete">
        <c:param name="action" value="userDelete" />
        <c:param name="usr_id" value="${user.id}" />
      </c:url>
      <c:url value="Auth" var="urlActive">
        <c:param name="action" value="userActive" />
        <c:param name="usr_id" value="${user.id}" />
        <c:param name="roleFilter" value="${roleFilter}" />
        <c:param name="usr_active" value="${!user.active}" />
      </c:url>
      <c:url value="Auth" var="urlChatDisconnect">
        <c:param name="action" value="userChatDisconnect" />
        <c:param name="usr_id" value="${user.id}" />
        <c:param name="roleFilter" value="${roleFilter}" />
      </c:url>
      <c:url value="Auth" var="urlChatDisconnect">
        <c:param name="action" value="userChatDisconnect" />
        <c:param name="usr_id" value="${user.id}" />
        <c:param name="roleFilter" value="${roleFilter}" />
      </c:url>
      <!--a href="${urlCreate}"-->
      <img id="create" class="task" src="img/action/new.png" alt="New user" title="New user" />
      <!--a href="${urlCreate}"-->
    </c:forEach>
  </tbody>
</table>
<tr class="${row.index % 2 == 0 ? 'even' : 'odd'}">
    <td width="20px">${row.index + 1}</td>
    <td id="td-userid-${row.index}">${user.id}</td>
    <td>${user.name}</td>
    <td>${user.email}</td>
    <td id="td-role-${row.index}">
        <c:forEach var="role" items="${user.roles}">
            ${role.id}
        </c:forEach>
    </td>
    <td>${user.profile}</td>
    <td class="c">
        <c:choose>
            <c:when test="${db}">
                <a href="${urlActive}">
                    <img src="img/true.png" alt="Active" title="Active" />
                </a>
            </c:when>
            <c:otherwise>
                <a href="${urlActive}">
                    <img src="img/false.png" alt="Inactive" title="Inactive" />
                </a>
            </c:otherwise>
        </c:choose>
    </td>
    <td align="center">
        <c:if test="${db}">
            <a href="${urlEdit}">
                <img id="edit" name="edit-${row.index}" class="task" src="img/action/edit.png" alt="Edit" title="Edit" />
            </a>
        </c:if>
        <a href="${urlConfig}">
            <img src="img/action/config.png" alt="User config" title="User config" />
        </a>
        <a href="${urlMail}">
            <img src="img/action/email.png" alt="Mail accounts" title="Mail accounts" />
        </a>
        <a href="${urlTwitter}">
            <img src="img/action/twitter.png" alt="Twitter accounts" title="Twitter accounts" />
        </a>
    </td>
</tr>
Appendix F

Invoice Example - Figures

```xml
<workflow-forms>
  <workflow-form task="run_config">
    <input name="invoice" label="Invoice" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="reviser">
    <button name="review" label="Need revision" transition="userReview" />
    <button name="cancel" label="Cancel" transition="cancel" />
    <button name="accept" label="Accept" transition="reviewed" />
  </workflow-form>
  <workflow-form task="approver">
    <button name="review" label="Need revision" transition="review" />
    <button name="cancel" label="Cancel" transition="cancel" />
    <button name="accept" label="Accept" transition="approved" />
  </workflow-form>
  <workflow-form task="user">
    <button name="accept" label="Accept" transition="review" />
  </workflow-form>
</workflow-forms>

Figure F.1: Form
Redesign and Improvement of KM Software

Figure F.2: Source code
Redesign and Improvement of KM Software

```java
package com.openkmworkflow.decision;

import org.jbpm.graph.exe.ExecutionContext;
import org.jbpm.graph.node.DecisionHandler;
import com.openkm.bean.form.Input;

public class InvoiceDecision implements DecisionHandler {
    @Override
    public String decide(ExecutionContext executionContext) throws Exception {
        // TODO Auto-generated method stub
        Input number = (Input) executionContext.getContextInstance().getVariable("invoice");
        if (number!=null){
            int value = Integer.valueOf(number.getValue());
            if (value>1000){
                return "review";
            }else{
                return "approve";
            }
        }else{
            return "approve";
        }
    }
}
```

Figure F.3: Decision handler

```java
package com.openkm.workflow.action;

import org.jbpm.graph.def.ActionHandler;
import org.jbpm.graph.exe.ExecutionContext;

public class InvoiceCancelAction implements ActionHandler {
    @Override
    public void execute(ExecutionContext executionContext) throws Exception {
        // TODO Auto-generated method stub
        System.out.println("Invoice has been canceled");
    }
}
```

Figure F.4: Action handler
Appendix G

Modified Files

All the files that have been modified for this project can be found in the following link: https://dl.dropboxusercontent.com/u/4038287/Last%20Report.rar
Bibliography

[1] wiki.openkm.com
[2] openkm.com
[3] wikipedia.com
[4] jqueryui.com
[5] javascriptsource.com
[6] codecademy.com
[7] stackoverflow.com
[8] www.colormatters.com
[9] knightdalehs.wcpss.net/student_life/fine_arts/Departments/Fine_Art/Collin/Documents/
[10] www.webdesign.org