FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Web Collaboration for Software Engineering

Tiago Mourão Teixeira

Report of Dissertation

Master in Informatics and Computing Engineering

Supervisor: Ademar Manuel Teixeira de Aguiar (PhD)

Second Supervisor: Nuno Honório Rodrigues Flores (MsC)

28^{th}July, 2009

Web Collaboration for Software Engineering

Tiago Mourão Teixeira

Report of Dissertation
Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: António Manuel Lucas Soares (PhD)

External Examiner: António Manuel Ferreira Rito da Silva (PhD)
Internal Examiner: Ademar Manuel Teixeira de Aguiar (PhD)

28^{th}July, 2009

Abstract

In a global market scenario and with an increasing demand for more complex and large projects, software engineers must be able to execute computer-supported collaborative work in multiple and distributed locations as effectively as possible.

The worldwide acceptance of the Internet makes web-based tools the ideal solution for this problem. However, there are two scientific gaps that contribute to the slower migration of Software Engineering tools to the Web. On the one hand, there is a lack of Web-based environments covering the entire life cycle of a project. On the other hand, there are few studies quantifying the pros and cons of using, in that environment, several integrated collaboration features.

In order to fill the first gap, this thesis began by researching the areas of collaborative software – mainly Groupware and Computer-Supported Collaborative Work – to have insight how people work in a team, which technologies to choose to aid teamwork, and how to adapt those to the team's needs. Afterwards, a study of the pros and cons of collaborative software was performed, concerning tools specifically designed for, or adaptable to, Software Engineering activities. The next phase was to narrow the research to Software Engineering, where an thorough investigation of the collaboration features of existing software products was made. Those results were then used to rationally choose the set of features to implement in a prototype. The prototype was developed on Ruby on Rails and had those features integrated as plugins in Redmine, a web-based project management software.

For the purpose of filling the second gap, an experiment was designed to test the prototype in a Software Engineering project. The experience should be executed as soon as possible, due to the fact that were wasn't time to conduct it in the context of this thesis.

Resumo

Num cenário de globalização e perante a exigência crescente por projectos cada vez maiores e mais complexos, os engenheiros de software têm que ser capazes de executar actividades colaborativas suportadas por computadores, em múltiplos locais geograficamente distribuídos, o mais eficazmente possível.

A globalização da Internet permite às ferramentas Web serem a solução ideal para resolver este problema. Contudo, há duas lacunas científicas que contribuem para a lentidão da migração de ferramentas usadas em Engenharia de Software para a Web. Por um lado, há falta de ferramentas Web que abranjam todo o ciclo de vida de um projecto. Por outro lado, há poucos estudos quantificando os prós e contras de usar funcionalidades de colaboração integradas numa ferramenta Web.

De forma a colmatar a primeira lacuna, esta tese começou por investigar as áreas de software colaborativo – nomeadamente Groupware e Computer-Supported Collaborative Work – para perceber como decorre o trabalho em equipa, quais as tecnologias a escolher para o suportar e como as adaptar às necessidades da equipa. De seguida, foi realizado um estudo dos prós e contras de software colaborativo, relativamente a ferramentas concebidas especificamente para, ou adaptáveis a, actividades de Engenharia de Software. A próxima fase consistiu em restringir a investigação a domínios específicos de Engenharia de Software, resultando num estudo exaustivo das funcionalidades de colaboração existentes em produtos de software. Estes resultados foram posteriormente usados para escolher o conjunto de funcionalidades a implementar num protótipo. O protótipo foi desenvolvido na framework web Ruby on Rails e teve as suas funcionalidades integradas como *plugins* no Redmine, um software web para gestão de projectos.

A segunda lacuna foi parcialmente resolvida através da definição de uma experiência, com o objectivo de testar o protótipo num projecto de Engenharia de Software. A experiência deverá ser executada assim que possível, uma vez que não houve tempo para o fazer no âmbito desta tese.

Acknowledgements

I would like to thank my supervisor, Prof. Ademar Manuel Teixeira de Aguiar, and co-supervisor, Prof. Nuno Honório Rodrigues Flores, for their expertise, ideas, feedback, time and encouragement.

I want to express my gratitude to my family (namely my parents, Américo and Eugénia, and to my brother, Miguel) and friends, who helped and encouraged me throughout my life.

I would like to thank André Cardoso, Hector Dantas, Jorge Abreu and Pedro Loureiro for providing valuable assistance during this thesis.

I am grateful to the undergraduate students from the Master in Informatics and Computing Engineering who developed the plugins for Redmine while in the discipline of Software Engineering Laboratory.

Special thanks to all the people who along the way believed in me.

Tiago Mourão Teixeira

Contents

List of Figures

Abbreviations

Ajax	Asynchronous JavaScript and XML
BPMN	Business Process Modeling Notation
CCB	Change Control Board
CRUD	Create, Retrieve, Update and Delete
CUBiT	Centralized Unified Build, Integration, and Test
CVS	Concurrent Versioning System
CSCW	Computer-Supported Collaborative Work
CSV	Comma-separated value
CSS	Cascading Style Sheet
DOM	Document Object Model
DSL	Domain Specific Language
DTD	Document Type Definition
DVCS	Distributed Version Control Systems
EJB	Enterprise JavaBean
EMS	Electronic Meeting System
FTP	File Traner Protocol
FTPS	FTP Secure
GDSS	Group Decision Support Systems
GoF	Gang of Four
HCI	Human-Computer interaction
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
IM	Instant Messaging
ISBN	International Standard Book Number
ISSN	International Standard Serial Number
JPS	JavaServer Pages
LDAP	Lightweight Directory Access Protocol
MDD	Model Driven Development
MDA	Model Driven Architecture
MVC	Model-View-Controller
OCL	Object Constraint Language
OMG	Object Management Group
ORM	Object-relation mapping
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistants
PHP	Hypertext Preprocessor

QoS	Quality of Service
QVT	Query View Transformation
RDBMS	Relational database management system
RIA	Rich Internet application
RSS	Really Simple Syndication
SCC	Source Code Control
SCM	Software configuration management
SDG	Single display groupware
SFTP	Secure File Transfer Protocol
SOA	Service-oriented Architecture
SODA	Services Oriented Development of Applications
SVN	Subversion
SysML	Systems Modeling Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
VCS	Version Control System
VoIP	Voice over Internet Protocol
XMI	XML Metadata Interchange
WBS	Work Breakdown Structure
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
WYSIWYG	What You See Is What You Get
WebDAV	Web-based Distributed Authoring and Versioning
WWW	World Wide Web

Chapter 1

Introduction

We live in a world where the value of information depends on a myriad of factors: how easily can we access and filter the information we are searching for; how effectively can we exchange the information with other people and increase the collective knowledge; and how flexible the information adapts to the needs and desires of our constantly changing societies.

The Internet is now the most effective and widely used solution for accessing and sharing information, providing a solid ground for facing the problems that arise from the globalization of the software industry. The distribution of Software Engineering teams across multiple locations – especially in large software projects – and the need to work collectively requires special attention. Software engineers must be able to communicate independently of their location and communication devices, coordinating efforts and creating a shared knowledge of the information and each other's work.

In order for the software industry to improve the collaboration between geographically distributed people (but also with co-located individuals), the use of technology – and in particular the Internet – is not enough. Technology must also be adapted to the people's specific problems, which requires the analysis of how humans interact in a social environment. To cope with this two-sided challenge, two research areas emerged: Groupware, in the late 1970s, and Computer-Supported Collaborative Work (CSCW), in the 1980s.

This thesis has two main objectives. The first is to provide a research on the collaboration features of existing software applications, across a subset of areas of a Software Engineering project. The second is to develop a prototype that integrates in a common web-based environment a selection of collaboration features capable of aiding the teamwork of software development teams.

1.1 Context

This dissertation spans four areas: CSCW, Groupware, Social Software, and Software Engineering.

CSCW is the interdisciplinary area that concerns the study of computer technologies to support teamwork, the behavior of people integrated in a team, and the extent to which computer technologies affect group behavior.

Groupware is the area whose goal is to facilitate human-computer interaction, by assisting groups of people – through the use of technology – in communicating, collaborating and coordinating their activities.

Social Software refers to those "*online-based applications and services that facilitate information management, identity management, and relationship management by providing (partial) publics of hypertextual and social networks*" [Schmidt, 2006].

Software Engineering is the area concerning the "*application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*" [IEEE, 1990].

1.2 Motivation and Objectives

Throughout history, software engineers have been using communication technologies to aid them on their software development processes. In the global software development environment we live on, the greatest challenge faced by software engineers is the coordination of their efforts through distance. The solution to this challenge is accomplished by following two steps.

Firstly, there is the need to understand how people work in a team, which technologies to choose for addressing teamwork's problems, and how to adapt the chosen technologies to the team.

Secondly, software applications should migrate to the Web, due to its worldwide access and establishment of standards. This phenomenon has already taken place, with the appearance of various tools with Ajax based user interfaces, enabling software engineers to collaboratively write documents, track the progress of a bug, design UML diagrams and many more.

The goals of this dissertation, which are closely related to the previously mentioned challenge, are the following:

- To present the theoretical concepts concerning the use of technology to support teamwork.
- To research the advantages and disadvantages of CSCW and Social Software tools.

- To research the collaboration features of existing software applications in a subset of areas of Software Engineering.
- To rationally choose and develop in a prototype a set of collaboration features, to integrate as plugins in Redmine, a Web-based project management software application.
- To study the impact of using collaboration features integrated in a single Web-based environment through the validation, in an experimental scenario, of the developed prototype.

The fact that there is, at the present time, a lack of integrated Web-based environments covering the entire software development life cycle and few studies on the impact of using several awareness and communication tools integrated in a single Web-based environment, was an important source of motivation for working on this master thesis.

1.3 Thesis outline

This thesis is composed by eight chapters, including the current chapter.
Chapter 2 presents the definitions and classifications of collaborative software.
Chapter 3 exhibits the state of the art of collaborative software. The chapter begins by presenting the advantages and disadvantages of CSCW and Social Software tools. It is then followed by a brief review of the inclusion and usage of collaborative features in Software Engineering Software.
Chapter 4 presents an exhaustive research of the collaboration features of existing software products, which spanned a subset of Software Engineering areas. Afterwards, it shows a summary of the research results, along with a list of identified open issues.
Chapter 5 exhibits a statement of the problem this dissertation intends to solve, a justification of the problem's uniqueness, and a discussion of why the problem is worthwhile to be solved.
Chapter 6 presents the sequence of steps that were followed for developing a prototype capable of addressing the first gap identified in the problem statement. It starts with the assay of Redmine, and is followed by the analysis of the collaboration features studied in the chapter 4, the requirements analysis, the architecture design, the technological decisions, the objectives fulfilled by the implemented prototype, and finally the implementation details of the prototype.
Chapter 7 shows the details of the experiment that was designed concerning the use of prototype, according to the second gap identified in the problem's statement.
Chapter 8 states the conclusions that were made of the developed work and presents notes about future work.

Chapter 2

Collaborative software: definitions and classifications

Teamwork analysis and its connection to software have been a research topic for more than three decades. In a global market scenario, software development is facing an increasing demand for coordinating the work of stakeholders across the globe as seamless as possible. In this context, it is of vital importance knowing and understanding how people work in a group and how to map the people's needs to the software potentially capable of supporting them.

This chapter provides an historical evolution of the definitions and classifications of collaborative software. Initially a groupware historical background is presented, concerned with the use of software tools to increase teams productivity. Afterwards a computer-supported collaborative work centered perspective is presented, focused on a interdisciplinary analysis of how teams work and how technologies affect human behavior. Finally, the background information of collaboration in Software Engineering is presented.

2.1 Groupware

This section presents the definitions, classifications, advantages and disadvantages of the Groupware research area.

2.1.1 Definitions of groupware

Peter and Trudy Johnson-Lenz were the first to coin the term *groupware*, on their research notes, in October 4, 1978, during their work with Murray Turoff and S. Roxanne Hiltz at the New Jersey Institute of Technology [Johnson-Lenz and Johnson-Lenz, 1998].

In February 2, 1979, Peter and Trudy Johnson-Lenz wrote about the term in an informal paper, "On CC and Citizen Participation", and afterwards, in 1981, they published the definition, "*intentional group processes plus software to support them*", in the paper "Consider the Groupware: Design and Group Process Impacts on Communication in the Electronic Medium".

Later, in January 1981 [Johnson-Lenz and Johnson-Lenz, 1981], these authors developed their theory of groupware in more detail in "The Evolution of a Tailored Communications Structure: The Topics System".

According to [Johnson-Lenz and Johnson-Lenz, 1981] "*If the software alone were considered, evaluation of the use of the complete groupware system would be insensitive to either the characteristics of the group itself or the process and procedures being followed by the group. Ineffective group work might well be the result of an inappropriate match between the group's needs and the selected software. It might also be the result of inappropriate procedures being used by the group, software aside*".

To address this situation, the authors presented a model of the groupware design process, shown in Figure 2.1. Only after the group's processes and procedures have been agreed upon, the software is selected or, if the group's needs aren't met by the available software, designed, implemented and tailored to support the defined processes and procedures. The problem of this definition is that it narrows the scope of group work to a set of processes.

Another definition, a more detailed one, came from Johansen, in 1988: "*Groupware... a generic term for specialized computer aids that are designed for the use of groupware work groups. Typically, these groups are small project-oriented teams that have important task and tight deadlines. Groupware can involve software, hardware, services, and/or group process support*". This definition is less process-oriented, explicitly associating groupware also to software, hardware, and services. The drawbacks of this definition is the exclusion of categories of products that weren't designed specifically for supporting group work – like email or shared databases – and also the restriction to small teams [Johansen, 1988].

A broader definition came from [Ellis et al., 1991], in 1991: "*computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment*". This definition is less restrictive on the scope, but also too broad. Despite excluding multi-user systems (such as time-sharing systems where users don't share the same goal), this definition includes shared database systems.

An even broader definition of groupware was made by [Nunamaker, 1995], in 1995, as "*any technology specifically used to make a group more productive*" and [Coleman, 1995], in the same year, as "*an umbrella term for the technologies that support person-to-person collaboration; groupware can be anything from email to electronic meeting systems to workflow*".

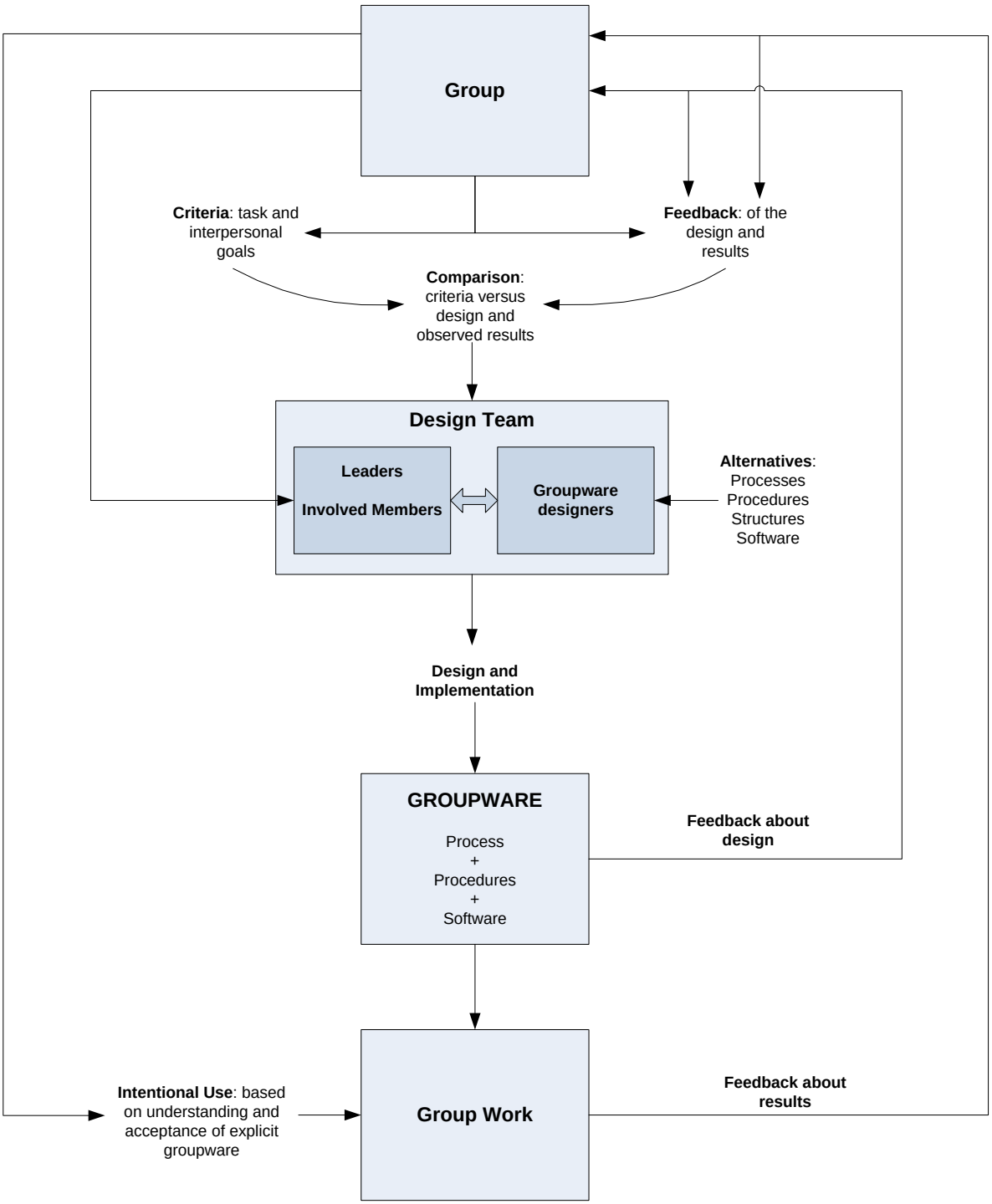

Figure 2.1: Groupware design process (adapted from [Johnson-Lenz and Johnson-Lenz, 1981]).

	Communication Support	Deliberation Support	Information Access Support
Concerted Work level			
Coordinated Work level			
Individual Work level			

Figure 2.2: Groupware Grid (adapted from [Nunamaker, 1995]).

It is clear that, as Grudin points out in [Grudin, 1994], groupware means different things to different people. Crowley [Ensor, 1990] argued that the single greatest impediment to computer support for workgroup collaboration is the lack of software permitting interaction across networked PCs (Personal Computers), and thus network file servers and related software should be considered groupware. Grudin and Poltrock [Grudin and Poltrock, 1991] considered multi-user software groupware because it provides informative success cases. Kraul [Grudin, 1994] argued that e-mail is the only successful CSCW application. Allen [Allen, 1995] defended that e-mail is itself a substrate for groupware products, but shouldn't be called groupware due to its insensitivity to organizational or group qualities.

Putting aside the differences between all the above definitions, there is one concept common to them all: *group work*. As [Ellis et al., 1991] stated, "*Groupware reflects a change in emphasis from using the computer to solve problems to facilitate human interaction*", and this change can only be made real if the participants work together. Having human interactions three key elements – *communication*, *collaboration* and *coordination* – the goal of groupware is then to assist groups in communicating, collaborating and coordinating their activities.

2.1.2 Groupware Grid

In [Nunamaker, 1995], Nunamaker presented the Groupware Grid, shown on Figure 2.2, "*which can serve as theory-based heuristic model for evaluating the contributions of groupware technology to team productivity*", particularly for the managers. As exhibited in this figure, the Groupware Grid contains nine cells, three for each axis.

The horizontal axis of the Groupware Grid derives from the Team Theory of Group Productivity. Team Theory "*asserts that team members divide their limited attention resources among three cognitive processes: communication, deliberation, and information access*" [Nunamaker, 1995]. These processes interfere with one another, limiting group productivity.

The *communication process* relates to people devoting attention to choosing words, behaviors, images and artifacts, presenting them through a medium to other team members.

The *deliberation process* relates to people devoting cognitive effort to forming intentions toward accomplishing the goal. This process includes the classic problem-solving activities, such as making sense of the problem, developing and evaluating alternatives, selecting and planning a course of action, monitoring results, etc.

The *information process* relates to the attention demands of finding, storing, processing, and retrieving the information the group members need to support their deliberation. A key function of information process is to increase the likelihood that one expects of the outcome obtained by choosing one course of action over another.

Information's value has dichotomic sides. On the positive side, information has value to the extent that it is available, accurate, and complete. On the negative side, there is the cost of acquiring, storing, processing, and retrieving it.

Another important concept of Team Theory is *goal congruence* – "*the degree to which the vested interests of individual team members are compatible with the group goal*" [Nunamaker, 1995]. The cognitive effort required by the three processes characterized above is motivated by this concept, because when team members interests are aligned with the group's interests, they have to employ less effort to achieve the goal than those whose interests are not served by the group's goal. However, the Groupware Grid doesn't address goal congruence, because this concept is more associated with technology's use than with its inherent nature.

Therefore, the horizontal axis addresses the potential for technology to reduce the cognitive costs of joint effort. That said, groupware may improve productivity to the degree that it reduces the attention costs of communication, deliberation, or information's processes.

The vertical axis of the Groupware Grid consists of three levels of group effort. For each one, Nunamaker made an analogy to sports racing (see Figure 2.3).

In the *Individual Work Level*, as with people in a 100 meter dash, the individual efforts require no coordination and the group productivity is simply the sum of individual results.

In the *Coordinated Work Level*, as in a relay race, the team works paying special attention to the coordination between otherwise independent individual efforts.

In the *Concerted Work level*, as in a rowing race, the team must make a continuous concerted effort. The demands placed on the team vary depending on the level of work in

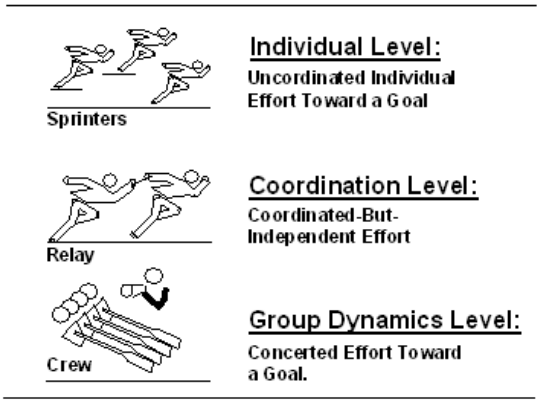

Figure 2.3: Three levels of group effort (extracted from [Nunamaker, 1995]).

which they are engaged.

At all these three levels, there is groupware technology to support teamwork.

The usefulness of the Groupware Grid arises when mapping the contributions of a single groupware tool or an entire groupware environment into the cells of the grid, with a given technology probably providing support in more than one cell. The comparison of the potential for productivity of different environments – achieved by comparing their respective grids – is also possible.

According to [Nunamaker, 1995] "*a team database like Lotus Notes offers little support at the Concerted Work level but offers strong support for communication and information access at the Coordination Level. A team database offers little deliberation support at the Coordination Level, but project management and workflow automation system offers strong deliberation support at that level*".

2.1.3 Advantages of groupware

The advantages of working in a group are [Davis, 1969]:

- The group can potentially increase performance through redundancy, if the problem demands that everyone work at the same task and if individual performance is associated with some probability of error. In this situation, group work with duplication provides a check on the quality of the group's output.
- If the task requires several pieces of information, using the unique but relevant information that each person possesses, will allow groups to potentially solve problems that one person alone couldn't solve successfully.

- If there is the need to break a task into sub-problems, then different group members can simultaneously work at the different portions of the task, thus accelerating the work.
- Questioning and debating during social interaction may stimulate new or intra-individual thought processes that the unvarying environment of the individual might not provide.
- The presence of others is motivating, thus being an advantage for the execution of some tasks.

Another advantage of groupware relates to the continuous improvement of tools developed to support teamwork in terms of reliability, functionality, and usability. Moreover, the networks that enable groupware are spreading throughout the world, increasing in flexibility, with wireless networking and accessibility – good access to the Internet is now possible in hotels, homes, coffee shops, and many other places [Sears and Jacko, 2007].

As [Hiltz and Turoff, 1993] pointed out, the overall conclusion is that "*groups are usually superior to individuals in proportion of correct solutions (quality) and number of errors, but somewhat less often are groups superior in terms of time required to reach an answer*". This conclusion is especially true if the number of person-minutes expended, rather than the elapsed time from problem presentation to solution, is computed.

2.1.4 Disadvantages of groupware

The disadvantages of teamwork are numerous and can't be ignored [Nunamaker, 1995] [Shaw, 1971].

On the one hand, there is a multiplicity of psychological factors (see Figure 2.4) inherent to the group's behavior, capable of compromising the project's success. For example, participants may fail to understand their goals, may lack focus, may have hidden agendas, or may be afraid to speak up, while others may dominate the discussion. Moreover, misunderstandings can occur, through differences of language, gesture, or expression.

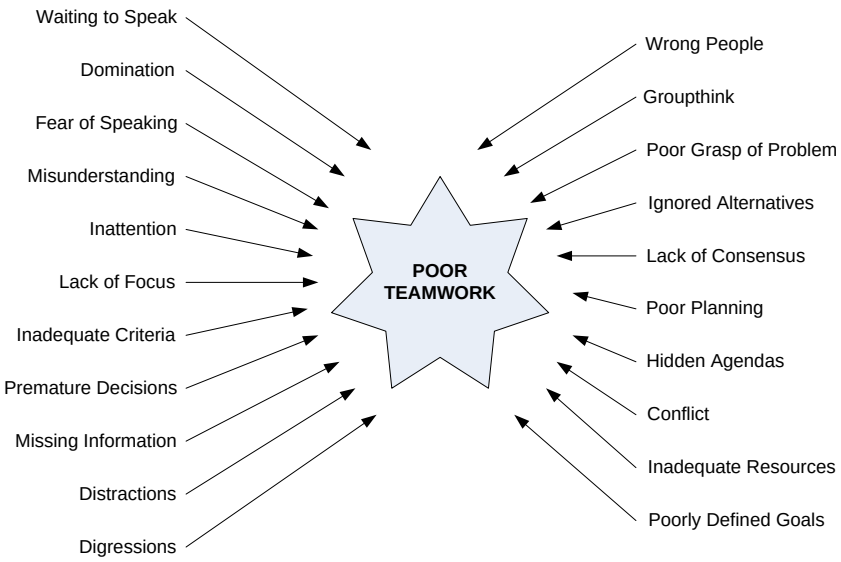

Figure 2.4: The disadvantages of groupware (adapted from [Nunamaker, 1995]).

On the other hand, teamwork is both difficult and expensive in time and money. For example, Panko [Panko, 1992] pointed out that, in 1992, a meeting between several managers or executives could cost upwards of $1000 per hour in salary costs alone. This numbers revealed to be intimidating, since back then there were more than 11 million formal meetings per day in the United States alone, with more than three billion meetings per year. In terms of time expended, managers spent about 20% of their time in formal meetings of five people or more, and up to 85% of their time communicating.

Furthermore, the dependency of groupware on its network infrastructure brings on a number of problems. For instance, doing web conferencing when some participants are on slow dial-up lines and others are on fast networks requires careful coordination [Sears and Jacko, 2007].

However, the necessity for people to collaborate so that tough problems can be solved, as well as the globalization of business, overshadows the disadvantages of groupware. As stated by [Nunamaker, 1995], "*for all the expense, teams will not go away*".

2.2 CSCW

This section presents the definitions, core concepts, classifications and future research topics of the CSCW research area.

2.2.1 Definitions of CSCW

Computer-Supported Collaborative Work emerged in 1984, when Iren Greif of MIT and Paul Cashman of Digital Equipment Corporation organized a workshop, which brought together 20 people from different fields to explore technology's role in the work environment, coining CSCW to describe it. Since then, thousands of researchers and developers across the globe (including the United States, Europe and Asia) have replied to this initiative [Grudin, 1994].

There are several definitions for CSCW, although very similar ones [Koch and Gross, 2006]. Bowers and Benford [Bowers and Benford, 1991] provided probably the most general description. They stated that "*In its most general form, CSCW examines the possibilities and effects of technological support for humans involved in collaborative group communication and work processes*".

Some researchers emphasize the aspect of group work or group activity in CSCW. For instance, Greif [Greif, 1988] defined CSCW as a "*computer-assisted coordinated activity such as communication and problem solving carried out by a group of collaborating individuals*".

Finally, other researchers balance the technology and group work components. Wilson [Wilson, 1991] defined CSCW as a "*generic term, which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques*", while Greenberg [Greenberg, 1991a] defined it as a "*scientific discipline that motivates and validates groupware design. It is the study and theory of how people work together, and how computer and related technologies affect group behaviour*".

In order to fully perceive the scope of CSCW, the technologies and associated hardware, software, services and techniques, as well as the group work and social interaction should be considered [Koch and Gross, 2006].

An essential characteristic of CSCW is that it's an interdisciplinary field, where researchers from various fields contribute with different perspectives and methodologies on the acquisition of knowledge on how teams work and how to support them [Greenberg, 1991a]. For instance, computer scientists bring in their knowledge of networks, messaging services, and distributed systems; social scientists contribute their sociological and anthropological knowledge for questions of design; psychologists, cognitive scientists, cognitive ergonomists and human-computer interaction researchers assist with their knowledge of human factors in computing [Gross and Traunmuller, 1996].

The line dividing Groupware and CSCW isn't clear, since both research areas study the use of technology to support collaboration between groups of people. Perhaps the difference can be seen if one analyzes the scope's depth of each one. While Groupware

focuses on technologies aiding groups to be more productive, CSCW goes further, studying not only the technologies and tools used in Groupware but also how people work together and how computer and related technologies affect group behavior. CSCW concerns with simultaneously offering technologies and tools to support group-work but more importantly on understanding how a group works and what is the impact on using tools to aid the group's results.

Kling pointed out in [Kling, 1991] "*In practice, many working relationships can be multivalent with and mix elements of cooperation, conflict, conviviality, competition, collaboration, commitment, caution, control, coercion, coordination and combat*". This "mix of elements" – which can only be considered by CSCW broader scope – makes clearer that group relationships are not based only on communication, collaboration and coordination, which are the three human interaction components addressed by Groupware.

Greif [Grudin, 1988a] concluded in 1988 that in the future a great number of software systems will have collaborative functionality and will be groupware to some extent. However, she noticed that in midst of this evolution, CSCW as a research field will continue to exist, because it addresses larger questions about the design and refinement of groupware.

2.2.2 Core concepts of CSCW

Over the years, CSCW researchers – responsible for designing and building the systems – have analyzed systems inside their community or through studies of existing systems, resulting in the identification of core concepts of cooperative work, the most relevant being:

Awareness

Individuals working together need to be able to gain some level of *shared knowledge about each other's activities* and therefore a *context for their own activities* [Dourish and Bellotti, 1992]. This context ensures that individual contributions are relevant to the group's activity as a whole, and to evaluate these contributions against group's goals and progress. In this scenario, information can then allow the group to manage the process of collaborative working.

It is of utmost importance to consider that the context within which group members collaborate is composed of both the content of individual contributions and their significance with respect to the whole group and its goals. Only then the system will enable each participant to make sense of others' activity and tailor their own work accordingly.

Awareness information is always required to coordinate group activities, whatever is the task's domain being considered. There are various approaches to the provision of awareness information. These approaches are classified in categories: the information is *explicitly generated*, directed and separated from the shared work object; or *passively collected and distributed*, presented in the same shared work space as the object of collaboration, and then applied to either *synchronous systems* or *synchronous and asynchronous systems*.

Articulation work

Cooperating individuals must somehow be able to partition work into units, divide it amongst themselves and, after the work is performed, reintegrate it [Schmidt and Bannon, 1992].

Appropriation (or tailorability)

Like individuals, different groups operate in different ways and in different settings, and may need to adapt the technologies to suit those circumstances. So, in looking at customization in CSCW, researchers have explored the balancing of the needs of the group against the needs of the individuals who make up that group [Greenberg, 1991b].

Studies in CSCW have revealed that, even with single-user technologies, customization is an inherently collaborative phenomenon and emerges within workgroups and social settings. The *situated action* perspective, as introduced by Suchman [Suchman, 1987], explains why customization is inherent to collaboration.

This perspective defines the sequential organization of human action as a moment-by-moment improvised affair, emerging in response to the circumstances of its production: physical circumstances, social circumstances, organisational circumstances and so forth. In this scenario, everyday action continuously incorporates elements of those settings as a means to accomplish the work at hand. Consequently, features of the setting and the artifacts around which inherently shared working practices are organized, are continually reconfigured, repurposed and incorporated into the way in which those practices develop. Since technologies are part of those settings, they also need to be reconfigured and repurposed.

To avoid confusion with the perspective of customization within Human-Computer interaction (HCI), Dourish [Dourish, 2003] refers to it as *appropriation*, as is the way in which technologies are adopted, adapted and incorporated into working practice. The appropriation might involve customization in the traditional sense – i.e., the explicit reconfiguration of technology for the purpose of suiting local needs – but it might also simply involve making use of the technology for purposes beyond those for which it was originally designed, or to serve new ends.

2.2.3 Classifications of CSCW

The complexity of CSCW domain makes it difficult to produce results that work in every situation, because the success of CSCW is greatly conditioned to the social context in which it operates. Since this context is hard to be generalized, CSCW systems based on the design of successful ones may fail to be appropriated in other seemingly similar contexts [Grudin, 1988b].

Hence, as a means to making CSCW effective, there is the need to diminish the social-technical gap between what is obligatory to support socially and what it can be supported technically, managing the nuance, flexibility and ambiguity properties inherent to human activity and absent in the systems.

In [Reinhard et al., 1994], Weber developed a taxonomy to evaluate CSCW tools and their relationship to collaborative processes, serving as a platform for specifying functional and technical requirements for CSCW systems in certain cooperative work processes.

The criteria used to identify CSCW systems were divided into three major groups:

Application

From an application viewpoint, certain tasks are generically present in many scenarios, from general-purpose tasks (such as brainstorming, note taking and shared agenda features) to more dedicated domains where there is the need for tailored tools. To the user, a CSCW system appears complete only when specialized and generic tools are integrated.

Functional

A CSCW system relates functional features with the social aspects of teamwork. The features have an impact on both the work behavior and efficiency of the entire group using the system. Issues such as interaction, coordination, distribution, user-specific reactions, visualization and data hiding must be taken into consideration. However, the psychological, social, and cultural processes active within groups of collaborators are the real keys to the acceptance and success of CSCW systems.

Technical

This group comprises hardware, software and network support. It divides the architecture of a CSCW system into four classes of classes or features: *input*, *output*, *application*, and *data*. Each can be centralized or replicated.

For all of these groups, concerns such as *flexibility*, *transparency*, *collaboration* and *sharing* are addressed and guidelines for their support are presented.

Another approach on categorizing and defining CSCW was made by Wilson [Wilson, 1994], which characterized CSCW into four human categories and four main technology categories.

On one side, regarding the human categories, Wilson considered:

1. *Individual aspects*, such as conversation patterns and the making of commitments.
2. *Organizational aspects*, for example the structure and culture of organizations.
3. *Group work design aspects*, such as involving the user in the work design process, rapid prototyping and usability testing.
4. *Group dynamic aspects*, for instance group decision making and the collaboration process.

On the other side, regarding the technology categories, Wilson considered:

1. *Communication systems*, which enables geographically distributed people to communicate with each other (e.g. telephone, e-mail, and videoconferencing).
2. *Shared work space facilities*, enabling people to view and work simultaneously on the same electronic space (e.g. electronic whiteboard and remote screen sharing).
3. *Shared information facilities*, which enables people to view and work on a shared set of information (e.g. multi-user databases).
4. *Shared activity support facilities*, to increase group work processes (e.g. multi-user editors and idea generation).

The different technology categories are not mutually exclusive and can be combinations of several parts of categories.

Finally, one of the most common approaches to the conceptualization of a CSCW system regards the context of the system's use. In this approach – which was first introduced by Johansen [Johansen, 1988] and later reviewed in [Baecker et al., 1995] – the context can be considered along two dimensions (see Figure 2.5). Along the vertical axis, collaboration can be co-located (*same place*) or geographically distributed (*different place*). Along the horizontal axis, collaboration can occur synchronously (*same time*) or asynchronously (*different time*).

	Same time (Synchronous)	Different time (Asynchronous)
Same place (Co-located)	Electronic meeting systems Roomware Single display groupware …	Large public displays Team rooms …
Different place (Remote)	Collaborative real-time editors Instant Messaging Videoconferencing Voice over Internet Protocol …	Blogs E-mail Group Calendars Internet forums Microblogs Version Control Systems Wikis …

Figure 2.5: CSCW Matrix (adapted from [Baecker et al., 1995]).

2.2.4 The Future of CSCW

In the future, CSCW features will most certainly converge to the growing market of mobile technologies. The possibility of collaborative work through devices independent of the users' location is very appealing. Inside the mobile devices, mobile phones will be an interesting device for providing CSCW capabilities, mainly because while some of us have notebooks, almost all have a mobile phone.

Other areas that will have a promising research future ahead are ubiquitous computing, where everyday life objects are interconnect with technology, as well as Web 2.0 and Social Software – we are already witnessing the enormous growth of web sites that give users the power to collaboratively form a network of shared knowledge, such as Facebook[1] and Twitter[2].

2.3 Collaboration in Software Engineering

This section presents the limitations of software engineers, collaboration techniques of Software Engineering, classification of collaboration in Software Engineering, categories of Software Engineering tools, and the future of collaboration in Software Engineering.

[1] http://www.facebook.com/
[2] http://www.twitter.com/

2.3.1 Limitations of software engineers

Today it is unimaginable to think of a Software Engineering project done by individuals without any contact between themselves, not only because human beings are inherently social but also because projects are too complex and large to be dealt individually.

Therefore, software engineers must coordinate their efforts, developing a shared understanding surrounding multiple artifacts, each with its own model, over the entire development process. This focus on model-oriented collaboration, embedded within a larger process, is what distinguishes collaboration research in Software Engineering from Groupware and CSCW research, which study context-independent collaboration technologies [Whitehead, 2007].

As humans, we have several limitations that affect our ability to create almost any piece of software, both when working individually and in groups of people.

When working alone, and having to face high levels of abstraction – as when writing requirements, designing software, writing code, or creating test cases – we are slow and error-prone. To solve this problem, we have to work in a group as a means to finish projects on time and within budget.

When in a group, although there is a greater chance others will notice mistakes that would have pass by when working alone, problems arise in this situation, such as the expressive but frequently ambiguous characteristic of humans' natural language, memory's limitation to remember all the details of a project, the inability to be aware of what everyone is doing in a group and thus risk duplicating the work of others, and the architecture and design of large systems being strongly dependent on the unique interpretation of each individual.

2.3.2 Collaboration techniques of Software Engineering

To address the referred limitation problems, the collaboration techniques of Software Engineering have evolved. This evolution is reflected by multiple goals spanning the entire life cycle of development, which can be grouped into three categories:

Communication

This category encompasses techniques for engineers to interchange ideas with stakeholders throughout the project. These include project's requirements descriptions (e.g. brainstorming and negotiation, relying on the chosen development mode [McConnell, 1996]) and ensuring a single system architecture and design, crossing organizational boundaries both inside and outside the company [Grinter, 1999].

Dependencies

This category includes the management of dependencies among activities, artifacts, organizations and engineers. Project management tasks play an important role in controlling the plan of activities, as well as their coordination. Coordination is an important factor on this management process and its effects are often clearly noticed not when it is present but when it is lacking [Malone and Crowston, 1994]. Reducing dependencies among engineers to a minimum – and therefore diminishing the need for collaboration – can be achieved with Software configuration management systems (SCM), which allows developers to work in individual workspaces and thereby isolate their changes from others.

Memory

This category comprises approaches to release stakeholders of tracking themselves the evolution a project. Identification, recording and resolution of errors can be achieved by inspections and reviews, testing, submission of reports by the users – whether in explicit beta testing programs, or through normal use – and bug tracking (issue management) systems. Project participants can also create a source of organizational memory, supplying their knowledge for future reference, through techniques such as SCM change logs and documentation's project repositories [Ackerman and Mcdonald, 2000].

2.3.3 Classification of collaboration in Software Engineering

Collaboration in Software Engineering can be characterized in two ways, regarding how structured is the communication between individuals.

The first category of collaboration, characterized by the *unstructured nature of communication*, has been adopted by software engineers for both project use and to hold meetings.

On the one hand, communication and collaboration technologies used to assist the coordination in every stage in a project's life cycle includes telephone, teleconferences, email, voice mail, discussion lists, the Web, instant messaging (IM), voice over Internet Protocol (VoIP), and videoconferences. On the other hand, software engineers hold meetings in meeting rooms and conduct informal conversations in hallways, doorways, and offices; while these discussions concern the development of a formal system – a piece of software – the conversations themselves are not formally structured.

The second category of collaboration, characterized by the *structured nature of communication*, concerns the collaboration in formal and semi-formal artifacts – such as requirements specifications, architecture diagrams, Unified Modeling Language (UML)

diagrams, Business Process Modeling Notation (BPMN) diagrams, source code, and bug reports – each being a different model of the ongoing project.

Software Engineering collaboration can thus be understood as artifact-based or model-based collaboration, where the focus of activity is on the production of new models, the creation of shared meaning around the models, and the elimination of error and ambiguity within the models.

This model-based collaboration is a hallmark of Software Engineering collaboration, distinguishing the study of collaboration within Software Engineering from the more general study of collaboration, which tends to lack this focus on model creation.

The reason why this model approach to Software Engineering collaboration is so important derives from its structuring effect. Models provide a *shared meaning* that engineers use when coordinating their work, as when engineers working together consult a requirements specification (the *model*) to determine how to design a portion of the system. Engineers also use models to create *new shared meaning*, such as when a UML diagram is discussed, and thereby have a better understanding of its meaning and implications to the project. In addition, models reveal *ambiguity* by making it possible for one engineer to clearly describe their understanding of the system – in doing so, it's clearer to software engineers that when the description is confusing to others, ambiguity is present.

Without the structure and semantics provided models, recognizing differences in understanding the system among software engineers would be a more difficult job.

2.3.4 Categories of Software Engineering tools

Software engineers have developed a wide range of tools to support collaborative work on their projects. According to [Whitehead, 2007], this tools fall into four broad categories: *model-based collaboration tools*, *process support tools*, *awareness tools*, and *collaboration infrastructure tools*.

Below, it will be given a brief overview of the application's context of each area. This dissertation focused primarily on model-based collaboration tools.

Model-based collaboration tools

Within projects, software engineers have to create artifacts of multiple types, including requirements specifications, architecture descriptions, testing plans, and the end product. Since each type of artifact has its own semantics – ranging from free form natural language to the semi-formal semantics of UML, or the formal semantics of a programming language – creating artifacts is in reality creating models.

During this process of creating artifacts, tools can be used to aid software engineers of being aware of their peers' activities. This need for collaboration is especially pertinent

during requirements and testing phases, where engineers work with customers to ensure the artifacts reflect their needs.

Therefore, it is possible to conclude that these tools are in reality assisting the creation of specific models, hence supporting model-based collaboration.

There is a set of collaboration tools to support the creation of every kind of model found in typical Software Engineering practices. In this dissertation, there were studied a set of tools that span the life-cycle of a Software Engineering project, which can be seen in chapter 4.

Process-based collaboration tools

Whether software engineers are dealing with small or large software projects, the use of a structure for specifying the sequence of actions to be executed, the roles of each participant, and the artifacts that must be received and those that must be supplied is of major relevance. This structure – useful for project management and for the reduction of the collaboration maintenance effort in short-term (in the first projects) and long-term basis (regularizing points of collaboration) – is a software process model and can take many forms, such as the Waterfall model, Spiral model, or Agile model.

Process centered software development environments provides means for writing software process models in a process modeling language and afterwards executing these models in the context of the environment. For instance, the environment can manage the assignment of tasks to engineers, monitor their completion, and automatically invoke the appropriate tools.

Examples of such systems are: Arcadia [Kadia, 1992], Oz [Ben-Shaul, 1993], Marvel [Ben-Shaul et al., 1992], ConversationBuilder [Kaplan et al., 1992], and Endeavors [Bolcer and Taylor, 1996].

Awareness tools

Nowadays, most medium to large-scale projects involve multiple stakeholders that may or may not be co-located. The stakeholders' geographical distribution is related to the awareness concept, that is, knowing who else is working on the project, what they are doing, which artifacts they are or were manipulating, and how their work may impact the work of others.

Examples of systems that provide awareness during software development are: Augur [Kadia, 1992], Seesoft [Eick et al., 1992], Palantir [Sarma et al., 2003], Jazz [Hupfer et al., 2004] and Lighthouse [da Silva et al., 2006] .

Jazz was a research project at IBM which focused on a specific set of collaborative features for the Eclipse IDE. Designed for small, informal teams, the objective of Jazz

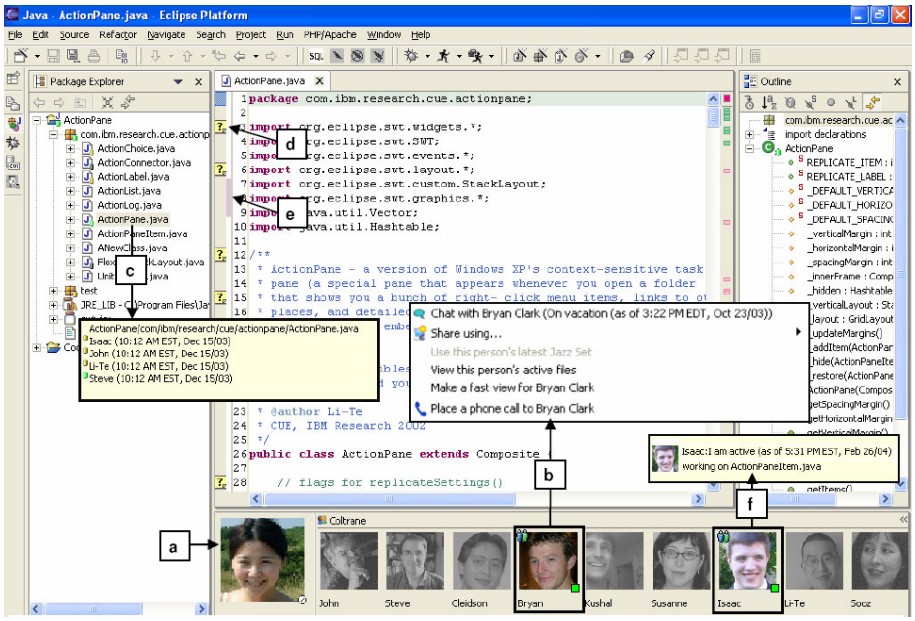

Figure 2.6: Jazz's features: a) Jazz Band showing teams, members, and status icons, b) menu offering communication options, c) decorators and tooltips on resources, d) anchored chat marker, e) code modification indicator, and f) team member's status message (extracted from [Hupfer et al., 2004]).

was to make software development a social activity, while capturing the team's artifacts to provide context for communication.

Jazz's features [Hupfer et al., 2004] [Cheng et al., 2004] (see Figure 2.6) included:

- Instant Messaging buddy list to monitor who of the user's team members were online and those that were coding. Members were represented by images, decorated by status icons at the bottom right indicating, whether the person was online, away, or busy.
- Instant Messaging status messages that automatically incorporated contextual information (e.g. what file the developer was working at a given time).
- After right-clicking a teammate's image, developers could choose from a popped up menu to initiate a text chat, voice-over-IP, or screen sharing session.
- A chat discussion could be saved as code annotation or into a discussion forum.
- Files and other resources were decorated with colored icons to indicate what other developers were doing with their local copies of the files. Green indicated that a file was open and being edited, yellow that a file was modified locally but not checked back into the source control repository, and black indicated that a file was not checked back in.

- Hovering over a resource brought up a tooltip displaying who was responsible for the changes to that resource.
- A developer could highlight a region of code in the editor, right-click it, and then initiate a chat about the selected piece of code. When the discussion ended, a transcript could be saved and would appear a marker next to the associated code. Teammates could later review the chat by clicking on the marker.
- Hovering over a marker – which was associated to a piece of code modified by team member(s) – would show the difference between the local code and the remote code on the teammate's desktop.

Collaboration infrastructure tools

In order for software tools to coordinate their work, there are integration technologies, including *data integration*, which ensures that tools can exchange data, and *control integration*, ensuring that tools have a sense of awareness and use that knowledge to take action. For example, nowadays most Integrated Development Environments (IDEs), once an engineer has finished editing their source code, store the file on a central repository (data integration) and then automatically call the proper compiler (control integration). Tools like Eclipse, Microsoft Visual Studio, Marvel [Ben-Shaul et al., 1992], and Web-based Distributed Authoring and Versioning (WebDAV[3]) [Whitehead and Goland, 1999] already implement these behaviors.

Today, the majority of software products use SCM systems to satisfy their need for data integration by managing files in separate workspaces. However, data integration standards and SCM can be intertwined, as in the case of Subversion[4], which uses WebDAV as the data integration technology in their implementation.

As for control integration support, there are two main approaches: *direct tool invocation* and *event notification services*. In direct tool invocation, a primary tool in an environment (e.g. Eclipse) directly calls another tool to perform some action. However, when multiple tools need to be coordinated, a message passing approach is a better choice. In this case, tools exchange event notification messages using some form of event transport, as demonstrated in The Field Programming Environment [Reiss, 1995] and the Sienna system [Carzaniga et al., 2001].

2.3.5 The Future of collaboration in Software Engineering

Today, we live in the Web 2.0 era. Until Google Maps, launched in 2005, the most frequent criticism to web-based applications was the lack of user interface interactivity.

[3]WebDAV is a set of extensions to the Hypertext Transfer Protocol (HTTP) that allows users to edit and manage files collaboratively on remote World Wide Web (WWW) servers.

[4]http://subversion.tigris.org/

The emergence of Google Maps was a landmark on the history of Web 2.0 applications, a family of products which tend to have desktop-like user interface interactivity within a web browser, as well as facilities for other sites to integrate their data into the application or vice-versa.

Although there are still a significant number of collaboration tools operating on PCs, there is a trend for the migrating of tools to the Web. There are several reasons why this is happening.

Firstly, because there is the possibility of the tools being accessible anytime, anywhere, which is a major plus in collaboration, since cooperation and communication aren't obliged to a restricted operation medium.

Secondly, because of AJAX, whose appearance provided more uniformity in JavaScript's capabilities across web browsers, and also because of the increasing processing power of web browsers [Whitehead, 2007].

Finally, not having to proceed to deployment activities for each PC that will use the tool is a big plus, both in terms of money and time.

The strengths of the Web are the weaknesses of the desktops and vice-versa. Therefore, in the future, it is expected the mixture of web-based tools with desktop-based ones. In order to this scenario of convergence to be fulfilled, there is the need to create interface standards by which desktop-based applications can access the Web-based services [Zeller, 2007]. For example, the possibility of accessing to bug tracking data within desktop IDEs would boost the productivity and allow a more quick resolution, since the user wouldn't have to switch between the IDE and bug tracking tool.

Another future research area concerns the use of collaboration tools in a global software development environment [Herbsleb, 2007]. An example of a tool that emerged to face the problems that can arise in distributed software development is EGRET, an Eclipse-based global requirements tool [Sinha et al., 2006].

2.4 Summary and Discussion

This chapter presented the background information concerning collaboration, first in a context-independent perspective – with an elucidation of groupware and CSCW areas – and then in a Software Engineering-oriented viewpoint.

Groupware is a research area focused on teamwork and whose main goal is to assist teams in the coordination, communication and collaboration of their activities. In order for a group to successfully follow groupware practices, there is the need to match the group's needs to the software features, use the appropriate groupware procedures in the group, and take into consideration the group's elements psychological factors. Despite its substantial time and money requirements, the role of groupware on enabling groups reach better and less error-prone solutions for though problems makes this area indispensable.

CSCW is an interdisciplinary field research area, where researchers from various fields – including computer, psychologists, cognitive and social scientists – contribute with different perspectives and methodologies for acquiring knowledge of how teams work and how they could be supported. CSCW core concepts are: *awareness*, where individuals are able to gain shared knowledge about each other's activities; *articulation work*, where cooperating individuals partition work into units, divide it and reintegrate it; and *appropriation*, by which individuals adapt the technologies to suit those different teamwork settings. CSCW can be classified in a number of different ways, the most popular being the context of a system's use, where classification is made along a vertical axis, concerning stakeholders location, and an horizontal axis, concerning the communication's synchronization.

Groupware and CSCW are two similar concepts, distinguished when the scope's depth of each one is analyzed. CSCW studies not only the technologies and tools used in groupware but also how people work together and how computers' technologies affect group behavior.

Collaboration in Software Engineering is a requisite for software engineers to coordinate their efforts and develop a shared understanding surrounding multiple artifacts, each with its own model, over the entire development process. This focus on model-oriented collaboration, embedded within a larger process, distinguishes collaboration in Software Engineering from Groupware and CSCW research, which studies context-independent collaboration technologies.

Analyzing the current trends of the relation between humans and technology, it is possible to predict that in the future, collaboration features will gradually transit from the PC to the other more flexible technological mediums.

One of those mediums will be mobile technologies, since they offer the possibility of collaborative work through devices independent of the users' location. Inside the mobile devices, mobile phones will probably be the most used, because of their worldwide acceptance.

Another promising medium will be ubiquous computing, due to its potential of widening the collaboration features to the environment, allowing teams to interact through devices embed in physical objects.

Finally, the collaboration features will progressively migrate to the Web. The possibility of accessing the features anytime and anywhere (and thus minimizing the negative effects of global software development), the growth of Web 2.0 and Social Software, the use of Ajax, and the creation of interface standards by which desktop-based applications will have access to web-based services will be some of the causes for this migration to happen.

Chapter 3

Collaborative software

This chapter presents the state of the art on collaborative software, with a special focus on Software Engineering. It begins by reporting a study on the advantages and disadvantages of software tools (i.e. categories of software) that can be used for collaboration purposes in any CSCW context. Then, it exhibits the Software Engineering areas where software tools can be used, either by adaptation or specifically, for aiding collaborative work.

3.1 General Purpose Software

This section presents, in accordance to the conceptualization model introduced by Johansen [Johansen, 1988], shown in Figure 2.5, the main advantages and disadvantages of each general purpose software tool that was studied in the context of this dissertation, mainly in a collaboration perspective. Those software tools were called general since they can be used in any CSCW context, which can or not be a Software Engineering context.

3.1.1 Asynchronous and Different Place CSCW

Regarding software used in the context of asynchronous and different place CSCW, this sub-section presents an analysis of: blogs, e-mail, electronic bulleting boards/internet forums, group calendars, microblogs, version control systems/source code management, and wikis.

Blogs

When compared to the previous technologies, blogs (a contraction of *weblogs*) are a relatively new collaboration tool. One of the first blogs remotes back to the year 1994, when Justin Hall, today an American freelance journalist, began writing a personal blog.

Blogs can be seen as a form of websites. They take a wide variety of types, such as personal or corporate, and can have a wide variety of content – in addition to text, it supports photographs (*photoblog*), sketches (*sketchblog*), videos (*vlog*), music (*MP3 blog*), audio (*podcasting*), etc. In this massive world of information, the *permalinks* feature, broadly used in blogs, is a major help on maintaining the integrity of resources and links over time.

The openness of blogs is one of its strongest points.

In the case of personal blogs, it gives voice to an otherwise unheard voice and merges the person with the mass media, resulting in some cases in a type of "citizen journalism".

In the case of corporate blogs, blogs can be used internally or externally. When used internally, blogs can improve employee participation, allowing direct connection between various layers of an organization and therefore augmenting collective intelligence and awareness. When used externally – often as a marketing tool – blogs can be used to announce new products and services or for serving as a reaction on public criticism, being a window to the company culture. For instance, General Motors Corp and General Electric Co. have posted recently posts on their corporate blogs in response to concerns about their financial performances to tranquilize shareholders' concerns.

A blog is a powerful social tool, allowing users to comment, tag, blogroll and more. It also by increases information's context, by allowing the explicit grouping of information, achieved through the aggregation of posts into categories.

The last advantage of blogs is its mainstream popularity. According to Nielsen Online [Johnson, 2009], blogs, along with social networks, have become the 4th most popular online category and are now surpassing e-mail in popularity.

On the negative side, blogs' liberty of speech can cause defamation or liability acts, while not providing effective protection of the blogger's name or place of work and also having a negative impact on employer branding[1].

E-mail

E-mail's origins remote back to the 1960's, predating the beginning of the Internet.

The low learning curve of e-mail is one of the reasons why this technology is a widely used communication tool. Being an asynchronous technology, e-mail can be used by a very large set of users without compromising the efficiency of communication.

Another advantage of e-mail is its reduced resources and communication requirements. At the resources level, e-mail has a minimal set-up cost and time, as well as good interoperability (since there is no requirement on the person's machine, Operating Systems (OS) or e-mail account). At the communication level, e-mail makes it

[1] According to Brett Minchinton, from The Employer Brand Institute, employer branding is the "*the image of the organization as a 'great place to work' in the minds of current employees and key stakeholders in the external market (active and passive candidates, clients, customers and other key stakeholders*" [Minchington and Estis, 2009].

easier to attract shy people, due to being a more impersonal form of communication than a face-to-face meeting.

On the negative side, e-mail provides a limited degree of interaction between users. In some situations the message's tone can be interpreted, perhaps incorrectly, as insolent or insulting, since the receiver cannot see the facial expressions of the sender.

A recurring complaint about e-mail relates to the little opportunity for users to exchange cues about when to expect a response or when are good times to be reached by email [Tyler and Tang, 2003].

As a technology tool, it has a very limited set of capabilities, namely at cooperation and collaboration levels. For example, it is difficult to use it as a document manager (easily achieved with wikis), since there is no control over the history of a document. This flaw is easily seen when one counts how much time it is lost on finding a specific version of the document, making a modification, attaching the document to a new email message and sending it to a group of people, when with a wiki the user only had to analyze the document's history and make the changes.

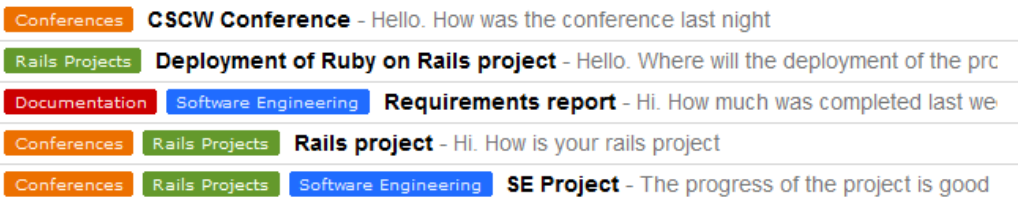

Figure 3.1: Screenshot of Gmail's labels.

Another recurrent flaw of e-mail relates to the loss of information context, since there is no connection between messages. For the purpose of improving this loss of context and provide better organization, one of today's leading solutions, Gmail, offers the possibility of labeling email messages (see Figure 3.1). In addition, e-mail suffers from information overload – due to being a push technology – and information duplication.

A lesser serious problem is spamming vulnerability, because of the extremely low cost of sending a huge number of messages.

Finally, while e-mail is one of the most effective communication mediums available today, a major weakness is the lack of awareness and social components. There is no access to the inbox of other people and there is no possibility to classify and evolve the messages value throughout the time, achieved for example with tags.

Electronic Bulletin Boards/Internet forums

Bulletin Board Systems (BBS), which later evolved to Internet forums, appeared in the late 1970's. BBS allowed logged-in users to perform functions such as downloading or

uploading software and data, reading news, and exchanging messages with other users, either through electronic mail or in public message boards.

Nowadays, Internet forums popularity overshadows BBS systems. Forums are used mainly for searching a solution, inside a particular category, to a problem.

The advantages of forums emerge not only due to its approach of categorizing topics, ideal for reducing the effort of filtering information, but also to its control over the content and for providing asynchronous communication.

A user with higher permissions can control the contents added or edited by other members, in accordance with the forum's rules. It is also possible to send to one or more users a private message.

Lastly, unlike chat rooms and IM, in a forum the participants do not have to be online simultaneously to receive or send messages. Also the messages posted to a forum are publicly available for some time, which is uncommon in chat rooms.

On the negative side, in most cases forums requires the users to be registered, which may discourage the potentially newcomer who doesn't want to give personal information.

For registered users, there is the distinction between posters and moderators, contrarily to a blog or a wiki, where the user can express his ideas without having them be subjected to the assessment of others. Activities such as deleting/moving posts or adding/editing/removing polls of threads are normally available only to moderators. There is also a limited liberty of speech, due to the word censoring system.

Another disadvantage of forums is *crossposting* - the act of posting the same message to multiple forums - when the message is not suitable for the different forums.

Finally, in terms of the posts content itself, there is a more strict relation between the posts and its context. Contrarily, blogs offer more liberty to the user, giving him the option to write and publish about any context, without having to find which group of posts fits the most his post.

Group Calendars

Group calendars are, like the name suggests, calendars used in the context of a group.

The pros of using a group calendar are mainly two.

The first is the possibility of sharing a calendar with a selected group of people or leaving it open to everyone. Similarly to an individual calendar, there are appointments a person wants to maintain private and others that are only useful when in the context of a group.

The second is the display in one calendar of events common to a group of people with different availabilities, minimizing the conflicts between schedules and the time that would be spent if there was the need for each person to communicate their own calendar.

The cons of a group calendar concern the good usage of it, how much is required from the users and who benefits from these efforts. There is the possibility of the calendar not being up to date, since people might forget to update it. In addition, according to [Grudin, 1988b], frequently in a company managers and executives are responsible for the calendar's initial construction, while secretaries do the hard work of maintaining it – the application fails because it requires additional work from those who don't perceive or receive a direct benefit from the use of the calendar.

Microblogs

A very recent and promising variant blogging is microblogging, which reduces the "rich media" of blogs to short messages with a maximum of 140 characters, known as *tweets* in today's most famous microblogging application, Twitter.

A microblog quickly captures the moments of our everyday lives, hence being a powerful awareness tool. This sharing of the daily life activities may encourage people to share informal and personal content, while in other communication mediums, like e-mail, they would have not. Without microblogging, these "small" issues would most certainly pass by.

One of the reasons why microblogging is becoming so popular resides in the fact that users aren't obliged to use a computer as a means to update the microblog or receive updates [McGiboney, 2009]. By the contrary, users are stimulated to send or receive messages via SMS or IM.

Twitter's popularity, the popular micro-blogging site created in 2006, is increasing rapidly: the growth in February 2009 relative to the growth in February 2008 was over of 1000%, while on Facebook, the leader of social networking sites – according to [Alexa, 2009], it is the 5th most visited site on the Web –, was approximately of 300% [Compete, 2009].

On the negative side, microblogging is a recent form of multimedia blogging, which limits its acceptance when compared to blogging.

Due to the messages' briefness – 140 character maximum for tracking the user's actions or observations – the content's richness of each message is very limited. On the contrary, a post of a blog can be very detailed.

Privacy issues are more likely to occur in a microblog than in a blog, since normally in a microblog there is a following/followers mechanism and one can easily send, without knowing it, a compromising message. For example, an employee has to be careful of the implications that arise when posting a new message, if his boss in his readers list, since microblogging applications sends the user's updates to all his / her subscribers.

Finally, as pointed out in [Honeycutt and Herring, 2009], the current Twitter interface presents a number of challenges that need to be addressed, namely: automatic refresh

intervals of not less than one minute, causing responses potentially to be missed; a limited history of tweets from followers, which is not preserved; lack of an interface view that displays tweets directed to and received from the same users; and the absence of a search function.

Version Control Systems/Source Code Management

Revision control systems (also known as Version Control Systems (VCS) or Source Code Management) are commonly used in software development when there is the need to manage the changes made by a group of people to the same computer files.

There are two types of VCS: Centralized and Distributed (DVCS). The advantages and disadvantages of each are presented next.

Centralized: This is the classic version control system, which can be applied in files as diverse as documentation and source code. Each version/revision is uniquely identified, allowing users to revert to previous versions; this feature is particularly relevant when a problem (e.g. a bug) is detected in a version, since it's possible to return to a baseline version.

Version Control Systems are especially useful when users work concurrently, for guaranteeing that there are no disruptive effects in the versions of a file after a change is made. For controlling concurrent access, VCS provide features like *delta compression* and source management models *file locking*/*version merging*. Delta compression retains only the differences between successive versions of a file. File locking and version merging avoid undesirable changes when two or more users are altering the same file at the same time.

Not only are VCS able to perform complex version control tasks – which if were done by developers would be very error-prone and counterproductive – but also represent a powerful awareness mechanism, since it's possible for a person to track the activities of another person.

Albeit source management models avoid undesirable changes to files, in some situations the models can have unwanted effects, such as when the *file locking* is used. In this situation, a version locked for a long time can lead other users, waiting for the unlocking, to frustration. This scenario can be especially problematic, if the locking is accidental and other users who have urgency to add relevant content must wait.

Finally, regarding centralized VCS, the common operations – such as commits, viewing history, and reverting changes – are slower than in DVCS, for the reason that unlike the former there is the need to communicate with a central server.

Distributed: Distributed Version Control Systems allows users to work more productively, even when not connected to a network. They can commit changes to their localized repositories as new revisions while being offline, whereas in a centralized version control system they must be online to be able to commit changes to the repository.

As mentioned above, in DVCS most operations are executed much faster, since there is no need to communicate with a central server. Instead of depending on a single central repository for synchronization purposes, each peer has a working copy of the codebase and synchronizes with other users by exchanging patches (called *change-sets*).

Furthermore, there isn't the concept of *folder level permissions*. Every user can add, remove and change the project's files. Additionally, when working on large projects, these systems are the ideal choice, given that users can explicitly work on parts of the project.

A critical drawback of DVCS comes with the impossibility for a user of knowing who has the latest version. This situation can be resolved if users define which repository holds the most recent version.

Another drawback is that since each user has total control over its own working copy, future merge conflicts are greater than in those encountered in the centralized version control systems.

Finally, the staff needs to be trained on how to use DVCS. This change is time and money consuming, and requires both the effort to understand the infrastructure behind DVCS and the willingness to make the adaptation.

Wikis

Wikis are one of the most widely adopted and effective collaboration tools available today. According to [Alexa, 2009], Wikipedia is the 7th most visited site on the Web.

The growth of wikis can be seen not only on the number of visitors, but also in the range of wikis that have been appearing. Besides Wikipedia, today's web users can visit wikis as diverse as Wikitionary ("*a collaborative project to produce a free-content multilingual dictionary*")[2] and Wikitravel ("*a project to create a free, complete, up-to-date, and reliable worldwide travel guide.*")[3], named one of TIME Magazine's 50 Best Websites of 2008 [Hamilton, 2007].

The potential value of a page is huge, because it can receive the input of a significantly large number of participants and immediately output that change to the Web (this immediacy is impossible in books, for example). This value is augmented by the very nature of wikis, which give free access to accurate and updated information [Wired, 2005] [Giles, 2005]. According to [Ebersbach et al., 2005] "*It has never been so easy to become a "correspondent" on the Internet, because the technical hurdles have been reduced to a minimum.*"

[2] http://en.wiktionary.org
[3] http://wikitravel.org

In a wiki, a user can edit any page or create new pages within the wiki, using only a Web browser. Pages' content can be easily associated with another page by the use of link creation, which quickly shows to the user if the link exists or doesn't [Leuf and Cunningham, 2001].

Wikis also makes it possible to use *backlinks*, that is, a list of linking to a given page. DokuWiki is one that provides this feature [DokuWiki, 2009]. Another feature is the use of CamelCase to name wiki pages. CamelCase is a form of markup for capitalizing words in a phrase and removing the spaces between them, making the reading of each letter easier and complying to PCs where the letters in a name must be contiguous [Definition, 2002]. Examples of wikis that use this convention are JSPWiki [JSPWiki, 2008], Trac [Trac, 2006] and PmWiki [PmWiki, 2005].

One of the most valuable features of wikis concerns the history of a document (see Figure 3.2, which shows the history of Wikipedia's homepage). The major part of them give the possibility for the user to write a text that summarizes the changes made to a wiki page, serving as a type of log message. Rollbacking to a previous version of a wiki page, highlighting the changes between two revisions through the diff feature – which is very useful when the page has a huge amount of content and authors and there is the need to quickly locate whether or not a change is necessary – and permalinks are examples of features pertaining document's history.

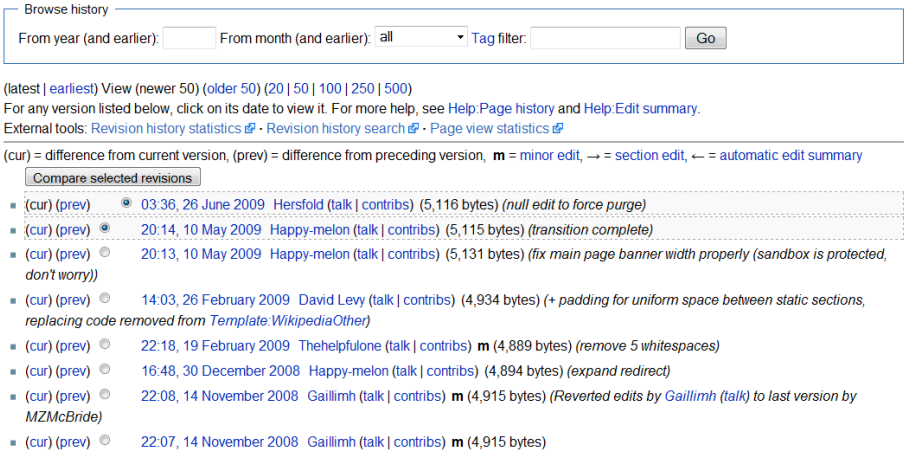

Figure 3.2: Screenshot of Wikipedia, showing the revision history of Wikipedia's homepage.

On the negative side, wikis suffers from problems such as loss of control over the pages and lack of usability. They require users to know a simplified markup language, which adds more complexity to the usage of the tool, especially when the user is familiarized with WYSIWYG Graphical User Interfaces (GUI), like Microsoft Office Word. Limited access to HTML and CSS fo a wiki, links written in a form that deviates

from the standard spelling due to CamelCase, and pages locked for a long time due to file locking, exemplify some of restrictions of a wiki.

Regarding users' behaviors, the open philosophy of wikis may result in trolling. In a study made in the context of Wikipedia articles [Kittur et al., 2007], it was concluded that on one side the number of edits on articles is decreasing: the percentage of edits made to article pages has decreased over the years from over 90% of all edits in 2001 to roughly 70% in July of 2006. On the other side, discussion, procedure, user coordination, and maintenance activity – including anti-vandalism – are increasing. Another interesting conclusion is that vandalism, which appears to be increasing as a proportion of all edits, shows a low level of occurrences (1-2% of all edits) and a mean survival time of 2.1 days.

The same paper observed that when a wiki possesses a high number of articles and users, and a certain quality level in articles is expected, coordination efforts require special attention. For example, nearly 40% of all edits in Wikipedia involve communication, consensus building, and development of policies and procedures [Kittur et al., 2007].

Another relevant observation was pointed out in [Kittur and Kraut, 2008], where it was stated that adding more editors to an article improved its quality only when users used implicit coordination techniques and was harmful when they did not. The coordination techniques were based on workgroup structure, unspoken expectations and shared mental models of the document to be completed. Furthermore, having more editors wasn't associated with improvements in article quality when the work was distributed evenly among editors or when they used direct communication on the article page.

3.1.2 Asynchronous and Same Place CSCW

Regarding software used in the context of asynchronous and same place CSCW, this subsection presents an analysis of: large public displays and team rooms.

Large public displays

Public displays of big dimensions are a good choice to make if there is the need to hold multiple work areas. They are easy to see and can be manipulated directly via touch or speech.

One example of a large public display tool came from [Russell et al., 2002], where "BlueBoard" was presented. This system, shown in Figure 3.3, was a large plasma display designed to simplify small group collaborative work. It was touch-sensitive and had a badge reader to identify individuals using the board.

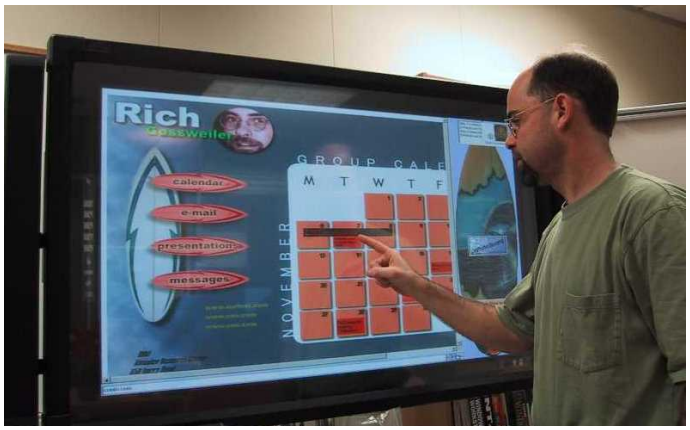

Figure 3.3: The Blue Board (extracted from [Russell et al., 2002]).

The onboard software acted as a client giving access to web-based content of each participant and possessed a set of tools for rapidly exchanging content between users.

Figure 3.4: A typical BlueBoard personal display (extracted from [Russell et al., 2002]).

As shown in Figure 3.4, a user could share information with a drag-and-drop movement of the content (e.g. a document or a page) to the personal icon of another user; the content was then e-mailed to the user when the content's owner left the session.

Despite its growing ubiquity, large display groupware often fails to be integrated into workgroup practice. In [Huang et al., 2006], the authors recognized patterns in design and deployment of large public displays that affected subsequent adoption success, and developed a framework of guidelines based on these patterns.

The set of guidelines included the design and deployment of tools to support both specific and general tasks, stimulating users to observe other users interact with the display for discovering potential uses of the system, allowing desktop interaction to

minimize barriers inherent to large displays (e.g. having to go to the display's physical location), and defining a dedicated core group of users early in the deployment.

Team rooms

Traditional team rooms are the ideal choice when there is the need for team members to be physically close and have a permanent shared space. Unlike meeting-centered models where documents do not exist after the meeting, team rooms are physical places and so they can store the meeting's documents for future meetings.

Like all co-located interactions, there is the possibility for each participant to observe the gestural communication movements of his companions, perceiving behaviors and states of mind that would easily pass by in remote interactions.

Being a work area, team rooms offers a set of common tools for collaboration, such as whiteboards and overhead projectors, while team members can bring their own personal "tools", like PCs, notebooks, books, documents, progress reports, task lists, and many more.

Despite being a good way for supporting co-located work, there are times when team rooms are not the most suitable choice. Nowadays, the trend towards more flexible organizations, with members scattered over geographical areas, makes the use of traditional team rooms an impossible choice. In these cases, *Remote Interaction* and *Communication and Coordination* CSCW technologies are more recommended.

An example of a team room tool was proposed by [Roseman and Greenberg, 1996]. The groupware system – called "TeamRooms" – filled the role of a team room for groups whose members could work both collocated and remotely, either synchronously or asynchronously. Like in a physical team room, users could customize their shared electronic space with both generic and specific tools to suit their needs, while all the artifacts in the electronic room persisted.

Generic tools, available in every room, included:

- *Chat*.
- *Shared whiteboard*, where users could select different colored pens or the eraser for producing freehand drawings.
- Awareness features: a *list of users in the current room*, with users identified by image stills or video snapshots; *telepointers*, one for each user, that communicated gestures to provide awareness of the actions of other users; and finally a *radar view*, which provided a miniature overview of the entire room.

Specific tools, each one being a special-purpose groupware application, included:

- *Post-it*, where users inside a room could write text sticky notes (which could be changed at any time). The changes were immediately seen on other users' displays.
- *Outline tool*, allowing team members to hierarchically organize a set of notes or ideas.
- *Concept map*, which organized information as a graph.
- *Database*.
- *File transfer*, which allowed external files to be placed in a room.
- *Web browser*.

All this tools can be seen in Figure 3.5.

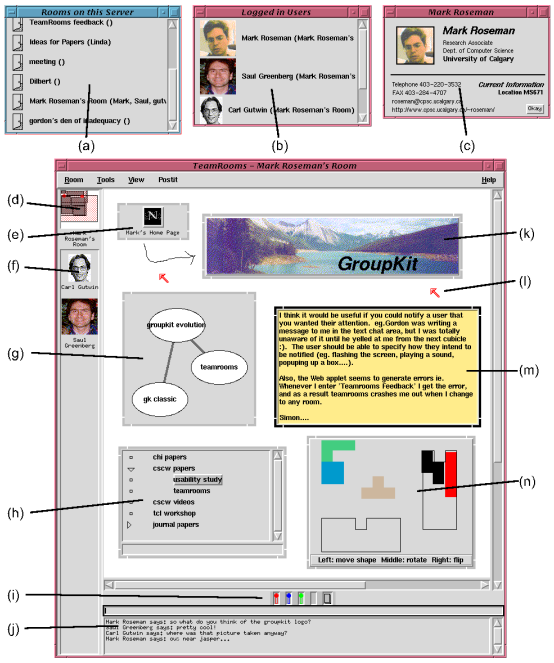

Figure 3.5: TeamRooms user interface, showing (a) available rooms; (b) connected users; (c) business card; (d) radar overview of room; (e) URL reference applet; (f) users in room; (g) concept map applet; (h) outliner applet; (i) whiteboard pens; (j) text chat area; (k) image applet; (l) telepointer; (m) postit applet; (n) tetrominoes applet (extracted from [Roseman and Greenberg, 1996]).

3.1.3 Synchronous and Different Place CSCW

Regarding software used in the context of synchronous and different place CSCW, this sub-section presents an analysis of: collaborative real-time editors/multi-user editors, instant messaging, voice over Internet protocol, and videoconferencing.

Collaborative real-time editors/Multi-user editors

Collaborative real-time editors allows several people to edit a file, at the same time, using different computers. Not only text documents can be edited in real-time. For example, with UNA, a real-time collaborative development environment developed by N-BRAIN, Inc, software engineers can edit wikis, text documents, spreadsheets, and source code[4].

One big advantage of a collaborative real-time editor is the revision history feature, which allows to know who made a change, what was the change, and at what time.

At present time there are several collaborative real-time editors, such as Google Docs and Ether Pad. The latter possesses impressive features when compared to other tools: allows editing on any platform and major browsers(Internet Explorer, Safari, and Firefox), provides revision saving coupled with restoration from stored versions, and incorporates a chat among participants in a document (including chronological archived discussions). Additionally, when compared to Google Docs, Ether Pad doesn't require e-mail or accounts to share a document, highlights who typed what, and doesn't lose undo history whenever someone else makes as change.

The drawbacks of this tool are fundamentally related to resources limitations.

The editing is not strictly instantaneous, because of the communication lag. To make matters even worse, there are limitations on the number of users that can collaborate at the same time, the total number of registered collaborators, and the files' size. For example Google Docs only permits 10 people to edit a document or presentation at the same time, and 50 people in the case of a spreadsheet. The number of people that can share a document, presentation or spreadsheet is limited to 200. A document can have up to 2MB per embedded image, a spreadsheet can only have up to 256 columns and 200,000 cells, and a presentation in .ppt and .pps formats can have a maximum size of 10MB.

Instant messaging

Instant messaging is one of the oldest collaboration tools, appearing for the first time on multi-user operating systems like CTSS and Multics in the mid-1960s.

Today, Instant messaging is widely used, posing however security breaches (IM Security Center recorded 1100 discrete attacks in 2004-2007). According to [Shiu and Lenhart, 2004] 42% of (Internet users – more than 53 million American adults – report using instant messaging. Certainly, IM's worldwide adoption was propelled by its easy and inexpensive installation, namely when compared to video conferencing.

Although the main feature of Instant messaging is allowing a form of real-time communication between users, saving conversations for future reference, at the present time IM offers much more than that. Mobile IM, business applications (e.g. Microsoft

[4] http://www.n-brain.net

Office Live Communications Server) and the possibility of using awareness devices, such as Webcams, are some example of the evolution of IM.

As stated by a study of the habits of computer-using workers [Garrett and Danziger, 2007], "*instant messaging in the workplace simultaneously promotes more frequent communications and reduces interruptions*" and more importantly "*(...) workers are developing effective strategies for using IM technologies in positive ways, even when more negative workplace impacts seem equally possible*". Another study provided the same insight, with the authors stating that "*chat was used overwhelmingly for work discussions or for articulation work to coordinate projects and meetings, and to negotiate availability*" [Handel and Herbsleb, 2002].

By focusing on the use of Instant messaging in a physically co-located group, IM can be useful not only for work in distributed teams [Quan-Haase et al., 2005]. Besides creating higher connectivity and a sense of community, in some cases IM also functions as a barrier. Employees use IM partially as a way to create distance between them and their superiors, which can be helpful when difficult decisions have to be made or sensitive topics have to be discussed.

The drawbacks of Instant messaging are mainly related to erroneous users' behaviors. Some examples of the dangers posed by IM are: *inappropriate IM use*, such as sending abusive messages in the workplace by hiding behind the informal, immediate, and anonymous nature of IM; *intellectual property leakage*, where employees using IM leak confidential information to an outside source; and *unethical users' behavior*, with users lying about their intentions and representing a danger or embarrassment to others.

Being a type of real-time communication, Instant messaging pose some threats. It can be a source of distraction [Farmer, 2003] and it is not fit for long and detailed conversations – in these situations, using voice communication is preferable.

To conclude, depending on the users reaction, Instant messaging can be annoying. Unlike chats, when a conversation begins the user receives from another user a pop-up window interrupting his work. The user has then to decide between delaying the reply until his work is done and interrupting the work and initiating the conversation, which when finished can reveal to be shallow.

Voice over Internet Protocol

Voice over Internet Protocol is a general term for a family of transmission technologies used for delivery of voice communications over IP networks.

VoIP can be used both for personal and business use (e.g. Skype[5] offers business VoIP applications).

VoIP greatest benefit is its low cost, both in installation as well as in operational terms.

[5] http://www.skype.com

Calls to other users of the service are free. This advantage is significant when the users are geographically distributed. However, free calls don't apply when calls are made to other landlines and mobile phones – these calls are subject to a fee.

One major cost saving characteristic of Voice over Internet Protocol is that unlike Public Switched Telephone Network (PSTN), numbers are not location or distance dependent.

Another advantage of VoIP is the integration with other technologies – it can easily and without any cost integrate with other services available over the Internet or mobile related. For example, Skype, the famous VoIP software, allows users to make video and instant messaging conversations, and also to send SMS messages directly from Skype or to use it on a cell phone.

However, VoIP's Quality of Service (QoS) is a challenging problem, since the quality of a VoIP connection depends on the available bandwidth. Furthermore, being the underlying IP network unreliable – in contrast to the circuit-switched Public switched telephone network (PSTN) – VoIP is vulnerable to latency and jitter. However, to address this problem, a number of protocols have been defined, such as RFC3611 and IEEE 802.11e-2005.

Another problem comes when in an emergency situation. Unlike PSTN, if there is no power a user cannot make a call.

To conclude, like any computer-based technology, VoIP systems are susceptible to attacks, such as denial-of-service attacks or recording conversations without the user's knowledge.

Videoconferencing

A videoconference is a set of interactive telecommunication technologies which allows two (point-to-point) or more locations (multi-point) to interact via two-way video and audio transmissions simultaneously.

High-speed Internet connectivity – a pre-requisite to have a communication with few delays and good video and audio quality – and hardware evolution have been opening doors for this tool. However, high-speed Internet is yet to be available for all users.

Videoconferencing is particularly helpful when there is the need to join more people into a single meeting room than those that would fit in just one place, since each party uses its own facilities. Another logistics advantage comes with the absence of needing to wait and pay for each person to travel to the conference's place.

The specter of areas where videoconferencing can be used is broad, reaching education (e.g. Virtual Learning Environments can bring opportunities to students in geographically isolated locations or economically disadvantaged), medicine and health, business, law, and media relations.

As a communication medium, videoconferencing is richer than any other tool that supports distributed groupware. Deaf and Hard hearing individuals, as well as blind or low vision individuals, can use this tool – Video Relay Service, for example, is a telecommunication service that allows this type of communication [FCC, 2008]. Moreover, it's a more personal type of communication, since it's possible to simultaneous hear and see the other person's response, including its gestures.

According to [Vertegaal and Ding, 2002], "*(...) with 22% more speech and 26% more talkspurts, subjects were significantly more likely to speak when gaze behavior of partners was synchronized with conversational attention. Task performance was also 46% higher in this condition*".

A problem observed during a videoconferencing session relates to the fact that "*the focal point of the screen is not the focal point of the camera, and it is therefore impossible to both look at the person you are talking to, and see them as well*", resulting in a very unnatural conversation [Meggelen, 2005].

At a human level, videoconferencing exposes the anonymity and privacy of the person, hence probably restraining potential users.

At a technical level videoconferencing presents intimidating challenges. The technical complexity drives back users who don't have technical knowledge and just want a simple interface. To avoid this problem, a support team, capable of providing fast and effective technical assistance, must be present during video conferencing meetings. Another criticism has been the lack of interoperability, where some systems aren't readily interconnected, such as ISDN and IP systems requiring a bridge.

3.1.4 Synchronous and Same Place CSCW

Regarding software used in the context of synchronous and same place CSCW, this subsection presents an analysis of: electronic meeting systems, single display groupware, and roomware.

Electronic meeting systems

According to [Nunamaker et al., 1991], who coined the term, Electronic meeting system (EMS) is "*a new form of meeting environment (...) which strives to make group meetings more productive by applying information technology*", "*designed to directly impact and change the behaviour of groups to improve group effectiveness, efficiency, and satisfaction*". In that paper, Nunamaker et al. alerted to the fact that their definition of meeting is broad, including any activity where people come together, either collocated and synchronously, or remotely located and asynchronously.

These systems (see Figure 3.6, to observe an EMS sequence of use) increase teams' productivity [Nunamaker et al., 1996]. Each user can contribute to the same shared object

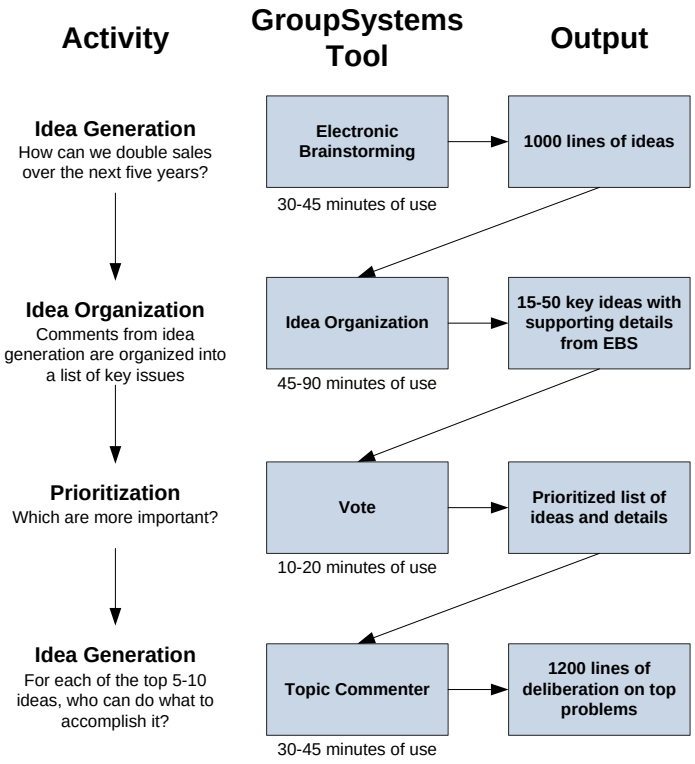

Figure 3.6: One sequence of use of an EMS (adapted from [Nunamaker et al., 1991]).

at the same time, thus nobody needs to wait for a turn to speak. Everyone has equal opportunity for participation, with their privacy protected, since each user contributes anonymously. The most important is not who said what and the fear of the repercussions – such as when an employee states that its boss is wrong – but the ideas that result from the discussion.

An EMS reduces memory demands associated with tracking groupware discussions. It offers access to external information (e.g. a database) and also the development of an organizational memory, by recording inputs, outputs and results in one repository. In addition, the group can prioritize a list of key issues into a short list, using the Alternative Evaluator, a multi-criteria polling tool [Nunamaker et al., 1991].

Another advantage of an EMS is that it boosts awareness. Participants' comments ate displayed on larger screens at the front of the room, as well as on each workstation.

On the negative side, an EMS requires networked computers, a projection screen, and EMS software. Additionally, it is less useful when there is the need for complex decision making – to address this scenario, Group Decision Support Systems (GDSS) are more indicated. Finally, the presence of a facilitator or group leader is necessary to design a step-by-step process for the team to follow so that the goal is achieved.

Roomware

According to [Prante et al., 2004], roomware "*integrates information and communication technology into room elements such as walls and furniture*". Roomware exemplifies, on a small scale, the potential of ubiquitous computing[6].

Roomware combines real objects of a room with computer-based support. Components of a building – such as walls and tables – use technology support to cooperate with the building's inhabitants by adapting to changing situations and providing context-aware information and services.

The collaborative potentialities of roomware are enormous. Roomware represents a shift from desktop-based interaction to human-information interaction and human-human cooperation based on real world objects, providing meetings and group work the special computer-based support they need, especially with respect to creative activities.

Prante et al. proposed a dedicated software infra-structure called Beach (see Figure 3.7), which acted as a framework for collaborative software and also provided support for interaction with roomware components.

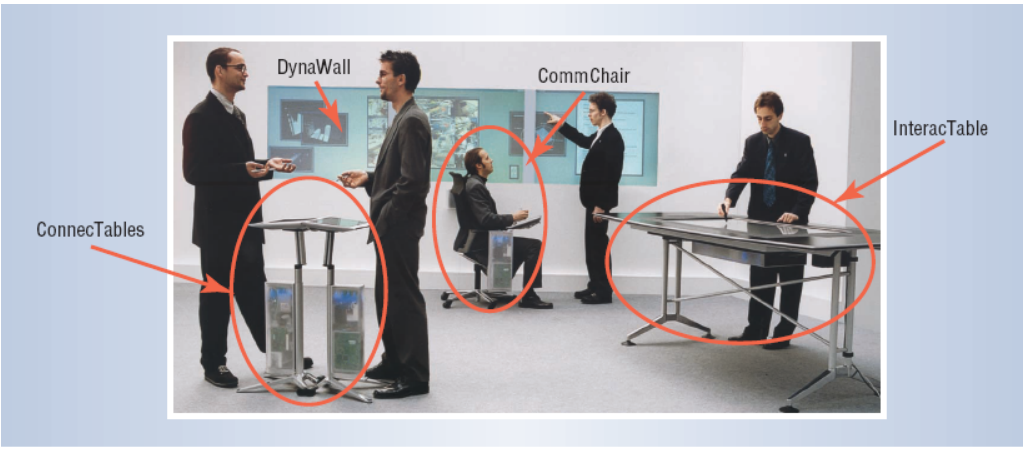

Figure 3.7: Beach, an example of second-generation roomware components (extracted from [Prante et al., 2004]).

The components developed were:

- *DynaWall*, a large touch-sensitive information display and interaction device integrated in a room's wall. This component enabled users to work in parallel on a shared document, in a contiguous interaction area composed of three screens and controlled by separate computers.
- *InteracTable*, a touch-sensitive plasma display integrated into a tabletop.

[6]Ubiquitous computing is a term used for referring the use of very tiny, almost invisible, devices, either mobile or embedded in almost any type of object imaginable, communicating through interconnected networks

- *CommChair*, a chair with the functionality of a pen-based computer, connected to all other roomware components via a wireless network.
- *ConnecTable*, a modular version of the previous component, in the form of a table, which allowed extending its workspace area to chairs located nearby.

Finally, being a recent research area, the applicability of roomware is limited. Moreover, the technology that supports roomware is complex, restricting even more its acceptance. For example, Beach required a network infrastructure to provide the connectivity between the components, a software infrastructure in order to have a wide range of cooperative sharing capabilities, and a sensing and localization infrastructure for extending the mobility of team members and roomware components in the whole building [Streitz et al., 1991].

Single display groupware

Single display groupware (SDG) is a term coined by [Stewart et al., 1999], who defined it as "*computer programs that enable co-present users to collaborate via a shared computer with a single shared display and simultaneous use of multiple input devices*".

SGD's technology primes for its multi-purpose context applicability. A group of people can interact with a single display groupware at the same time and in the same place, choosing one of the multiple input devices, such as a mouse, a keyboard, and many more.

Not only PCs can be used as an interaction medium. The Pebbles project investigated the use of hand-held Personal Digital Assistants (PDAs) as portable input devices in an SDG, allowing the users to share the same screen [Myers et al., 1998].

Limiting the collaboration to a single display can be problematic. When many users want and need to share the display, it can be confusing to perceive who is doing what and what is being used.

3.2 Social Software

This section presents the main advantages and disadvantages of the social software tools that were studied in the context of this dissertation: feeds, folksonomies, social bookmarking, social cataloging, and social online storage.

3.2.1 Feeds

A web feed (or news feed) is a data format used for providing users with frequently updated content. Users subscribe content syndicated by distributors and make collections of web feeds through an Internet aggregator application.

Web feeds not only reduce the time and effort necessary for the users to memorize and search the content of their interest, but also extend the content's diversity. Web feeds can be used in many types of websites (e.g. websites, social bookmarking websites, weblogs, schools, and podcasters) and in heterogeneous contexts (e.g. news, business, sport, entertainment, tech and science, computers and videogames, and shopping).

Using web feeds as an advertising or marketing tool is advantageous to both the user, who is actually interested in receiving product updates, and to the website owner, since advertising becomes targeted.

When compared to receiving frequently published content via e-mail, web feeds are less vulnerable to threats like spam, viruses, phishing, and identity theft, since users do not disclose their e-mail address. Another comparison that can be done to e-mail is that unlike an e-mail inbox, where all mails are mixed-in together, feeds are automatically "sorted" (i.e., each feed URL has its own sets of entries).

Other strong point of web feeds is its flexibility, to the user and as a technology.

That said, if the user wants to stop receiving news, he doesn't have to send an unsubscribe request – he just has to remove the feed from the aggregator. Hence, users have the power to easily subscribe/unsubscribe at any time their feeds. Additionally, there is no need for the user to answer to questions on why he is unsubscribing and then confirm the unsubscribing.

On a technology viewpoint, there is a wide variety of Really Simple Syndication (RSS) readers available for multiple platforms – such as Windows, Linux, Mac OS X, cross-platform, web-based, and mobile – and multiple purposes, like email converters, plugins for browsers, managers, mixers, directories, and many more [Schroeder, 2007].

It's not just PCs that are gaining RSS attention, but also mobile devices. For example: DoYouFeed.com[7] transforms any RSS feed into an iPhone friendly site; LifeFeeds[8] lets users bookmark articles to their del.icio.us account, email articles to their friends and post articles to their blog; NewsGator Go![9] allows the users to read the feeds while offline and to synchronize the posts they already have read between devices; and Google Reader[10] allows the users to share content with their friends.

The main disadvantages of web feeds pertain their lack of usability. One problem is that since RSS feeds do not display the actual URL or name of the website, it can get confusing on what feed a user is actually reading. Another problem is that the user can't read, in the RSS feed reader, the full text of the feed but only a summary (in the cases there is one).

[7] http://www.doyoufeed.com
[8] http://www.litefeeds.com
[9] http://www.newsgator.com
[10] http://www.google.com/mobile/default/reader.html

Other drawbacks are the requirement of an RSS feed reader for using web feeds and the publishers not being able to determine how many and how frequently users subscribe to their feeds [Kamble, 2008].

3.2.2 Folksonomies

The origin of the term *folksonomy* retraces to 2004, when Thomas Vander Wal defined the informal social classification present in services like Furl (launched in 2003), Flickr and Del.icio.us (both launched in 2004).

According to Vander Wal, "*Folksonomy is the result of personal free tagging of information and objects (anything with a URL) for one's own retrieval. The tagging is done in a social environment (shared and open to others). The act of tagging is done by the person consuming the information*" [Wal, 2007].

The existence of three fundamental data in a folksonomy tool – the *person* tagging, the *object* being tagged as its own entity, and the *tag* being used on that object – can be of extreme value, because there is the possibility of using two elements to find the remaining one. For example, if the object is know (e.g. the web page being tagged) as well as its tag, it's possible to find other individuals who used the same tag on that object, which may lead to somebody who has the same interest and vocabulary as the user [Wal, 2005].

Folksonomies can be classified in two ways [Smith et al., 2005]. The first is the number of people tagging one resource: it can be *broad* (such as in the case of del.icio.us[11] or CiteULike[12] applications) where many users tag the resource, or it can be *narrow* (such as in the case of Flickr[13] or Technorati[14]) where few users tag the resource. The second (shown in Figure 3.8) is using two axes: one for distinguishing if the content being tagged belongs to the user (*my stuff*) or to other user (*other's stuff*), and another for distinguishing if the content being tagged is only for the user's benefit (*private tags*) or also for other's benefit (*public tags*).

Tagging is an activity with a very low learning curve – when adding tags, people use their own vocabulary, associating explicit meaning to the object. Users create tags as quickly as they create content and they are immediately added to the ontology [Kroski, 2005]. This user-based tagging allows not only the user to organize his data but also and most importantly allows the user to share the data with other users, hence augmenting collective intelligence. The network effects, which result from users' contributions, are the key to market dominance in the Web 2.0 era [O'Reilly, 2005], where everyone's vocabulary is included, reflecting everyone's needs (the democracy nature of folksonomies), without cultural, social, or political bias.

[11] http://delicious.com
[12] http://www.citeulike.org
[13] http://www.flickr.com
[14] http://technorati.com

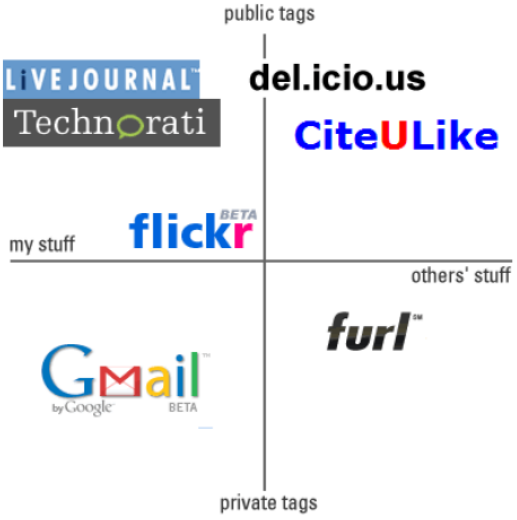

Figure 3.8: Folksonomies two-axis classification (extracted from [Smith et al., 2005]).

According to a study of the del.icio.us website conducted by the Information Dynamics Lab at HP Labs [Golder and Huberman, 2005] "*because stable patterns emerge in tag proportions, minority opinions can coexist alongside extremely popular ones without disrupting the nearly stable consensus choices made by many users*". The sustainable growth of diversity among tags, directly linked to users' tagging unlimited liberty, lets to the discovering of unknown and unexpected resources [Kroski, 2005]. This discovery is especially powerful when the tagging is a social activity.

In a traditional classification scheme, one category term is selected which includes all related terms. When future objects are cataloged, it must be determined that either they fit into a particular category or they do not. In a folksonomy, these items can fit into multiple categories and, like in a traditional classification schemes, tags have diverse application domains, ranging from social bookmarking to site navigation. Additionally, like a traditional classification scheme, tags can be searchable.

The enormous amount of information that is now being published in the Web through new mediums, such as blogs and wikis, make a traditional taxonomy and controlled vocabulary an impossible solution, because of the cost and the amount of manpower that the situation would require [Kroski, 2005].

When compared to web search engines, tags are in principle more reliable. In a tag, Internet resources are classified by human beings, who have a real understanding of the resource's content, as opposed to software, which algorithmically attempts to determine the meaning of the resource by extracting words from the titles, headings, or special fields called meta tags.

Since tags are created through users' own vocabulary, they can be seen as a kind of

social bookmarking. According to del.icio.us' Joshua Schachter, tagging it is "*basically a way to remember in public*", that is, a way for the users to be able to recall their information at a later time [Kroski, 2005].

Users' liberty on choosing tags according to their own vocabulary is also related to tags' main criticisms.

The absence of *synonym control* is one of biggest problems of tags. For example, if a user tagged himself in a photo with his own name or with "me", then the two tags would be interpreted as distinct. A serious consequence of this is *lack of recall*, i.e., a system's lack of ability to return all resources related to a topic. This phenomenon occurs because a folksonomy search doesn't return a complete results list, in consequence of using similar tags. For example, a search for "cat" will not retrieve resources tagged with "kitten", "feline", "tabby", or even "cats" [Kroski, 2005]. However, as Shirky pointed out [Shirky, 2004], synonym control suffers from the fact that "*related terms (...) cannot be trivially collapsed into a single word without loss of meaning, and of social context*".

Another problem of tags is their quality, when compared with a controlled vocabulary. Examples of such disadvantages are as follows:

- The absence of a standard for the structure of tags (e.g.: singular vs. plural, capitalization, etc). On the contrary, the use of *controlled vocabulary* – an orderly approach to cataloguing applied for example in libraries – allows for both the validation and quality control of known terms to be registered within an information system [Hammond et al., 2005].

- Mistagging due to spelling errors.

- No mechanism for users to indicate hierarchical relationships between tags has both its downsides("*The top few categories of a traditional hierarchy give us a much better idea of the contents of a media collection than thousands of individual tags, even if these tags are ranked by their frequency in the collection*" [Heymann, 2008]) but also its upsides("*A formal classification system needs generally to be predictive both of the ordering of terms that are used within it, and of the terms that will be allowed (or tolerated) by it.*" [Hammond et al., 2005]). There have been studies that research ways of constructing hierarchies in social tagging systems, like [Heymann and Garcia-Molina, 2006], which proposed an algorithm for leveraging notions of similarity and generality, implicitly present in the data generated by users as they annotate objects.

Therefore, there is no oversight as to how resources are organized, with the quality of the tags depending on users' knowledge, intentions and imagination.

Although folksonomies can be seen as discovery systems, they lack the powerful search capacity of a hierarchical taxonomy. Consequently, is nearly useless for searching out specific, accurate information [Sterling, 2005].

Finally, collective tagging has the potential to exacerbate the problems associated with the fuzziness of linguistic and cognitive boundaries. Systematic variation across individuals, in what constitutes a *basic level* (that which is most directly related to humans' interactions with them), can have significant consequences in tagging [Tanaka and Taylor, 1991]. For example, a document tagged with "perl" and "javascript" may be too specific for some users, while a document tagged with "programming" may be too general for others [Golder and Huberman, 2005].

3.2.3 Social bookmarking

We live in a time where the Web is an apparently unlimited source of information – on 24 May 2009, the indexed web contained at least 41.23 billion pages [WorldWideWebSize, 2009]. The real challenge comes when we need to manage this information overload, capturing the information that we seek and keeping that information for future reference.

Internet bookmarks appeared in 1993, with the launch of Mosaic browser, to eliminate the problem of viewing a page and not being able to save the hyperlink of the page. However, this feature had a critical problem, which accentuated with the passage of time: the bookmarks only lived in one PC. If the user switched to another PC, all the bookmarks he saved in the other machine would be lost.

Online bookmarking emerged has the solution for this problem. By saving the bookmarks in the Web, the user was no longer dependent of a particular machine, thus being capable of maintaining one single repository for the bookmarks, accessible everywhere.

Created in 1996, itList, one of the first online bookmark services, allowed the users not only to store and organize their bookmarks online but also to share them with other users; users could choose whether or not to make their list public [Project, 1999].

A list of social bookmarking services was launched until the release of del.icio.us on 2003. Del.icio.us (see Figure 3.9), one of the most popular social bookmarking services available today, pioneered tagging and coined the term *social bookmarking*. Del.icio.us' success is explained not by its features but by its usability. The integration of del.icio.us with browsers, like Mozilla Firefox, is also responsible for this success. Other social bookmarking services are Furl, social citation services Citeulike and Connotea, the recommendation system Stumbleupon, and social news website Digg.

In a social bookmarking system, users can bookmark web pages not only for themselves but also to others, giving users the opportunity to express differing perspectives on

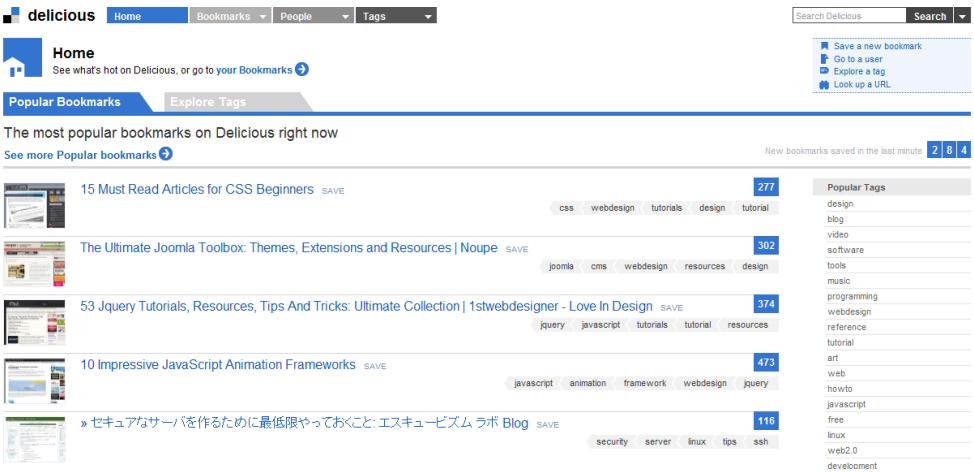

Figure 3.9: Screenshot of social bookmarking site del.icio.us.

information and resources through informal organizational structures [Educause, 2005]. As pointed out in [Hammond et al., 2005], the more social bookmarking is used, greater is the value that accrues to the system itself, and thereby to all who participate in it.

Being a social activity, social bookmarking can foster collaboration. For example, del.icio.us has the *Network* concept, helping groups share useful links. Instead of just one person finding relevant and useful information, communities within and across organizations can make use of this simple capability and enhance their collaboration practices.

One major advantage of social bookmarking, over browser bookmarks, is the capability of organizing bookmarks in one place for portability. In addition, in order to maintain user's privacy, bookmarks can either be public or private, and RSS feeds can used to keep track of bookmarks activity.

Heymann [Heymann et al., 2008] published an interesting article regarding the benefits of using social bookmarking for searching the Web. Some conclusions were:

- As a data source for search, social bookmarking has URLs that are often actively updated and prominent in search results. Tags can be made by the community to guide users' to valuable content before search engine indices are updated with web sites related to the tags.
- As a source for new pages, social bookmarking may help search engines discover pages it might not otherwise.
- While some users are more prolific than others, the top 10% of users only account for 56% of posts.
- Del.icio.us has relatively little redundancy in page information.

- Popular query terms and tags overlap significantly, though tags and query terms are not correlated. One likely reason the two are uncorrelated is that search queries are primarily navigational, while tags tend to be used primarily for browsing or categorizing.

- Tags are on the whole accurate. Roughly 7% of tags were deemed "irrelevant", few tags were deemed "subjective" – less than one in twenty for all users – and almost no "spam" was found in the dataset.

- Although there was a reasonable degree of overlap between query terms and tags, there was no positive correlation between popular tags and popular query terms. The explanation why the two are uncorrelated is that search queries are primarily navigational, while tags tend to be used primarily for browsing or categorizing.

The contexts where social bookmarking can be used aren't limited to websites. For instance, libraries can use social bookmarking tools to create web-based subject bibliographies – PennTags, an initiative from the Library at the University of Pennsylvania, encourages the use of tagging.

In the same article, Heymann et al. mentioned some problems associated to social bookmarking:

- The number of registered posts per day was relatively small, with approximately 120,000 URLs posted each day to Del.icio.us. Moreover, the number of total posts was also relatively low – there were roughly 115 million public posts, coinciding with about 30-50 million unique URLs. Thus, URLs produced by social bookmarking are unlikely to be numerous enough to impact the crawl ordering of a major search engine.

- The tags which annotate URLs, while relevant, are often functionally determined by context. Tags were present in the page text of 50% of the pages they annotated and in the titles of 16% of the pages they classified. Hence, tags are unlikely to be much more useful than a full text search emphasizing page titles.

Due to the community structure of social bookmarking, the doors are wide open to spammers, who misuse the popularity and the high PageRank of social bookmarking systems for their purposes. All they need is an account – then they can freely post entries which bookmark the targeted spam web site. To confront this risk, which can diminish the quality of social bookmarking systems, Krause [Krause et al., 2008] introduced features to eliminate spam in social bookmarking systems using machine learning approaches.

3.2.4 Social cataloging

Social cataloging is a fairly recent web concept, when compared to social bookmarking, and it has the potential to revolutionize how people organize information. This potential is reflected in the possibility of cataloging virtually anything, ranging from books to scholarly citations.

One of the most popular social cataloging applications is Library Thing[15]. Founded in 2005, Library Thing offers to anyone the possibility of storing and sharing personal library catalogs and book lists, counting more than 600,000 book lovers and 35 million books [Kiss, 2008].

Another popular social cataloging application is Last.fm[16], a music community website founded in 2002, that had by 2008 over 21 million active users based in more than 200 countries. Although users are not allowed to upload copyrighted audio files, commercially available albums are regularly added by Last.fm staff. Users have also the possibility of tagging songs and music groups to better organize their tracks.

Other applications include social movie site Flixster[17] and scholarly citations sites Bibster[18], CiteULike and Connotea[19].

Social cataloging is similar to social bookmarking, especially on the use of tags and the notion of sharing resources in a community. Thus, most of the pros and cons related to tags and sharing that existed in social bookmarking also apply to social cataloging. That said, the advantages and disadvantages of social cataloging exhibited here refer mainly to online catalogues and their respective domains of application, like books, scholarly citations or music.

Social cataloging can make cataloging a revolutionary experience and a more flexible process. For example, in the case of LibraryThing, members can [Bell, 2007]:

- Catalog their books.
- Connect to people with the same books.
- Receive suggestions for what to read next.
- See the names of the members with the 50 most similar libraries (“Members with your books” feature).
- See how many and which books are shared in common between two users, when viewing another member’s profile or library.
- Leave a comment on another member’s profile.

[15] http://www.librarything.com
[16] http://www.last.fm
[17] http://www.flixster.com
[18] http://bibster.semanticweb.org
[19] http://www.connotea.org

- Add a book's review, conversation topic and rating.
- Get a book from or give a book to other person, through book swapping sites ("Swap this book" feature).
- Import information from external libraries.

Social cataloging opens the door for creating a social network of virtually any type of resource, where anyone can catalogue (not only librarians).

As a web-based tool, social cataloging can agglomerate information distributed in multiple sites, both physical and virtual resources. LibrayThing permits the cataloging of books from Amazon, the Library of Congress and 690 other world libraries and import them from anywhere. CiteULike supports more than 60 sources, such as Amazon, CiteSeer, IEEE Digital Library and SpringerLink. Last.fm's music library contains over 3.5 million individual audio tracks from artists on all the major commercial labels.

In some social cataloguing applications there is repetitive complaint: the automated selection of a user's email account's entire address book so that an invitation to all the contacts can be sent. Examples of such applications include the social movie site Flixter [Wikipedia, 2009] and the social network for booklovers site Shelfari [Wegman, 2007].

3.2.5 Social online storage

Considering online storage solely, its advantages are related to having a resource available independently of the user's location, while the disadvantages are a consequence on depending on a third party.

On the positive side less storage devices is required, since the third party is responsible for it. Online data can be accessed from any part of the world with just a computer and an Internet connection.

An application of online storage is backup. The data of a user is safe and protected at a location that is far his own, which can be a big plus in situations where some sort of accident or event (e.g. fire, burglaries, and natural disasters) causes permanent damage to hard drives and other records that are kept around the office or home [Edwards, 2008].

On the negative side, since the data is stored by a third party, the user has less direct control over it than what he could have had if the data was onsite. Another problem is that if the company is discontinued or modifies its service, there is the possibility of losing the data, without prior notice and without indication of the reasons.

Considering social online storage exclusively, users can share files with friends or everyone. Changes performed on one computer will also be visible instantaneously on another computer.

An example of a social online storage tool is Wuala[20]. Being a P2P social online storage, it gives the possibility of choosing who of the user's friends can access to the folder or making the folder visible to all, and to share a folder by password and send a link plus password by email or in an instant message. If the user doesn't want to restrict the access to the file, it is possible to copy the link of the public file and post it anywhere (email, chat, blog, forum, etc.), and to point people to the user's latest pictures, scientific papers, freeware applications, or to his files from MySpace, Facebook, etc.

In Wuala, it's easy to communicate through comments, do joint editing on documents, upload files with no size or type limitation, set roles and permissions, stream videos and assign them to members, and harness the speed of a P2P network [Brenner, 2009].

A disadvantage of social online storage is that data whose reproduction presents an infringement of the copyrights can very quickly be made public.

3.3 Software Engineering Software

The Software Engineering Body of Knowledge guide [Abran et al., 2004] organizes Software Engineering as a discipline spanning ten Knowledge Areas: Software requirements, Software design, Software construction, Software testing, Software maintenance, Software configuration management, Software engineering management, Software engineering process, Software engineering tools and methods, and Software quality.

Throughout time, within each of these areas, software engineers have been integrating general purpose and social software – presented in sections 3.1 and 3.2 – into their teamwork activities.

E-mail has been broadly adopted as an electronic communication medium, due to the combination of no set-up costs, asynchronous nature and the capability of sending the same message to a great number of recipients. One common usage of e-mail is as a notification system, such as when a new bug is added or when a new file is uploaded (e.g. project management tool Basecamp).

In the past years, blogs have been used in IT departments for easing the communication between employees, both within projects and for personal notes. For example, after Google acquired the blogging service Blogger in 2003 and deployed an internal blog for its employees, there was a growth of communication and awareness, with employees keeping track of meeting notes, sharing diagnostics information, sharing snippets of code, and sharing personal notes.

Microblogs are beginning to be used for the quick sharing of work and personal information, founded on the principle of a software engineer following the activities of their peers or the tools they use. For example, bug tracking system MantisBT, by

[20] http://www.wuala.com

integrating with micro-blogging service Twitter, keeps the followers of Mantis updated with all issues that are resolved in the bug tracker as well as manual updates.

Wikis are used by software engineers for the collaborative writing of documents, such as drafts of designs and implementations, design tradeoff discussions, requirements gathering, and user guides.

Videoconferencings, Voice over Internet Protocol, and Instant Messaging have been used for establishing communications between remotely located software engineers. For example, in NetBeans IDE developers can discuss source code with other remote developers.

Web feeds have been growing in popularity as a substitute notification system to e-mail, due to its increased flexibility on executing subscribe/unsubscribe actions and to the organization it implies. Contrarily to e-mail, where the subscribed content is mixed with other messages of the inbox, each URL has its own set of entries. The area on which feeds are currently more frequently used is **Software engineering management**, for monitoring the activities of a project (e.g. in project management tool Microsoft Project).

However, software engineers have also been using tools specifically defined for addressing the particular needs of Software Engineering. For example, Source Control Management can be found nowadays in almost all IDEs, giving developers the ability to work concurrently on files, to merge changes with other developers' changes, and to track and audit changes that were requested and made. Over the time, SCM has been used not only within the software development phase, but also within other phases, such as software documentation (e.g. in applications like word processors, spreadsheets, and wikis), software requirements (e.g. in Rational RequisitePro), and software design (e.g. in web-based tool Gliffy, allowing users to save the history of changes made to a diagram).

3.4 Summary and Discussion

This chapter presented the state of the art of the collaborative software in three categories: **General Purpose Software**, **Social Software**, and **Software Engineering Software**.

General Purpose Software tools can either be applied in a Software Engineering environment or on other contexts of use. The tools were classified according to the context of its use and agglomerated into four groups of tools: **Asynchronous and Different Place CSCW**, **Asynchronous and Same Place CSCW**, **Synchronous and Different Place CSCW**, and **Synchronous and Same Place CSCW**.

The collaboration benefits received from a tool depends on the match between the user's needs and the tool's potentialities – for maximizing the number of receivers of a message e-mail would be the best choice, for a document manager a wiki should be chosen, for augmenting awareness microblogging would the ideal choice, and for

providing video and audio transmissions where there are users geographically located, videoconferencing would be recommended when the resources are abundant and VoIP/IM with Webcams when the resources are scarce.

A software tool can and should be used in conjunction with other tools when there is complementariness of benefits, such as when tags are used in conjunction with e-mail to reduce the loss of information context, single display groupware and large public display are integrated in a roomware system, or when VoIP is combined with IM to make clearer conversations difficult to be putted into words.

Social Software tools are a family of programs that allow users to interact and share information with other users. The analyzed tools were: **Web feeds**, a powerful awareness mechanism; **Folksonomies**, which refers to tagging, an action intensively used in CSCW and other social software tools for user-based organization of content; **Social bookmarking**, for saving web sites via a browser; **Social cataloging**, which extends Social bookmarking to virtually any type of item; and **Social online storage**, useful for saving and make available resources in the Web. Usually all social software embeds social networking features, which explicitly construct networks of users liked by shared interests.

Over the past decades, software engineers have been adapting general purpose and social software to their needs or using tools specific to the domain of Software Engineering.

Software engineers have been integrating tools as diverse as e-mail, blogs, microblogs, wikis, videoconferencing, VoIP, IM, and web feeds to collaborate more efficiently. E-mail is used not only as a communication tool but also – as with web feeds – for making notifications, when integrated with other tools. Internal blogs are used for easing the communication between employees, at a work and personal levels. Microblogs allows users to quickly share personal and work information, thus augmenting the awareness of each individual. Wikis are used for the collaborative writing of documents. Videoconferencing is best suited for establishing communications between remotely located software engineers when the bandwidth is high, while VoIP and IM are more appropriated when the bandwidth is medium or low.

Moreover, software engineers have been using tools created solely for Software Engineering purposes. An example of such tools is Source Control Management, a tool frequently used in multi-elements projects and in various phases (e.g. software development, software documentation, software requirements, and software design), where there is the need for managing the changes made by each element to the files of a project.

Chapter 4

Collaboration on Software Engineering Tools

This chapter exhibits the tables summarizing in-depth research made for software tools within a subset of Software Engineering areas. The details of the research of each tool can be seen in Appendix A.

The tools areas chosen to be studied were: **Bug Tracking**, **Construction**, **Design**, **Engineering Management**, and **Requirements**. A special type of tool area – which aids Software Engineering projects – was considered, called **"Collaboration Tools"**, corresponding to tools specifically developed for collaborative purposes, spanning the previously mentioned areas and also integrating general purpose software and social software tools (analyzed in sections 3.1 and 3.2). Except **Bug Tracking** and **Collaboration**, all the areas match the knowledge areas of the Software Engineering Body of Knowledge guide [Abran et al., 2004].

There were studied a total of 47 tools, with the following distribution by each area: 5 within Bug Tracking, 9 within Construction, 13 within Design, 8 within Engineering Management, 5 within Requirements, and lastly 7 within Collaboration.

Each tool was analyzed according to three criteria: *Collaboration*, *Integration*, and *Other characteristics*.

The first criterion, *Collaboration*, was divided in three parameters: *Awareness*, *Communication*, and *Collective Knowledge*.

Awareness regarded *Team awareness* ("Who is doing what, when and where?", "Who knows about my activity?"), *Context awareness* ("What and who relates to what I am doing?"), and *Resources awareness* ("Which resources relate to what I am doing or seeing?"). Basically, as shown in Figure 4.1, *Team awareness* considered uniquely people's activity, *Resources awareness* considered uniquely resources' (e.g. a

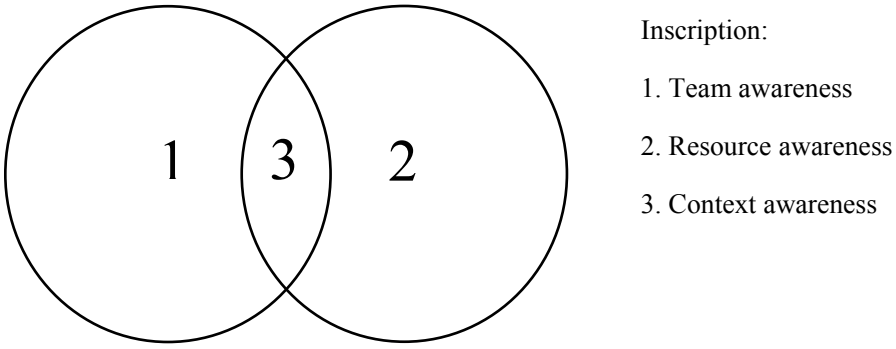

Figure 4.1: Awareness parameters.

document) activity, and *Context awareness* considered the intersection between people's and resources' activities.

Communication regarded how the user reached his peers (e.g. chat, e-mail, IM, etc.) and how he expressed himself to others (e.g. comments).

Collective Knowledge regarded to what extent was collective knowledge harnessed (e.g. folksonomies, tagging, ranking, polls, etc).

The second criterion, *Integration*, considered to what extent each tool provided integration for, or received integration from, other tools, either as an application (taking the form of a plugin) or through data (e. g. importing a file from another application). If the integration had collaboration purposes, the integration's information was added only to the *Collaboration* criterion.

In the cases where a tool doesn't provide collaboration or integration features in a criterion, the criterion is ommitted.

Finally, the third and last criterion, *Other characteristics*, presented additional information to the first and second criteria, such as when the tool is paid and how many days are provided in its trial. This additional information is shown in Appendix B.

Each tool was characterized according to twenty-seven features potentially capable of supporting collaborative activities. The bullet symbol means that the feature on the vertical axis was found in the tool on the horizontal axis.

The majority of the features were already studied in sections 3.1 and 3.2. However, there are some new features:

- **Comments** refers to users having the possibility of adding comments or notes inside the application.
- **Contact Management** encompasses functionalities related to saving and managing contact information (e.g. name, mobile phone, etc.) of a user.

- **File sharing** refers to performing file uploading activities inside a group.
- **Folder sharing** refers to aggregating files in one folder accessible to a group of people.
- **Online whiteboard** is a digital whiteboard available online for users to execute activities normally reserved to a physical whiteboard.
- **Recommender** is a recommendation system, a type of information filtering technique that presents information items (such as documents and books) that are likely of interest to a user. In order to make a recommendation, the system compares the profile of a user to some reference characteristics, thus predicting the "rating" the user would give to items not yet considered. The profile can be built either through *explicit data collection*, where the system asks the user to communicate his preferences regarding one or more items, or through *implicit data collection*, where the user's behavior is recorded so that his interests are discovered.
- **Screen sharing** refers to sharing the same screen for co-located (in this case similar to a SDG system) or distributed users.
- **Team Timeline** is a more restricted Gantt graph that shows a graphical representation of a chronological sequence of events (normally milestones in Software Engineering).
- **To-Do list** is a list of tasks that need to be performed, according to a priority and within a time interval.
- **Voting** refers to a set of functionalities that enable users to vote, such as polls or classifying an item in a scale from 1 to 5, and afterwards to filter information (e.g. in rankings or recommendations).

4.1 Bug Tracking Tools

A bug tracking tool is a software application designed to improve software's quality assurance, aiding programmers in tracking reported software bugs (i.e. errors, flaws, mistakes, failures, or faults in a computer program preventing it from behaving as intended). The analyzed bug tracking tools were: BUGtrack[1], Bugzero[2], Bugzilla[3], JIRA[4], and MantisBT[5].

[1] http://www.bugtrack.net
[2] http://www.websina.com/bugzero
[3] http://www.bugzilla.org
[4] http://www.atlassian.com/software/jira
[5] http://www.mantisbt.org

Figure 4.2 exhibits the table which summarizes the research of the analyzed bug tracking tools.

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Char	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
BUGtrack					•	•																	•				
Bugzero					•																		•				
Bugzilla		•			•																		•				
JIRA		•		•	•																		•			•	
Mantis					•	•					•	•											•				•

Figure 4.2: Bug Tracking tools summary.

4.2 Construction Tools

Software construction refers to the "*detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging*" [Abran et al., 2004].

The analyzed construction tools were: Aptana Studio[6], Eclipse[7], IntelliJ IDEA[8], JCreator[9], KDevelop[10], Komodo[11], Microsoft Visual Studio[12], NetBeans[13], and Zend Studio[14].

Figure 4.3 exhibits the table which summarizes the research of the analyzed construction tools.

[6] http://www.aptana.com
[7] http://www.eclipse.org
[8] http://www.jetbrains.com/idea
[9] http://www.jcreator.com
[10] http://www.kdevelop.org
[11] http://www.activestate.com/komodo
[12] http://www.microsoft.com/visualstudio
[13] http://www.netbeans.org
[14] http://www.zend.com/en/products/studio

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Chat	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
Aptana Studio																							•				
Eclipse				•	•	•	•		•		•			•						•			•	•			
IntelliJ IDEA		•		•	•						•												•			•	
JCreator																							•				
KDevelop																							•				
Komodo								•						•									•				
Microsoft Visual Studio 2008		•		•	•	•	•																•				
NetBeans		•			•				•	•	•			•									•	•	•		•
Zend Studio																							•				

Figure 4.3: Construction tools summary.

4.3 Design Tools

Software design is both "*the process of defining the architecture, components, interfaces, and other characteristics of a system or component*" and "*the result of [that] process*" [IEEE, 1990].

This dissertation focused on the tools used for constructing visual models – such as UML or BPMN diagrams – during the project's life-cycle, and therefore in the definition of the parts of a system or component.

The analyzed design tools were: Altova UModel[15], ArgoUML[16], Borland Together[17], BOUML[18], EclipseUML[19], Enterprise Architect[20], Gliffy[21], Magic Draw[22], Microsoft Visio[23], Rational Rose Data Modeler[24], StarUML[25],Telelogic Rhapsody[26], and Visual Paradigm[27].

[15] http://www.altova.com/products/umodel/uml_tool.html
[16] http://argouml.tigris.org
[17] http://www.borland.com/us/products/together/index.html
[18] http://bouml.free.fr
[19] http://www.uml2.org/index.html
[20] http://www.sparxsystems.com.au/products/ea/index.html
[21] http://www.gliffy.com
[22] http://www.magicdraw.com
[23] http://office.microsoft.com/en-us/visio/default.aspx
[24] http://www-01.ibm.com/software/awdtools/developer/datamodeler
[25] http://staruml.sourceforge.net
[26] http://www.telelogic.com/products/rhapsody/index.cfm
[27] http://www.visual-paradigm.com

Figure 4.4 exhibits the table which summarizes the research of the analyzed design tools.

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Chat	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
Altova UModel								•															•				
ArgoUML																						•					
Borland Together																											
BOUML							•																•				
EclipseUML																							•				
Enterprise Architect								•						•									•				
Gliffy					•		•	•												•			•				
Magic Draw								•															•				
Microsoft Visio		•						•		•											•						
Rational Rose Data Modeler																							•				
StarUML																											
Telelogic Rhapsody							•																				
Visual Paradigm														•									•				

Figure 4.4: Design tools summary.

4.4 Engineering Management Tools

Software engineering management is the "*the application of management activities - planning, coordinating, measuring, monitoring, controlling, and reporting - to ensure that the development and maintenance of software is systematic, disciplined, and quantified*" [Abran et al., 2004].

This dissertation focused on the project management activities embed in this area. There were studied the following project management tools: Basecamp[28], Gantt Project[29], Intellisys Project[30], Microsoft Project[31], Project KickStart[32], Rally Enterprise (only the Project Management module)[33], Trac[34], and Wrike[35].

[28] http://www.basecamphq.com
[29] http://www.ganttproject.biz
[30] http://project-management-software-review.toptenreviews.com/intellisys-project-review.html
[31] http://office.microsoft.com/en-us/project/FX100487771033.aspx
[32] http://www.projectkickstart.com
[33] http://www.rallydev.com/agile_products/lifecycle_management/project_management
[34] http://trac.edgewall.org
[35] http://www.wrike.com

Figure 4.5 exhibits the table which summarizes the research of the analyzed project management tools.

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Char	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
Basecamp	•	•		•	•	•	•				•											•	•				
Gantt Project																											
Intellisys Project					•				•	•											•						
Microsoft Project		•			•	•				•																	
Project KickStart		•			•																						
Rally Enterprise				•	•	•					•												•				•
Trac		•				•															•		•				
Wrike		•		•	•	•	•															•					

Figure 4.5: Engineering Management tools summary.

4.5 Requirements Tools

Software requirements "*express the needs and constraints placed on a software product that contribute to the solution of some real-world problem*" [Kotonya and Sommerville, 1998].

There were studied the following requirements management tools: Borland Caliber Analyst[36], Contour[37], Rational RequisitePro[38], RavenFlow[39], and Telelogic DOORS[40].

For a more comprehensive study, consult the INCOSE Requirements Management Tools Survey[41]. This survey has been studying requirements management tools since 1990's, counting on 20 May 2009 forty-one studied tools.

Figure 4.6 exhibits the table which summarizes the research of the analyzed requirements tools.

[36] http://www.borland.com/us/products/caliber
[37] http://www.jamasoftware.com/contour
[38] http://www-01.ibm.com/software/awdtools/reqpro
[39] http://www.ravenflow.com/products
[40] http://www.telelogic.com/products/doors/doors
[41] http://www.paper-review.com/tools/rms/read.php

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Chat	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
Borland Caliber Analyst									•														•				
Contour		•		•	•															•							
Rational RequisitePro					•				•																		
RavenFlow		•						•															•			•	
Telelogic DOORS		•			•				•					•												•	

Figure 4.6: Requirements tools summary.

4.6 Collaboration Tools

The analyzed collaboration tools were: CollabNet TeamForge[42], EGroupWare[43], IBM Lotus[44], Jazz[45], Lighthouse[46], Mylyn[47], and Socialtext[48].

Figure 4.7 exhibits the table which summarizes the research of the analyzed collaboration tools.

[42] http://www.open.collab.net/products/sfee
[43] http://www.stylite.de/EGroupware
[44] http://www-01.ibm.com/software/lotus
[45] http://www-01.ibm.com/software/rational/jazz
[46] http://www.lighthouseapp.com
[47] http://www.eclipse.org/mylyn
[48] http://www.socialtext.com

	Blogs	Comments	Contact Management	Dashboard	E-mail	Feeds	File sharing	Folder sharing	Forums	Group Calendar	Instant Messaging/Chat	Microblogs	Online whiteboard	Real-time editor	Recommender	Screen sharing	Social bookmarking	Social cataloguing	Social networking	Tags	Team timeline	To-Do list	Version Control	Version Control	Video conferencing	VoIP	Voting
CollabNet TeamForge		•		•	•				•														•				•
EGroupWare			•		•			•		•													•			•	•
IBM Lotus			•		•	•	•	•		•	•																
Jazz	•	•		•					•		•									•			•		•		
Lighthouse				•	•	•				•										•			•				
Mylyn		•			•					•													•				
Socialtext	•	•	•	•	•	•						•		•					•	•							•

Figure 4.7: Collaboration tools summary.

4.7 Open issues

The analysis of the summary tables revealed open issues.

The e-mail service provided in the studied software tools depended on external e-mail providers and was used primarily as notification system. The consequence of this situation is that software engineers must switch between tools to execute e-mail related actions – which could be avoided if the tool itself had an internal e-mail service, such as EGroupWare collaboration tool – and also are limited to using e-mail to be notified of changes made in the tool, an event resolved more effectively by Web feeds.

There was a lack of group calendars in the Engineering Management Tools, with only 2 out of 8 presenting a team calendar. Project Management requires the constant monitoring of the progress of project and therefore software engineers need to schedule meetings – without a group calendar engineers must communicate with their colleagues so that they can be aware of their schedules.

Microblogs appeared only in MantisBT bug tracking tool and Socialtext collaboration tool, which denotes a clear gap between the growth of micro-blogging on the Web and its adoption in Software Engineering. The lack of studies concerning the use of micro-blogging in workplaces and the fact that micro-blogging appeared very recently might serve as a justification for this absence. In the future, research should be directed to determining how well microblogs enhance the conversation and collaboration between individuals and for what purposes microblogs are used. Only then proof can be made of microblogs being more than just a trend and knowledge can be gained on what contexts this technology should be used in detriment of similar ones, such as IM/Chats.

Online whiteboard and screen sharing were not found in any of the studied tools. The first feature has the potential to substitute a physical whiteboard, an object commonly used in meetings, and thus its usefulness should be researched among **Design Tools**, where it could serve as a draft to a diagram, **Requirements Tools**, where it could be used to clarify an idea (e.g. while in a brainstorming session), or in **Collaboration Tools**, where it could be filled simultaneously or not by a group of Software Engineering. The second feature would boost awareness.

Social bookmarking and social cataloguing were also not found in any tool. The inexistence of these features, within a tool or integrated with external services, is an impediment for organizing the knowledge collected outside the tool in the web. Moreover, the lack of these features – more severe while in the documentation phase – implies that the user must seek communication tools as a means to exchange information about web pages or other content they visited.

Other notable registered absences were: recommender feature, which could be implemented based on a tagging [Hung et al., 2008] or a rating system; videoconferencing and VoIP features; and voting.

Surprisingly the number of Software Engineering tools that include some of the most popular collaboration software available on the Web is very small. Out of 47 tools, only 3 provided a Blog, 2 a Microblog, 1 Social networking capabilities, 6 a Wiki, and 6 a user-based tagging system. The collaborative software more frequently supported were E-mail and Version Control Systems; the first is used primarily as notification system and as the result of integration with external e-mail applications, while the second was found mainly in the **Bug Tracking**, **Construction**, **Design**, and **Collaboration** areas.

4.8 Summary and Discussion

The Software Engineering tools that were analyzed spanned the **Bug Tracking**, **Construction**, **Design**, **Engineering Management**, and **Requirements** areas. A special type of tool area – which aids Software Engineering projects – was considered, called **"Collaboration Tools"**, and corresponded to the tools specifically developed for collaborative purposes, spanning the previously mentioned areas and integrating General Purpose Sofware and Social Software. Each tool was analyzed according to three criteria: *Collaboration*, concerning the research of *awareness*, *communication* and *collective knowledge* features; *Integration*, concerning the interchangeability of data between applications and the integration of applications as plugins on other applications; and *Other characteristics*, concerning features not collected in the other criterions.

Almost every design tool studied in this dissertation was desktop-based, meaning that UML diagram editing is limited to physical constraints of the PC, while the majority of the collaboration tools were web-based. Requirements and bug-tracking tools are

already conquering the Web, while code editing is for now restricted to desktop – the global acceptance of tools like Visual Studio and Eclipse as desktop IDEs reflect this phenomenon.

Chapter 5

Problem statement

The problem this dissertation aims to solve is to fill two gaps of the current state of art of web collaboration tools for Software Engineering:

- Integrate in a single web-based environment a rationally chosen set of features, capable of supporting collaborative activities between software engineers and covering a subset of areas of Software Engineering.
- Study on the impact of using several awareness and communication tools integrated in a single web-based environment, by designing and conducting an experiment.

This chapter presents the justification for the problem's uniqueness and a discussion of why the problem is worthwhile to be solved.

5.1 Justification for the problem's uniqueness

The uniqueness of the problem addressed in this dissertation is justified by the fact that none of the collaboration tools studied in section 4.6 provided support for the entire life cycle of a project (see section B.6 to consult which areas are spanned by each collaboration tool). Furthermore, the majority of the collaboration tools spanned areas of Software Engineering by integration with other tools, and not by providing themselves support for Software Engineering activities.

Another factor contributing to the problem's uniqueness is the lack of studies validating, through an experiment, the benefits introduced by collaboration features. The investigation on the state of the art revealed the existence of studies measuring the effectiveness of collaboration features in a workplace, but none had those features integrated together in a web-based environment nor were they monitored through the life cycle of a Software Engineering project.

5.2 Discussion of the problem's worthiness

The proposed web-based tool would be useful for minimizing the time and effort spent by software engineers, every day, on using separate software products for having access to collaboration features: opening the e-mail inbox, adding a bookmark to their social bookmarking service, writing a document in a wiki and notifying others, chatting about a project's artifact and modifying it throughout the conversation, writing a micropost on the progress of a task, and many more. Google Wave[1] will be a new model for communication and collaboration on the web, including features such as real-time collaboration and natural language support, which will try to achieve that same goal. It is expected to be launched later this year.

When compared to desktop-based tools, the very nature of the proposed tool would be an advantage, since it would be accessible anytime, anywhere, a common requisite in Software Engineering projects.

The study on the impact of using collaboration features integrated in a single web-based environment would boost research in the areas of CSCW, Groupware, Social Software and Software Engineering by increasing confidence in positive results. These results would make it easier to convince teams to adopt technologies emerging on the Web and also to reuse old ones.

5.3 Summary and Discussion

This chapter presented the statement of the problem addressed by this dissertation, a justification for the problem's unique characteristics, and a discussion of the value inherent to the problem's resolution.

The problem is to integrate a set of collaboration features in a single web-based environment and then to study, by designing and conducting an experiment, the impact of introducing those features in a Software Engineering team.

The uniqueness of the problem is justified by the lack, at the present time, of collaboration tools providing support for the entire life cycle of a project and of studies validating the benefits introduced by collaboration features in a web-based environment. Moreover, the collaboration tools spanned the areas of Software Engineering by integration with other tools.

Finally, the proposed web-based tool would be useful for: minimizing the time and effort spent by software engineers on using separate software products for having access to collaboration features; being accessible anytime, anywhere; and for convincing software engineers, through the experiment's results, to adopt technologies emerging on the Web and also to reuse old ones.

[1] http://wave.google.com/

Chapter 6

Prototype

Following the first gap identified in chapter 5, this chapter presents the defined steps for developing a prototype capable of integrating, into a single Web-based environment, some of the features referred in chapter 4.

The features were integrated as plugins into Redmine, a project management web application.

6.1 Redmine

The first step was to research the features of Redmine, a project management web application launched in June 2006 and written using Ruby on Rails open-source web framework.

Redmine's features include:

Multiple projects support:

- Users can have different roles and set of permissions on each project.
- Each project can be declared as public or private, i.e, visible by project members only.
- Modules (e.g. Wiki, Repository, etc.) can be enabled/disabled on each project.

Issue tracking system:

- Users can create an issue – i.e. a "task" which can be a *Bug*, *Feature* or *Support* – and associate statuses to the issue, which can be *New*, *Assigned*, *Resolve*, *Feedback*, *Closed* or *Rejected*.

- Each issue has a priority (ranging from *Low* to *Immediate*), a person responsible for it, start and due dates, files associated, estimated and spent time, a group of watchers – i.e., users that follow the activity of the issue – and a percentage of completed work.
- Issues can be created via Web or e-mail.

Gantt chart and calendar

News, documents & files management

Feeds & email notifications:

- Project activity, change sets, news, issues and issue changes are available as Atom feeds.

Per project wiki and forum:

- Wiki's features include textile syntax and diff/annotate views.

Time tracking functionality:

- Time can be entered at project or ticket levels.
- Generation of reports to view time per user, issue type, category or activity.

SCM integration (SVN, CVS, Git, Mercurial, Bazaar and Darcs):

- Users can browse the contents of the repository and view/search change sets.

Multiple LDAP authentication support

Multilanguage support

Multiple databases support

- Redmine runs with MySQL, PostgreSQL or SQLite.

As shown in Figure 6.1, Redmine is composed of three menus and a sidebar:

1. **Top left menu**: gives quick access to the home page (which includes the latest news and projects), the user's personal page (a type of dashboard, with drag-and-drop components, such as issues assigned to the user, issues watched by the user, documents, etc.), the list of projects on which the user is a member, and a help page.

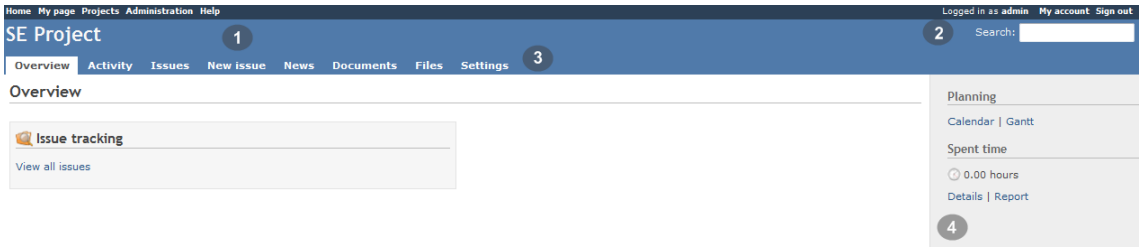

Figure 6.1: Redmine's screenshot.

2. **Top right menu**: gives to the current user access to his account page (with personal information, e-mails notifications and preferences) and to the sign in/sign out links.

3. **Main menu**: located bellow the top menus, it shows the modules – i.e., the core features of Redmine – visible by the user, either at the project or application levels (visible independently of the projects).

4. **Right sidebar**: displays features related to the selected module.

6.2 Features analysis

After collecting and understanding the features provided in Redmine, the next stage was to analyze the cost and benefits of implementing each of the features gathered in the state of the art phase (see chapter 4). The results of this analysis, which included an ordered list of the features to implement, can be read in Appendix C.

The analysis of each feature was defined in accordance with four criteria:

1. **Time:** Estimate of the number of days required to implement the feature.
2. **Value:** The benefit of implementing the feature.
3. **Innovation:** The innovation value of implementing the feature.
4. **Preference:** The preference of the author in implementing the feature.

Criteria 2, 3 and 4 were evaluated in a scale with three values: *Low*, *Medium*, and *High*.

Criterion 2 was defined taking into consideration the investigation's results of the advantages and disadvantages of General Purpose Software (section 3.1) and Social Software (section 3.2).

Criterion 3 was defined considering the number of software applications that supported the feature – the greater the number, less was the innovation of implementing the feature (see chapter 4 to consult which tools supported each feature).

Certain features were not evaluated in terms of the criteria 1 and 4, either because were already present in Redmine and consequently there was no interest in using them as

plugins (“Comments”, “Forums”, “Version Control”, and “Wiki ”) or they were too time consuming to implement within the available time (“Video conferencing”, “VoIP”, “Online whiteboard”, and “Screen sharing”).

6.3 Requirements analysis

After ordering the list of features to implement, there were defined the following requirements for each plugin:

Blog

1. The system must allow a post to be created with rich media content, accessible via a textile syntax GUI.
2. Each post must have an author, title, date and time of submission, date and time of the last update, body and a set of associated tags.
3. All posts must be listed and ordered from the newest to the oldest, with each page displaying a maximum of 5 posts.
4. Editing or removing a post must be restricted only to the post’s author.
5. When the user is creating a new post or editing an existent one, the system must inform him of the characters left to use – out of a maximum of 120 – in the post’s title.
6. When reading a post, the user must be able to add comments via a textile syntax GUI, view the total number of comments added so far, and navigate to the (existing) previous and next posts.
7. Removing a comment must be restricted only to the comment’s author.
8. When the user communicates his intent of removing a post or comment, the system must inquire the user for him to confirm his intention.

Contact Management

1. The system must permit the user to add and edit his contact information, which includes the organization the user works on (name, position, department, office, street address, postal code, locality, state, country, and city), supervisors’ names, phone numbers (business, home, and mobile), instant messaging usernames (Windows Live Messenger, Skype, Yahoo, AIM, GTalk, and Meebo), and finally

links to his web pages (blog, personal homepage, and Linkedin/Facebook/Twitter profiles).

Dashboard

1. The system should have a dashboard with the following widgets: feeds of chats, tasks, and artifacts; the user's milestones, tasks, iterations, and chats; and the user's top artifacts.
2. The dashboard's widgets should be dragged and dropped, minimized and maximized (i.e. clicking in an icon should hide or show the widget's content), and reset to the default position.

E-mail

1. When creating an e-mail, the user should be able to send the e-mail to multiple users and compose the message's body with a textile syntax GUI.
2. The system must allow the user to use an internal e-mail to receive e-mails, send e-mails, and view the sent e-mail.
3. The system must show how many new messages the user received and distinguish those from the rest.
4. The system should allow the user to delete, mark as unread/read, forward, and reply to an e-mail.

Feeds

1. The system must use Atom feeds – already used in other modules of Redmine – in the **Social bookmarking**, **Social cataloging**, **Blog** and **Microblog** plugins.
2. The system must allow users to subscribe/unsubscribe chats, tasks, and artifacts. These feeds should be used for internal use only and are essentially "bookmarks" for rapidly accessing the content subscribed.
3. The most recent feeds, as well as all the feeds, should be accessed via the sidebar.

IM/Chat

1. The system must allow the user to create a chat room, invite users and associate elements (iterations, artifacts, chats, milestones, and tasks).
2. The system should allow the user to view the list of user chats, active chats and closed chats.

3. The system must allow the user to change the name of the chat room, send text messages to other users, view the messages of all users inside the chat room, view the status – defined in the sidebar – of the users invited to the chat through the use of colored icons (green for online, red for offline, and green with a stop for busy), add notes (unique for each chat), and view the annotations of other users.

Microblog

1. The system should allow a micropost to be created with text content.
2. Each micropost must have an author, date and time of submission, date and time of the last update, and message.
3. All microposts must be listed and ordered from the newest to the oldest, with each page displaying a maximum of 10 microposts.
4. When the user is creating a new micropost or editing an existent one, the system must inform him of the characters left to use – out of a maximum of 140 – in the micropost's message.
5. Editing or removing a micropost must be restricted only to the micropost's author.
6. When the user communicates his intent of removing a micropost, the system must inquire the user for him to confirm his intention.

Polls

1. The system must allow the user to create a poll with an author, dates of creation and closure, title, configurable number of options, and question.
2. All polls must be listed and ordered from the newest to the oldest.
3. The system should show, via an icon, if a poll is currently open or closed.
4. The system must show when the user voted, the number of votes per option and the total number of votes.
5. The number of votes the user can make must be restricted to one.
6. The system must allow the user to add comments to polls.
7. When the user communicates his intent of removing a poll, the system must inquire the user for him to confirm his intention.

Ratings

1. The user must be allowed to vote, more than once, on bookmarks, catalog entries and artifacts with stars (1, 2, 3, 4 or 5) via an Ajax star rating system.
2. The system must show the numbers of votes for an artifact.
3. The system should make rankings with the top rated artifacts.

Recommendations

1. The system should make recommendations of bookmarks and catalog entries (books, conference articles, documentation reports, miscellaneous, and thesis) the user doesn't possess based on a predicted rating, evaluated by finding one or more bookmarks or catalog entries the user has in common with other users.
2. A bookmark must be considered common to another bookmark if both possess the same URL.
3. A catalog entry must be considered common to another catalog entry if both possess the same title, author's first name, and author's last name.
4. The system should show how many recommended items of each type the user has.
5. For each recommended item, the system must show, based on the item's ratings gave by other users, the mean rating and the number of votes.
6. After the user accepts the recommended item, the system should redirect to the page that adds the item, filling the obligatory fields and using the predicted rating as the initial rating.
7. The recommendations should be made either by the system or the user; in the latter scenario, the user should be able to write a note enunciating the reason for making the recommendation.

Search

1. The system must allow users to search the **Blog**, **Social bookmarking**, **Social cataloging**, **Microblog**, and **Software Engineering** plugins.
2. In the case of searching the blog's posts, the user should be able to search according to one of the criterions: body, title, or by the name of the user who added the post.
3. In the case of searching the bookmarks, the user should be able to search according to one of the criterions: description, title, URL address, or by the name of the user who added the bookmark.

4. In the case of searching the catalog's entries, the user should be able to search according to one of the criterions: title, at least one of the authors (each with first and last names), publisher, publication year, ISBN and ISSN codes, volume, issue, chapter, initial and final pages, bibtex, the personal description/opinion of the user who added the entry, or by the name of the user who added the entry.

5. In the case of searching the microblog's microposts, the user should be able to search according to one of the criterions: message or by the name of the user who added the micropost.

6. In the case of searching the Software Engineering plugin, the user should be able to search: artifact types by name, permission groups by name, and artifacts/iterations/tasks by name or description.

7. The system should allow users to go back (without having to press the browser's back button) to redefine which type of content they wish to search or with which criterion they wish to search by.

8. The system should show all the database's records if the user doesn't write anything.

9. The system must search the database's records independently of the typed keywords' order or if all or only part of the keywords were matched. For example, if the user searches the blog's posts added by users named "John" and there are two records (i.e. posts), one added by the user named "John Mills" and another by the user named "Jack John", the system must return them.

10. The page with the search's results must show a message with the format "<N> results for <Keywords>", with *N* being the number of records found in the database that matched the search and *Keywords* the sequence of one or more words/numbers used in the search. In addition, the system should have a hyperlink to the searched plugin.

Social bookmarking

1. The system must allow the user to create a bookmark with a title, a URL (which must begin with http://, https://, ftp:// or ftps://), and a personal description.

2. Each post must also contain an author, date and time of submission, date and time of the last update, rating, and a set of associated tags.

3. All bookmarks must be listed and ordered from the newest to the oldest, with each page displaying a maximum of 10 bookmarks.

4. When the user communicates his intent of removing a bookmark, the system must inquire the user for him to confirm his intention.
5. Editing or removing a bookmark must be restricted only to the bookmark's author.
6. The system should show the rating the author gave to the bookmark.
7. When the user is creating a new bookmark or editing an existent one, the system must inform him of the characters left to use – out of a maximum of 120 and 1000 – in the bookmark's title and description, respectively.

Social cataloging

1. The system should allow the user to create a new catalog entry with one of the following types: *book*, *conference*, *documentation*, *miscellaneous* (for an entry that doesn't fit in any of the other types), or *thesis*.
2. The system must allow the user to create, for the chosen type, a catalog entry with the item's title, at least one of the authors (each with first and last names), publisher, publication year, International Standard Book Number (ISBN) and International Standard Serial Number (ISSN) codes, volume, issue, chapter, initial and final pages, bibtex entry, and finally the personal description/opinion.
3. Each catalog entry must also contain the author (i.e., the user who submitted it), date and time of submission, date and time of the last update, rating and a set of associated tags.
4. All catalog entries must be listed and ordered from the newest to the oldest, with each page displaying a maximum of 10 entries.
5. When the user communicates his intent of removing a catalog entry, the system must inquire the user for him to confirm his intention.
6. Editing or removing a catalog entry must be restricted only to the entry's author.
7. The system should show the rating the author gave to the catalog entry.
8. When the user is creating a new catalog entry or editing an existent one, the system must inform him of the characters left to use – out of a maximum 1000 – in the catalog entry's description.

Software Engineering

1. The system must allow the user to add artifact types (e.g. "Requirements Engineering report", "Architecture report" ...) and to remove them, after confirming his intention.

2. The system must permit the user to edit the name of an artifact type/artifact.
3. The user must be able to add artifacts, providing a name, description, type (i.e. one of the artifact types, such as "Requirements Engineering report"), a file that represents the artifact (e.g. a PDF document), user's notes (e.g. "This is the first version of the document"), and the tasks related to the artifact (e.g. "Write requirements report").
4. After having added the artifact, the user should have the possibility of submitting new versions of the same artifact, composed with a new file and notes to describe the changes made in relation to the previous version.
5. The system should permit the creation of permission groups, i.e., logical groups which specify the actions the members of the groups can execute in an artifact (view, edit, and remove) and in an artifact type (create, view, edit, and remove).
6. After a new permission group has been created, the system must implicitly add the user to the group's members and allow him to add other users to the group.
7. The system must permit the user to delete the members, artifact types, and artifacts associated with the group.
8. The user must be able – while within a specific artifact – to add comments to an artifact, view the related artifacts/tasks, view the artifact's historical (i.e. its different versions), view the related groups, and restrict the permissions of users related with the artifact.
9. The system must permit the user to create iterations – with a name, description, and associated manager – and associate tasks, users, and artifacts to them.
10. The system must allow the user to add a new task to an iteration, with a name, description, start and due dates, initial progress, and associated users/tasks/artifacts.
11. An iteration must have three states: a *pending* state, if all the tasks of the iteration have a progress of 0%; a *current* state, if at least there is one task with a progress greater than 0%; and a *finished* state, if all the tasks have a progress of 100%. The system should filter the shown iterations by the combination of possible states (e.g. showing only the finished iterations, showing the current and pending iterations, etc).
12. A task must have three states: a *pending* state, if the task has a progress of 0%; a *current* state, if the task has a progress greater than 0%; and a finished state, if the task has a progress of 100%. The system should filter the tasks by the combination of possible states and allow the user to easily increment and decrement the progress of a task.

Social networking

1. The system must allow the user to add another user to his network of friends (after searching by fist and/or last names) and consequently be notified, in the application, when the new friend added or edited a post, micropost, bookmark, catalog entry, task or artifact.
2. The system must list the most recent items that were created or edited, the user responsible for the creating or editing them, the event's date and finally on which project the event took place.
3. The system must allow the user to remove a friend, after the user confirmed his intention.
4. The system must show the list and number of users following the user as well as those being followed by him (i.e., his friends).
5. In the case the user searches for himself or for a user that is already his friend, the system must warn him and hide the link for adding that user.

Tags

1. The system must allow the user to add tags – separated by commas – to artifacts, tasks, iterations, blog's posts, catalog's entries, and bookmarks.
2. The system should allow the user to view the posts/bookmarks/catalog entries tagged with the same tag of a specific post/bookmark/catalog entry, either through clicking on the tag displayed in a tagcloud or in a list.
3. The system must show a tag cloud, i.e., a visual depiction of the tags.

Common to all the modules is English and Portuguese language support.

With the exceptions of **E-mail**, **Social network** and **Recommendation** modules, all the other modules interact with information related to individual projects.

6.4 Architectural design

Architectural decisions

The architecture of the prototype was defined considering two patterns: **3-tier** and **Model-View-Controller (MVC)**.

The first pattern (see Figure 6.2) is a client-server architecture which defines a system in three separate layers: *Graphical User Interface*, the topmost layer, displays information to the user; *Business Logic*, the middle layer, controls the systems functionalities, serving

as the bridge between the two other layers; and *Data*, which stores and retrieves the system's data by executing Create, Retrieve, Update and Delete (CRUD) operations. The advantages of using this pattern include: independence between layers and avoidance of mixing functionalities into different logical layers; maximization of the system's modularity; and encapsulation of the implementation details of one layer from the other layers. In a web application, the *Graphical User Interface* is the web browser, the *Business Logic* is the application server and the *Data* is the database server.

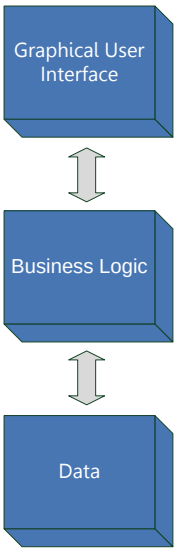

Figure 6.2: 3-tier architecture.

The second pattern is the basis for any Rails application. Originally described in 1978 by Trygve Reenskaug, MVC (see Figure 6.3) was invented to bridge the gap between the human user's mental model and the computer's digital model, giving the user the illusion of seeing and manipulating the domain information directly. Back then, MVC was conceived as a general solution to the problem of users controlling large and complex data sets.

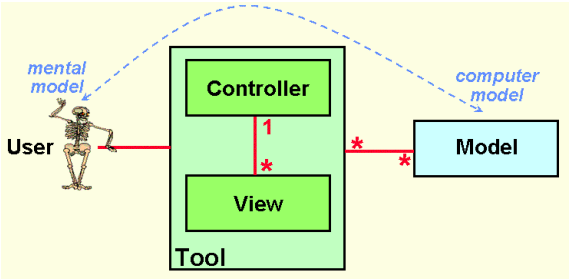

Figure 6.3: MVC architecture model (extracted from [Reenskaug, 1978]).

In Reenskaug's design, an application was broken into three types of components:

model, *view*, and *controller*. The model maintains the state of the application, by enforcing that all the business rules apply to the application's data. The view generates the GUI, frequently based on the data contained in the model. The controller receives events from the outside world (normally user input), interacts with the model, and displays the appropriate view to the user.

In a Rails application, incoming requests are first sent to a router, which parses the request. Then the request is forward to a specific place of the application (the controller) and within these to a particular method (an action). This method will prepare the view for rendering information to the user.

Figure 6.4 shows an example of MVC's usage in a Rails application, where the user clicked the button – linking to http://my.url/store/add_to_cart/123 – that adds a product to a shopping cart. The routing component picks the first part of the path, *store*, as the name of the controller, the second part, *add_to_cart*, as the name of the action to invoke in the controller, and lastly the last part of the path, *123*, as the id of the product to be added.

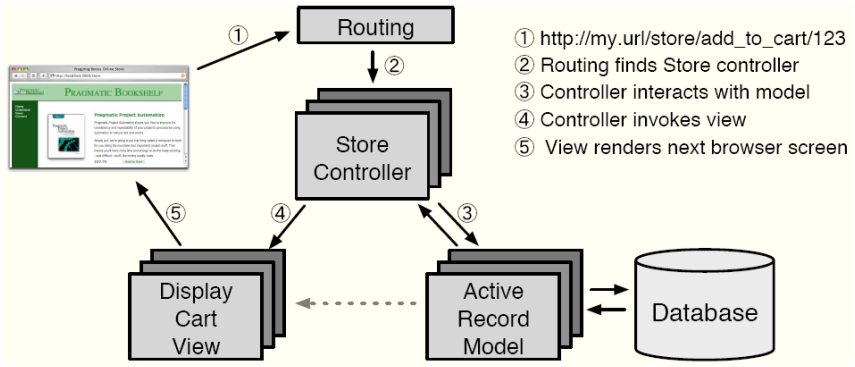

Figure 6.4: MVC architecture in Ruby on Rails (extracted from [Thomas et al., 2006]).

Logical architecture

In accordance with the design patterns previously mentioned, the prototype's logical architecture was defined, as shown in Figure 6.5. The logical architecture divided the system into four layers: *Graphical User Interface*, *Business Logic*, *Data access* and *Data*. The first three layers were the same as the ones defined in a 3-tier architecture, while the fourth referred to the database management system and the files repository. Inside the Graphical User Interface layer, the system functionalities were divided and grouped into functional areas that traversed the other layers; the description of each group can be seen in section 6.3.

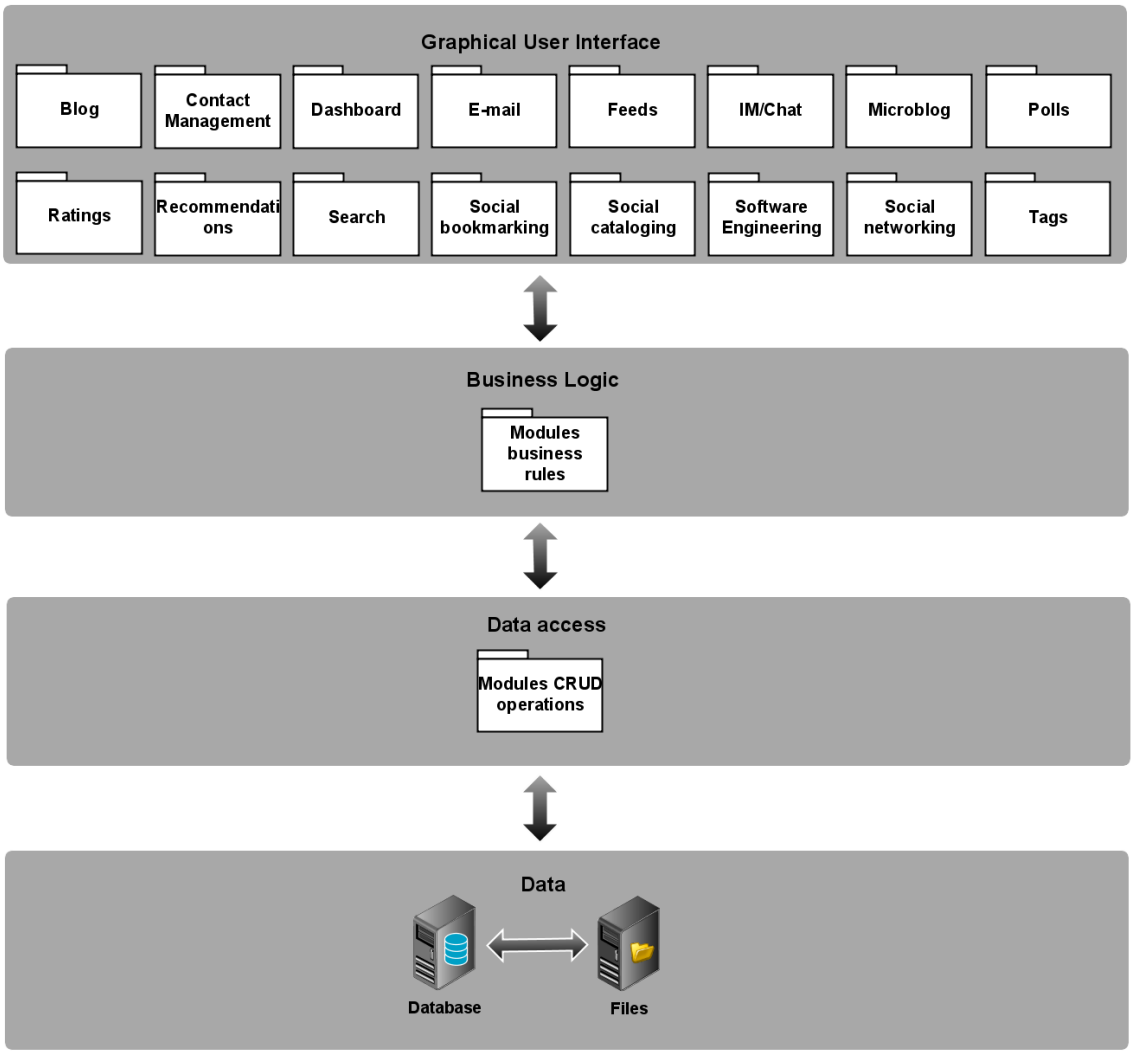

Figure 6.5: Prototype's logical architecture.

Physical architecture

There was also defined the prototype's physical architecture, shown in Figure 6.6, which captures the hardware topology needed to execute the software components defined in the logical architecture.

The client – a machine with an Internet browser and with the associated technologies JavaScript, XHTML, Ajax and CSS – connects to a Mongrel web server running the prototype. Then, through Ruby on Rails Active Record Object-relation mapping (ORM) library – which connects business objects and database tables – the Web server connects to the Database server. Finally, the Database server, running a MySQL relational database management system (RDBMS), accesses a File server, with one of the repositories supported by Redmine.

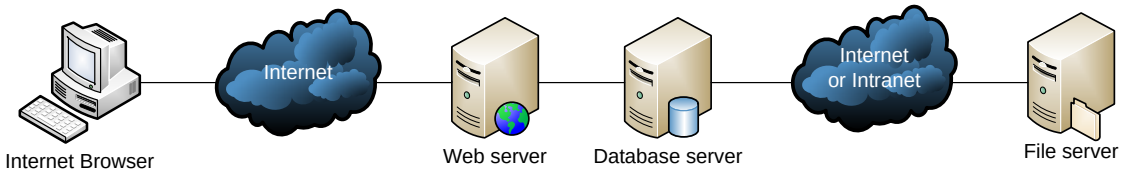

Figure 6.6: Prototype's physical architecture.

6.5 Technological decisions

Except the Ruby on Rails framework, no other technology was defined in the dissertation proposal as obligatory to use in the prototype development.

The technological decisions were made according to the author's previous acquired experience, the requirements and architecture design, and the study of the currently available technologies.

Base technologies

Launched in July 2004, Ruby on Rails is an open-source web framework for making the development of web applications quicker to perform and easier to maintain. Rails relies mainly in two principles: *Convention over Configuration* and *Don't Repeat Yourself*. The first is a design paradigm for reducing the number of configuration files needed to maintain – such as the XML configurations files of a Java web application – through the use of naming convention (e.g. the table of a model has the same name as the model, is pluralized and in lowercases; and primary keys are called "id"). The second is a principle for guarantying that every piece of knowledge of the system – including code, database schemas, test plans, the build system, and even documentation – has a single representation, thus augmenting the unambiguity of the system.

Other important features of Rails are: the organization of a system, at the architecture level, according to the MVC paradigm; *scaffolding*, a command for generating views and controllers from a specific model; and its suitability to agile software development.

The reason why Ruby on Rails was chosen as the prototype's programming language relates to the fact that Redmine was written in Ruby on Rails.

Being Ruby on Rails a packaged Ruby application, the prototype's programming language was Ruby, a dynamic open-source object-oriented programming language. Ruby is a reasonably well-known programming language and it supports multiple programming paradigms, including functional, object-oriented, imperative and reflective.

Browser technologies

In terms of web technologies there were used: JavaScript, for validating forms on the client-side and for Ajax; XHTML for writing web pages; Ajax for avoiding the reload of the entire page when only a small part was changed, hence reducing the time spent on refreshing the page; and CSS, for defining the presentation of XHTML web pages.

Business logic and Database technologies

The RDBMS chosen was the popular open source MySQL. The reasons behind the choice were the cross-platform support, author's familiarity, open-source license, performance, and ease of use.

Development technologies

There were considered two IDEs to develop the prototype: Eclipse with a Rails plugin and NetBeans. Netbeans was chosen due to the author's familiarity, great code completion and highlighting, good inline documentation and a flexible and usable search feature.

Version Control technologies

Subversion was the chosen version control system for development, due to being supported by Redmine and also due to its worldwide adoption and familiarity to the author.

Some big advantages of Subversion when compared to its predecessor, CVS, are: support for all file types, versioning of directories, file renaming, and atomic commits [Fishg, 2008] [Software, 2005].

Web server technologies

Although Ruby on Rails comes with a built-in web server, WEBrick, the chosen web server was Mongrel, due to being a more fast alternative [Overflow, 2009] [Blog, 2007].

6.6 Implemented prototype

The prototype successfully developed and/or integrated all the requirements described in section 6.3 with the exception of replying to an e-mail and the user making recommendations.

The **Dashboard**, **E-mail**, **Feeds**, **IM/Chat**, **Polls**, **Software Engineering**, and **Tags** plugins – previously developed by undergraduate students from the Master in Informatics and Computing Engineering (MIEIC) course at Faculdade de Engenharia da Universidade

do Porto (FEUP), while in the discipline of Software Engineering Laboratory – were integrated in the prototype. E-mail and Polls were modified at a visual level, while Tags was expanded to the Blog, Microblog, Social bookmarking, and Social cataloging plugins.

With the purpose of avoid zooming out to view all the plugins of the main menu, some plugins were agglomerated into one module: **Blogs** includes Blog and Microblog plugins; **References** includes Social bookmarking and Social cataloging plugins; and **SE** includes Software Engineering plugin. If the user clicked on one of these modules, the hyperlinks for each plugin would be shown.

Figure 6.7 shows a screenshot of the developed prototype with all the developed and/or integrated plugins.

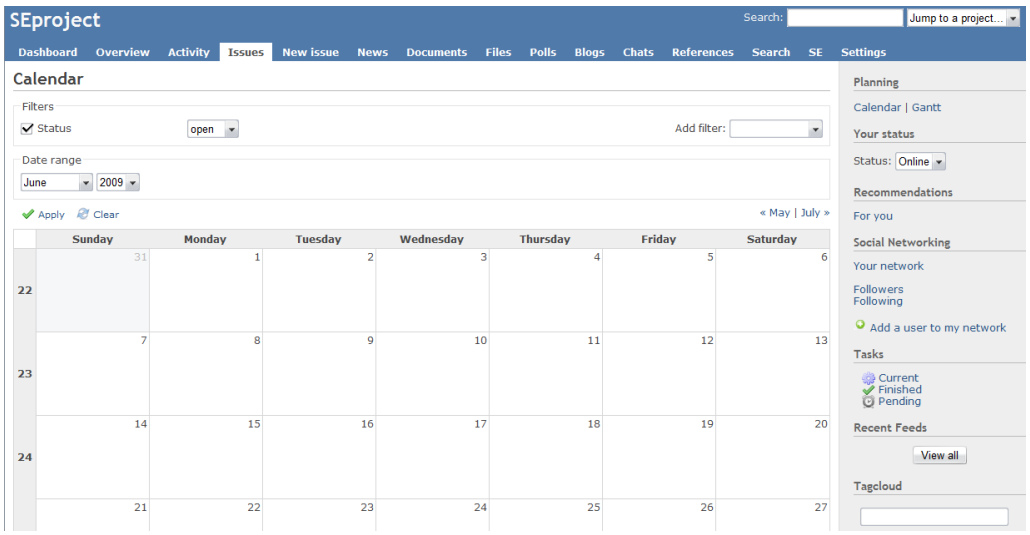

Figure 6.7: Screenshot of the developed prototype.

The following figures show one screenshot for each of the developed and/or integrated plugins of 6.3. More screenshots of the prototype can be seen on Appendix D.

Blog

Figure 6.8 shows the content of a blog's post.

Figure 6.8: Screenshot of a blog's post.

Contact Management

Figure 6.9 shows part of the form for adding contact information of a user.

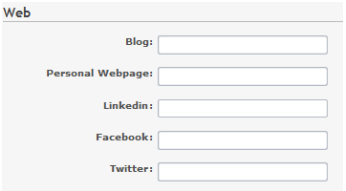

Figure 6.9: Screenshot of the Contact Management plugin.

Dashboard

Figure 6.10 shows the dashboard with the widgets in their default positions.

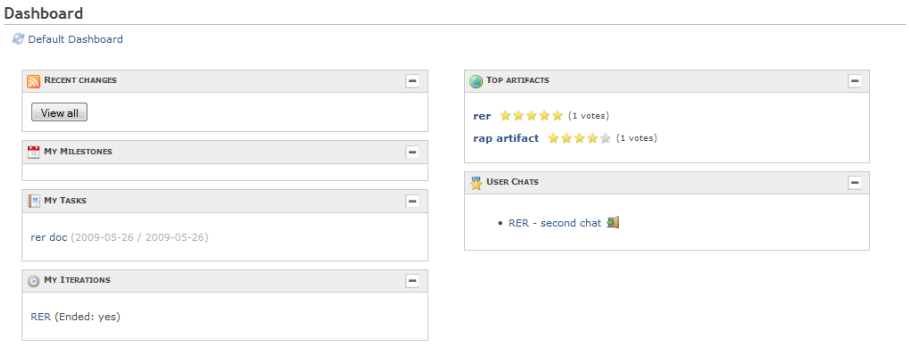

Figure 6.10: Screenshot of the Dashboard plugin with widgets in their default positions.

E-mail

Figure 6.11 shows the e-mail inbox with a new message.

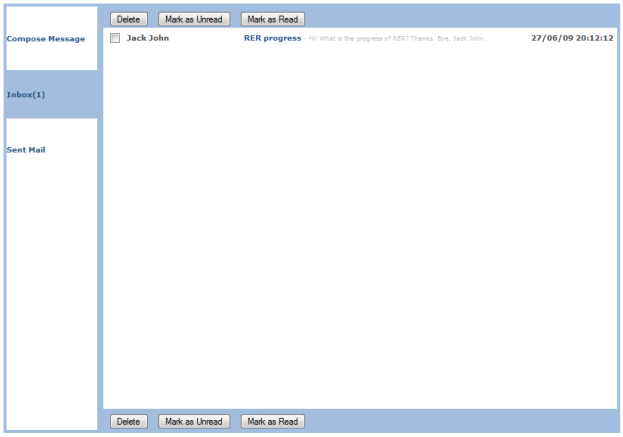

Figure 6.11: Screenshot of the E-mail plugin with a new message in the inbox.

Feeds

Figure 6.12 shows the feeds of the logged-in user. This page can be acessed either by clicking on the "View all" button of the "Recent feeds" category located in the sidebar or by clicking on the same button located in the dashboard. The content of each widget can be show (by clicking on the plus symbol) or hidden (by clicking on the minus symbol).

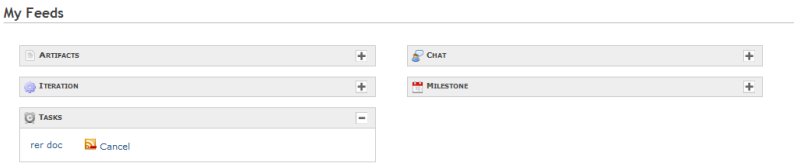

Figure 6.12: Screenshot of the Feeds plugin.

IM/Chat

Figure 6.13 shows a discussion between three users for chatting about the analysis of RER.

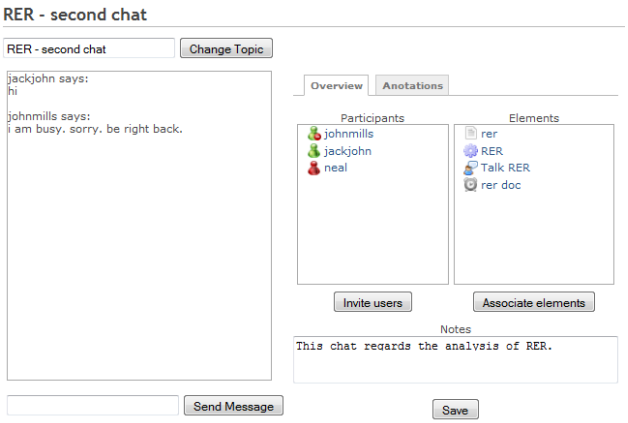

Figure 6.13: Screenshot of the discussion inside the created room.

Microblog

Figure 6.14 shows a micropost. In this figure, there are two icons for editing and removing the micropost, a notebook and a garbage can, respectively. It is also possible to create an Atom feed for the microblog.

Figure 6.14: Screenshot of a micropost.

Polls

Figure 6.15 shows a poll with three votes and one comment.

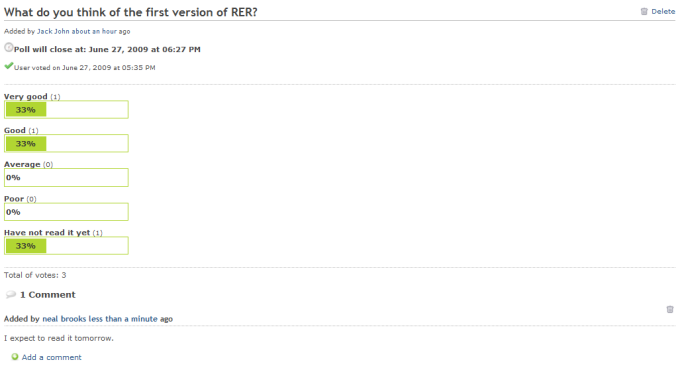

Figure 6.15: Screenshot of a poll.

Ratings

Figure 6.14 shows an example of a user rating an item, namely giving 4 out of 5 stars for the bookmark with title "Managing Requirements". After the rating has been given, the colors of the star turn from green to yellow, as can be seen in the bookmark with title "Requirements: an introduction".

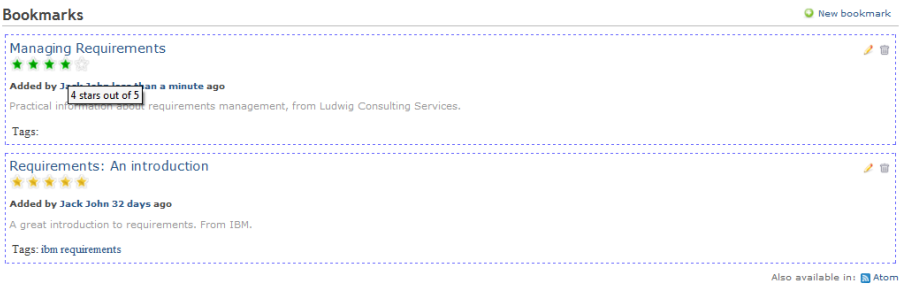

Figure 6.16: Screenshot of a bookmark being rated.

Recommendations

Figure 6.17 shows a recommendation on a bookmark.

Figure 6.17: Screenshot of the Recommendations plugin.

Search

Figure 6.18 shows the results of searching bookmarks belonging to users with “John” on its name.

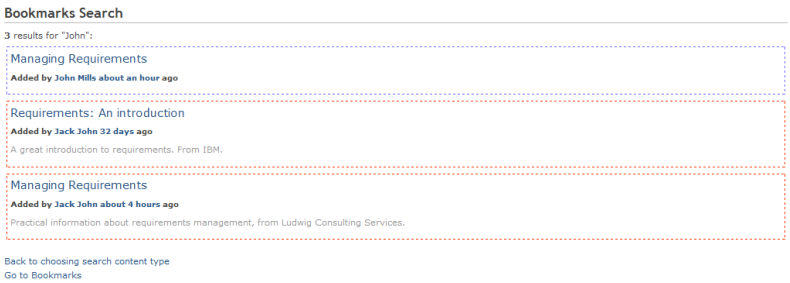

Figure 6.18: Screenshot of the search’s results.

Social bookmarking

Figure 6.19 shows three bookmarks added by the project’s users. The bookmarks added by the logged-in user are delimited by blue-dashed lines, while bookmarks added by other users are delimited by orange-dashed lines.

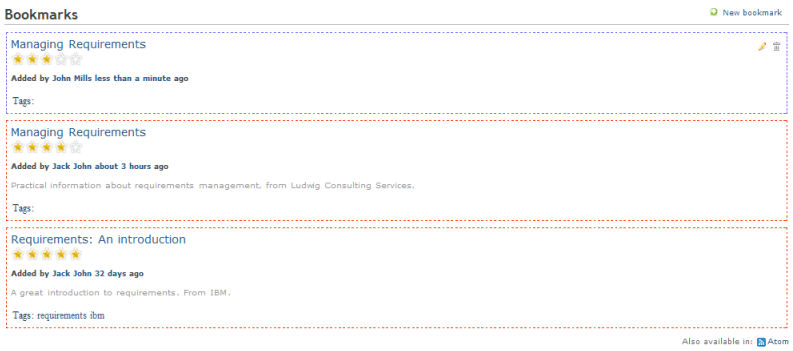

Figure 6.19: Screenshot of the Social bookmarking plugin.

Social cataloging

Figure 6.20 shows two catalog entries added by the project’s users.

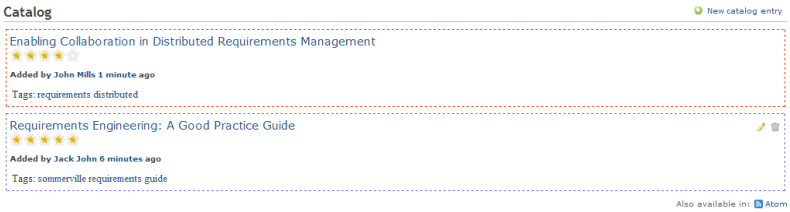

Figure 6.20: Screenshot of the Social cataloging plugin.

Software Engineering

Figure 6.21 shows the details of the task named “rer doc”.

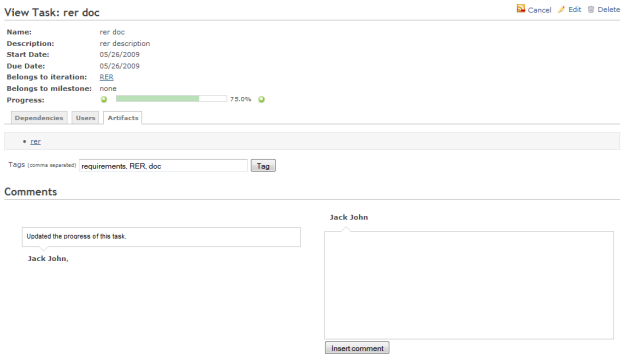

Figure 6.21: Screenshot of the details of the task named “rer doc”.

Social networking

Figure 6.22 shows the social network of the logged-in user.

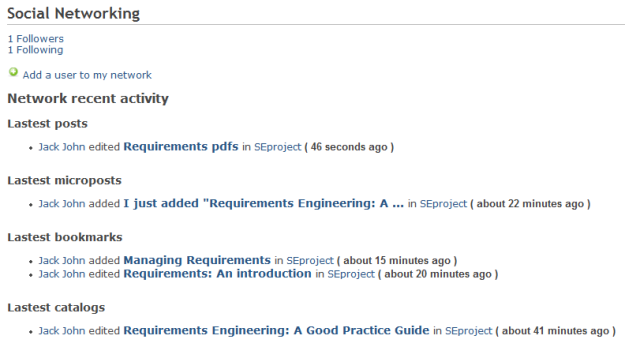

Figure 6.22: Screenshot of the Social networking plugin.

Tags

Figure 6.23 shows the project’s tagcloud.

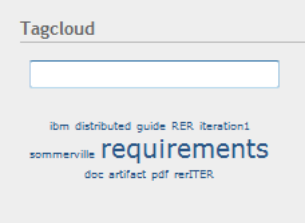

Figure 6.23: Screenshot of the project's tagcloud.

6.7 Implementation details

This section presents the implementation details of the **Recommendations** plugin, due to the complexity of its algorithm.

Research on recommendation algorithms

The first step that was followed was the research of existing recommendation algorithms. As a means to simplify the development, the research was limited to algorithms based on explicit data gathering. There were found two families of algorithms that could be implemented: **item-based collaborative filtering** or **rating-based collaborative filtering**.

The chosen algorithm was **Slope One**, a rating-based collaborative filtering technique, first proposed by [Lemire and Maclachlan, 2005]. Item-based collaborative filtering algorithms [Sarwar et al., 2001] – used for example in Amazon [Linden et al., 2003] – predict the ratings of one item based on the ratings on another item, typically using linear regression ($f(x) = ax + b$), hence requiring a considerable amount of storage and being prone to experience overfitting phenomenon[1]. On the contrary, Slope One uses predictors of the form $f(x) = x + b$, being easy to implement, efficient to query, reasonably accurate, and supportive of both online queries and dynamic updates.

The basis of Slope One schemes works on the principle of a *popularity differential* between items for users, where in a pairwise fashion, it is determined how much better one item is liked than another. This differential is simply obtained by subtracting the ratings of the two items. In turn, this difference can be used to predict another user's rating of one of those items, given their rating of the other. An example in shown in Figure 6.24, where there are two users (A and B) and two items (I and J). User A gave item I a rating of 1 and item J a rating of 1.5, whereas user B gave item I a rating of 2. Therefore, item J is rated more than item I by $1.5 - 1 = 0.5$ points and it could be predicted that user B would give item J a rating of $2 + 0.5 = 2.5$.

[1] In statistics, overfitting occurs when an excessive variance of the difference between the average value of an estimator and the correct value is produced.

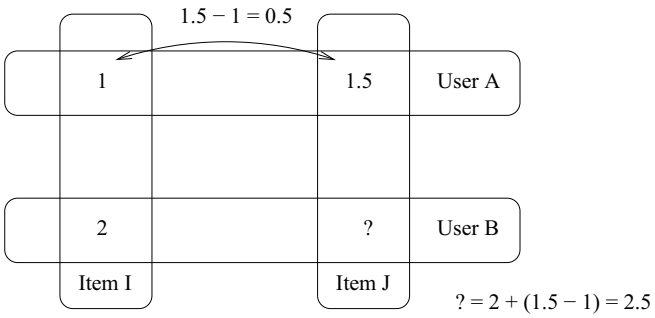

Figure 6.24: Basis of Slope One schemes: User A's ratings of two items and User B's rating of a common item is used to predict User B's unknown rating (extracted from [Lemire and Maclachlan, 2005]).

In the scenario where two items are rated by more than one user, Slope One considers the average difference in ratings between the two items. Another scenario is when a user rates several items, in which case the predicted ratings for each pair of items are combined using a weighted average; in this dissertation, the weight was considered as being the number of users having rated both items. An example of this situation is illustrated in Figure 6.25, with the interrogation symbol meaning that the item wasn't rated.

Users	Item 1	Item 2	Item 3
A	5	3	2
B	3	4	?
C	?	2	5

Figure 6.25: Example where two items were rated by more than one user and where users rated several items.

There are three users (A, B, and C) and three items (1, 2, and 3). For the purpose of obtaining the predicted rating C would give to item 1, the algorithm computes:

1. The average difference between items 1 and 2, adding afterwards the result to the rating C gave to item 2.
2. The average difference between items 1 and 3, adding subsequently the result to the rating C gave to item 3.
3. The weighted average.

In step one the algorithm first computes $((5-3)+(3-4))/2 = 0.5$, which means that on average item 1 is rated above item 2 by 0.5, and then adds 2. Next, in step two,

the algorithm first computes $(5-2)/1 = 3$ and then adds 5. In step three, the algorithm computes $(2*2.5+1*8)/2+1$, being 2 in the numerator/denominator the number of users (A and B) who rated both items 1 and 2, and 1 the number of users (A) who rated both items 1 and 3. In the final, the predicted rating would be 4.33.

Implementation

After researching the existing recommendation algorithms, there was implemented a rating system. Although the recommendation system of the developed prototype was applied to bookmarks and catalog entries, there was chosen a plugin – acts_as_rateable – which allows the association of a rating with any model.

Each record of the ratings database has: a unique identifier, the item's type (e.g. a bookmark), the item's identifier and the identifier of the user who rated it. For the purpose of allowing multiple ratings of an item, the prototype searches the previous rating of the item, deletes it and then adds a new record with the new rating.

Subsequently, a *ratings matrix* was constructed, which serves as the basic structure for the prototype's Slope One algorithm. The rows indexes contain the project's distinct users who added at least one bookmark/catalog entries. The columns indexes contain, for the bookmarks, all the project's distinct URLs associated with the added bookmarks, and for the catalog entries, all the project's distinct entries, with each entry grouped by the title, author's firstname and author's last name.

With the matrix constructed, the content of each element is then filled with the rating the user of the current row gave the item on the current column. If the rating doesn't exist, the element's content is set to 0 and the pair (row, column) is saved in an array (*zeros array*).

The prototype computes the predicted ratings for the unrated items according to the following pseudo-code:

1. Until all the positions of the *zeros array* are processed, pick a zero's position from the *zeros array*.

1.1 In the row of the zero's selected position and until all the columns are processed, search in the *ratings matrix* for elements with non-zero ratings. If an element with a non-zero rating is found, save the zero's position and then the position of the current non-zero value in the *path array*; otherwise, halt to next column of the *ratings matrix*.

1.2 If the size of *path array* is greater than 0, then until all the elements in *path array* are processed, pick an element from the *path array*.

1.2.1 In the column of the first non-zero value previously saved in *path array* and until all the lines are processed, search in *ratings matrix* for elements with non-zero ratings. If

an element with a non-zero rating is found, add to the element the position of the non-zero value to the *path array*; otherwise, halt to the next line of the *ratings matrix*.

1.3 If there was added at least one second non-zero rating to an element of the *path array*, then until all of its elements are processed, pick an element from that array.

1.3.1 If the rating in the same row as the second non-zero rating – i.e., the last position of the current element – and in the same column as the selected zero's position is non-zero, then add its position to the *path array*; otherwise, halt to the next element of that array.

1.4 If there was added at least one third non-zero rating to an element of the *path array*, then until all the elements in that array are processed, pick an element from the array.

1.4.1 Save in a hash – with the key corresponding to the zero's position of each element – the mean of the average differences between the third and second non-zero values, i.e., the predicted rating. If the mean is less than 0.0 or greater than 5.0, set the mean to 0.0 or 5.0, respectively.

2. Return the hash.

Finally, the last step was to transverse the hash returned by the Slope One algorithm and construct an array of recommendations. For each element of the hash, if the predicted rating is greater or equal than 2.5 and the logged-in user matches the user in the row of the element's key, then a recommendation is prepared, which is composed by the:

- The user for whom the recommendation is addressed.
- Recommended item's unique identifier (e.g. for a bookmark is its URL).
- All the items matching the recommended item's identifiers, along with the mean classification of the items and the number of votes.
- Predicted rating.

After adding all the recommendations to an array, the array is then sorted by rating in decreasing order. This array is posteriorly accessed in the view of the item's type (bookmarks or catalog), where the user can then accept the recommendation and be redirected to the page that adds the recommended item, with the obligatory field filled by default and the predicted rating used as the initial rating.

6.8 Summary and Discussion

This chapter presented the steps that were followed for developing the prototype.

The first step was to be aware of the features provided in Redmine, the web application that served as the base for developing the prototype. Launched in June 2006, Redmine is a project management web application written using Ruby on Rails open-source web framework, which by consequence was the framework used to implement the prototype's features.

The next step was to analyze the cost and benefits of implementing each of the features gathered in the state of the art phase. The analysis of each feature was defined according to four criteria:

- **Time**, which defined the estimated number of days for the feature's implementation.
- **Value**, concerning the value gained with the feature's implementation.
- **Innovation**, concerning the uniqueness acquired by implementing the feature when compared to existing software applications.
- **Preference**, concerning the author's preference on implementing the feature.

Then, after ordering the implementation priority of each feature, there were defined the requirements for each plugin.

Afterwards, the architecture of the prototype was defined considering two patterns: **3-tier** and **MVC**. The first is a client-server architecture which defines a system in three separate layers: *Graphical User Interface*, which displays information to the user; *Business Logic*, which controls the systems functionalities; and *Data*, which stores and retrieves the system's data by executing CRUD operations. The second, the architecture pattern of Ruby on Rails applications, breaks a system into three types of components: *model*, which guaranties that the business rules are applied; *view*, which generates the GUI; and *controller*, which receives user events and interacts with model and view.

The prototype's physical architecture was defined as client machine – with an Internet Browser (and associated technologies JavaScript, Ajax, XHTML, and CSS) – accessing a Mongrel Web Server. The Web Server, in his turn, connects through Rails Active Record ORM library to a MySQL Database server and these lastly to a File Server.

The developed prototype successfully implemented all the requirements, with the exception of replying to an e-mail and the user making recommendations.

Chapter 7

Experiment

This chapter exhibits the experiment that was designed for validating the developed prototype, according to the second gap identified in chapter 5.

The experiment's objective was to gather and interpret the performance results of a person using the implemented prototype to collaborate in a Software Engineering environment, when compared to the use of other applications.

7.1 Experiment design

The experiment was designed according to the specification of a scientific method, which was presented in [Zelkowitz and Wallace, 1998].

Experiments conducted using a scientific method study the effect that a method or tool, called a *factor*, has on an attribute of interest. An experiment with an effect previously assigned to a factor is called a *treatment*. The elements studied are called *subjects*. The goal of this type of experiments is to collect data from a sufficient number of subjects – all following the same treatment – as a means to establish statistical validity on the attribute of concern, when compared to some other treatment.

In the (replicated) experiment that was designed, there were defined two small groups of elements (subjects) to perform a similar set of tasks (treatments) in two scenarios (factors). The first scenario concerns the use of software applications in order to successfully fulfill Software Engineering tasks. The second scenario concerns the use of the developed prototype and possibly other external software applications to achieve the same goal.

7.2 Subjects

The subjects were divided in two homogeneous groups: the *control group*, which don't use the prototype, and the *experiment group*, which experience the prototype. The homogeneity of the subjects should be guaranteed by selecting subjects who don't have knowledge of the programming language they have to use so that the tasks are completed.

The selected subjects must verify the following conditions:

- **Be familiar of collaborative software applications**. This condition is an important requisite, since this experiment aims for comparing the productivity of teams from a CSCW perspective, both using and not using the developed prototype. If the subject isn't familiar with collaborative software, then the before and after the prototype comparison cannot be done.
- **Have participated at least in one Software Engineering project**. The prototype has the primary objective of aiding software engineering teams throughout their projects, although it is possible to use it on other contexts. In this way, the subjects must have both the knowledge and practice of the how to develop software and use the tools that most fit their needs along the project life cycle.
- **Be willing to work in a team**. This requisite might seem too social-oriented, but perhaps it is the most crucial one. The importance of this requirement relates to the very nature of why CSCW emerged – the real value of this family of applications arrive not when individuals use them to support their work, but mainly when there is the need of teamwork. If the subjects only exchange opinions and their own work output for personal benefit, these applications lose their social advantages.

7.3 Attributes of interest

There were defined two attributes of interest for the experiment:

- **Task completion time**. This is a simple but extremely effective measurement attribute, since it is objective and common to all the subjects.
- **Tools used**. This is fundamentally a behavioral attribute, reporting which collaborative tools – interpreted as collaborative software applications – each subject used while performing a task. The reason why this attribute was measured relates to the absence of behavioral context provided by the task completion time attribute. The time elapsed during a task does not give crucial information needed to understand, in a given context, which tools are most frequently used by the user and those that are not. For this attribute, both the name and type – according to the

collaboration features referred in chapter 4 – of the tool are subject to recording. For example, “Windows Live Messenger” as name and “IM” as type would be a possible record. Implicit to this recording is the number of different tools, grouped by type, used by a subject. This is an important indicator of how much the prototype reduces the effort and time spent on using and connecting information between different and isolated software applications – by integrating in a single software application multiple collaboration tools, the prototype has the potential of offering an all-in-one package to fulfill a task.

7.4 Tasks

The tasks – which span the Requirements, Design, and Construction areas of Software Engineering – were defined involving the use of Scratch[1](see Figure 7.1), an interactive programming language, to develop the arcade game Pong. This 2D sports game simulates a table tennis, with players controlling a paddle with the purpose of hitting a ball back and forth, losing a point when the ball isn’t returned.

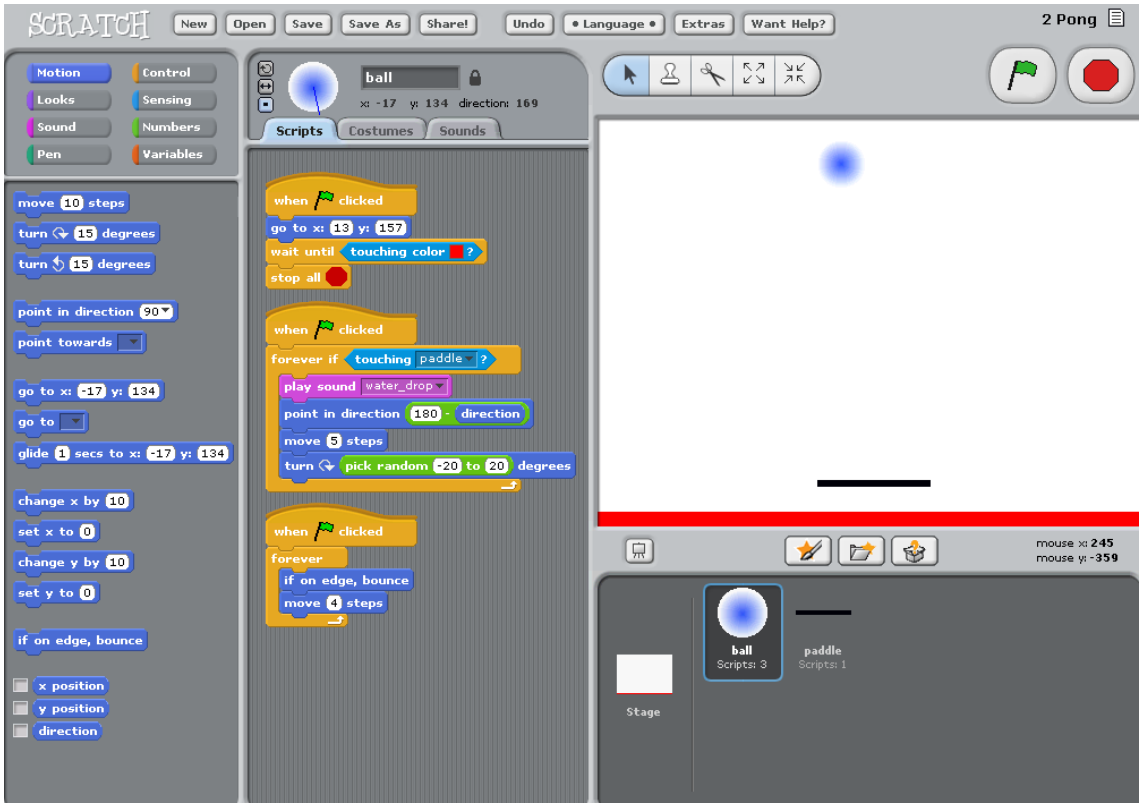

Figure 7.1: Screenshot of Scratch.

[1]http://www.scratch.mit.edu

Scratch is a desktop application available free of charge, designed to help young people (ages 8 and up) develop interactive stories, animations, games, music, and art. The projects can be shared and downloaded by others on the web.

The experiment was designed so that the subjects could complete the maximum number of tasks in a 3 hours session. Before the session begins, each subject should have already a computer with Scratch installed and an account created, while the subjects of the experiment group should have a Redmine project created, with accounts for each one.

The experiment consists of 9 tasks:

Task 1: Create a project in Scratch. In this task it is expected from the subjects to research about Scratch, for example by reading the guide available in Scratch's homepage.

Task 2: Define the functional and non-functional requirements of Pong.
This task has 5 sub-tasks:

1. Research about the game to develop.
2. Identify and document, individually, the game's requirements.
3. Publish the requirements from 2 and communicate the action to the others.
4. Identify and document, in a group, the game's requirements, discussing the requirements from 2.
5. Publish the requirements from 4.

The following deliverables were associated with this task:

- Brief description of each of the identified functional and non-functional requirements.
- Use cases diagrams.
- Actors of the system.
- Linkage between use cases and requirements.
- Prototype of the GUI.

Task 3: Define the logical and physical architecture.
This task has 4 sub-tasks:

1. Identify and document, individually, the game's architecture.

2. Publish the architecture from 1 and communicate the action to the others.
3. Identify and document, in a group, the game's architecture, discussing the architecture from 1.
4. Publish the architecture from 3.

The following deliverables were associated with this task:

- Brief description of the architecture and technologies used.
- Logical and physical architecture diagrams.

Task 4: Define the project plan.
This task has 2 sub-tasks:

1. For each task, define the start and due dates, estimated effort, priority and person responsible.
2. Publish the project plan.

The following deliverables were associated with this task:

- Project plan.

Task 5: Open the Pong project available in Scratch.
This task was defined in order for the user to be more familiarized with an example of simpler version of Pong.

Task 6: Define the static components of the game, such as the colors, background, position and dimensions of objects.

Task 7: Define the animated components of the game, such as objects movements, collisions, user's points, and timer.

Task 8: Document and publish the implemented features of the game.
This task obligatorily includes a relation between the implemented features and the requirements defined in task 2.

Task 9: Make the game available for the team.

Tasks 6 and 7 were defined so that the first was to be executed by one sub-group, while the second was to be performed by a second sub-group, disjoint from the previous

sub-group. This situation was defined for two reasons. The first was to separate the efforts, so that while one sub-group was implementing the game's scenario the other was researching the more difficult phase of how to program the game's animation. The second was to make the sub-group of task 7 communicate with the group from task 6 to acquire knowledge of their work.

Each task/sub-task was defined with different levels collaboration, which could be Low (L) or High (H) (see Figure 7.2).

Task / Sub-task	1	2.1	2.2	2.3	2.4	2.5	3.1	3.2	3.3	3.4	4	5	6	7	8	9
Collaboration Level	L	L	L	H	H	H	L	H	H	H	H	L	H	H	H	L

Figure 7.2: Collaborations levels of the experiment's tasks/sub-tasks.

7.5 Summary and Discussion

In the designed experiment, there were defined two small groups of elements to perform a similar set of tasks in two scenarios. The first scenario concerns the use of software to successfully fulfill the tasks and the second the use of the prototype and possibly other external software to achieve the same goal.

The subjects were divided in two homogeneous groups: the *control group*, which don't use the prototype, and the *experiment group*, which experience the prototype.

The selected subjects must be familiar of collaborative software applications, have participated at least in one Software Engineering project and be willing to work in a team. For measuring the performance of the subjects there were defined two attributes of interest: *task completion time* and *tools used*, with the latter compensating the absence of behavioral context provided by the first.

The experiment was designed for a 3 hours session and is composed of 9 tasks – which span the Requirements, Design, and Construction areas – involving the use of Scratch, an interactive programming language, to develop the arcade game Pong.

Since the experiment was not run yet, it is planned to refine it in small groups and then evolve it, for its execution to be done in an academic setting and later in an industrial context.

Chapter 8

Conclusions

The objectives defined in the problem statement were in its majority completely fulfilled. There were studied features potentially capable of augmenting collaboration between software engineers and also how to adapt groupware technologies to the needs of a project. These features were then searched inside Software Engineering applications, rationally chosen and in the final developed and/or integrated in Redmine, a web-based tool.

The depth and broadness of the research, the degree of integration between features (e.g. web feeds in the blog) and the high number of successfully implemented requirements were relevant achievements. For the purpose of studying the impact of the developed prototype a replicated experiment was defined, although it wasn't conducted, due to the unavailability of the subjects.

This chapter presents the contributions of new knowledge made by this thesis and the future research to be done after this dissertation.

8.1 Summary of contributions

The contributions of new knowledge made by this thesis were achieved in three phases: before, during and after the prototype.

In the first phase, there was performed a detailed research of software applications of various Software Engineering areas and later ordered the features to develop and/or integrate in the prototype. Both the research of applications on multiple areas and the use of the research's results to choose the features are new developments to the current state of the art.

In the second phase, the features were all integrated in a single web-based tool to provide support for the entire life cycle of a Software Engineering project. The

innovation was achieved not only through the prototype's goals, but also by its features – Recommender, Social bookmarking and Social cataloging are unique to the prototype.

Finally, in the third phase, it was designed an experiment for validating the impact of using the prototype in a real-life situation. The experiment can be replicated in the context of any Software Engineering project and is not restricted to the developed prototype.

8.2 Future Research

The most critical future development of this thesis relates to conducting the designed experiment. By running the experiment, insight would be given on the effectiveness of using the developed prototype, future improvements of the prototype would be identified, and proof would be obtained on differences between individual and collective work regarding the number of committed errors, the time elapsed during the tasks and the quality of subject's work.

In the future, researchers should focus their attention on the open issues identified in section 4.7. The reason why so few of the studied Software Engineering applications include some of the most popular collaboration software available on the Web – such as blogs, microblogs, social networks, wikis and tagging – should be brought out. Additionally, the inexistence of online whiteboard, screen sharing, recommender, social cataloging and social bookmarking features on any of the studied applications should stimulate researchers on determining the competitive advantage gained by its use in Software Engineering projects.

Some interesting fields of study to explore in the future should be the integration between desktop and web-based applications, the development of methods and applications to effectively support global software development environments (which will certainly become more frequent than nowadays), the migration of desktop applications to the Web, the definition of procedures for successfully adapting groupware technologies to the specific context of each project and the teams' needs, and the integration between multiple collaborative software when there is complementariness of benefits (e.g. embedding VoIP in IM, with the first providing a more suitable service for detailed conversations and the second for saving the history of a conversation).

During the dissertation there were collected ideas for future developments of the prototype. These include using tags in conjunction with e-mail to reduce the loss of information context, the personalization of groups of receivers in a microblog, receiving e-mail and feeds notifications of any changes made by "watched" users, making recommendations based on implicit data collection, uploading files to an e-mail, integrating contact management with e-mail, implementing live search in the search plugin, and extending recommender, social networking and tags to the Redmine modules.

Additionally, the remaining features which weren't developed on this version of the prototype should also be reviewed as further work.

To conclude, although the developed prototype provides support for all the phases of a project's life-cycle, this support is generic. Contrarily to the studied collaboration tools, there weren't developed or integrated tools specifically targeted to a particular Software Engineering area, such as in the case of the integration of CollabNet TeamForge with Microsoft Visual Studio and Eclipse construction tools.

Bibliography

[Abran et al., 2004] Abran, A., Bourque, P., Dupuis, R., Moore, J. W., and Tripp, L. L. 2004. Guide to the Software Engineering Body of Knowledge - SWEBOK. IEEE Press, Piscataway, NJ, USA, 2004 version edition.

[Ackerman and Mcdonald, 2000] Ackerman, M. S. and Mcdonald, D. W. 2000. Collaborative support for informal information in collective memory systems. Information Systems Frontiers, 2(3-4):333–347.

[Alexa, 2009] Alexa 2009. Alexa top 500 global sites. http://www.alexa.com/topsites. Last accessed on 20 March 2009.

[Allen, 1995] Allen, C. 1995. Definitions of groupware. http://www.alacrityventures.com/DoG.html. Last accessed on 23 May 2009.

[Baecker et al., 1995] Baecker, R. M., Grudin, J., Buxton, W., and Greenberg, S. 1995. Readings in Human-Computer Interaction: Toward the Year 2000. Morgan Kaufmann.

[Bell, 2007] Bell, A. 2007. Advantages of social cataloging. http://annamaebell.wordpress.com/2007/11/11/advantages-of-social-cataloging/. Last accessed on 24 May 2009.

[Ben-Shaul, 1993] Ben-Shaul, I. Z. 1993. Oz : A decentralized process centered environment. Technical report, Columbia University.

[Ben-Shaul et al., 1992] Ben-Shaul, I. Z., Kaiser, G. E., and Heineman, G. T. 1992. An architecture for multi-user software development environments. SIGSOFT Softw. Eng. Notes, 17(5):149–158.

[Blog, 2007] Blog, M. D. 2007. Mongrel vs. webrick. http://www.missiondata.com/blog/ruby/71/mongrel-vs-webrick/. Last accessed on 29 June 2009.

[Bolcer and Taylor, 1996] Bolcer, G. A. and Taylor, R. N. 1996. Endeavors: a process system integration infrastructure. In: ICSP '96: Proceedings of the Fourth International Conference on the Software Process (ICSP '96), Washington, DC, USA. IEEE Computer Society, page 76.

[Bowers and Benford, 1991] Bowers, J. M. and Benford, S. D. (Editors) 1991. Studies in computer supported cooperative work: theory, practice and design. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands.

[Brenner, 2009] Brenner, A. 2009. Effective wuala usage: Online collaboration. http://www.wuala.com/blog/2009/01/

effective-wuala-usage-part-3-online.html1. Last accessed on 24 May 2009.

[Carzaniga et al., 2001] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. 2001. Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst., 19(3):332–383.

[Cheng et al., 2004] Cheng, L.-T., de Souza, C. R., Hupfer, S., Patterson, J., and Ross, S. 2004. Building collaboration into ides. Queue, 1(9):40–50.

[Cheng et al.,] Cheng, L.-T., Rohall, S., and Patterson, J. Project: Bluegrass:virtual worlds for business. http://domino.watson.ibm.com/cambridge/research.nsf/99751d8eb5a20c1f852568db004efc90/1b1ea54cac0c8af1852573d1005dbd0c?OpenDocument. Last accessed on 25 May 2009.

[Coleman, 1995] Coleman, D. 1995. Groupware technology and applications: an overview of groupware. pages 3–41.

[Compete, 2009] Compete 2009. Site comparison of twitter.com (rank #47), facebook.com (#3). http://siteanalytics.compete.com/twitter.com+facebook.com/?metric=sess. Last accessed on 20 March 2009.

[da Silva et al., 2006] da Silva, I. A., Chen, P. H., Van der Westhuizen, C., Ripley, R. M., and van der Hoek, A. 2006. Lighthouse: coordination through emerging design. In: eclipse '06: Proceedings of the 2006 OOPSLA workshop on eclipse technology eXchange, New York, NY, USA. ACM, pages 11–15.

[Davis, 1969] Davis, J. H. 1969. Group performance. Addison-Wesley.

[Definition, 2002] Definition, C. 2002. What is camelcase? http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci824384,00.html. Last accessed on 24 May 2009.

[DokuWiki, 2009] DokuWiki 2009. backlinks [dokuwiki]. http://www.dokuwiki.org/backlinks. Last accessed on 24 May 2009.

[Dourish, 2003] Dourish, P. 2003. The appropriation of interactive technologies: Some lessons from placeless documents. Comput. Supported Coop. Work, 12(4):465–490.

[Dourish and Bellotti, 1992] Dourish, P. and Bellotti, V. 1992. Awareness and coordination in shared workspaces. In: CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work, New York, NY, USA. ACM, pages 107–114.

[Ebersbach et al., 2005] Ebersbach, A., Glaser, M., and Heigl, R. 2005. Wiki : Web Collaboration. Springer.

[Educause, 2005] Educause 2005. 7 things you should know about... social bookmarking. Educause. Last accessed on 24 May 2009.

[Edwards, 2008] Edwards, T. 2008. Advantages to online data storage backup. http://ezinearticles.com/?Advantages-To-Online-Data-Storage-Backup&id=919741. Last accessed on 24 May 2009.

[Eick et al., 1992] Eick, S. G., Steffen, J. L., and Sumner, Jr., E. E. 1992. Seesoft-a tool for visualizing line oriented software statistics. IEEE Trans. Softw. Eng., 18(11):957–968.

[Ellis et al., 1991] Ellis, C. A., Gibbs, S. J., and Rein, G. 1991. Groupware: some issues and experiences. Commun. ACM, 34(1):39–58.

[Ensor, 1990] Ensor, B. 1990. How can we make groupware practical (panel). In: Proc. Conf. on Human Factors in Computing Systems (CHI), Seattle, WA. ACM Press, pages 87–89.

[Farmer, 2003] Farmer, R. 2003. Instant messaging - collaborative tool or educator's nightmare!

[FCC, 2008] FCC 2008. Video relay services. http://www.fcc.gov/cgb/consumerfacts/videorelay.html. Last accessed on 6 June 2009.

[Fishg, 2008] Fishg, S. 2008. Comparison between cvs and subversion. http://versioncontrolblog.com/comparison/CVS/Subversion/index.html. Last accessed on 29 June 2009.

[Garrett and Danziger, 2007] Garrett, K. R. and Danziger, J. N. 2007. Im = interruption management? instant messaging and disruption in the workplace. Journal of Computer-Mediated Communication, 13(1):23–42.

[Giles, 2005] Giles, J. 2005. Access : Internet encyclopaedias go head to head : Nature. http://www.nature.com/nature/journal/v438/n7070/full/438900a.html#top. Last accessed on 24 May 2009.

[Golder and Huberman, 2005] Golder, S. and Huberman, B. A. 2005. The structure of collaborative tagging systems.

[Greenberg, 1991a] Greenberg, S. (Editor) 1991a. Computer-supported cooperative work and groupware. Academic Press Ltd., London, UK, UK.

[Greenberg, 1991b] Greenberg, S. 1991b. Personalizable groupware: accommodating individual roles and group differences. In: ECSCW'91: Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work, Norwell, MA, USA. Kluwer Academic Publishers, pages 17–31.

[Greif, 1988] Greif, I. (Editor) 1988. Computer-supported cooperative work: a book of readings. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Grinter, 1999] Grinter, R. E. 1999. Systems architecture: product designing and social engineering. SIGSOFT Softw. Eng. Notes, 24(2):11–18.

[Gross and Traunmuller, 1996] Gross, T. and Traunmuller, R. 1996. Methodological considerations on the design of computer-supported cooperative work. Cybernetics and Systems: An International Journal, 27(3):279–303.

[Grudin, 1988a] Grudin, J. 1988a. Perils and pitfalls. BYTE, 13(13):261–264.

[Grudin, 1988b] Grudin, J. 1988b. Why cscw applications fail: problems in the design and evaluationof organizational interfaces. In: CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work, New York, NY, USA. ACM, pages 85–93.

[Grudin, 1994] Grudin, J. 1994. Computer-supported cooperative work: history and focus. Computer, 27(5):19–26.

[Grudin and Poltrock, 1991] Grudin, J. and Poltrock, S. 1991. Computer-supported cooperative work. In: Conf. on Human Factors in Computing Systems (CHI) Tutorial notes.

[Hamilton, 2007] Hamilton, A. 2007. Wikitravel - 50 best websites 2008 - time. http://www.time.com/time/specials/2007/article/0,28804,1809858_1809957_1811566,00.html. Last accessed on 24 May 2009.

[Hammond et al., 2005] Hammond, T., Hannay, T., Lund, B., and Scott, J. 2005. Social bookmarking tools (i): A general review. D-Lib Magazine, 11(4).

[Handel and Herbsleb, 2002] Handel, M. and Herbsleb, J. D. 2002. What is chat doing in the workplace? In: CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 1–10.

[Herbsleb, 2007] Herbsleb, J. D. 2007. Global software engineering: The future of socio-technical coordination. In: FOSE '07: 2007 Future of Software Engineering, Washington, DC, USA. IEEE Computer Society, pages 188–198.

[Heymann, 2008] Heymann, P. 2008. Tag hierarchies. http://heymann.stanford.edu/taghierarchy.html. Last accessed on 24 May 2009.

[Heymann and Garcia-Molina, 2006] Heymann, P. and Garcia-Molina, H. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University.

[Heymann et al., 2008] Heymann, P., Koutrika, G., and Garcia-Molina, H. 2008. Can social bookmarking improve web search? In: WSDM '08: Proceedings of the international conference on Web search and web data mining, New York, NY, USA. ACM, pages 195–206.

[Hiltz and Turoff, 1993] Hiltz, S. R. and Turoff, M. 1993. The network nation: human communication via computer. MIT Press, Cambridge, MA, USA.

[Honeycutt and Herring, 2009] Honeycutt, C. and Herring, S. C. 2009. Beyond microblogging: Conversation and collaboration via twitter. In: HICSS '09: Proceedings of the 42nd Hawaii International Conference on System Sciences, Washington, DC, USA. IEEE Computer Society, pages 1–10.

[Huang et al., 2006] Huang, E. M., Mynatt, E. D., Russell, D. M., and Sue, A. E. 2006. Secrets to success and fatal flaws: The design of large-display groupware. IEEE Comput. Graph. Appl., 26(1):37–45.

[Hung et al., 2008] Hung, C.-C., Huang, Y.-C., Hsu, J. Y., and Wu, D. K. 2008. Tag-based user profiling for social media recommendation. In: Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, AAAI 2008.

[Hupfer et al., 2004] Hupfer, S., Cheng, L.-T., Ross, S., and Patterson, J. 2004. Introducing collaboration into an application development environment. In: CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 21–24.

[IEEE, 1990] IEEE 1990. Ieee standard glossary of software engineering terminology. Technical report, IEEE.

[Johansen, 1988] Johansen, R. 1988. GroupWare: Computer Support for Business Teams. The Free Press, New York, NY, USA.

[Johnson, 2009] Johnson, N. 2009. Social networking and blogs surpass email in popularity. http://blog.searchenginewatch.com/090310-082257. Last accessed on 23 May 2009.

[Johnson-Lenz and Johnson-Lenz, 1981] Johnson-Lenz, P. and Johnson-Lenz, T. 1981. The evolution of a tailored communications structure: The topics system. Computerized Conferencing and Communications Center, 1.

[Johnson-Lenz and Johnson-Lenz, 1998] Johnson-Lenz, P. and Johnson-Lenz, T. 1998. Groupware: coining and defining it. SIGGROUP Bull., 19(3):58.

[JSPWiki, 2008] JSPWiki, C. 2008. Camel case. http://www.jspwiki.org/wiki/CamelCase. Last accessed on 24 May 2009.

[Kadia, 1992] Kadia, R. 1992. Issues encountered in building a flexible software development environment: lessons from the arcadia project. SIGSOFT Softw. Eng. Notes, 17(5):169–180.

[Kamble, 2008] Kamble, S. 2008. Rss feeds advantages and disadvantages. http://www.samirkamble.com/rss-feeds-advantages-and-disadvantages/. Last accessed on 24 May 2009.

[Kaplan et al., 1992] Kaplan, S. M., Tolone, W. J., Carroll, A. M., Bogia, D. P., and Bignoli, C. 1992. Supporting collaborative software development with conversationbuilder. SIGSOFT Softw. Eng. Notes, 17(5):11–20.

[Kersten, 2007] Kersten, M. 2007. Mylyn - the task-focused interface. www.eclipse.org/mylyn/presentations/2007-12-mylyn-webinar.ppt. Last accessed on 21 May 2009.

[Kiss, 2008] Kiss, J. 2008. Last.fm widgets boost user numbers. http://www.guardian.co.uk/media/2008/feb/28/web20.digitalmedia. Last accessed on 24 May 2009.

[Kittur and Kraut, 2008] Kittur, A. and Kraut, R. E. 2008. Harnessing the wisdom of crowds in wikipedia: quality through coordination. In: CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 37–46.

[Kittur et al., 2007] Kittur, A., Suh, B., Pendleton, B. A., and Chi, E. H. 2007. He says, she says: conflict and coordination in wikipedia. In: CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA. ACM, pages 453–462.

[Kling, 1991] Kling, R. 1991. Cooperation, coordination and control in computer-supported work. Commun. ACM, 34(12):83–88.

[Koch and Gross, 2006] Koch, M. and Gross, T. 2006. Computer-supported cooperative work - concepts and trends. In: Proc. Conf. of the Association Information And Management (AIM). Bonner Koellen Verlag.

[Kotonya and Sommerville, 1998] Kotonya, G. and Sommerville, I. 1998. Requirements Engineering : Processes and Techniques (Worldwide Series in Computer Science). John Wiley & Sons.

[Krause et al., 2008] Krause, B., Schmitz, C., Hotho, A., and Stumme, G. 2008. The anti-social tagger: detecting spam in social bookmarking systems. In: AIRWeb '08: Proceedings of the 4th international workshop on Adversarial information retrieval on the web, New York, NY, USA. ACM, pages 61–68.

[Kroski, 2005] Kroski, E. 2005. The hive mind: Folksonomies and user-based tagging. http://infotangle.blogsome.com/2005/12/07/the-hive-mind-folksonomies-and-user-based-tagging/. Last accessed on 24 May 2009.

[Lemire and Maclachlan, 2005] Lemire, D. and Maclachlan, A. 2005. Slope one predictors for online rating-based collaborative filtering. In: Proceedings of SIAM Data Mining (SDM'05).

[Leuf and Cunningham, 2001] Leuf, B. and Cunningham, W. 2001. The Wiki way: quick collaboration on the Web. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. 2003. Amazon.com recommendations: item-to-item collaborative filtering. Internet Computing, IEEE, 7(1):76–80.

[Malone and Crowston, 1994] Malone, T. W. and Crowston, K. 1994. The interdisciplinary study of coordination. ACM Comput. Surv., 26(1):87–119.

[Marcelo Alvim and da Silva,] Marcelo Alvim, André van der Hoek, R. R. A. S. and da Silva, I. A. Side-by-side project awareness - exploring multi-monitor environments. http://www.ics.uci.edu/~{}andre/sidebyside.html. Last accessed on 25 May 2009.

[McConnell, 1996] McConnell, S. 1996. Rapid Development: Taming Wild Software Schedules. Microsoft Press, Redmond, WA, USA.

[McGiboney, 2009] McGiboney, M. 2009. Twitter's tweet smell of success. http://blog.nielsen.com/nielsenwire/online_mobile/twitters-tweet-smell-of-success/. Last accessed on 12 July 2009.

[Meggelen, 2005] Meggelen, J. V. 2005. The problem with video conferencing. http://www.oreillynet.com/etel/blog/2005/04/the_problem_with_video_confere.html. Last accessed on 24 May 2009.

[Minchington and Estis, 2009] Minchington, B. and Estis, R. 2009. 6 steps to an employer brand strategy. http://brandcoach.typepad.com/branddigest/brett-minchington-ryan-estis/. Last accessed on 20 March 2009.

[Myers et al., 1998] Myers, B. A., Stiel, H., and Gargiulo, R. 1998. Collaboration using multiple pdas connected to a pc. In: CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 285–294.

[Nunamaker, 1995] Nunamaker 1995. Electronic meeting systems: Ten years of lessons learned. In: Groupware: Technology and Applications. Prentice-Hall, Inc.

[Nunamaker et al., 1991] Nunamaker, J. F., Dennis, A. R., Valacich, J. S., Vogel, D., and George, J. F. 1991. Electronic meeting systems to support group work. Commun. ACM, 34(7):40–61.

[Nunamaker et al., 1996] Nunamaker, Jr., J. F., Briggs, R. O., Mittleman, D. D., Vogel, D. R., and Balthazard, P. A. 1996. Lessons from a dozen years of group support systems research: a discussion of lab and field findings. J. Manage. Inf. Syst., 13(3):163–207.

[O'Reilly, 2005] O'Reilly, T. 2005. What is web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=2. Last accessed on 24 May 2009.

[Overflow, 2009] Overflow, S. 2009. Mongrel vs. webrick. http://stackoverflow.com/questions/596902/mongrel-vs-webri. Last accessed on 29 June 2009.

[Panko, 1992] Panko, R. R. 1992. Managerial communication patterns. Journal of Organizational Computing, pages 95–122.

[PmWiki, 2005] PmWiki, C. 2005. Camelcasedlinks. http://www.pmwiki.org/wiki/Test/CamelCasedLinks. Last accessed on 24 May 2009.

[Prante et al., 2004] Prante, T., Streitz, N. A., and Tandler, P. 2004. Roomware: computers disappear and interaction evolves. Computer, 37(12):47–54.

[Project, 1999] Project, S. 1999. The scout report – september 17, 1999. http://www.mail-archive.com/scout-report@hypatia.cs.wisc.edu/msg00038.html. Last accessed on 24 May 2009.

[Quan-Haase et al., 2005] Quan-Haase, A., Cothrel, J., and Wellman, B. 2005. Instant messaging for collaboration: A case study of a high-tech firm. Journal of Computer Mediated Communication, 10(4).

[Reenskaug, 1978] Reenskaug, T. 1978. Mvc xerox parc 1978-79. http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html. Last accessed on 1 June 2009.

[Reinhard et al., 1994] Reinhard, W., Schweitzer, J., Völksen, G., and Weber, M. 1994. Cscw tools: Concepts and architectures. Computer, 27(5):28–36.

[Reiss, 1995] Reiss, S. P. 1995. The Field Programming Environment: A Friendly Integrated Environment for Learning and Development. Kluwer Academic Publishers, Norwell, MA, USA.

[Roseman and Greenberg, 1996] Roseman, M. and Greenberg, S. 1996. Teamrooms: network places for collaboration. In: CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 325–333.

[Russell et al., 2002] Russell, D. M., Drews, C., and Sue, A. 2002. Social aspects of using large public interactive displays for collaboration. In: UbiComp '02: Proceedings of the 4th international conference on Ubiquitous Computing, London, UK. Springer-Verlag, pages 229–236.

[Sarma et al., 2003] Sarma, A., Noroozi, Z., and van der Hoek, A. 2003. Palantir: raising awareness among configuration management workspaces. In: Software Engineering, 2003. Proceedings. 25th International Conference on. pages 444–454.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In: WWW '01: Proceedings of the 10th international conference on World Wide Web, New York, NY, USA. ACM, pages 285–295.

[Schmidt, 2006] Schmidt, J. 2006. Social software: Facilitating information-, identity- and relationship-management, pages 31–49. Books on Demand.

[Schmidt and Bannon, 1992] Schmidt, K. and Bannon, L. 1992. Taking CSCW seriously. Supporting articulation work. Computer-Supported Cooperative Work , (1):7–40.

[Schroeder, 2007] Schroeder, S. 2007. The ultimate rss toolbox - 120+ rss resources. http://mashable.com/2007/06/11/rss-toolbox/. Last accessed on 24 May 2009.

[Sears and Jacko, 2007] Sears, A. and Jacko, J. A. (Editors) 2007. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Second Edition (Human Factors and Ergonomics). CRC.

[Shaw, 1971] Shaw, M. E. 1971. Group Dynamics: The Psychology of Small Group Behavior. McGraw Hill, New York.

[Shirky, 2004] Shirky, C. 2004. Folksonomy. http://many.corante.com/archives/2004/08/25/folksonomy.php. Last accessed on 24 May 2009.

[Shiu and Lenhart, 2004] Shiu, E. and Lenhart, A. 2004. How americans use instant messaging. http://www.pewinternet.org/~/media//Files/Reports/2004/PIP_Instantmessage_Report.pdf.pdf. Last accessed on 24 May 2009.

[Sinha et al., 2006] Sinha, V., Sengupta, B., and Chandra, S. 2006. Enabling collaboration in distributed requirements management. IEEE Softw., 23(5):52–61.

[Smith et al., 2005] Smith, G., Merholz, P., Morville, P., and Wal, T. V. 2005. Sorting out social classification. In: Information Architecture Summit. Last accessed on 24 May 2009.

[Software, 2005] Software, P. 2005. Svn vs cvs. http://www.pushok.com/soft_svn_vscvs.php/. Last accessed on 29 June 2009.

[Sterling, 2005] Sterling, B. 2005. Order out of chaos. http://www.wired.com/wired/archive/13.04/view.html?pg=4. Last accessed on 24 May 2009.

[Stewart et al., 1999] Stewart, J., Bederson, B. B., and Druin, A. 1999. Single display groupware: a model for co-present collaboration. In: CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA. ACM, pages 286–293.

[Streitz et al., 1991] Streitz, N. A., Tandler, P., Müller-Tomfelde, C., and Konomi, S. 1991. Roomware: Towards the next generation of human-computer interaction based on an integrated design of real and virtual worlds, pages 553–578. Addison-Wesley.

[Suchman, 1987] Suchman, L. A. 1987. Plans and situated actions: the problem of human-machine communication. Cambridge University Press, New York, NY, USA.

[Tanaka and Taylor, 1991] Tanaka, J. W. and Taylor, M. 1991. Object categories and expertise: Is the basic level in the eye of the beholder? Cognitve Psychology, 23:457–482.

[Thomas et al., 2006] Thomas, D., Hansson, D., Breedt, L., Clark, M., Davidson, J. D., Gehtland, J., and Schwarz, A. 2006. Agile Web Development with Rails. Pragmatic Bookshelf.

[Trac, 2006] Trac, C. 2006. Camel case. http://trac.edgewall.org/wiki/CamelCase. Last accessed on 24 May 2009.

[Tyler and Tang, 2003] Tyler, J. R. and Tang, J. C. 2003. When can i expect an email response? a study of rhythms in email usage. In: ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work, Norwell, MA, USA. Kluwer Academic Publishers, pages 239–258.

[Vertegaal and Ding, 2002] Vertegaal, R. and Ding, Y. 2002. Explaining effects of eye gaze on mediated group conversations:: amount or synchronization? In: CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work, New York, NY, USA. ACM, pages 41–48.

[Wal, 2005] Wal, V. 2005. Folksonomy definition and wikipedia. http://www.vanderwal.net/random/entrysel.php?blog=1750. Last accessed on 24 May 2009.

[Wal, 2007] Wal, V. 2007. Folksonomy. http://www.vanderwal.net/folksonomy.html. Last accessed on 24 May 2009.

[Wegman, 2007] Wegman, J. 2007. Caught in shelfari's sticky web: No more friends, please! http://www.observer.com/2007/caught-shelfari-s-sticky-web-no-more-friends-please. Last accessed on 24 May 2009.

[Whitehead and Goland, 1999] Whitehead, Jr., E. J. and Goland, Y. Y. 1999. Webdav: a network protocol for remote collaborative authoring on the web. In: ECSCW'99: Proceedings of the sixth conference on European Conference on Computer Supported Cooperative Work, Norwell, MA, USA. Kluwer Academic Publishers, pages 291–310.

[Whitehead, 2007] Whitehead, J. 2007. Collaboration in software engineering: A roadmap. In: FOSE '07: 2007 Future of Software Engineering, Washington, DC, USA. IEEE Computer Society, pages 214–225.

[Wikipedia, 2009] Wikipedia 2009. Flixter. http://en.wikipedia.org/wiki/Flixster. Last accessed on 24 May 2009.

[Wilson, 1991] Wilson, P. (Editor) 1991. Computer Supported Cooperative Work: An Introduction. Springer.

[Wilson, 1994] Wilson, P. 1994. Introducing CSCW–what it is and why we need it. Ashgate Publishing, Brookfield, VT.

[Wired, 2005] Wired 2005. Wikipedia, britannica: A toss-up. http://www.wired.com/culture/lifestyle/news/2005/12/69844. Last accessed on 24 May 2009.

[WorldWideWebSize, 2009] WorldWideWebSize 2009. The size of the world wide web. http://www.worldwidewebsize.com/. Last accessed on 24 May 2009.

[Zelkowitz and Wallace, 1998] Zelkowitz, M. V. and Wallace, D. R. 1998. Experimental models for validating technology. Computer, 31(5):23–31.

[Zeller, 2007] Zeller, A. 2007. The future of programming environments: Integration, synergy, and assistance. In: FOSE '07: 2007 Future of Software Engineering, Washington, DC, USA. IEEE Computer Society, pages 316–325.

Appendix A

Tools' details

This appendix complements chapter 4 by showing the details of each of the analyzed software tools.

A.1 Bug Tracking Tools

A.1.1 BUGtrack

BUGtrack is a paid web-based bug tracking and project management system launched in 2001.

Collaboration/Awareness

BUGtrack's awareness features are few, when compared for example with the ones provided by JIRA or Bugzilla. The features are: **e-mail notifications** of new bugs or status changes; **reporting of new bugs by email**, automatically turning bugs into trackable records; **RSS feeds 2.0** subscriptions; and **time tracking**, which consists on registering the estimated time and the actual time spent to resolve a bug.

Similarly with the remaining bug-tracking tools, BUGtrack integrates with revision control systems, namely **CVS**, **Subversion** and **Microsoft Visual Source Safe**.

Integration

It can **import Comma-separated value (CSV)** and **Microsoft Excel** files.

A.1.2 Bugzero

Bugzero is a paid web-based bug tracking system launched in 2001.

Collaboration/Awareness

Bugzero's awareness features are: **e-mail bug submissions** for modifying an existing bug or creating a new one, including a spam-filter and support for e-mail attachments (per project and template based); and integration with **CVS** and **Perforce** source version control systems, through e-mail.

Integration

Bugzero's integration features include: **exporting of CSV files**, with support of Asian languages in Unicode; and **LDAP/Active Directory user authentication** support, available as add-on.

A.1.3 Bugzilla

Bugzilla is a free web-based bug tracking system launched in 1998.

Collaboration/Awareness

Bugzilla offers the most complete set of awareness features, which include: **email notifications** about any change made in Bugzilla – which notifications the user gets and on which bugs is defined by his preferences; **scheduled reports** (daily, weekly, hourly, etc.) **by e-mail**; **charts**; **time tracking**; **comments**, which can be marked as private (not visible outside the user's group); **creating/modifying bugs through e-mail**; **"watch"** other users (i.e., receive their e-mails); **control bug visibility/editing** with groups; **CVS**, **Perforce**, and **Subversion** version control systems integration; and **Mylyn** integration.

Integration

Bugzilla integrates with: **Bonsai**, a web-based tool for managing CVS; **Tinderbox**, a program that prepares HTML pages which display the history of different development variables; **JIRA**. It also imports/exports **CSV** files into/from spreadsheets.

A.1.4 JIRA

JIRA (see Figure A.1) is a paid web-based bug and issue tracking system launched in 2002.

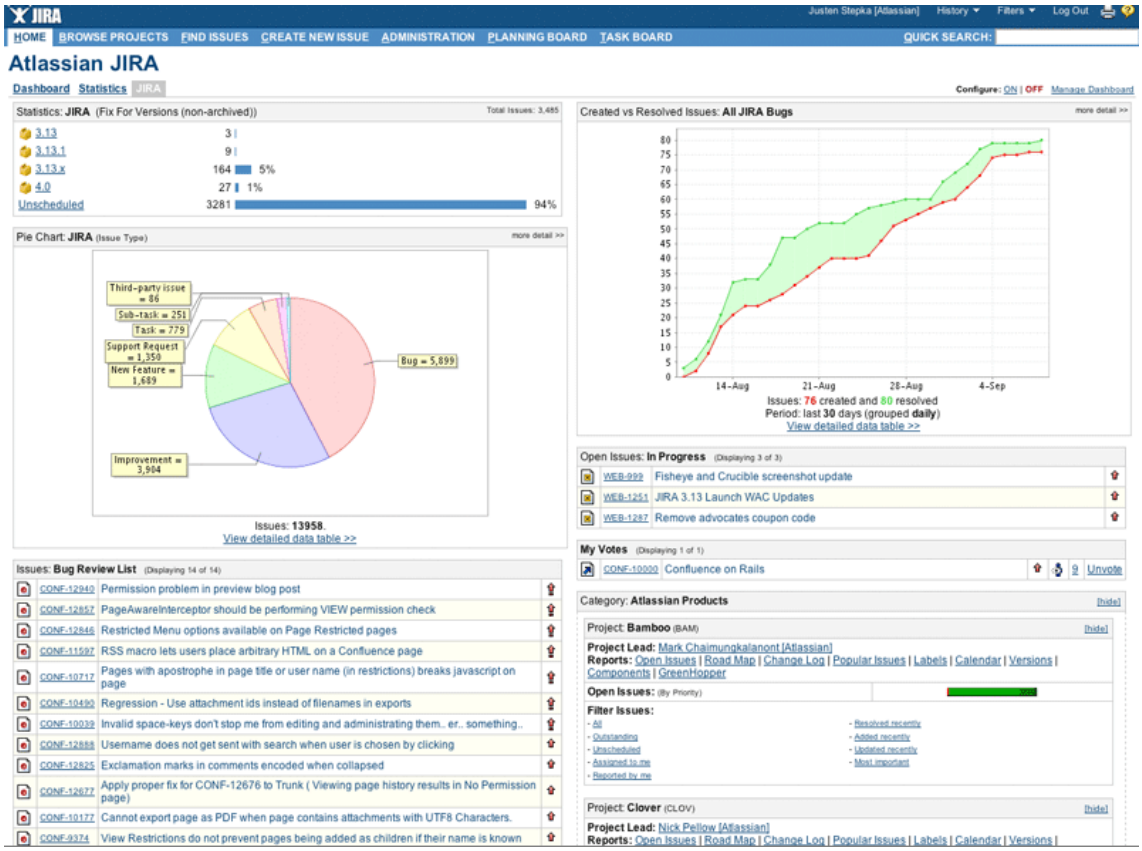

Figure A.1: Screenshot of the bug tracking tool JIRA (extracted from http://www.atlassian.com/software/jira).

Collaboration/Awareness

In JIRA, users' awareness is primarily achieved by the use of **e-mail** and **comments** features. The user can create or comment on issues using an e-mail account, receive the latest information to their inbox by subscribing to a saved filter, write comments for an issue, and choose which issues can be voted on and by whom (with the results posteriorly displayed in a "Popular Issues" report).

Like most bug-tracking tools, JIRA possesses a **time-tracking** feature. The built-in time tracking reports shows the project's current status and any deviation from original time estimates.

Two unique awareness features of JIRA are a **dashboard** – the first page the user sees when logged in to JIRA – and **Mylyn** integration, a task focused UI for Eclipse that makes working with both large and small workspaces as easy as possible. **Mylyn brings JIRA inside an IDE** – for example, the user can view his list of assigned issues in JIRA and edit and comment on JIRA issues, whether he is online or offline. Finally, Mylyn handles syncing up any changes made to JIRA when the user comes back online.

Collaboration/Communication

JIRA's communication features are limited to its integration with **Confluence**, an enterprise wiki, and also to **Gliffy**.

When the user activates trackback in both JIRA and Confluence, if a link is made from one application to the other, the link will automatically lead both ways. Additionally, when macros are activated, any JIRA search-result can be embedded in a Confluence page and any JIRA dashboard portlet can be embedded in a Confluence page.

When JIRA Confluence Portlet extension is installed, users can display a Confluence page and display it in the JIRA's portlet, which can be useful when showing news or selected RSS feeds from Confluence.

Regarding revision control systems JIRA integrates with: **CVS**, **Subversion**, **Perforce**, and **Accurev**.

Integration

JIRA can **import data** from various applications: **Bugzilla**, **FogBugz**, **Mantis**, **Microsoft Project**, **Microsoft Excel**, and **CSV** files.

One great advantage of JIRA, when compared with the other researched applications, is the diversity of the supplied plugins, including: JIRA **charts**, JIRA **calendar** (which includes exporting as an iCal file), **Timechart** (tracks the value of a particular statistic over a given period of time), **IntelliJ IDEA** IDE, JIRA **FishEye** (users can see the direct connections between JIRA issues and their source repositories, CVS or Subversion), JIRA **Bambo** (which automatically compiles and tests code as it changes), and **GreenHopper** (a project management tool).

A.1.5 MantisBT

MantisBT is a free web-based bug-tracking system.

Collaboration/Awareness

MantisBT's awareness capabilities are limited and include: **e-mail notifications**, **built-in reporting** (through reports or graphs), **RSS Feeds** (applied to news, issues matching saved filters, and issues matching a specific project), **reporting issues via e-mail** (available as a patch), **time tracking**, and Source Control Integration (**SVN** and **CVS**). An exclusive awareness feature of MantisBT is its integration with **Twitter**.

Collaboration/Communication

The sole communication functionality of MantisBT is a **chat**, for which integration is optional.

Integration

MantisBT integration include: exporting contents to **CSV**, **Microsoft Excel**, and **Microsoft Word** files; and **JIRA** integration.

A.2 Construction Tools

A.2.1 Aptana Studio

Aptana Studio is a paid IDE.

Collaboration/Awareness

When compared to other IDEs – such as IntelliJ IDEA or Eclipse – Aptana Studio has few awareness features. **"Remote Project Creation"** is a feature – available only on Aptana Studio Pro edition – that provides a wizard to create projects for remote sites. Other remote feature is **"Remote Project Import"**, offering a wizard to create local projects from a remote File Transfer Protocol (FTP), Secure File Transfer Protocol (SFTP), or FTP Secure (FTPS) location. The last feature, common to all IDEs, is SCC integration: users can install free plugins to integrate with **SVN** or **CVS**.

Integration

Aptana Studio has plugins for several tools and programming languages. One plugin is for **Adobe AIR**, a software solution for building rich Internet applications (RIAs). Another plugin is for **Eclipse** – being based on Eclipse, Aptana Studio can be plugged into Eclipse or other Eclipse-based IDEs. It is possible to extend Aptana Studio with other Aptana plugins or plugins for Eclipse (which counts over 1000).

Some languages and standards supported by Aptana Studio are: Ajax, Cascading Style Sheets (CSS), Document Object Model (DOM), HyperText Markup Language (HTML), JavaScript, Hypertext Preprocessor (PHP), Python, and Ruby on Rails.

In addition to the languages support, Aptana Studio also embeds ready to use application platforms for Ajax, Java, PHP, Python, and Ruby on Rails.

A.2.2 Eclipse

Eclipse (see Figure A.2) is a free IDE launched in 2004.

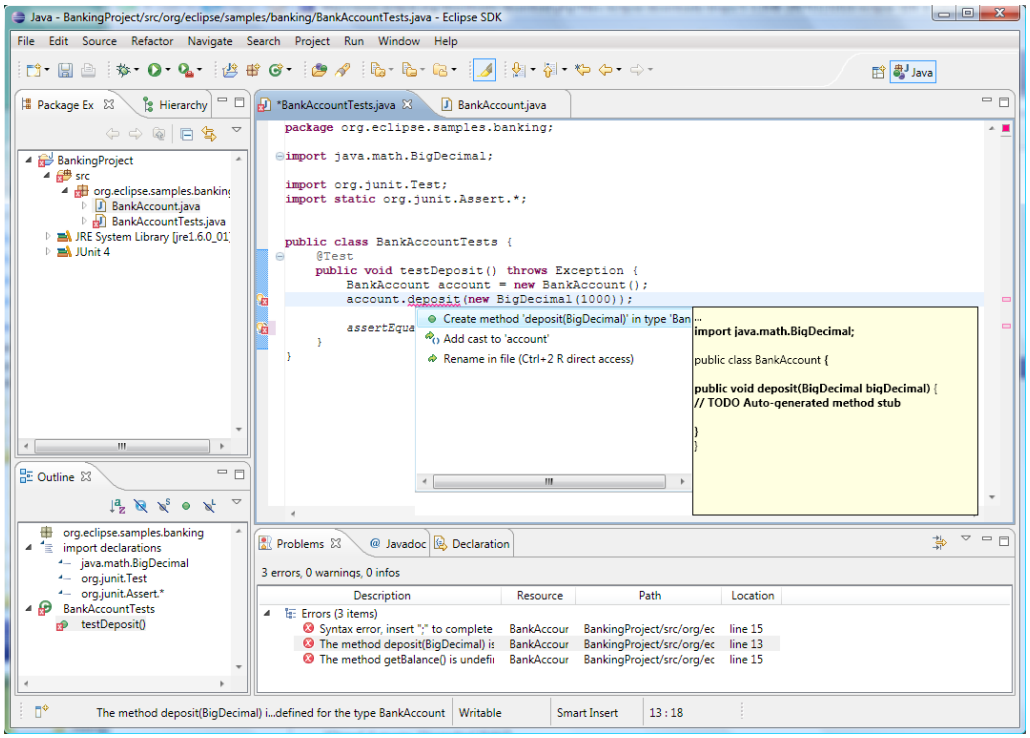

Figure A.2: Screenshot of the construction tool Eclipse (extracted from http://www.eclipse.org).

Collaboration/Awareness

Elipse's awareness features are abundant and diverse. Eclipse does not provide built-in awareness features – these are provided by its enormous collection of plugins. Examples of awareness plugins are: **Subclipse**, which enables Subversion support; **AccuRev AccuBridge**, which integrates SCM into Eclipse; **AT-Project**, a bug tracking solution for managing issues, tasks and changes along a project life-cycle; **Clearcase**, a plugin based on IBM's Clearcase, a tool with version control, workspace management, and parallel development capabilities; **CodeBeamer**, a Web-architected software development solution with collaborative features for development teams; **ProjectKoach**, which adds progress chart support for measuring Velocity and Burn-Up rates of a project; **StarTeam**, which integrates workflow known by Borland StarTeam users with Eclipse, such as allowing users to configure remote location or providing customized views of team-specific information; **Teamprise**, a plugin that provides access to Team Foundation Server, part of Microsoft's Team System; **Cola**, a Real-Time Shared Editing tool; **Code Colaborator**, a plugin based on Smartbear's application which enables peer review of source code changes; **Gmail-Clipse**, a Gmail client for using e-mail accounts inside and outside Eclipse; and **Eclipse RSS Reader**, which enables Eclipse's users to subscribe and browse various RSS feeds.

Not only are the plugins numerous but also they are based on applications of great quality – AccuRev was awarded both in 2005[1], with a Productivity Award, and in 2007[2], with a Jolt Award; and Code Collaborator was awarded in 2008[3] with a Jolt Award.

[1] http://www.ddj.com/showArticle.jhtml?documentID=sdm0506b&pgno=12
[2] http://www.ddj.com/architect/201001409
[3] http://www.sdexpo.com/2008/west/press/jolt.htm

Besides plugins, Eclipse integrates with: **Team Concert**, which includes a dashboard; **Jazz**; **ColabNet**; and **Borland Caliber Analyst**.

Collaboration/Communication

Eclipse's communication features are achieved via the plugin **Shareclipse**.

Shareclipse allows users to create at least one peer group and then to work together on "Shared-Documents" in real-time. Multimedia audio/video conferencing, flexible local/remote video switching, and a chat view with emotion support are also provided.

Collaboration/Collective Knowledge

Collective Knowledge is harnessed in Eclipse through **Code Beamer** plugin. This plugin provides Knowledge management support, with full text search, indexing and tagging to classify content.

Integration

Among the languages supported by Eclipse there are: C/C++, CSS, Cobol, Java, Java EE, Javascript, Perl, PHP, Python, PHP, Radrails, and Qt.

Eclipse also integrates with a significant number of tools, including: **Altova UModel**, **Borland Together**, **BOUML**, **Eclipse UML**, **Enterprise Architect**, **Magic Draw** and **Visual Paradigm** design tools; **Aptana Studio** and **IntelliJ IDEA** IDEs; and **OpenOffice** documents.

A.2.3 IntelliJ IDEA

IntelliJ IDEA is a paid IDE.

Collaboration/Awareness

IntelliJ IDEA provides few awareness features. These features are: the **synchronization of developers' configuration**, including code style settings, code inspection profiles, and project libraries configuration; a **diff tool**, which compares project files opened by two team members; integration with **AccuRev**, a SCM solution; integration with **Git version control system**, achieved by the plugin Git4Idea; **Mercurial and Bazaar distributed version control systems**; and VCS integration (**Team Foundation Server**, **Subversion**, **Perforce**, **CVS**, and **StarTeam Visual SourceSafe**).

Collaboration/Communication

IntelliJ IDEA is one of the three IDEs – the others are Eclipse and NetBeans – that has a **chat**. The bundled IM client provides not only standard actions like exchanging text messages and navigation through message history, but also unique code-related communications. These unique features include referencing to a particular point in the file, sending stack trace with direct links to underlying source code, viewing differences between project files opened by two team members, viewing a list of files opened in IntelliJ IDEA by any of the contacts, and managing others' access to the IDE project files.

Integration

Some of the languages supported by IntelliJ IDEA are: CSS, Flex, Groovy, HTML, Java, JavaScript, JRuby, Rails, Ruby, XHTML, XML, and XSL.

IntelliJ IDEA features are enhanced through the integration with the following applications: **JIRA**; **TeamCity**; **Eclipse**, allowing users to share the single code base in a mixed-IDE environment; **Ant** and **Maven**, two builder tools; **application servers** (Apache Tomcat versions 4.x and 5.x, Bea Systems WebLogic versions 7.x, 8.x, 9.x and 10.x, IBM WebSphere versions 5.1, 6.0 and 6.1, JBoss, Geronimo, Glassfish, and Resin); **Facets**, a plugin that encapsulate support for a specific framework or technology, thus ensuring a more easy configuration and management of Enterprise Applications; **Magic Draw** and **Visual Paradigm** design tools; and **Code Review**, which allows a developer to send a code review email that details any changes to their code compared with the repository, before changes are checked in.

A.2.4 JCreator

JCreator is a paid IDE.

Collaboration/Awareness

JCreator possesses only one awareness feature in consequence from its integration with **Ant** and **CVS** version control systems.

Integration

The main language supported by JCreator is Java. That being said, JCreator provides support for **JavaServer Pages (JSP)**, a Java technology that allows developers to create dynamically-generated web sites, with HTML, XML, or other document types. JSP technology allows Java code and certain pre-defined actions to be embedded into static content.

A.2.5 KDevelop

KDevelop is a free IDE launched in 2000.

Collaboration/Awareness

The only awareness feature present in KDevelop is the integration with **CVS**, **Clearcase**, **Perforce**, and **Subversion** version control systems. KDevelop provides GUI support – via context menu on files and directories of the project and also directly in the editor – and status coloring of the version control system in the project tree view.

Integration

KDevelop supports 15 languages, including: C/C++, Java, Perl, PHP, Python, and Ruby. It also has a plugin that integrates Ant, a build tool for Java projects.

A.2.6 Komodo

Komodo is a paid IDE.

Collaboration/Awareness

Komodo's awareness features are: **"multi-document editing"**, allowing users to work on multiple documents simultaneously using multiple tab groups, split views, and cross-document searches; support for **CVS**, **Perforce**, **Subversion**, **Git**, **Mercurial**, and **Bazaar** version control systems; **"multi-user support"**, a feature that shares important data – like a toolbox, templates, or configuration preferences – between multiple users with a common data directory; and **"project manager"**, a flexible organization of all project elements, with live folders for the current contents of corresponding file system directory and virtual folders for any project or Toolbox component.

Integration

Some of the languages supported by Komodo are: CSS, HTML, JavaScript, Perl, PHP, Python, and Ruby.

A.2.7 Microsoft Visual Studio 2008

Microsoft Visual Studio is a paid IDE launched in 1997.

Collaboration/Awareness

Team Foundation Server, a team collaboration platform, part of Microsoft Visual Studio Team System 2008, provides Microsoft Visual Studio 2008 the following awareness tools: version control; work item tracking; bug management; reporting and business intelligence on project status, performance and quality metrics; customizable process templates; integration with Office Excel and Office Project for project management; Team Concert, which includes a dashboard; and notification of actions assigned to services, using feed syndication, e-mail, or invoking other web service.

Visual Studio 2008 provides the following plugins: **Team Review**, an add-in for performing code reviews; **VisualSVN**, a professional Subversion integration; **TestPartner**, which automates functional testing for .NET and the Web; **Issue Tracking Anywhere**, a web-based issue tracking system designed for issue/work item tracking, bug tracking, customer support and project management; **Team Foundation Sidekicks**, a suite of tools for Team Foundation Server administrators and advanced users, providing GUI for administrative and advanced version control tasks in multi-user environments; and **TeamSpec**, which enables project requirement management activities directly inside Microsoft Word, by the use of Team Foundation Server project artifacts such as Scenarios, Risks, Issues, Bugs, and Tasks.

Microsoft Visual Studio 2008 can also integrate with the following tools: **Borland Caliber Analyst**, **RavenFlow**, **Telelogic DOORS**, and **CollabNet TeamForge**.

Collaboration/Communication

Microsoft Visual Studio 2008 provides the following communication features: a **project portal**, where users can communicate between themselves and link comments to various items; and **TeamCompanion for Outlook**, a plugin that integrates Microsoft Outlook with Team Foundation Server.

Integration

Some languages supported by Microsoft Visual Studio 2008 are: C#, C/C++, and CSS.

Microsoft Visual Studio integrates with: .NET Framework; LINQ, a set of language extensions to Visual Basic and Visual C#; Microsoft Office; SQL Server; Web, including ASP.NET AJAX; Windows Client; and Windows Mobile.

Finally, in terms of plugins, Visual Studio 2008 includes: **Nevron Diagram Designer**, a freeware diagram editor for .NET components; and **Altova UModel** design tool.

A.2.8 NetBeans

NetBeans is a free IDE launched in 1997.

Collaboration/Awareness

NetBeans's awareness features include: **automatic recognition of existing version-controlled directories**; integration with **CVS**, **Subversion**, **Mercurial**, and **Clearcase** version control systems; **"Versioning window"**, allowing users to view all recent changes in a directory, read commit log messages, and roll back changes to previous versions; **diff viewer**; **"Remote Project Sharing"**, which allows sharing NetBeans projects in real time over the network, reviewing partners' work, and making remote changes to the project; and **MigCalendar**, **VoIP**, and **CodeBeamer** plugins (including comments, forums, wiki, and e-mail).

Collaboration/Communication

NetBeans possesses a collaboration **chat**, giving developers the possibility of sending instant messages in plain text, XML, HTML, or Java, complete with syntax highlighting.

Integration

Some of the most relevant languages features are: C/C++, Java, Java EE and Web, JavaScript, PHP, Ruby/Ruby on Rails, and Visual Mobile.

NetBeans integration extends to **Magic Draw** and **Visual Paradigm** design tools and **NetBeans UML**, **Struts 2.x Framework**, and **OpenSwing** plugins. NetBeans UML is a tool for designing an UML model and generating source code from the UML model. Struts 2.x Framework is an extensible framework for creating enterprise-ready Java web applications. OpenSwing is an open-source project that provides a set of Swing graphics components manipulated inside the IDE.

A.2.9 Zend Studio

Zend Studio is a paid IDE.

Collaboration/Awareness

Zend Studio provides one awareness feature, which is the integration with source control systems **CVS** and **Subversion**.

Integration

Zend Studio integrates with **Eclipse**.

A.3 Design Tools

A.3.1 Altova UModel

Altova UModel is a paid desktop-base UML tool.

Collaboration/Awareness

Altova UModel product's awareness features are limited to: **sharing and reusing customized UML packages** across multiple projects or among developers, and integration with **version control systems**.

Integration

Like the majority of the design tools, Altova UModel provides interoperability through the use of eXtensible Markup Language (XML) and XML Metadata Interchange (XMI). XMI is a part of the UML specification that describes the method for software tools to save UML models in a common format (e.g. XML), enabling users to share UML models even if they don't use the same UML applications. That being said, Altova UModel 2009 supports **modeling of XML Schema in UML diagrams** and **XMI 2.1 model interchange**.

Altova UModel integrates with **Eclipse** and **Microsoft Visual Studio** IDEs; in the latter, the user can view and seamlessly switch between his UML software model and source code editing windows in the same application development environment where Visual Studio software projects are developed.

A.3.2 ArgoUML

Argo UMl is a free desktop-based UML tool launched in 2002.

Integration

ArgoUML provides integration to **Pramatic Studio 3.0**, an J2EE IDE, where the user can create a UML model, open a UML project, and perform reverse-engineering from code written within the IDE; and to **Eclipse**, with two plugins: **ArgoEclipse**, an open source

UML tool based on ArgoUML, and **Argo2Encore**, which converts UML models created with ArgoUML to the Eclipse UML dialect Ecore, thus generating Java code.

It supports XMI (**imports the UML1.4 formats XMI 1.1 and 1.2**, but only **writes XMI 1.2**) and **Object Constraint Language (OCL)**. Using OCL, ArgoUML provides constraint modeling support – syntax and type checking – on UML Classes.

A.3.3 Borland Together

Borland Together is a paid desktop-based design tool.

Integration

Borland Together provides integration to **Eclipse** and **Borland Caliber Analyst** (a Collaborative Requirement Definition and Management software).

A.3.4 BOUML

BOUML is a free desktop-based UML tool launched in 2005.

Collaboration/Awareness

BOUML's awareness features are: integration with version control systems (including **CVS**, **Subversion**, and **Clear Case**); **HTML documentation generator**; and **sharing project files**.

Integration

BOUML provides integration to **Eclipse** and **Rational Rose**. In case of Rational Rose, BOUML imports the models, although it creates the diagrams empty.

It provides support for **XMI 1.2 and 2.1 generation** and for **importing XMI 2.1** for a UML 2.0 or 2.1 file.

A.3.5 EclipseUML

EclipseUML is a paid desktop-based UML tool launched in 2002.

Collaboration/Awareness

EclipseUML's only awareness feature is its integration with version control system **CVS**.

Integration

EclipseUML provides integration to **Eclipse**. It also supports **XMI 2.1**: users can change, add or delete metamodel elements using the XMI 2.1 Editor view.

A.3.6 Enterpise Architect

Enterpise Architect is a paid desktop-based UML tool.

Collaboration/Awareness

Enterprise Architect's awareness features are among the most complete and include: integration with **version control systems**; **sharing Enterprise Architect's files** by placing them on a shared network drive and afterwards having small teams concurrently log on and work within the models; posting models on the Internet or on a local Intranet using the **HTML report generator**; and finally using Enterprise Architect's files replication feature for complex **distributed development**.

Integration

Enterprise Architect provides integration to **Eclipse** and **Microsoft Visual Studio .NET**. It also supports **XML schemas**, including: Reverse Engineer XML schema into UML models and Forward Engineer XML schema from UML models.

A.3.7 Gliffy

Gliffy (see Figure A.3) is a free/paid web-based UML tool.

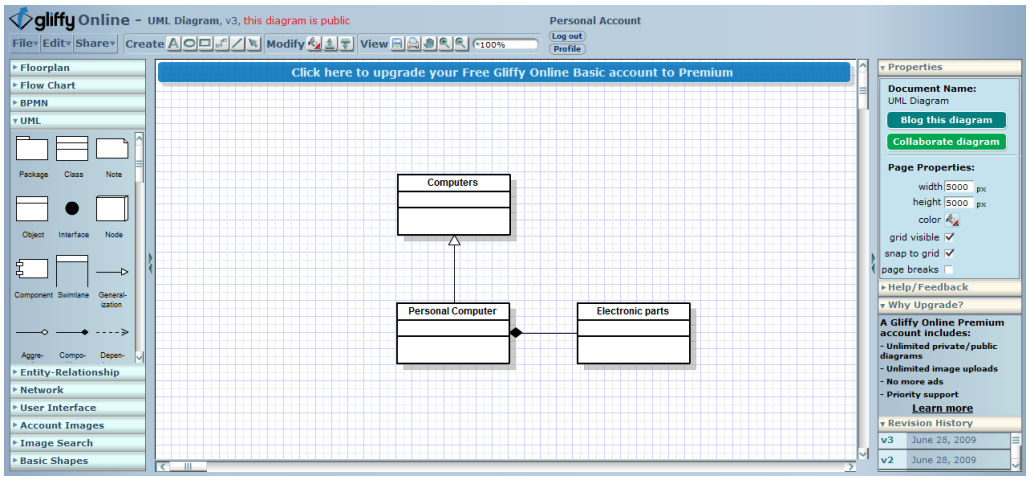

Figure A.3: Screenshot of the design tool Gliffy.

Collaboration/Awareness

Being the only web-based design tool analyzed, Gliffy offers one of the most complete set of awareness features, which encompasses: **revision control**, where every time a user changes a file, Gliffy saves a new version to the version's historical; **private workspaces** (only available however in a Premium account); **one-click publishing**, where users can share a URL with a read-only version of a diagram or even embed the diagram's image into a wiki, blog and hosted office applications; and finally inviting other users with **auto-email notification**.

Collaboration/Communication

Gliffy's communication features are achieved by its integration with **Confluence**, an enterprise wiki, making it possible for users to add diagrams to a Confluence wiki page, and also with **JIRA**, giving users the possibility of adding diagrams to a JIRA issue.

Collaboration/Collective Knowledge

Gliffy offers the possibility of **tagging a diagram** to improve search and organization.

A.3.8 Magic Draw

Magic Draw is a paid desktop-based UML tool launched in 1998.

Collaboration/Awareness

Magic Draw's awareness features include: **Teamwork Server**; declaring **shared packages** inside a project – which can be made visible in other projects – using the original project as a module; and **CSV import** plugin.

Integration

Magic Draw integrates with; **IntelliJ IDEA** 4.X or later; **NetBeans** 6.X or later; **Eclipse** 3.1 or later (JDT or Java IDE); **CodeGear JBuilder** 8.0, 9.0, X, 2005, 2006, 2007, 2008; and **IBM RAD** 7.0.

It also supports **XMI** (imports for versions 1.0, 1.1, and 1.2), **XML** (imports and exports) and **OCL**.

A.3.9 Microsoft Visio

Microsoft Visio is a paid desktop-based UML tool launched in 2000.

Collaboration/Awareness

Microsoft Visio's awareness features are: **document workspaces**; creating **timelines and calendars**, by using revision marks to track diagram changes; and checking diagrams in and out directly from **Microsoft Office Visio**. These features are achieved through its integration with **Microsoft Office SharePoint Server**.

Collaboration/Communication

Microsoft Visio's communication features are a **wiki**, a **blog** and a **portal**, all achieved through its integration with Microsoft Office SharePoint Server.

Integration

Microsoft Visio integrates with a significant number of tools, including: **Autodesk AutoCAD**; **Clip art**; **Microsoft Visual Studio**; **Microsoft Office Outlook 2007**; **Microsoft Office Project** integration with Visio Gantt charts and timelines; and **XML Web services** integration.

A.3.10 Rational Rose Data Modeler

Rational Rose Data Modeler is a paid desktop-based UML tool.

Collaboration/Awareness

Rational Rose Data Modeler possesses the following awareness features: **Web Publishing**; integration with any Source Code Control (SCC)-compliant **version control system**; a **repository**; and a method of **merging changes made by a team**.

Integration

Rational Rose Data Modeler integrates with **XML Document Type Definition (DTD)**, which defines the legal building blocks of a XML document. With XML DTD, independent groups of people can agree to use a common DTD for interchanging data.

A.3.11 StarUML

StarUML is a free desktop-based UML tool launched in 1996.

Integration

StarUML's integration features include: Microsoft Office Document Generation (in **Microsoft Word**, **Excel**, and **PowerPoint** formats); XMI 1.1: **UML 1.3 import** and **XMI export**; and **import Rational Rose** files.

A.3.12 Telelogic Rhapsody

Telelogic Rhapsody is a paid desktop-based UML tool.

Collaboration/Awareness

The awareness features of Telelogic Rhapsody include: **Systems Modeling Language (SysML)**[4] compliance; and team collaboration, including **model-based graphical differencing** and **merging**.

Integration

Telelogic Rhapsody integrates with: **Telelogic DOORS**, **Eclipse**, and **Rational RequisitePro**. It also supports **XMI**.

[4] A general purpose modeling language for systems engineering applications. It is a dialect of UML.

A.3.13 Visual Paradigm

Visual Paradigm is a paid desktop-based UML tool.

Collaboration/Awareness

Visual Paradigm's awareness features are narrow and include: VP Teamwork Server, used for example for **concurrent and collaborative modeling**; and integration with SCC **CVS**, **Subversion**, and **Perforce**.

Integration

Visual Paradigm's integration is vast and includes the following tools: **Eclipse**, **NetBeans** (e. g. importing NetBeans 6.x UML diagrams), **ItelliJ IDEA**, **BEA WebLogic Workshop**, **Borland JBuilder**, and **Oracle JDeveloper** IDEs; **Microsoft Visio** (e.g. importing Visio drawings); and **Microsoft Excel** (e.g. importing and exporting Excel files for UML diagrams).

It also supports importing and exporting **XMI /XML** files.

A.4 Engineering Management Tools

A.4.1 Basecamp

Basecamp (see Figure A.4) is a paid web-based project management tool.

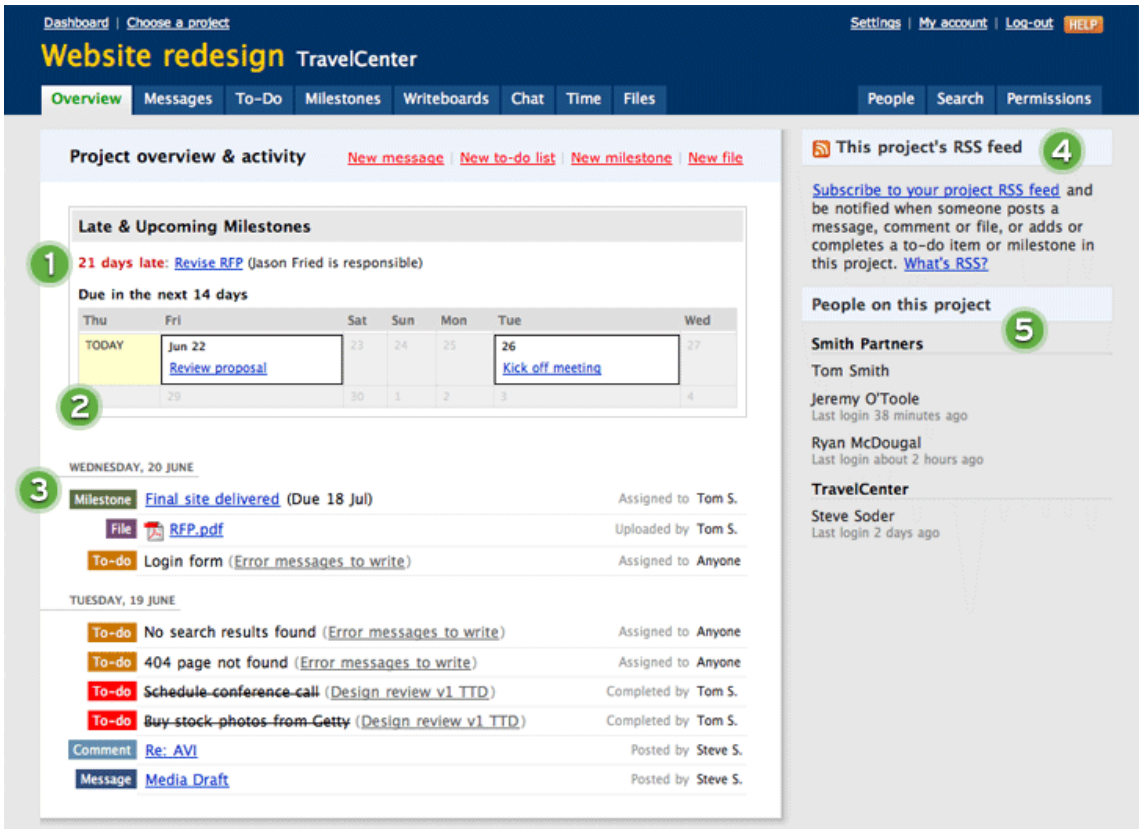

Figure A.4: Screenshot of the engineering management tool Basecamp, showing: 1) Due items in red at the top, 2) A calendar marking due items in the next 14 days, 3) A log of the project's recent activity, 4) The RSS feed for the project, and 5) Who's logged in and when (extracted from http://www.basecamphq.com).

Collaboration/Awareness

The awareness features of Basecamp are: **"File sharing"**, allowing users to upload files, categorize, sort, track versions, and share deliverables; **"To-do lists"**, where users can add to-do items, assign them to somebody and make the lists private or public; **"Milestones"**; and **"Writeboards"**, web-based text documents (with the possibility of being exported to HTML) which saves documents versions history.

Other awareness features include those provided by the plugins **Beanstalk**, **Telegraph**, and **Springloops**. The first is a hosted Subversion system that makes easier to setup, browse, and track Subversion. The second handles RSS and Atom feed subscriptions for Basecamp projects. The third is a SCM tool focused on web development teams.

Collaboration/Communication

Basecamp includes the following communication features: in **"File sharing"** it's possible to define who receives an e-mail with a notification when a new file is uploaded; in **"To-do-lists"** it's viable to send an e-mail notification to who is responsible; in **"Writeboards"** it's possible to send a version to an e-mail address; **"Messages"**, which

can be organized in categories, have comments associated, be defined as private and associated with a milestone, and trigger the sending of an e-mail with a notification when a new message or comment is posted; **"Message Boards"**, a blog; **posting messages, milestones and to-do lists to a Basecamp account via e-mail**, achieved by the plugin Mailmanagr; and finally a **chat** – a Campfire feature – where project's stakeholders can enter the project's chat room and chat with other users.

Integration

Basecamp can be integrated with various plugins, which include: **Fixx**, a bug tracking and issue tracking system; **Harvest**, a web-based time tracking for small businesses or team, based on client services; and **Campfire**, a plugin of the team collaboration tool developed by 37 signals – the same company that developed Basecamp – that permits users to upload images (which latter appear in the chat) and to see the history of a conversation.

A.4.2 Gantt Project

Gantt Project is a free desktop-based project management tool.

Collaboration/Awareness

Gantt Project possesses built-in and plugin-related awareness features.

Gantt Project allows users to **work with project files stored on a WebDAV server**, although preventing two or more users from editing the same file simultaneously.

In terms of plugins, **GanttProject Integrator** allows easy control of multiple Gantt project files in a corporate environment, while **ganttweb** provides a web representation of a GanttProject generated XML file's directory.

Integration

Gantt Project's integration is limited and comprises only data interoperability, including **XML (import and export)**, **Microsoft Project** files, **HTML and PDF reports**, and **CSV**.

A.4.3 Intellisys Project

Intellisys Project is a paid desktop-based project management tool.

Collaboration/Awareness

Intellisys Project's awareness features are two: **team calendar** and **team timeline**.

Collaboration/Communication

Intellisys Project provides a **forum** – the only application to do so – and **e-mail** integration.

Integration

Intellisys Project integrates only with **Microsoft Project**.

A.4.4 Microsoft Project

Microsoft Project is a paid desktop-based project management tool launched in 1985.

Collaboration/Awareness

The awareness features of Microsoft Project are a **team calendar** and **RSS feeds**. Microsoft Project also integrates with **CollabNet TeamForge** collaboration tool.

Collaboration/Communication

Microsoft Project's communication features include: **e-mail integration**; **adding notes** to a task, resource or project; and **creating a workspace site** for a project.

Integration

Like other Microsoft's products, Microsoft Project integration is achieved primarily with other Microsoft applications. Project integrates with: **Microsoft Excel and Visio**, used by Project's "Visual Reports" to produce PivotTable views, charts, graphs, and diagrams; **Microsoft Outlook**; and **Microsoft Windows SharePoint Services**, which allows sharing and managing documents of Microsoft Projects.

Microsoft Project also integrates with **Intellisys Project** and **Project KickStart**.

A.4.5 Project KickStart

Project KickStart is a paid desktop-based project management tool launched in 1992.

Collaboration/Awareness

Project KickStart's awareness features are minimal and include **issue tracking** and the **interlinking of documents** within specific projects.

Collaboration/Communication

The communication features of Project KickStart include: **email integration** (e.g. when a user sends a report, an e-mail in sent), **adding notes to tasks**, and **project tasks/Gantt status reports**, useful to answer questions like "Is the project on schedule/budget?" or "Who is working on what?".

Integration

Project KickStart integrates with: **Microsoft PowerPoint**, allowing users to export a project plan to PowerPoint; **Microsoft Outlook**, where users can import Outlook contacts and export a project plan into Outlook; **Microsoft Excel**; **Microsoft Word**; **Microsoft Project**; **ATC!**, a contact and customer manager software; **WBS Cart**, which display a

project using a tree-style diagram known as a Work Breakdown Structure (WBS) Chart; **MindManager**, which makes possible both creating mind maps and sending them to Project KickStart for assigning and scheduling, or creating Project KickStart plans and then visualize them in MindManager; and **Milestones Professional**, a tool that lets users – after exporting a project plan – to create taskbars, add titles, view task durations and percent completions, and more.

A.4.6 Rally Enterprise

Rally Enterprise is a paid web-based tool (with a project management module).

Collaboration/Awareness

Besides project management activities – which by nature are awareness activities – the only awareness features of Rally Enterprise's Project Management module are **SCM** and **RSS feeds** (which sends notifications to users).

Collaboration/Communication

What Rally Enterprise's Project Management module lacks in awareness features, it compensates in communication features, including: **instant messaging**; **customization of user's personal home page** to track progress, get alerts and view reports; and **sending notifications to users** via Rally and email.

Integration

Rally Enterprise integrates with several tools, including: **Eclipse** and **Microsoft Visual Studio Team Foundation Server** IDEs; **Microsoft Visual Studio Team Foundation Server** and **Subversion** SCM; **Mylyn** collaboration tool; **AccuRev**; and finally **Bugzilla**, **Fitnesse**, **HP Quality Center**, **JIRA**, and **Rational ClearQuest** test and defect management tools.

A.4.7 Trac

Trac is a free web-based project management and bug-tracking tool.

Collaboration/Awareness

Trac's awareness features include: **VCS integration** (GitPlugin, PerforcePlugin, Subversion, TracMercurial, TracDarcs, and Trac-Bzr); **GanttPlugin**; **Trac Timeline**, which provides a historical view of the project in a single report; **Trac Changeset Module**, a built-in functionality for visualizing changes (*diffs*) to files; **TracTickets**, a tracking system of issues and bugs within a project; **RSS feeds** (for subscribing to project events, be notified about important issue tickets, and be updated with changes to a specific file or directory); and **Mylyn**.

Collaboration/Communication

The communication features of Trac resumes to the possibility of **annotating a bug**, once a ticket has been entered into Trac.

Integration

Trac integrates with **DoxygenPlugin**, a plugin that integrates doxygen (a source code documentation generator tool) documentation, and with **TestCaseManagement**, a test case management plugin that uses Subversion as the test-case repository and also the ticket framework in Trac as a means to create test runs.

A.4.8 Wrike

Wrike is a paid web-based project management tool launched in 2006.

Collaboration/Awareness

Wrike's awareness features include: **editing a shared task**, such as changing the duration and uploading files; **automatic building of reports**; **automatically communicated changes in the project plan** to team members via e-mail notifications, iCalendar and RSS feeds; users' **to-do lists**; viewing **users' and projects' activity**; **collaborative web workspace**; and Wrike's **Intelligent Email Engine**, which allows a user to share a task with his teammates, after sending an e-mail with the task to wrike@wrike.com, thus adding it to its to-do list and to the project plan.

Integration

Wrike's integration with other tools include: **Microsoft Outlook**, **Microsoft Word**, **Microsoft Excel**, **Microsoft PowerPoint**, **BlackBerry**, and **Gmail**.

A.5 Requirements Tools

A.5.1 Borland Caliber Analyst

Borland Caliber Analyst is a paid desktop-based requirements tool.

Collaboration/Awareness

Borland Caliber Analyst – a Borland application comprised of two products for defining and managing software requirements, Borland Caliber DefineIT and Borland CaliberRM, respectively – has the fewest awareness features of all the analyzed tools, which are: a **centralized repository** and **change-propagation** mechanism between Caliber DefineIT and CaliberRM (a change in one afterwards is merged with the other).

Collaboration/Communication

Borland CaliberRM offers **discussion groups** for scenarios/requirements.

Integration

In the case of Caliber DefineIT, there is integration at the application life cycle level through: **Borland Silk**, **Mercury Quality Center**, **Visual scenarios and requirements synchronization**, **Test case generation and tracing**, **Requirement to UML/BPMN transformation**, **Borland StartTeam**, and **Borland Together** design tool. At the client's applications level, there is integration solely with **Eclipse** platform.

In the case of CaliberRM, there is integration at the application life cycle level through: **Borland Silk**, **Mercury Quality Center**, **Visual scenarios and requirements synchronization**, **Test case generation and tracing**, **Borland StartTeam**, and **Borland Together** design tool. At the client's applications level, there is integration with: **Microsoft Visual Studio Team System**, **Eclipse** platform, **Borland Delphi**, **Borland JBuilder**, **Web and Windows** clients, and **Microsoft Visual Studio 2005**.

A.5.2 Contour

Contour (see Figure A.5) is a paid web-based requirements tool.

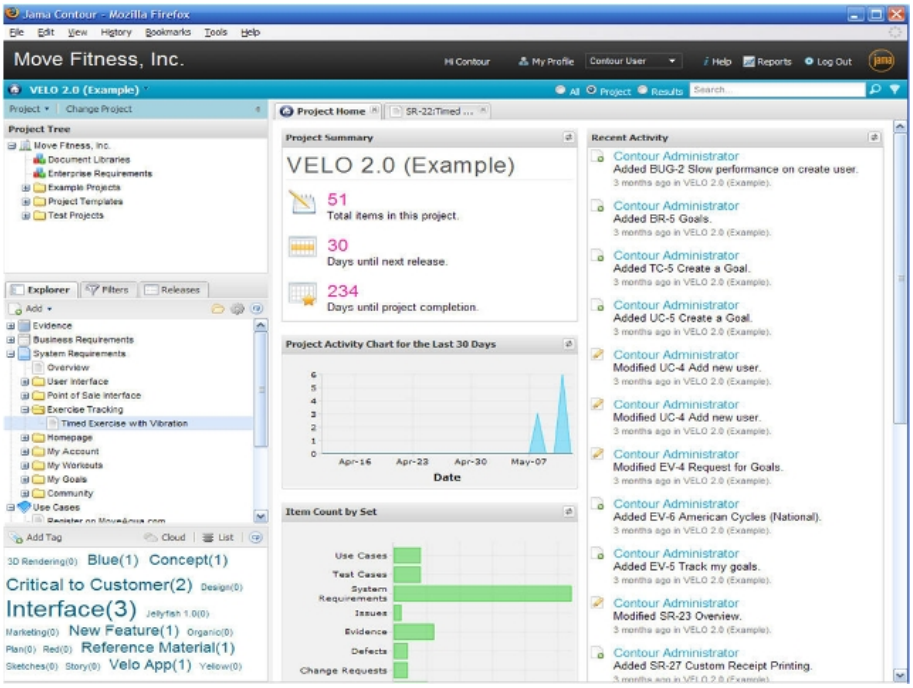

Figure A.5: Screenshot of the requirements tool Contour (extracted from http://www.jamasoftware.com/contour).

Collaboration/Awareness

Contour's awareness features include: seeing immediately **"who and what" is affected by the daily tasks** of a user; **e-mail notification of groups of requirements** and other items; **unlimited reviewer user accounts** for collaboration across large teams; **displaying recent work on projects**; tracking the complete **version history of requirements** and other items; and a **dashboard**.

Collaboration/Communication

Contour's communication features includes the possibility of initiating **team discussions within artifacts** and also **adding comments to items**.

Collaboration/Collective Knowledge

Contour is the only tool that possesses a collective knowledge feature, which is **tagging**.

Integration

Contour integrates with multiple browsers (**Mozilla Firefox, Internet Explorer, Safari, and Google Chrome**) and **Lightweight Directory Access Protocol (LDAP)**, an application protocol for querying and modifying directory services running over TCP/IP.

A.5.3 Rational RequisitePro

Rational RequisitePro is a paid desktop-based/web-based requirements tool.

Collaboration/Awareness

Rational RequisitePro's awareness features include: a user can **work on a document offline** and then, when online, **merge the document with the previous version**; support for **review and markup of a Microsoft Word document**, if the user uses the dynamic Word integration, and **generation of static Microsoft Word documents** for review purposes, if the user doesn't use the dynamic Word integration; **access control at the document, requirement, attribute and attribute value levels**; support for **email-enabled discussion groups** to track new or changed requirements; **discussion groups**, allowing users to submit proposed changes prior to committing them to the repository; and **Web access** for viewing, authoring and managing requirements, as well as project administration activities.

Integration

Rational RequisitePro integrates with: **Rational ClearQuest** (a SCM software); **Rational Software Architect** (a model-driven development tool); **Rational Software Modeler** (a collaborative platform for visual modeling and design); **Rational Application Developer** (an extension for Eclipse); **Rational Data Architect** (a collaborative data design solution to discover, model, relate, and standardize diverse and distributed data assets); **Rational Systems Developer** (a model-driven development tool for software products and systems development); **WebSphere Business Modeler** (a business process modeling software); **WebSphere Integration Developer** (a user-friendly authoring environment for end-to-end integration in the user's Service-oriented Architecture (SOA)); **Rational SoDA** (a project-wide documentation automation tool); **Rational Unified Process** (a process framework that provides industry-tested practices for software and systems delivery); **Rational TestManager** (a tool for test activity management, execution and reporting); and **RavenFlow**.

A.5.4 RavenFlow

RavenFlow is a paid desktop-based/web-based requirements tool launched in 2000.

Collaboration/Awareness

RavenFlow's awareness features include: creation of a corporate **library of validated specifications, accessible via Web**; a **versioning system** for keeping track of requirements changes; **reviewing the text and diagrams** of a requirement; an executive **dashboard** for aiding project managers in finding project risks at the requirements stage; and **submitting comments and issues**. These features are available via RAVEN's **web-based collaboration portal and collaboration server**.

Collaboration/Communication

The only communication feature of RavenFlow is the **storing of all the comments and issues associated with the requirements for auditing and compliance purposes**, through the use of RAVEN's collaboration server.

Integration

RavenFlow's integration with other tools includes: **RequisitePro**, for the elicitation and definition of requirements in RavenFlow and then automatically transferring them into IBM Rational RequisitePro; **Telelogic DOORS**, enabling eliciting and defining requirements in Raven and automatically transfer them into IBM Telelogic Doors; **HP Quality Center Integration**, which populates into Quality Center complete requirements and test cases from RavenFlow; **Microsoft Visual Studio 2005**, enabling authoring and validating requirements in RavenFlow and afterwards transform them as work items into Microsoft Visual Studio 2005 Team Foundation Server; and **Microsoft Word**, integrating directly into Word the requirements from RavenFlow.

It is also possible for a user to **transform requirements' text into a graphical model**, and to **detect missing requirements**, correcting those before inserting the model.

A.5.5 Telelogic DOORS

Telelogic DOORS is a paid desktop-based/web-based requirements tool.

Collaboration/Awareness

Telelogic DOORS's awareness features include: **multi-user concurrent write access to the same document**; **group-reviewing of changes proposed by multiple users**, through DOORS' built-in change proposal system (CPS) and its integration with Telelogic Change; **grouping dependent changes together or creating single changes that affect many requirements**, through the use of CPS; **sending an email to a user when the status of his change proposal is updated**; **Distributed Data Management (DDM)**, a feature that allows portions of the DOORS database to be re fetched, worked on and returned for resynchronization with the database; **participation of users from remote locations** in making changes and creating new requirements directly to the DOORS

database, using Internet or Intranet; and **approval/disapproval of changes either online or by a Change Control Board (CCB)**, with the accepted changes being afterwards promoted into the document or dataset automatically.

Collaboration/Communication

In Telelogic DOORS, users can open **discussion topics** regarding a requirement, and make **comments** about them. The environment is open to Telelogic DOORS desktop and Web clients.

Integration

Telelogic DOORS integrates with: **HP QualityCenter**, for large-scale test environments; **Telelogic Tau**, for traceability between requirements and software models; **Telelogic Focal Point**, to balance what is most important to the customers against resource availability and cost, thereby tracing the requirements to the features; **Microsoft Team Foundation Server**, to create and maintain traceability between requirements in Telelogic DOORS and Team Foundation Server work items in Visual Studio; **Telelogic System Architect**; and **RavenFlow**.

A.6 Collaboration Tools

A.6.1 CollabNet TeamForge

CollabNet TeamForge (see Figure A.6) is a paid collaboration tool.

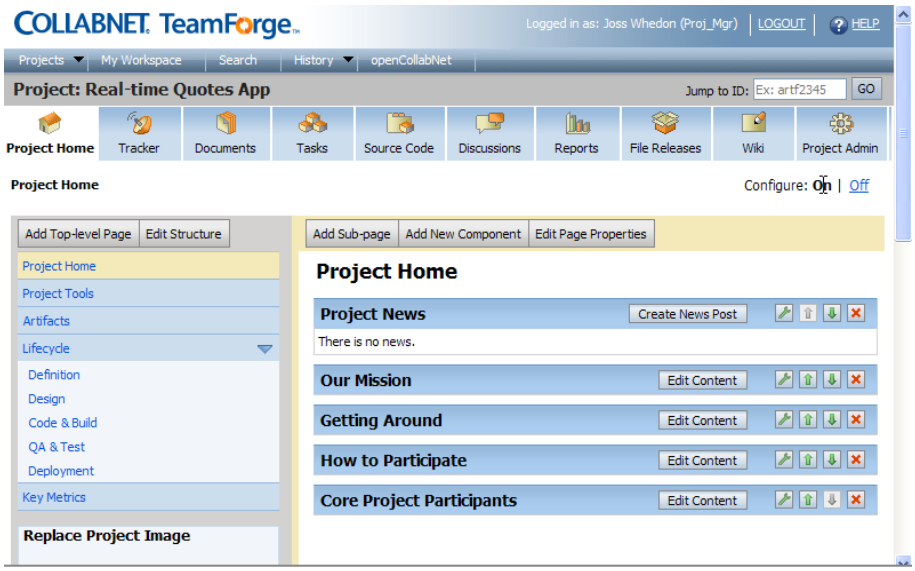

Figure A.6: Screenshot of the collaboration tool CollabNet TeamForge (extracted from http://www.open.collab.net/products/sfee).

Collaboration/Awareness

CollabNet TeamForge includes the following awareness features: **viewing new messages automatically** on the project's homepage for insight across multiple projects; **monitoring of documents, tasks, trackers, releases, and wiki pages for changes**; **managing document reviews** – which can be office documents, image files, and binary files – via Web or email, including comments, approvals, and publication; **file locking**, for preventing undesirable changes on a file being edited; associating discussion posts, wiki pages, and documents with any project item; **revision diffs** of documents and wiki pages ; **Web-based access** to all its collaboration features; **Microsoft Office plug-in** to save artifacts directly from Office applications or any email client; **SCM support** (CVS, Perforce, and Subversion out of the box) **and integration** (IBM Rational ClearCase and Borland StarTeam); **SCM notifications**, with users receiving instant e-mail notifications of changes; automating software build processes with Apache **Ant** integration; **associating artifacts with other objects** for traceability purposes; optional **automated alert emails**, reminding team members when tasks are overdue; **controlling who can create/view/edit documents, discussions, source code, tasks, trackers, and wiki pages**; performing **global, cross-project, cross-application** (across all discussions, documents, source code, releases, wiki pages, news, tasks, and tracker artifacts, including attachments and comments) and **cross-version searches**, through the use of keywords or artifact IDs; **tracking the relationships among different project items**; using **discussion archives** to maintain the evolution of the project and record critical reasons for decisions; and **external access** to view, edit, or manage discussion forums, forum topics/posts, documents, and document folders, achieved through the integration with SOAP SDK and services like DiscussionApp and DocumentApp.

Collaboration/Communication

CollabNet TeamForge's communication features include **discussion forums** and **mailing lists**. In a discussion forum – which can be accessible by Web or email – posts can be attached with media types and discussions can be monitored for new posts, with the option of receiving instant notifications or daily summaries.

Integration

CollabNet TeamForge integrates with: **HP Quality Center**; **CruiseControl**, a continuous integration tool and an extensible framework for creating a custom continuous build process; **CUBiT**, acronym for "Centralized Unified Build, Integration, and Test", a CollabNet TeamForge product allowing users to provision their own systems from a pool of centralized virtual and physical resources based on reusable profiles; **Eclipse**; **Microsoft Visual Studio**; and **Microsoft Project**, which includes the feature of synchronizing task data from a project plan with a CollabNet TeamForge task folder.

A.6.2 EGroupWare

EGroupWare is a paid collaboration tool.

Collaboration/Awareness

EGroupWare's awareness features include: **Task Management module** (for creating tasks and delegating them to team members; linking tasks to other data types such as addresses, contacts, and projects; and getting e-mail notifications and/or popup messages if the status or content of a task has been changed), **Team Calendar module** (for managing own and shared appointments; getting e-mail notifications about any changes in the calendar; managing invitations; allocating resources to an appointment; alarm and reminder functions; linking appointments to other data like contacts, tasks or attached files; and performing full-text search), **Contact Management module** (for creating multiple address books, viewing contacts and organizations by using filters and advanced search, importing/exporting address data material, and filling documents automatically with contact data), **File Manager module** (for providing a team folder, an individual home data area for each user, and direct access to the file manager from the personal computer using WebDAV), **Knowledge Management module** (for publishing information by releasing articles which contain descriptions, links to other information and attached files; and ordering articles by categories), **Wiki "What YouSee Is What You Get" (WYSIWYG) module** (including index based full text search and protect information by user and group rights on page level), **WebDAV support**, **News module** (an intranet information system allowing users to publish news on a group specific level and also to use filter and search options), **Resources module** (including the documentation and management of resources, and ordering resources by categories), **Project Management module** (for visualizing project status/time line in a Gantt chart, creating sub projects, and linking contacts, tasks or other kinds of information to the project), **Time Sheet module** (for managing and auditing working time documentation), and **Tracking system module** (for managing and auditing service, maintenance, change management and other kinds of processes which need a delegation structure and escalation procedures; creating functional areas (*tracker queues*) and types of tasks to be tracked; and linking tracker entries to Task Management, Project Management and Time Sheet modules).

Collaboration/Communication

The only communication feature of EGroupWare is **e-mail**, which includes the possibility of sharing folders, saving e-mails as tasks, managing multiple identities and email accounts, and referring to addresses in contact management for composing emails more quickly.

Collaboration/Collective Knowledge

EGroupWare's assesses collective knowledge through **polls**.

A.6.3 IBM Lotus

IBM Lotus is a family of paid collaboration tools. Despite the various collaboration products Lotus software is composed of, this research focused only on one product: IBM Lotus Notes.

Collaboration/Awareness

Through its integration with IBM Lotus Quickr – a team collaboration and content sharing tool – IBM Lotus Notes provides the following awareness features: **saving e-mail attachments in shared or personal Lotus Quickr places** and afterwards **sending links to Lotus Quickr places**; and **drag-and-drop entire emails with attachments** into Lotus Quickr places.

IBM Lotus Notes also provides **RSS and AtomWeb Feeds** support, which feeds external content, such as blogs, news sites, and podcasts.

Collaboration/Communication

IBM Lotus Notes' communication features include: a **calendar**, displayed in the sidebar; a **scheduling system** that selects meeting times based on the availability of all or just the required attendees and lets users accept or decline meeting invitations, propose alternate times, or delegate attendance to a colleague; **e-mail**, including automatic spell check (highlighting errors as users type), flagging mechanism, definition of follow up dates for when a reply is needed, definition of alarms to remind the recipient of an impending follow up date, signing and encrypting messages to verify sender authenticity and protect confidential data, recalling an email sent in error, matching user input with their most recent contacts, and viewing the inbox by conversation, where threads of related email are grouped together; **contact management**, which includes viewing all recent collaborations with a contact, and maintaining personal and professional data for a contact – including a photograph – and subsequently organizing them into related groups and categories, therefore serving for supplying contact information necessary to initiate e-mail and IM communications; **alerts**, warning senders of incoming email and meeting invitations that the user is unavailable; and finally **sharing documents without having the need to purchase an application**, through the use of the Open Document Format.

A.6.4 Jazz

Jazz is a free collaboration tool.

Collaboration/Awareness

Jazz's awareness features include: **viewing who else besides the user is logged in and what they are working on**; and receiving **automatic notifications of changes, inputs and milestones** that impact the work of the user.

There are two research projects that were initiated for serving as Jazz-plugins.

One was a research involving the use of **3D virtual worlds to support distributed work**. Named **"Bluegrass"**, it was a Jazz plugin that allowed developers to meet online, discuss their work, and collaboratively create designs for their projects. The plugin took Jazz artifacts and represented them in the virtual world. The prototype included chat bubbles as persistent "post-it note" objects that could be used to create diagrams, transportation mechanisms, and visualizations to show presence and crowds in the world [Cheng et al.,].

The other is a research for exploring the use of **multi-monitor environments** to increase awareness, using Jazz's hooks and listeners to obtain the information that the visualizations of the project's state need [Marcelo Alvim and da Silva,].

Finally, Jazz Integration Architecture provides a support for several IBM Rational applications, including: **Rational Team Concert**, a collaborative work environment with work item, source control and build management, process management and iteration planning; **Rational Quality Manager**; and **Rational Requirements composer**, which provides textual and visual requirements definition, tags and comments support in a collaborative environment.

Collaboration/Communication

Jazz's provides project-integrated presence and messaging, including a **chat** and a **blog**.

Integration

Jazz integrates with **Microsoft Visual Studio**, **Eclipse**, and a significant number of Rational software, including Application Developer, AppScan, Asset Manager, Build Forge, ClearCase, ClearQuest, Functional Tester, Method Composer, Performance Tester, Policy Tester, RequisitePro, Self-Check, Software Analyzer, and Software Architect.

A.6.5 Lighthouse

Lighthouse is a paid collaboration tool.

Collaboration/Awareness

Lighthouse's awareness features are, in the context of a Ticket (Lighthouse's equivalent to a task): adding **comments**; defining **who's responsible** for the ticket; associating **milestones** to tickets and defining the **person responsible for the milestone**; **changing the ticket's state** (such as "new" or "resolved"); **finding tickets according to pairs of keywords** and respective values, such as "watched:me", which displays the tickets being watched by the logged-in user – by default it uses all of the words that are entered in the search; **filtering tickets with a few common custom searches**: "Tickets I'm watching" (which shows all the tickets the currently logged-in user is watching), "Todays tickets" (which displays all the tickets created on a given day), "Assigned to me" (which presents all the tickets assigned for the current user), "All Tickets" (which finds the most recently updated tickets), "Open Tickets" (which finds all tickets that haven't been resolved) and "Closed Tickets" (which finds all tickets that have been resolved, put on hold, or marked as invalid); a pop open **ticket history** window, showing the tickets the logged-in user is working on; and **"watching" a ticket**, which notifies the user – either when at the creation stage (by e-mail) or when the ticket is already created (in the application) – of all the changes made to the ticket.

Lighthouse also possesses: a **dashboard** (showing the opened, resolved, and week's tickets; the scheduled, unscheduled, and completed milestones; the messages posted by the logged-in user; the comments made by other users to the user's messages; the homepages created by the user; and the number and percentage of tickets the user

completed); **RSS feeds**; and SCC integration – **Subversion** – through Beacons plugin, allowing users to modify their ticket properties from a revision log.

Collaboration/Communication

Lighthouse's built-in communication features include a **"Messages" module, a type of microblog per-project**. Other communication features result from its integration with Beacons plugins and include: **e-mail** integration, where users can get notifications in their inboxes or even update tickets directly from the e-mail; and a **team calendar**, as a consequence of its integration with Google Calendar.

Collaboration/Collective Knowledge

The sole collective knowledge feature of Lighthouse is **creating and searching tags**.

A.6.6 Mylyn

Mylyn is a free collaboration tool launched in 2005.

Collaboration/Awareness

Being a task-focused interface for Eclipse, Mylyn's primary awareness feature is **task-context**. By saving users' interaction in tasks and allowing users to share the tasks with other members of the team, Mylyn reduces the time and effort spent on sending emails or instant messages.

Mylyn monitors the work activity on the users' tasks to identify information relevant to the task at hand, creating a task context, i.e. the **set of all artifacts related to his task** (e. g. documents the user browsed, methods he edited, APIs he referred to, etc.). When the user finishes the task and returns afterwards, Mylyn shows all the interaction related to the task.

Other awareness features, also related to tasks, are **scheduling and degree-of-interest model** [Kersten, 2007].

Scheduling defines a time period the users have to complete a given task.

In the case of the degree-of-interest model, task context management is based on the idea that each element in the system is weighted according to its relevance to the task at hand. The more the user interacts with the element, the higher its degree of interest to the task becomes. When the degree is high, a bookmark is implicitly created.

Mylyn also connects with other tools, including **Codebeamer** and **Tasktop** plugins. Codeamer gives users access to **bug/issue trackers in a project**, and SCC **Bugzilla, Trac and JIRA**, if users desire to work with tasks stored in one or more task repositories and not to store tasks locally in his workspace. Tasktop allows users to **track the documents and web pages** that are relevant for each task, integrates with **calendars** and **email** (Outlook and Gmail), automates **time tracking**, provides facilities to **adjust and upload task times to a JIRA or Bugzilla server**, and makes easier the **installation of connectors** to access JIRA, Rally or CollabNet.

Mylyn helps the user on working with source repositories by **managing change sets automatically**. Once a task is activated, a change set for the task is added and any changes

made when working on the task are added to the change set. Incoming changes made by team members show up grouped by task.

Since Mylyn saves not only the changes made during a task but also all the interactions, when the user turns over the task to a team member, all the actions he made, such as documentation he consulted, will be available and serve as a good starting point. The need to send an e-mail to detail the steps a user took is no longer necessary.

If the user is using a task repository connector that supports attachments, he can share task context by attaching it to a bug report.

Collaboration/Communication

Mylyn's only communication feature is users' possibility on adding **comments** to tasks.

A.6.7 Socialtext

Socialtext (see Figure A.7) is a paid desktop-based/web-based collaboration tool launched in 2001.

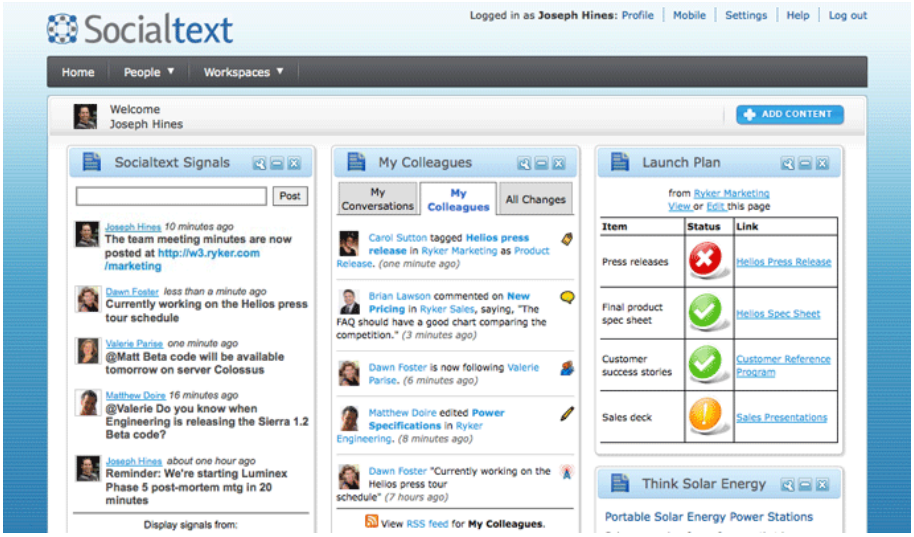

Figure A.7: Screenshot of the collaboration tool Socialtext (extracted from http://www.socialtext.com).

Collaboration/Awareness

Socialtext's features are included in four modules: **"Socialtext People"**, **"Wiki workspaces"**, **"SocialCalc"**, and **"Socialtext Dashboard"**.

In "Socialtext People", each person has a profile page, with their **contact information and skills**, a list of their **friends and followers**, and the **workspaces** the person belongs to. On each profile it is also displayed the **most recent actions** (edits, comments, and tags) performed by the person and their list of friends – people the person is following – and a directory providing a list of all the members in the person's company. By adding a user to his network of followed users, a person can read the user's comments from a widget on the dashboard or from "Socialtext Desktop", a desktop application for using Socialtext.

Information from Active Directory or an LDAP directory can be automatically populated into "Socialtext People".

In "Wiki workspaces", and like most wikis, there is a **revision history**, which gives an historical context on the users actions. It is possible to **write a document in rich text (WYSIWYG) or wiki text**. When a person is creating an e-mail, it's possible to have the contents of the e-mail posted to a workspace or create a new wiki page automatically by putting the new page name in the subject of the e-mail. **Feeds** (for pages or and colleagues activities updates), configurable **alerts** (e.g. 'Watch' feature enables the person to receive an email when the page is updated), and the possibility of **working offline or on a mobile** are other relevant wiki features. Users can download pages while they have web access, edit them while offline, and then synchronize changes back to the server when he comes back online.

In "SocialCalc", a person can **publish a spreadsheet that everyone updates** on a workspace page, each person updating a different segment. The "SocialCalc" audit trail shows who made what changes and gives the ability to **roll back to an earlier revision**. Integrated **notifications** alerts others when spreadsheets are added or updated. **Authoring, linking, and searching of documents** are also provided.

"Socialtext Dashboard" is customizable, allowing users to **drag-and-drop standards-based widgets**. At any time, it is possible to change the widgets included in the dashboard and their position. Socialtext pre-built widgets include: **"My Conversations"**, which shows changes others have made to any workspace page authored, edited, or commented on by the logged-in user; **"My Colleagues"**, which shows recent updates made by people subscribed by the logged-in user; **"Workspaces"**, showing workspaces the user accessed, as well as their activity metrics; **"Workspace Tags"**, a tag cloud of all the tags on pages in the context of a particular workspace; and **"All People Tags"**, a tag cloud of all tags on people registered in Socialtext.

Since Socialtext supports the Google OpenSocial standard, any widget that is written to the standard can be added to the "Socialtext Dashboard" module. Over 100.000 widgets support the Google OpenSocial standard, including widgets to access and display information from Outlook, Google Calendar, Twitter, MapQuest, Wikipedia, Del.icio.us.

Socialtext also provides **RSS feeds support**. Feeds are automatically generated for every page, tag, search query, watchlist and workspace, and can be used for tracking micro-blogging messages and also for monitoring the weblog conversations from a dashboard's widget.

Collaboration/Communication

Socialtext provides communication features in **"Socialtext Desktop"** and **"Collaborative Weblogs"** modules.

"Socialtext Desktop" is built on Adobe AIR and is available for Windows, Mac and Linux. This application has two main features: **"Signals"** and **"Activity"**.

"Signals" are a king of **social messaging/micro-blogging** – where messages are 140-characters long – and has three goals: sharing links and information, providing a "Questions and Answers" mechanism, and sharing information on what the user is doing and where he is.

"Signals" integrates also with **social networking**, which allows users to follow other users and thus serve as a filter, showing the messages of the user's most important stakeholders.

"Activity" has two goals: **monitor page updates and social networking** events; and keep current on the **activities of user's colleagues**.

In Socialtext, weblogs are intended to facilitate dynamic conversations among colleagues. Regarding a weblog post, users can **search, add in-line comments, and view the full history of all the revisions** of a post. An update to a weblog post triggers notifications to those who subscribe to the workspace, to the person or the page. A new weblog post can be added via **e-mail** or with the synchronization of changes made to a post when the user was offline.

Collaboration/Collective Knowledge

Socialtext provides the users – in "Socialtext People", "Wiki workspaces", and "SocialCalc" modules – the opportunity of **tagging**. The posts of a weblog can also be tagged, with a **tag cloud** being provided.

Integration

Socialtext Desktop integrates with **Adobe AIR**, while Socialtext integrates with **Microsoft SharePoint** and **Lotus Connections** through connectors.

In the case of Microsoft SharePoint, users can: publish the final version of a workspace page as a Microsoft Word document stored inside of SharePoint; link to files in the SharePoint repository from any Socialtext workspace page; and perform single sign-on access to both the SharePoint server and Socialtext workspaces.

In the case of Lotus Connections, all members of the Lotus community can collaboratively and securely author, review, and share information with each other.

Appendix B

Tools' other characteristics

B.1 Bug Tracking Tools

This section provides additional information to the bug tracking tools' collaboration and integration features studied in section A.1.

BUGtrack

- Free 14-day trial.
- 24/7 Web Access.

Bugzero

- Try-edition, with no time limit but constrained to 100 bug reports and 5 named user accounts.
- Cross database (standard SQL).
- Cross platform (code base written in Java and J2EE).

Bugzilla

- Runs on MySQL and PostgreSQL DBMS.

JIRA

- Free 30-day trial.
- Runs on the following DBMS: SQL Server, MySQL, PostgreSQL, Oracle, DB2, Firebird, Sybase, and HSQLDB.

MantisBT

- Written in PHP.

- Runs on following DBMS: MySQL, Microsoft SQL Server, PostgreSQL, and Oracle (currently at an experimental stage).
- Unlike the majority of the bug-tracking applications studied in this dissertation, it provides wiki integration (DokuWiki, MediaWiki, XWiki, and TWiki) and support for mobile devices (MantisWAP).

B.2 Construction Tools

This section provides additional information to the construction tools' collaboration and integration features studied in section A.2.

Aptana Studio

- Free 31-day trial.
- Windows, Mac, and Linux compatibility.

Eclipse

- Eclipse Public License.
- Windows, Mac OS X, and Linux compatibility.

IntelliJ IDEA

- Free 30-day trial.
- Windows, Mac OS X, and Linux compatibility.

JCreator

- Free 30-day trial.
- Windows (98, 2000, XP, and Vista) compatibility.

KDevelop

- GNU General Public License.

Komodo

- Free 21-day trial.
- Windows, Mac OS X, and Linux compatibility.

Microsoft Visual Studio 2008

- Free 90-day trial.
- Windows compatibility.

NetBeans

- Open-source.
- Windows, Linux, Mac OS X, and Solaris compatibility.

Zend Studio

- Free 30-day trial.
- Windows, Mac OS X, and Linux compatibility.

B.3 Design Tools

This section provides additional information to the design tools' collaboration and integration features studied in section A.3.

With the purpose of avoiding repetition, each UML diagram was mapped to a number:

1. Activity diagram.
2. Class diagram.
3. Collaboration diagram (from UML 1.x).
4. Communication diagram.
5. Component diagram.
6. Composite structure diagram.
7. Deployment diagram.
8. Interaction overview.
9. Object diagram.
10. Package diagram.
11. Sequence diagram.
12. State machine diagram.
13. Timing diagram.
14. Use case diagram.
15. Profile diagram.

Altova UModel

- Compliant with Linux, MacOS, and Microsoft Windows operating systems.
- Free 30-day trial.
- Supports UML 2.2 version and almost all diagrams (1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15).
- BMPN support.

- C#, Java, and VB.NET forward and reverse engineering.
- UML Roundtrip Engineering, which synchronizes code-writing and UML diagrams-revisions activities.

ArgoUML

- Compliant with Mac OS X, Microsoft Windows, and Unix/Linux operating systems.
- UML 1.x version support. Alongside Rational Rose Data Modeler, it is the only of the 13 UML tools studied in this dissertation which provides support uniquely to UML 1.x version.
- As an UML 1.4 compliant tool, ArgoUML can create only a few number of diagrams: 1, 2, 3, 7, 11, 12, and 14.
- C#, C++, Java, PHP4, and PHP5 forward engineering.
- Java reverse engineering.
- Runs on virtually any platform, since it is written entirely in Java and uses the Java Foundation Classes.
- Exports diagrams as GIF, PNG, PS, EPS, PGML and SVG.
- Zooming diagrams.
- "To Do" list, to help designer keep track of their tasks
- Checklists, made specific to the selected design element (e.g., Class, Attribute, Operation, Association).
- BSD License.

Borland Together

- Free 15-day trial.
- UML 2.x support.
- Customizable template-based document generation, capable of assembling content from all model types and requirements.
- BPMN support.
- Model Driven Architecture (MDA) features, including OMG's Query View Transformation (QVT).
- Domain Specific Language (DSL) toolkit, which increases modeling's value by using domain-specific languages that users can create and deploy.
- Automating design and code reviews, including audits and metrics at the model and code level.

BOUML

- Open-source.
- UML 2.0 support, including diagrams 1, 2, 3, 5, 7, 9, 11, 12, and 14.
- C++, Idl, Java,PHP, and Python forward engineering.
- C++, Java, PHP reverse engineering.
- Unix/Linux/Solaris, MacOS X, and Windows compatibility.
- "Project control", a tool that manages applicative write access to the project files – at most one user per file, to avoid having to merge modifications made by several users – without having to depend on OS capabilities.
- "Project synchro", a tool associated to "Project control" that allows users to synchronize a repository project with the changes made by several users.
- Use Case wizard, that when applied to a use case allows to specify textually a summary, context, pre-conditions, description, port-conditions and exceptions.

EclipseUML

- Free 30-day trial.
- UML 2.1 support, including almost all diagrams (1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14).
- Database Reverse Engineering.
- "Documentation view" feature, where each UML element has its own documentation saved in the UML metamodel.
- MDA support.
- Zooming, exporting and printing diagrams.

Enterpise Architect

- Free 30-day trial.
- UML 2.1 support, including almost all diagrams (1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14).
- BPMN support.
- C#, C++, Delphi, Flash ActionScript, Java, PHP, Python, VB.Net, Visual Basic forward engineering.
- Java and VB.Net reverse engineering.
- Word compatible documentation for post-editing and linking into Word Master Documents.

- Linking rich text documents to model elements and editing them directly using built-in Rich Text editor.
- Output in rich text format.
- Windows and Linux compatibility.
- Traceability reporting.
- Database modeling.
- Support for MDA Transformation, Baseline, Requirements Management, Testing, and Project Management.

Gliffy

- UML 2.x support, including diagrams 1, 2, 3, 5, 7, 9, 11, 12, and 14.
- BPMN support.
- Platform-independence and no software to install.
- Free (permanently) and paid accounts.
- Extensive shape library.
- Exports diagrams as JPG, PNG, and SVG pictures.
- Printing diagrams.
- Management of users and workspaces.
- Uploading Web images.

Magic Draw

- Free trial.
- UML 2.x support, including diagrams 1, 2, 4, 5, 6, 7, 8, 11, 12, and 14.
- BPMN support.
- C++, Corba IDL, Java, .NET, and WSDL forward and reverse engineering
- Generating reports in normal text, RTF, HTML, Spreadsheet template (which need to be saved as HTML format) and XML template (DocBook or FO) formats.
- During reverse engineering, Javadoc style comments in code are collected and stored as documentation of the model element.
- C style comments in code are collected as documentation of the model element.
- Runs in any Java 5 or 6 compatible virtual machine.
- Estimation tools, Model Driven Development (MDD), and Requirements tools support.

- Open Application Programming Interface (API).
- Exporting diagrams as bitmap (JPEG, PNG) or vector (TIFF, EMF, WMF, EPS, SVG) images.
- Model and diagram search engine.
- Adding hyperlinks to any model element.
- Retrieving database structure via a JDBC connection.

Microsoft Visio

- Free 30-day trial.
- UML 2.x support, including diagrams 1, 3, 5, 11, 12, and 14.
- BPMN support.
- SQL Server 2005 or 2008 database forward engineering.
- Database modeling diagrams with reverse engineering of any Open Database Connectivity-compliant data source.
- Web site mapping and documentation, including auto-generated Web site maps.
- Diversified set of supported diagrams, including: Directory services, Engineering, ITIL, Logical network, Network rack, Pivot, Sample, Building, Space and floor plans, Workflow shapes (3-D), Timelines and calendars, and Organization charts.
- Wizards for generating diagrams from existing data.
- Support for PDF and XPS files.
- Microsoft Windows XP with Service Pack (SP) 2, Windows Server 2003 with SP1, or later operating system compatibility.

Rational Rose Data Modeler

- Only UML 1.x support, including diagrams 2, 5, 7, 11, 12, and 14.
- Services Oriented Development of Applications (SODA) integration.
- Printing diagrams.
- Windows, HP, Unix, and Linux compatibility.

StarUML

- UML 2.0 support, including diagrams 1, 2e, 3, 5, 6, 7, 11, 12, and 14.
- BMPN support.
- C#, C++, and Java forward and reverse engineering.
- Windows compatibility.
- Plug-in Architecture Customizable Code Generation.
- MDA support (UML profiles and customizable diagrams).
- Open API.
- User-Interface with VS.NET look and feel and Dockable windows.
- Pattern Support: Gang of Four (GoF), EnterpriseJavaBean (EJB), and User-defined patterns.

Telelogic Rhapsody

- Free 30-day trial.
- UML 2.1 compliance.
- Ada, C, and C++ forward engineering.
- Automatic and customizable documentation, synchronized with the implementation.
- Model-driven testing, which identifies design errors early through simulation and requirements-based testing.
- Linux and Windows compatibility.

Visual Paradigm

- Free 31-day trial.
- UML 2.1 support, including diagrams 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14.
- BPMN support.
- Ada, C#, C++, Delphi, Flash ActionScript 3.0, IDL, Java, Objective-C, Object Definition, Perl, PHP 5.0, Python, Ruby, and VB.NET forward engineering.
- Ada 9x, C++, Java, .NET, CORBA IDL, XML, XML Schema, Databases with JDBC, Hibernate mapping files, PHP 5.0, and Pyhton reverse engineering.
- Exporting reports in PDF and DOC formats.
- Project publisher.

- Ad Hoc report creation.
- Rich text.
- MacOS X and Windows compatibility.
- Support for Requirements Management, Mind Mapping, Database Modeling (e.g. Entity relationship diagrams (ERD) and Object-relational mapping (ORM) diagrams), and Design Patterns (e.g. GoF).
- "Animacian", a package of animation features, such as animation of sequence diagrams.
- "Model Transitor", a useful set of features for making transitions between diagrams, such as tracing the origin of a diagram element.
- Intuitive GUI, such as Flexible zooming, which can zoom to a selected region.
- Exporting diagrams as JPG, PNG, SVG and EMF image files.
- Microsoft Windows (98, 2000, XP, or Vista), Linux, Mac OS X, Solaris or all other Java-enabled platforms compatibility.

B.4 Engineering Management Tools

This section provides additional information to the engineering management tools' collaboration and integration features studied in section A.4.

Basecamp

- Free 30-day trial.

Basecamp

- Open source.
- Windows, Linux and MacOSX compatibility.

Intellisys Project

- Windows, Mac and Linux compatibility.

Microsoft Project

- Free 60-day trial.
- Microsoft Windows XP with Service Pack (SP) 2, Windows Server 2003 with SP1, or later operating system compatibility.

Project KickStart

- Free trial.
- Windows Me / NT / 2000 / XP / Vista compatibility.

Rally Enterprise

- Free trial, with limited features.

Trac

- BSD license.

Wrike

- Free 30-day trial.

B.5 Requirements Tools

This section provides additional information to the requirements tools' collaboration and integration features studied in section A.5.

Borland Caliber Analyst

- Free 30-day trial.
- Windows 2000 Pro / XP Pro / Server 2003 compatibility.

Contour

- Of all the requirements tools analyzed, Contour is the only one to provide a totally Web based interface, more precisely a Web 2.0 (Ajax) interface.
- Free 14-day trial.

Rational RequisitePro

- Fully functional free 15-day trial.
- Windows 2003 Enterprise SP1, Windows 2003 Enterprise SP2, Windows 2008, Windows Server 2003 Enterprise Edition SP2, Windows XP Professional SP2, Windows Vista Business, Windows Vista Enterprise, and Windows Vista Ultimate compatibility.

RavenFlow

- Fully functional free 30-day trial.
- RAVEN SQA module, which automatically generates tests from requirements.
- Microsoft Windows compatibility.

Telelogic DOORS

- Linux, UNIX, and Microsoft Windows compatibility.

B.6 Collaboration Tools

This section provides additional information to the collaboration tools' collaboration and integration features studied in section A.6.

CollabNet TeamForge

- Free 30-day trial, limited to 25 users.
- Spans **Configuration Management**, **Development**, and **Engineering Management** Software Engineering areas.

EGroupWare

- Fully functional free 30-day trial.
- Spans **Configuration Management** and **Engineering Management** Software Engineering areas.

IBM Lotus

- Free 90-day trial.

Jazz

- Spans **Requirements**, **Design**, and **Configuration Management** Software Engineering areas.

Lighthouse

- Free account, limited to one project and two users.
- Spans **Configuration Management** and **Engineering Management** Software Engineering areas.

Mylyn

- Eclipse Public License.
- Spans **Configuration Management** Software Engineering area.

Socialtext

- Free 30-day trial.

Appendix C

Features analysis

The features (see Figure C.1) implementation order was made by dividing, on each feature, the sum of the **Value**, **Innovation** and **Preference** values by the number of estimated days that would be needed for the feature to be implemented. The higher the value, the greater was the priority for implementing the feature.

H, *M* and *L* are the diminutives for *High*, *Medium* and *Low*, respectively. In the calculus, each was mapped to an integer – *High* to 3, *Medium* to 2 and *Low* to 1.

In the **Innovation** column, the percentage of tools (represented between 0.0 an 1.0) that didn't support the feature was computed and converted into *H* if the percentage was greater than 0.66, *M* if the percentage was between 0.33 and 0.66, and *L* if the percentage was less than 0.33.

		Evaluation											
		Time (T)	Value (V)	Innovation (I)								Preference (P)	(V + I + P) / T
				Bug Tracking	Construction	Design	Engineering Management	Requirements	Collaboration	Percentage	Scale		
Features	**Blog**	1	H	0	0	0	1	0	2	0.98	H	H	9.00
	Comments	-	M	2	3	1	5	3	4	0.70	H	-	-
	Contact Management	3	L	0	0	0	0	0	3	1.00	H	M	2.00
	Dashboard	7	M	1	3	0	3	1	4	0.83	H	L	0.86
	E-mail	2	M	5	4	1	6	3	5	0.60	M	M	3.00
	Feeds	1	H	2	2	0	5	0	3	0.81	H	H	9.00
	File sharing	5	M	0	2	3	2	0	1	0.85	H	M	1.40
	Folder sharing	15	M	0	1	5	0	1	2	0.85	H	L	0.40
	Forum	10	M	0	2	0	1	3	2	0.87	H	L	0.60
	Group Calendar	3	M	0	1	1	2	0	4	0.91	H	M	2.33
	Instant Messaging/Chat	2	H	1	3	0	2	0	2	0.87	H	M	4.00
	Microblog	3	H	1	0	0	0	0	1	0.98	H	H	3.00
	Online whiteboard	-	H	0	0	0	0	0	0	1.00	H	-	-
	Real-time editor	15	H	0	3	2	0	1	1	0.87	H	L	0.47
	Recommender	5	H	0	0	0	0	0	0	1.00	H	H	1.80
	Screen sharing	-	L	0	0	0	0	0	0	1.00	H	-	-
	Social bookmarking	2	M	0	0	0	0	0	0	1.00	H	M	3.50
	Social cataloguing	3	M	0	0	0	0	0	0	1.00	H	H	2.67
	Social networking	5	H	0	0	0	0	0	1	1.00	H	H	1.80
	Tags	1	H	0	1	1	0	1	3	0.94	H	H	9.00
	Team Timeline	14	L	0	0	1	2	0	0	0.94	H	L	0.36
	To-Do list	2	L	0	0	1	2	0	0	0.94	H	M	3.00
	Version Control	-	H	5	9	8	3	2	5	0.43	M	-	-
	Video conferencing	-	M	0	2	0	0	0	0	0.96	H	-	-
	VoIP	-	H	0	1	0	0	0	1	0.98	H	-	-
	Voting	5	M	1	1	0	0	2	1	0.91	H	M	1.40
	Wiki	-	H	1	1	0	1	0	3	0.94	H	-	-

Figure C.1: Analysis of collaboration features.

Appendix D

Developed prototype

This appendix complements section 6.6 by showing more screenshots of the developed prototype.

Blog

Figures D.1 and D.2 show the content of the Blogs tab (which provides links for the Blog and Microblog plugins) and a comment for the post, respectively.

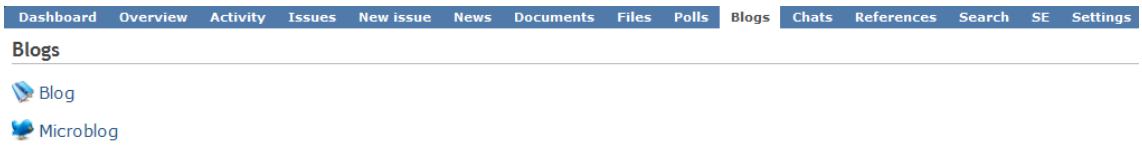

Figure D.1: Screenshot of the Blogs tab.

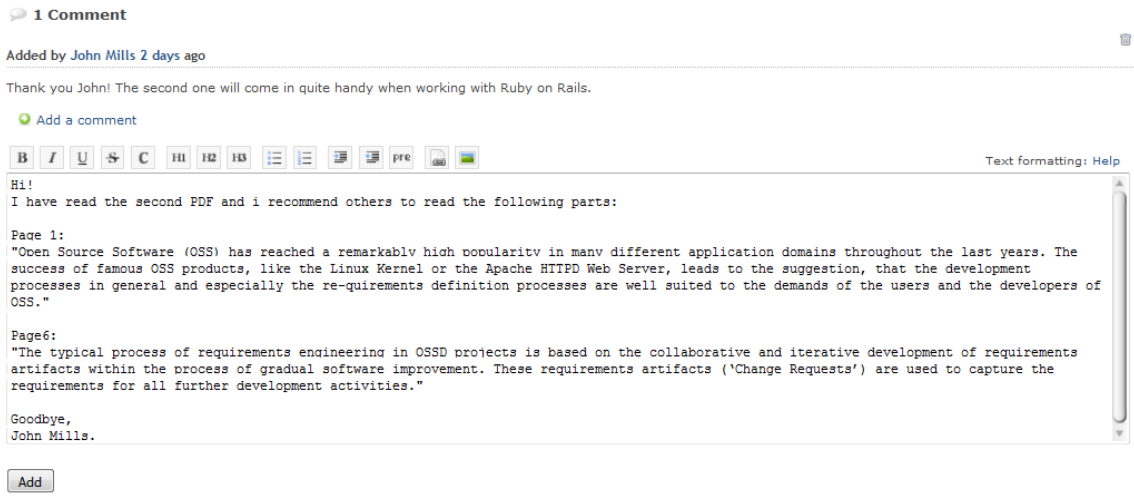

Figure D.2: Screenshot of a comment for the blog's post.

Dashboard

Figure D.3 shows the dashboard after dragging and dropping some widgets (see the widgets' initial positions on Figure 6.10). The colors of the moved widgets are subdued.

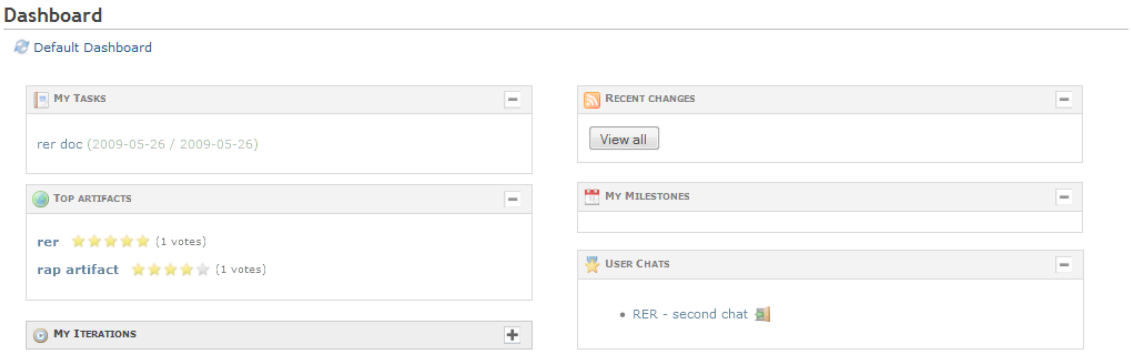

Figure D.3: Screenshot of the Dashboard plugin with moved widgets.

E-mail

Figure D.4 shows the content of a received message (see Figure 6.11).

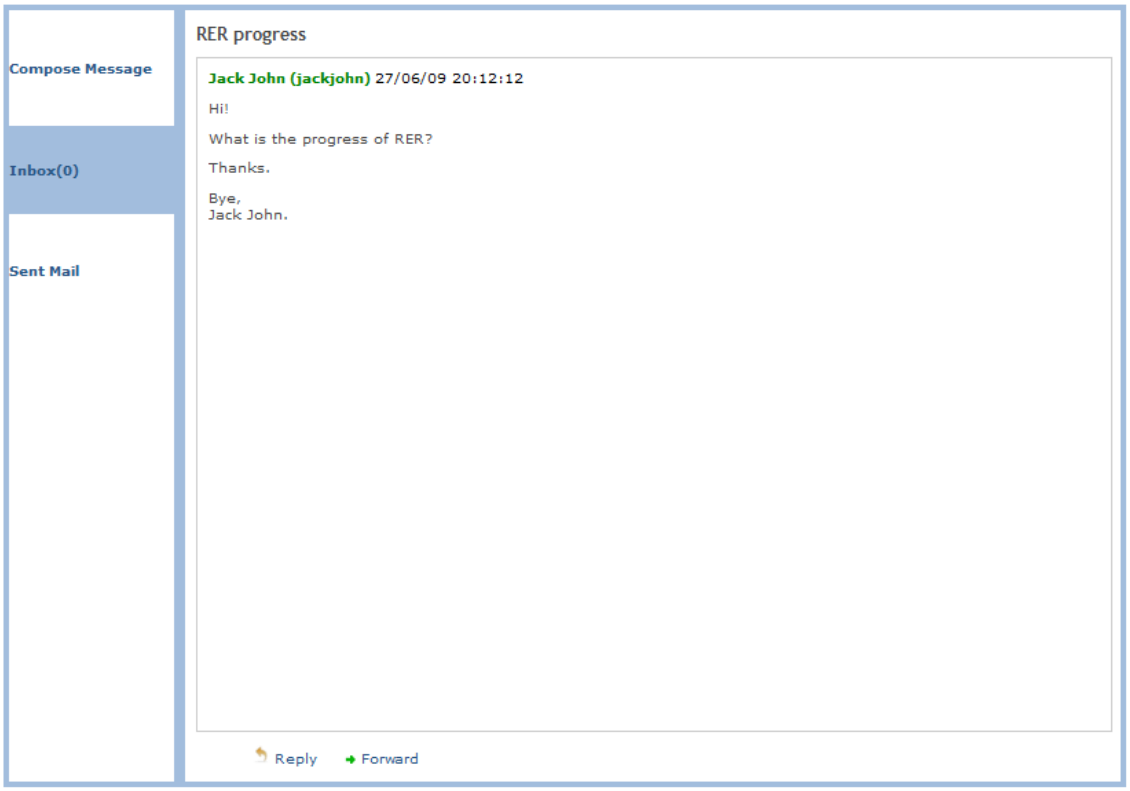

Figure D.4: Screenshot of the E-mail plugin with the content of the received message.

IM/Chat

Figures D.5 and D.6 show the content of IM/Chat plugin and the creation of a new chat room, respectively. In the second figure, the widget's elements selected for the chat have blue background.

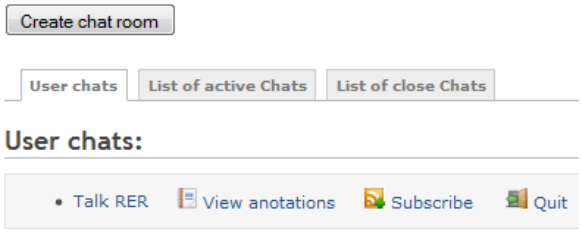

Figure D.5: Screenshot of the IM/Chat plugin.

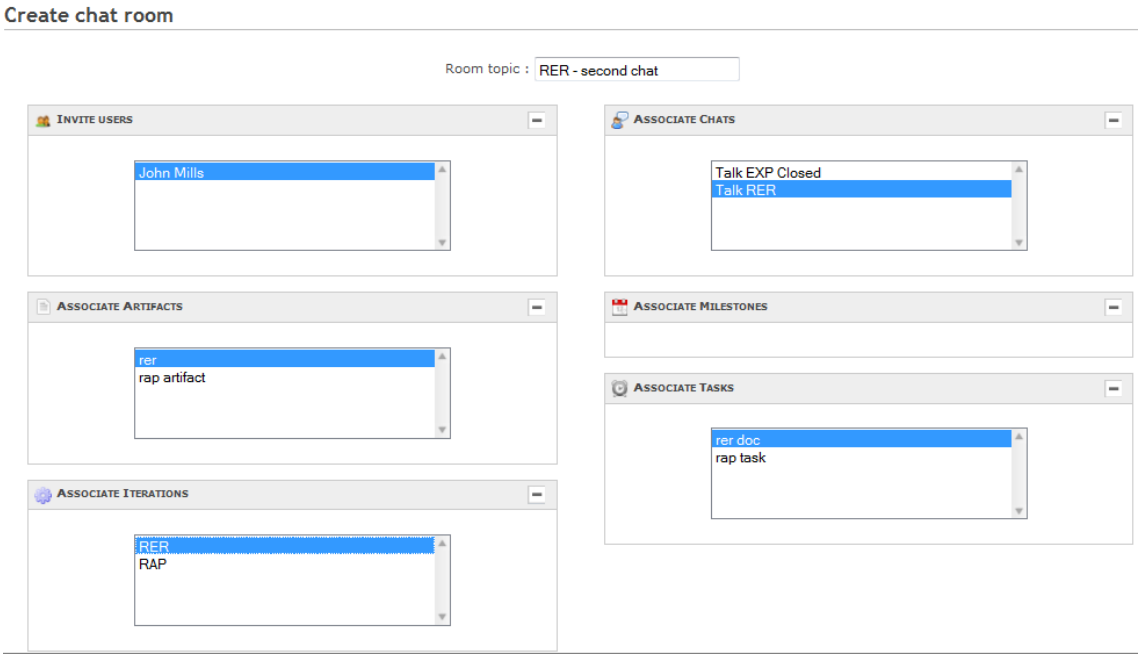

Figure D.6: Screenshot of the creation of a new chat room.

Microblog

Figure D.7 shows the characters left to use while writing a new micropost.

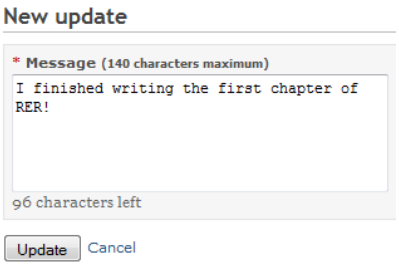

Figure D.7: Screenshot of the characters left to use while writing a new micropost.

Recommendations

Figure D.8 shows more information concerning the recommended bookmark of Figure 6.17. This figure, which was obtained by clicking on the blue colored information icon located next to the hyperlink of the recommended bookmark, shows the notes written by one user (Jack John) that possesses a bookmark with the same URL.

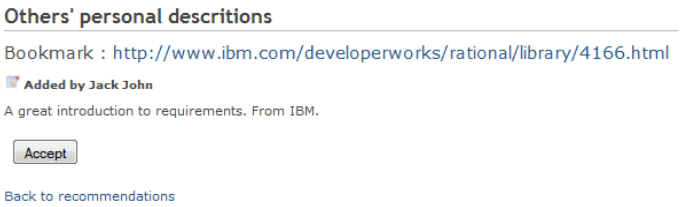

Figure D.8: Screenshot of more information on a recommended bookmark.

Search

Figures D.9 and D.10 show the two previously executed steps for searching bookmarks belonging to users with “John” on its name (see the final step on Figure 6.18).

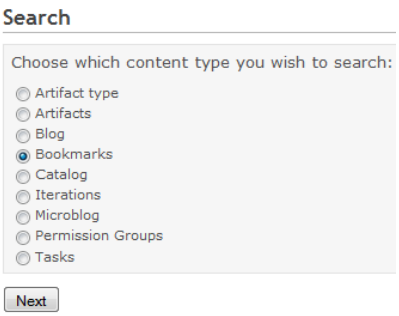

Figure D.9: Screenshot of choosing bookmarks as the content type of a search.

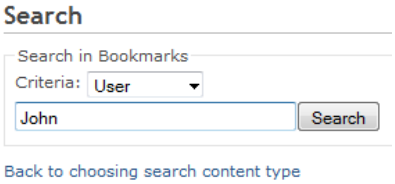

Figure D.10: Screenshot of choosing to search for bookmarks by its owner.

Social bookmarking

Figure D.11 shows the content of the References tab, which provides links for the Social bookmarking and Social cataloging plugins.

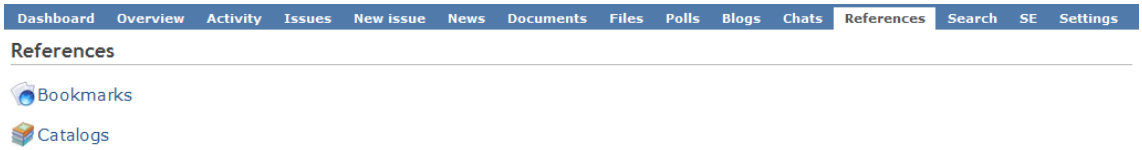

Figure D.11: Screenshot of the References tab.

Software Engineering

Figures D.12, D.13, D.14, D.15, D.16, D.17 and D.18 show the content of the Software Engineering tab, the listing of the project's iterations, the details of the iteration named "RER", all the project's current tasks, all the project's artifacts, the details of an artifact, and the permissions for artifact types associated with the permission group named "RER", respectively.

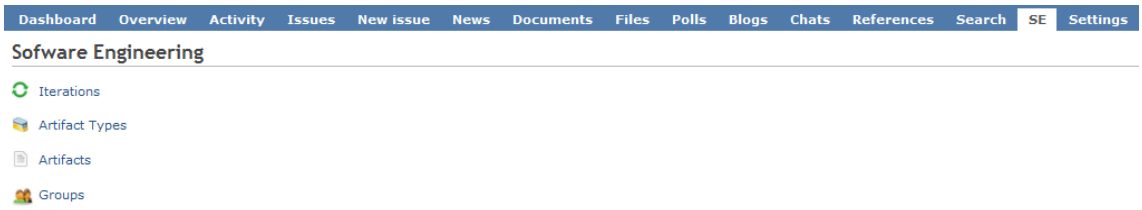

Figure D.12: Screenshot of the Software Engineering tab.

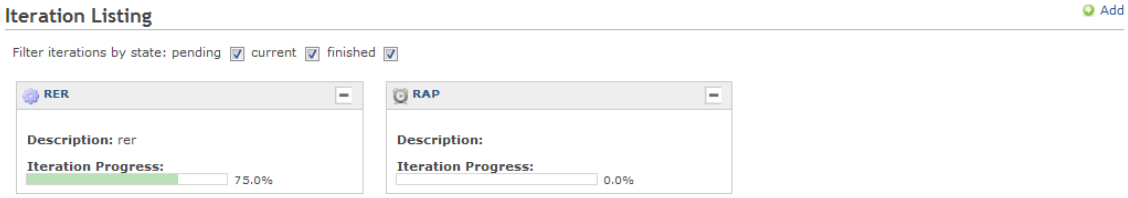

Figure D.13: Screenshot of the listing of the project's iterations.

Figure D.14: Screenshot of the details of the iteration named "RER".

Figure D.15: Screenshot of all the project's current tasks.

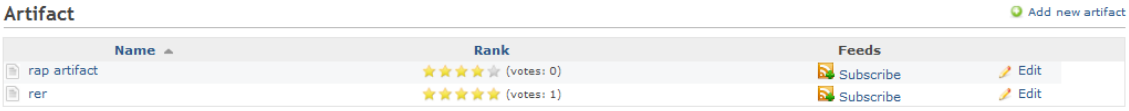

Figure D.16: Screenshot of the all the project's artifacts.

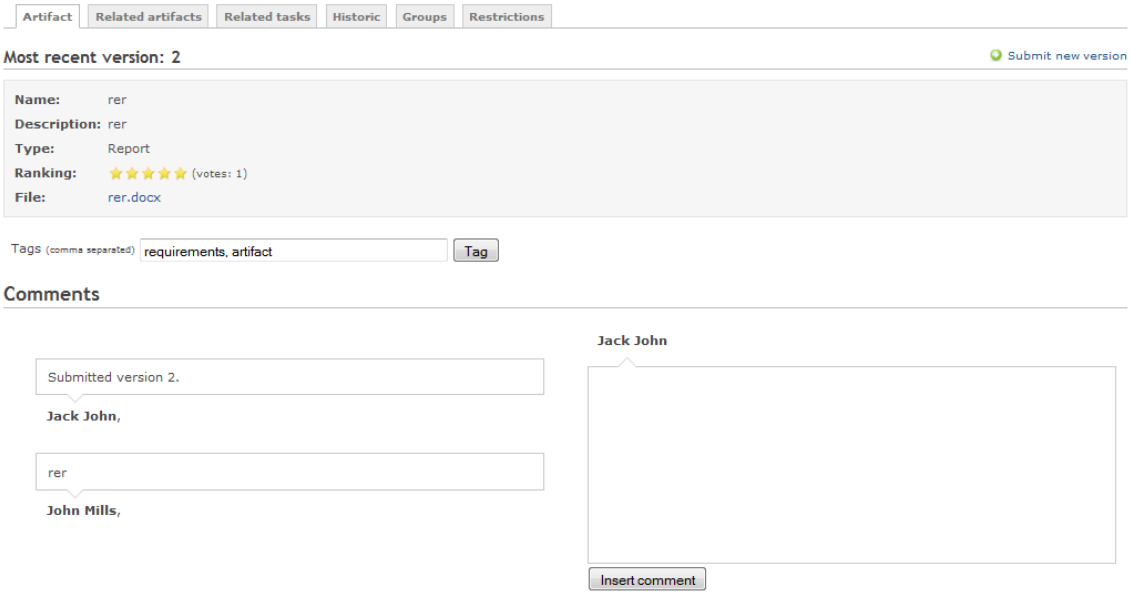

Figure D.17: Screenshot of the details of an artifact.

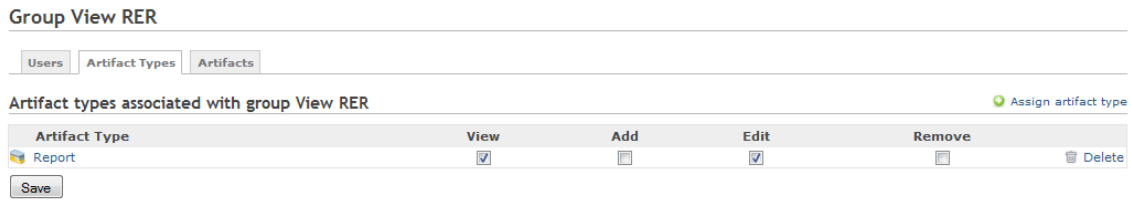

Figure D.18: Screenshot of the permissions for artifact types associated with the permission group named “RER”.

Social networking

Figures D.19, D.20 and D.21 show the results of searching for users with “john” in its name, the users “John Mills” is following and his followers, respectively.

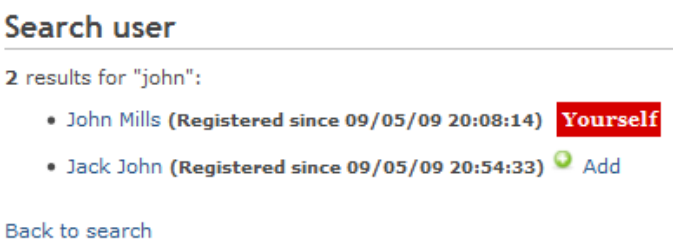

Figure D.19: Screenshot of searching the user with “john” in its name.

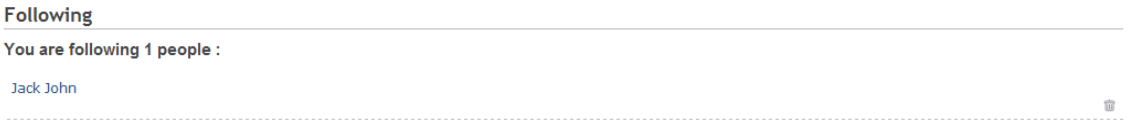

Figure D.20: Screenshot of users followed by the logged-in user.

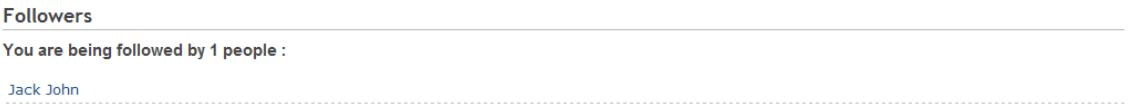

Figure D.21: Screenshot of users following the logged-in user.

Tags

Figures D.22 and D.23 the tagcloud's tags (see Figure 6.23 to view the tagcloud's initial stage) beginning with "re" and all the artifacts, tasks, posts, bookmarks and catalog entries tagged with "requirements".

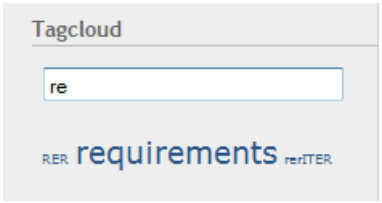

Figure D.22: Screenshot of tags in the project's tagcloud beginning with "re".

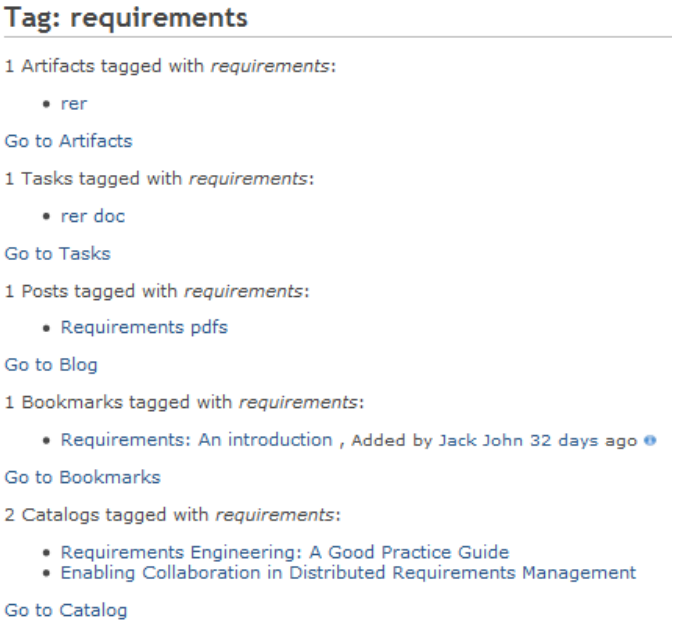

Figure D.23: Screenshot of the artifacts, tasks, posts, bookmarks and catalog entries tagged with "requirements".