

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Software Knowledge Management using Wikis: a needs and features analysis

Rafael Araújo Pires

FINAL VERSION

Report of Dissertation

Master in Informatics and Computing Engineering

Supervisor: Ademar Manuel Teixeira de Aguiar (PhD)

Co-Supervisor: João Carlos Pascoal de Faria (PhD)

March 2010

**Software Knowledge Management using Wikis:
a needs and features analysis**

Rafael Araújo Pires

Report of Dissertation
Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Maria Cristina Ribeiro (PhD)

External Examiner: Isabel Ramos (PhD)

Internal Examiner: Ademar Manuel Teixeira de Aguiar (PhD)

March 25, 2010

Abstract

Knowledge Management is a sub-section of management that pursues improvement in business performance. It increases the organization's ability to learn, to innovate, to solve problems and aims to enhance organizational knowledge processing. It is looking for the available and required knowledge assets that are divided as tangible and intangible. Tangible assets, which are related to explicit knowledge, include manuals, customer information, news, employees, patent licenses, proposals, project artifacts, etc. Intangible assets, which are related to tacit knowledge, include skills, experiences and knowledge of employees within the organization. It is a method that simplifies the process of sharing, distributing, creating, capturing, and understanding a company's knowledge.

The challenges involved to implement knowledge management in organizations include technological, organizational and individual issues.

Knowledge management systems (KMS) are described as IT-based systems developed to support and to enhance the organizational processes of knowledge creation, storage/retrieval, transfer and application. KMS are fundamental to implement a knowledge management initiative.

With the spread and growth of wikis, they soon have become used in various environments, such as education (teaching and distance learning), business (employees share knowledge) and engineering (tool for project management). Although they are already very useful, a better support for knowledge management in software projects can be achieved.

This dissertation identifies needs and features of a wiki that will provide a good solution for a software knowledge management. They are based on literature reviews and on identified problems from some organizations. The focus is not to find the necessities of a specific software development model, but to take the common parts among them and improve the way how they are managed. The necessities identified can be implemented through features, providing the necessary functionalities for successful software knowledge management. The features described were compared with the 25 most popular wikis in order to verify if they are already implemented. Looking at the analysis, in 19 wikis are found 14 features implemented, which represents half of them. It means that more than half of the wikis analyzed implement more than half of the features. On the other hand, there are 9 features that are only implemented by 5 or less wikis and 2 features are not implemented. It was also possible to confirm the idea that wikis by themselves, with the features provided by default, already implement a considerable part of the functionalities needed.

Resumo

A gestão de conhecimento é uma vertente da gestão que procura a melhoria no desempenho de um negócio. Aumenta a capacidade de aprender, de inovar e de resolver problemas. Ambiciona a melhoria do processamento de conhecimento numa organização. Procura o conhecimento disponível e necessário, que pode ser classificado como tangível e intangível. Os aspectos tangíveis do conhecimento, que estão relacionados com conhecimento explícito, incluem manuais, informação proveniente de clientes ou colaboradores, notícias, patentes, propostas, etc.. Os aspectos intangíveis, que estão relacionados com o conhecimento tácito, incluem competências, experiência e conhecimento dos funcionários da organização.

Foram identificados os desafios envolvidos na implementação da gestão de conhecimento nas organizações. Estes incluem níveis como o tecnológico, organizacional e individual.

Os Sistemas de Gestão de Conhecimento (SGC) são necessários para fomentar a prática da gestão de conhecimento. Os SGC são descritos como sistemas baseados em tecnologias de informação, para suporte e melhoria dos processos de criação, armazenamento, recuperação, transferência e aplicação de conhecimento numa empresa.

Com a disseminação e crescimento das wikis, estas rapidamente se tornaram úteis em diversos campos. São disso exemplo a educação, tendo utilidade no ensino à distância; as empresas, para partilha de conhecimento entre funcionários, e na engenharia, como ferramenta na gestão de projectos. Apesar de serem já bastante úteis, é possível atingir um melhor apoio na gestão de conhecimento para desenvolvimento de software.

Esta dissertação pretende identificar as necessidades e funcionalidades de uma wiki capaz de fornecer uma solução para a gestão de conhecimento na área da engenharia de software. O estudo baseia-se numa revisão da literatura e em problemas identificados em algumas organizações. O objectivo não é encontrar as necessidades de um modelo específico de desenvolvimento de software, mas identificar as necessidades comuns entre eles, visando melhorar a forma como são geridos. As necessidades identificadas serão cobertas pelas funcionalidades posteriormente definidas. As funcionalidades descritas foram procuradas nas 25 wikis mais populares. Através da análise, verificou-se que em 19 das 25 wikis, foram implementadas pelos menos 14 das funcionalidades estabelecidas. Isto significa que mais de metade das wikis analisadas implementam mais de metade das funcionalidades identificadas. Por outro lado, 9 funcionalidades foram implementadas apenas por 5 ou menos wikis, sendo que 2 das funcionalidades não são implementadas. Foi também possível confirmar que, por si só, e atendendo às funcionalidades inerentes a uma wiki, esta garante já uma grande parte das funcionalidades necessárias.

Acknowledgements

I would like to thank my supervisors, Professor Ademar Manuel Teixeira de Aguiar and Professor João Carlos Pascoal de Faria, for sharing their wisdom.

A special thank is due to Pedro Veloso Gomes, for monitoring this work and for sharing his experience.

I owe special thanks to my Strongstep colleagues, for their trust and support.

I wish to express my gratitude to my parents, grandparents and my brother, for all their support and comprehension, as well as all my friends for their patience and friendship.

Finally, a special thanks to Daniela, for her encouragement, support, and care.

Thank you all
Rafael Araújo Pires

**“Knowledge Management is expensive,
but so is stupidity”**

Davenport

Contents

1 Introduction.....	1
1.1 Context.....	1
1.2 Motivation and Objectives.....	1
1.3 Dissertation Structure.....	2
2 Knowledge Management.....	3
2.1 Introduction.....	3
2.2 What is knowledge?.....	3
2.3 What is Knowledge Management?.....	6
2.3.1 Reviewing technocratic studies.....	9
2.3.2 Reviewing behavioral studies.....	10
2.4 Organizational Learning - OL.....	12
2.4.1 Kolb's - Experiential learning.....	12
2.4.2 Argyris and Schön - Double-loop learning.....	13
2.4.3 Wenger's - Communities of practice.....	13
2.4.4 Nonaka and Takeuchi's - Knowledge spiral.....	15
2.4.5 Experience Factory Organization – EFO.....	16
2.5 Implementing Knowledge Management.....	17
2.6 Knowledge Management Systems – KMS.....	19
2.6.1 Knowledge management software.....	21
2.6.2 Knowledge management systems' architectures.....	25
2.6.3 Wikis.....	28
3 Problem: Software Knowledge Management.....	33
4 Software Knowledge Management using Wikis.....	38
4.1 Features.....	38
4.2 Wikis and implemented features	52
5 Conclusions and Future work.....	55
5.1 Goals achieved.....	55
5.2 Future work.....	56
References.....	57

List of Figures

Figure 2.1 - Knowledge evolution cycle, from (Rus et al, 2002)	7
Figure 2.2 - Kolb's model for experiential learning, adapted from (Atherton, 2009).....	12
Figure 2.3 - Single and Double-loop learning architecture, adapted from (ProvenModels, 2010)	13
Figure 2.4 - Why focus on communities of practice?, adapted from (Wenger, 2006).....	14
Figure 2.5 - Knowledge spiral, from (Nonaka et al, 1995).....	15
Figure 2.6 - Flow of information through the Experience Factory Organization, from (Rus et al, 2002).....	16
Figure 2.7 - Architecture of a centralized KMS, from (Maier, 2006).....	25
Figure 2.8 - Architecture of server and peer, from (Maier, 2006).....	27
Figure 2.9 – Wiki Design Principles, from (Cunningham, 2010).....	31
Figure 4.1 – Prototype for the login feature.....	39
Figure 4.2 - Prototype of the user profile.....	40
Figure 4.3 - Prototype for the blog feature.....	40
Figure 4.4 - Prototype for the rich text feature while creating a new article in a blog.....	41
Figure 4.5 - Prototype for the version control feature.....	42
Figure 4.6 - Prototype of the contributions of a user	43
Figure 4.7 - Prototype of the repository feature.....	43
Figure 4.8 - Prototype of one community of practice.....	44
Figure 4.9 - Prototype of a workflow definition.....	45
Figure 4.10 - Prototype of a dashboard with microblogging and widgets.....	47
Figure 4.11 - Prototype of a project section.....	48
Figure 4.12 - Prototype of the user allocation feature.....	49

List of tables

Table 2.1 - Knowledge perspectives and their implications, from (Alavi et al, 2001).....	4
Table 2.2 - Knowledge taxonomies and examples, from (Alavi et al, 2001).....	6
Table 2.3 - Schools of knowledge management, from (Dingsøy et al, 2009).....	8
Table 2.4 - Main findings for technocratic studies, adapted from (Bjørnson et al, 2008).....	9
Table 2.5 - Main findings for behavioral studies, adapted from (Bjørnson et al, 2008).....	11
Table 2.6 - Categories of knowledge management software: summary table, from (Carvalho, 2006).....	24
Table 2.7 - Type of KMS and type of KM initiative, from (Mayer, 2006).....	28
Table 2.8 - Types of wikis, adapted from (Grace, 2009).....	29
Table 2.9 - Conversational technology overview, from (Wagner, 2004).....	29
Table 4.1 - Traceability matrix between needs and features.....	51
Table 4.2 - 25 most popular wikis (WikiMatrix, 2010).....	53
Table 4.3 - Wikis and implemented features.....	54

Abbreviations and acronyms

BI	Business Intelligence
BPR	Business Process Re-engineering
CBR	Case-Based Reasoning
CI	Competitive Intelligence
CMM	Capability Maturity Model
CMS	Content Management Systems
CSCW	Computer-Supported Cooperative Work
DBMS	Database Management System
EFO	Experience Factory Organization
ICT	Information and Communication Technologies
IS	Information Systems
IT	Information Technology
KM	Knowledge Management
KMS	Knowledge Management Systems
OL	Organizational Learning
OLAP	Online Analytical Processing
P2P	Peer-to-Peer
SDLC	Software Development Life Cycle
SE	Software Engineering
SECI	Socialization, Externalization, Combination and Internalization
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability Determination
SWEBOK	Software Engineering Body of Knowledge
WYSIWYG	What You See Is What You Get

1 Introduction

1.1 Context

This work is conducted within the Master in Informatics and Computing Engineering at Faculty of Engineering of Porto University. It aims to study the knowledge management in organizations that develop software using an already useful tool, wikis.

This topic of dissertation appears within the organization where I have been working in. It is a software quality consultancy company that regularly faces the challenges of software development companies. Sometimes those challenges are related to problems that can be solved with better knowledge management solutions. It was in this context that the idea of studying software knowledge management emerged. Concerning to the wiki component of this dissertation, I believe that wikis, with their inherent features, largely satisfy the needs to manage knowledge in software engineering, but they can and must be improved. Thus, it is appropriate to evaluate its applicability and to build what's missing on top of it.

1.2 Motivation and Objectives

Software engineering is complex and implies many people working together in different phases, activities and places. The work involved is dynamic with new technologies supporting it rising very fast. Knowledge is created every day and with the emergence of software in several sectors of activity it covers several domains. It is hard for organizations to keep track of all that knowledge, trying to know “the what”, “the where” and “the who” about knowledge. A good management of this knowledge is crucial for the efficiency and success of organizations and that's here where my motivation appears.

The aim of this dissertation is to identify the necessities and features of a wiki that will provide a good solution for a software knowledge management. It is important to refer that the

Introduction

necessities and features identified will cover the needs related to knowledge managements in software engineering and not the support for all the processes of software development. This necessities will appear from literature analysis and from challenges identified with the costumers of the company where I am working in.

This dissertation will address the following topics:

- Necessities for software knowledge management.
- Features that covers all the necessities identified.
- An analysis of the most popular wikis relating their features and the ones defined in this dissertation.

1.3 Dissertation Structure

Besides the Introduction, this dissertation contains 4 more chapters.

On chapter 2, Knowledge Management, is described the meaning of knowledge and knowledge management. The most common models of organizational learning are also addressed in this chapter, as well as the issues and solutions of knowledge management initiatives. To finish the chapter, knowledge management systems are described. Then, in chapter 3, Problem: Software Knowledge Management, are introduced some challenges that software organizations are facing in the knowledge management context. The necessities for a successful software knowledge management are also shown here. Moving forward to the chapter 4, Software Knowledge Management using Wikis, the features necessary to achieve the necessities of software knowledge management are listed. An analysis of the most popular wikis regarding the features defined is also included. The last chapter, number 5, Conclusions and Future work, presents the conclusions and next steps to be followed.

2 Knowledge Management

2.1 Introduction

This chapter is dedicated to the clarification of concepts about Knowledge Management and Knowledge Management Systems. The related studies described in literature are also focused here.

2.2 What is knowledge?

In the Oxford English Dictionary, “Knowledge” is defined as (i) “expertise, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject”, (ii) “what is known in a particular field or in total; facts and information” or (iii) “awareness or familiarity gained by experience of a fact or situation” (Soanes et al, 2005).

Davenport and Prusak in their book “Working knowledge” describe knowledge as “a fluid mix of framed experience, values, contextual information and expert insights and grounded intuitions that provide a framework for evaluating and incorporating new experiences and informations. It originates and is applied in the minds of the knower. In software organizations, it often becomes embedded not only in documents or repositories, but also in organizational routines, processes, practices, and norms” (Davenport et al, 1998).

Three refinement levels of knowledge can be addressed: data, information and knowledge. *Data* corresponds to unorganized or factual information about events but not its meaning or relevance. When organized, data constitutes useful *information* to users to perform tasks. Through the human factor, a correct comprehension of the existing information contributes to the construction of *Knowledge*. It's not just the comprehension of information, but also the relationship among them. Applying knowledge several times creates *wisdom* (Rus et al 2002). Considering these refinements and the software development environment, it's easy to match the

Knowledge Management

concepts. Data refers to metrics collected from a project, data from lessons learned, etc. Analyzing this data and collecting it from several projects, creates the possibility to build models with information able to be applied into new projects. Looking at the next level, knowledge, it is represented by “best practices” or standards in software engineering.

The data, information and knowledge perspectives have been widely cited in the literature, but there are some alternative perspectives as shown in Table 2.1 (Alavi et al, 2001).

<i>Table 2.1 - Knowledge perspectives and their implications, from (Alavi et al, 2001)</i>			
<i>Perspectives</i>		<i>Implications for Knowledge Management (KM)</i>	<i>Implications for Knowledge Management Systems (KMS)</i>
Knowledge vis-à-vis data and information	Data is facts, raw numbers. Information is processed / interpreted data. Knowledge is personalized information.	KM focuses on exposing individuals to potentially useful information and facilitating assimilation of information	KMS will not appear radically different from existing IS (Information Systems), but will be extended toward helping in user assimilation of information
State of mind	Knowledge is the state of knowing and understanding.	KM involves enhancing individual's learning and understanding through provision of information	Role of IT (Information Technology) is to provide access to sources of knowledge rather than knowledge itself
Object	Knowledge is an object to be stored and manipulated.	Key KM issue is building and managing knowledge stocks	Role of IT involves gathering, storing, and transferring knowledge
Process	Knowledge is a process of applying expertise.	KM focus is on knowledge flows and the process of creation, sharing, and distributing knowledge	Role of IT is to provide link among sources of knowledge to create wider breadth and depth of knowledge flows
Access to information	Knowledge is a condition of access to information.	KM focus is organized access to and retrieval of content	Role of IT is to provide effective search and retrieval mechanisms for locating relevant information
Capability	Knowledge is the potential to influence action.	KM is about building core competencies and understanding strategic know-how	Role of IT is to enhance intellectual capital by supporting development of individual and organizational competencies

Knowledge Management

The knowledge views referred before suggest different perceptions of knowledge management. Each perception invokes a different strategy for managing knowledge. These different strategies for knowledge management and respective systems can be seen in Table 2.1 (Alavi et al, 2001).

Two dimensions of knowledge were defined considering Polanyi's and Nonaka's work. First dimension, *tacit knowledge*, combines technical and cognitive components of an individual. It includes concrete know-how, skills that apply to a specific context and individual beliefs and paradigms. The second dimension, *explicit knowledge*, also known as codified knowledge, is manipulated in symbolic and/or natural language. It consists on information that can be easily transmitted or documented (Polyani, 1967; Nonaka, 1994). Codification of tacit knowledge is very hard, since this knowledge is personal and difficult to transfer, contrasting with the explicit one, which has a formal codification and application and can be easily processed (Nonaka et al, 1995). Some authors suggest that tacit knowledge is more valuable than explicit, and vice versa, but these two dimensions can't be separated. The explicit knowledge is only understood if an individual has the necessary background structures formed, which represents tacit knowledge (Alavi et al, 2001).

Despite tacit and explicit knowledge had been widely recognized in the literature, there are a few more dimension cited. Table 2.2 resume them (Alavi et al, 2001).

It is essential to internalize these types of knowledge. They will affect the way how knowledge is managed and what will be managed. Consequently, the design of knowledge management systems will be also affected (Alavi et al, 2001).

Software engineering has several kinds of activities with different types of associated knowledge:

- Organizational knowledge - activities related to the company or business objectives. This type of knowledge in software companies usually doesn't differ from other companies' knowledge.
- Managerial knowledge - considers planning, staffing, tracking and leading a project.
- Technical knowledge - related to technical activities like requirements analysis, designing, programming, testing, etc.
- Domain knowledge - this knowledge regards to the specific field where the software fits (Rus et al, 2001).

Knowledge requires awareness about the existing knowledge in an organization and the consciousness of the missing and needed knowledge (Agresti, 2000). In fact, knowledge is an important resource in a company, which relevance and meaning depend on the context. It refers to specific places, people and activities. This characteristic is called knowledge *scope* and leads us through individual, group, organizational, multiple organizational or industry-wide dimensions (Rus et al, 2001; Rus et al, 2002).

Knowledge Management

Table 2.2 - Knowledge taxonomies and examples, from (Alavi et al, 2001)

<i>Knowledge Types</i>	<i>Definitions</i>	<i>Examples</i>
Tacit	Knowledge is rooted in actions, experience, and involvement in specific context	Best means of dealing with specific customer
Tacit (cognitive tacit)	Mental models	Individual's belief on cause effect relationships
Tacit (Technical tacit)	Know-how applicable to specific work	Surgery skills
Explicit	Articulated, generalized knowledge	Knowledge of major customers in a region
Individual	Created by and inherent in the individual	Insights gained from completed project
Social	Created by and inherent in collective actions of a group	Norms for inter-group communication
Declarative	Know-about	What drug is appropriate for an illness
Procedural	Know-how	How to administer a particular drug
Causal	Know-why	Understanding why the drug works
Conditional	Know-when	Understanding when to prescribe the drug
Relational	Know-with	Understanding how the drug interacts with other drugs
Pragmatic	Useful knowledge for an organization	Best practices, business frameworks, project experiences, engineering drawings, market reports

2.3 What is Knowledge Management?

Knowledge Management (KM) is a subsection of management that pursues improvement in business performance. It increases the ability to learn, to innovate, and to solve problems and aims to enhance organizational knowledge processing (KMCI, 2007). The Knowledge Management concept in this dissertation will focus on software engineering knowledge processing.

According to Thomas Davenport and Laurence Prusak, KM is defined as “a method that simplifies the process of sharing, distributing, creating, capturing, and understanding a company’s knowledge” (Davenport et al, 1998).

Knowledge Management

Knowledge management is looking for the available and required knowledge assets that are divided in tangible and intangible. Tangible assets, which are related to explicit knowledge, include manuals, customer information, news, employees, patent licenses, proposals, project artifacts, etc.. Intangible assets, which are related to tacit knowledge, include skills, experiences and knowledge of employees within the organization. Although explicit knowledge is generally easier to handle, knowledge management must also deal with tacit knowledge (Rus et al, 2001; Rus et al, 2002).

Agresti and Wiig have defined the knowledge evolution cycle to support a knowledge management strategy. This cycle includes the following phases (Figure 2.1):

- Originate/create knowledge: employees develop knowledge through several activities like learning, problem solving, innovation, creativity and also through importation from outside sources.
- Capture/acquire knowledge: employees acquire information about knowledge through explicit knowledge.
- Transform/organize knowledge: companies organize and transform knowledge through documents and knowledge bases.
- Deploy/access knowledge: distribution through training programs, automated knowledge-based systems or expert networks.
- Apply knowledge: it's the ultimate goal and the the most important phase. It aims to make knowledge available whenever it is needed (Agresti, 2000; Rus et al, 2002; Wiig, 1999).

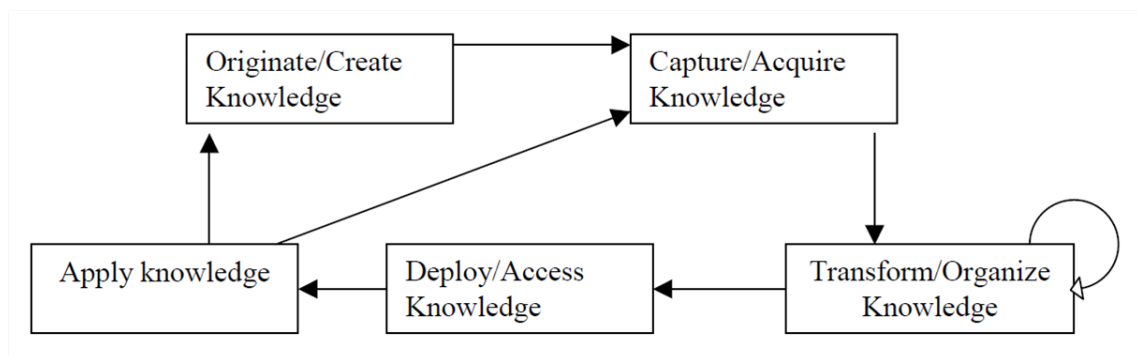


Figure 2.1 - Knowledge evolution cycle, from (Rus et al, 2002)

KM pursues the elevation of the individual knowledge to an organizational level. The accessibility of knowledge allows its spreading in a group and through different organizations, mitigating problems resolution and avoiding duplication of tasks. So, learning is fundamental on KM (Rus et al., 2002). Senge refers that “organizations learn only through individuals who learn. Individual learning does not guarantee organizational learning. But without it no organizational learning occurs” (Senge, 1990).

Knowledge Management

Michael Earl has distinguished work in knowledge management into three categories, technocratic, economic and behavioral. Each one of these categories include schools with a particular focus as shown in Table 2.3 (Earl, 2001).

These schools listed in Table 2.3 can not be seen as competitors, rather the contrary. A good KM strategy must include characteristics from multiple schools (Dingsøy et al, 2009).

Technocratic schools take advantage of information technologies to support the knowledge work. The *systems* school focuses on technology to aim knowledge sharing through knowledge codification into databases or repositories. The *cartographic* school concerns to knowledge maps or directories that improve knowledge exchange. The idea is to make people with expert knowledge accessible to all employees for advice or knowledge exchange. This feature is often called “yellow pages”.

Table 2.3 - Schools of knowledge management, from (Dingsøy et al, 2009)

<i>Category</i>	<i>School</i>	<i>Focus</i>	<i>Aim</i>	<i>Unit</i>
Technocratic	Systems	Technology	Knowledge bases	Domain
	Cartographic	Maps	Knowledge directories	Enterprise
	Engineering	Processes	Knowledge flows	Activity
Economic	Commercial	Income	Knowledge assets	Know-how
Behavioral	Organizational	Networks	Knowledge pooling	Communities
	Spatial	Space	Knowledge exchange	Place
	Strategic	Mind-set	Knowledge capabilities	Business

The *engineering* school addresses processes and knowledge flows ensuring their availability to the organization. Economic category has just the *commercial* school, which relates to the relationship between incomes in organizations and knowledge assets. As example, Dingsøy, Bjørnson and Shull reported the tracking of employee knowledge (intellectual capital) to capitalize on intellectual property rights. Finally, the behavioral category that addresses knowledge sharing management. This category covers three schools: the *organizational* school that focuses on networks (“knowledge communities”) for knowledge sharing inside the organization; the *spatial* school which analyzes the physical space of the office in order to encourage the knowledge sharing; and the *strategic* school which focuses on the definition of the organization strategy throw the value that knowledge can bring to the company (Bjørnson et al, 2008; Dingsøy et al, 2009).

Knowledge Management

As seen, multiple approaches for knowledge management have been made. It is therefore important to check how they were applied in software engineering.

Dingsøy, Bjørnson, and Shull conducted a review of studies in the area of software engineering that can be introduced in these categories of knowledge management. Next is presented a summary of this review.

2.3.1 Reviewing technocratic studies

Starting with the *system* school, it usually evaluates the usefulness of knowledge repositories. Inside software engineering field, engineers have started using repositories even before they became popular. But in some other fields this strategy leads to expensive repositories that finish in “information graveyards”. *Cartographic* school in software engineering has been followed by skills-management tools to help in resources allocation, experts identification and upgrade skills needed. Inside *engineering* school some studies focus on fostering software development processes to manage tacit and explicit knowledge. These studies argue that the processes should be integrated with collaborative social processes to enable the organization learning. Moreover, other studies found a different strategy making central learning processes through project postmortem reviews and retrospectives. In fact, most companies develop software through projects. So the easiest way to start with the organizational learning, is by exchanging experiences across projects.

When software companies decide to go through software process improvement (SPI), this school can be a success factor with the creation of organizational knowledge (Bjørnson et al, 2008; Dingsøy et al, 2009).

<i>School</i>	<i>Concepts</i>	<i>Main findings</i>
Systems	Development of knowledge repositories and initial use	Approach to supporting risk in project management
		Users should be involved in development
		Approach to support design activities
	Use of knowledge repositories over time	Benefits can be realized quickly, tool remains useful over time, and more benefits accrue over time
Cartographic	Use of cartographic system	Tool was used for: allocating resources, searching for competence, identifying project opportunities and upgrading skills
		Tool enabled learning practice at both individual and company level

Knowledge Management

Table 2.4 - Main findings for technocratic studies, adapted from (Bjørnson et al, 2008)

<i>School</i>	<i>Concepts</i>	<i>Main findings</i>
Engineering	Managing knowledge on the software development process	It is feasible to use knowledge management as underlying theory to develop key process areas to supplement the CMM (Capability Maturity Model)
		No matter what knowledge management approach you pursue in SPI, you need to create both tacit and explicit knowledge. Tacit is necessary to change practice, explicit is necessary to create an organizational memory
		A techno-centric approach to SPI may impose unnatural work practices on an organization and fails to take account of how process improvements might occur spontaneously within a community of practice
		It is possible to define and implement software process in a beneficial and cost-efficient manner in small software organizations. Special considerations must be given to their specific business goals, models, characteristics, and resource limitations
	Managing knowledge through formal routines	Formal routines must be supplemented by collaborative, social processes to promote effective dissemination and organizational learning
	Mapping of knowledge flows	Knowledge mapping can successfully help an organization to select relevant focus areas for planning future improvement initiatives
		Casual maps for risk modeling contributes to organizational learning
	Process for conducting project reviews to extract knowledge	Creating a suitable environment for reflection, dialogue, criticism, and interaction is salient to the conducting of a postmortem
		The organizational level can only benefit from the learning of project teams if the knowledge and reasoning behind the process improvement is converted into such an explicit format that it can also be used for learning in organizational level
	Implications of social interaction on knowledge sharing	The focus on the pure codified approach is the critical reason of Tayloristic team failure to effectively share knowledge among all stakeholders of a software project

2.3.2 Reviewing behavioral studies

The studies for the *organization* school report the advantages organizations can take with social networks. The connection between employees through social networks improve the knowledge assimilation, when new subjects are presented. Relating to the *spacial* school, no studies were found. When it comes to the *strategic* school, studies refer that organizations must

Knowledge Management

address not only organizational and technological issues when implementing a strategy for KM. They also must consider human factors. It is also important to keep in mind that, besides implementation, a strategy for the maintenance is essential to achieve best results. Even within this school, it is possible to identify two different knowledge management types: “Organic” knowledge management, that looks for innovation, and the “Mechanistic” knowledge management that handles the existing knowledge. Finally, this school also enhances the relevance of handling both explicit and tacit knowledge, in order to define a good strategy for KM (Bjørnson et al, 2008; Dingsøy et al, 2009).

For the economic category and therefore the commercial school, no studies were found. Here is an opportunity to researchers to discover useful information about the return on investment from a knowledge management system based on this school.

The revision of the literature led the authors to observe that managers give more value to explicit knowledge than to the tacit one. Organizations in the software engineering field should engage in solutions that cover both types of knowledge which are essential to software development processes. Given the schools discussed and the software development practices, it's possible to map the KM approaches used. Traditional software development is related to the technocratic school, while agile development practices relate to behavioral schools. The agile methods for software development have some practices like retrospectives and frequent meetings that are already good practices. However there is a lack of support for the knowledge outside development teams (Dingsøy et al, 2009).

Table 2.5 - Main findings for behavioral studies, adapted from (Bjørnson et al, 2008)

<i>School</i>	<i>Concepts</i>	<i>Main findings</i>
Organizational	How networks are used in software engineering	Networks should be used in addition to other activities when introducing new software engineering methods
		Description of the role of networks
		Networks built on existing informal networks are more likely to be successful
Strategic	Which factors contribute to successful knowledge management	Suggested model, including technological, organizational and human resource factors
	Which learning processes are used in practice	Ongoing interaction between different learning processes important to improve practice
	Which strategies exist for managing software engineering knowledge	Found evidence of strategies for codification and personalization in software companies

2.4 Organizational Learning - OL

To achieve an effective organizational learning some models were defined in the literature. These models explain how knowledge can be learned, not only in the individual level but also at organizational. The most widely referred models are presented bellow (Bjørnson et al, 2008):

2.4.1 Kolb's - Experiential learning

According to Kolb, learning from experiences should be done through four distinct learning modes. Starting with the way how people retain experiences, Kolb defines the first and second learning modes (Atherton, 2009; Bjørnson et al, 2008):

- Comprehension - knowledge about. Depends on the symbolic representation (Abstract Conceptualization)
- Apprehension - direct practical experience. Qualities identified tacitly in direct experiences (Concrete Experience)

The other two learning modes are inserted in the way how people transform/understand experiences (Atherton, 2009; Bjørnson et al, 2008):

- Intension - Internal reflexion. What the experience means to the person (Reflective Observation)
- Extension - External manipulation. Testing it in practice (Active Experimentation)

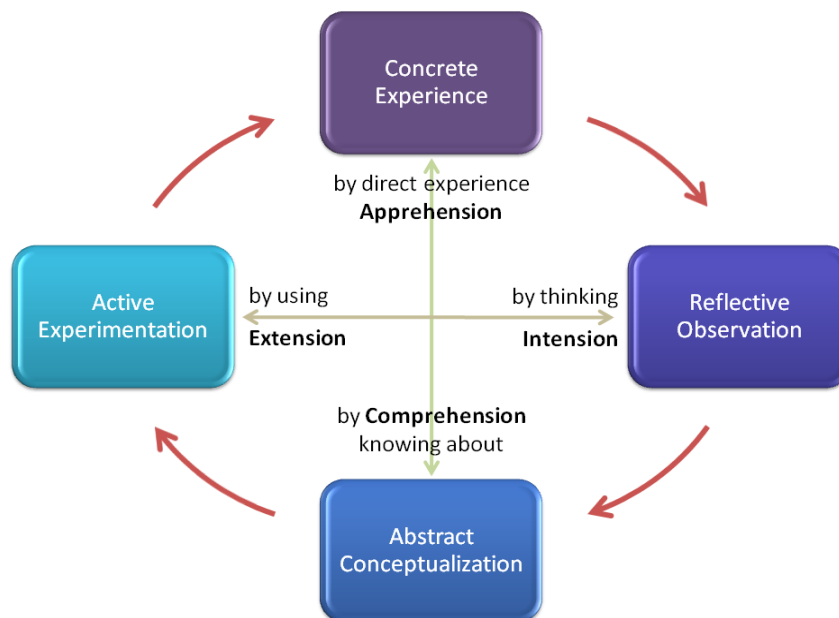


Figure 2.2 - Kolb's model for experiential learning, adapted from (Atherton, 2009)

The experiences referred in Kolb's model can be interpreted as knowledge. In Figure 2.2 is possible to see the interactions between the four learning modes.

2.4.2 Argyris and Schön - Double-loop learning

One of the companies mistakes is the tendency to face learning as a problem solving activity. So, they try to identify and correct the errors. Correcting those errors is fundamental, but employees have also to look to themselves. A self-assessment of their behavior can identify how they, unconsciously, are a part or a contribution to the organization's problems. Thus, they may improve the way how they work. Following this idea, Argyris has “coined” the terms “single loop” and “double loop”, explaining it with the analogy: “a thermostat that automatically turns on the heat whenever the temperature in a room drops below 68 degrees is a good example of single-loop learning. A thermostat that could ask, “Why am I set at 68 degrees?” and then explore whether or not some other temperature might-more economically achieve the goal of heating the room would be engaging in double-loop learning” (Argyris, 1991).

Most successful professionals seldom fail on their tasks. But failure contains an evolution potential. The inability to accept failure may turn professionals unable to learn from their own mistakes. In fact, the difficulty on overcoming criticisms might reduce the potential of learning (Argyris, 1991).

As seen above, Argyris defined two levels of learning (Figure 2.3):

- Single-loop learning - At this level, people follow the process and solve problems, but don't bother about what is the origin of the problem.
- Double-loop learning - This level takes advantages with past actions feedback. Besides feedback, employees also question and know why they do things like that and not in an alternative way.

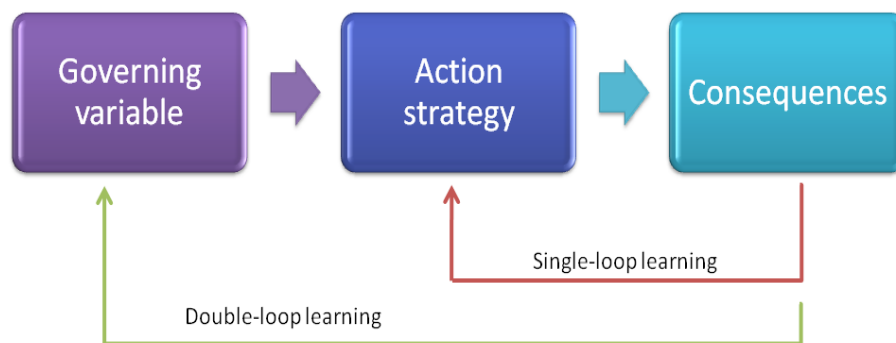


Figure 2.3 - Single and Double-loop learning architecture, adapted from (ProvenModels, 2010)

2.4.3 Wenger's - Communities of practice

Wenger defines communities of practice as “groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly” (Wenger, 2006). A community of practice is always improving its own “practices, routines, rituals,

Knowledge Management

artifacts, symbols, conventions, stories and histories” (Bjørnson et al, 2008). He claims attention not to confuse the concept with simple communities since not every community constitute a community of practice. As example, Wenger referred a neighborhood that is often a community, but not necessarily a community of practice.

There are three main characteristics that communities of practices need to have (Wenger, 2006):

- **Domain** - A well defined shared domain with interest to the community. It cannot be just a club or network of people. Being a member implies to have interest in the domain.
- **Community** - Must go after the domain interests. Members may share relevant information and participate in discussions and activities. That interaction allows a global learning between members. Having a simple website or sharing a job with someone doesn't form a community of practice, only if a regular interaction occurs between the members.
- **Practice** - It is not only a community of interest. Members need to be practitioners and to get involved in the community by sharing experiences, tools and solutions for recurring or punctual problems. Those interactions may be documented in order to create a knowledge base about the community domain.

Communities of practice improve learning at three different levels. First, individuals learn by becoming a member and contributing. Second, communities learn with member's participation, which leads to a practice improvement. At last, organizations learn with the maintenance of interconnected communities of practice (Bjørnson et al, 2008).

Why focus on communities of practice?		
	short-term value	long-term value
members	<ul style="list-style-type: none">• help with challenges• access to expertise• confidence• fun with colleagues• meaningful work	<ul style="list-style-type: none">• personal development• reputation• professional identity• network• marketability
organization	<ul style="list-style-type: none">• problem solving• time saving• knowledge sharing• synergies across units• reuse of resources	<ul style="list-style-type: none">• strategic capabilities• keeping abreast• innovation• retention of talents• new strategies

Figure 2.4 - Why focus on communities of practice?, adapted from (Wenger, 2006)

2.4.4 Nonaka and Takeuchi's - Knowledge spiral

Organizational goals are achieved through tasks made by employees. So, it is fundamental to learn and foster knowledge at individual level. Senge refers in his book that the only way to make organizations learn is with individual learning. It's not a guarantee, but without individual learning organizations won't learn. Individual learning is made through several transformations of knowledge. Employees may learn from colleagues or from manuals, which means, from tacit or explicit knowledge (Senge, 1990) (Rus et al, 2001).

These transformations were defined by Nonaka and Takeuchi:

- Socialization - Interaction between employees that lead to knowledge sharing. With observation and practice tacit knowledge is transferred to another person.
- Externalization - Creating information about knowledge. This transformation implies the conversion of tacit knowledge into explicit by creating documentation, giving presentations or teaching.
- Combination - Aggregating/connecting explicit knowledge. Can be made by merging knowledge from different sources (documents, meetings, call notes) in order to create new knowledge.
- Internalization - Transforms explicit knowledge into tacit one. People understand the information from explicit knowledge, transforming it into new individual knowledge (tacit).

The Figure 2.5 represents the knowledge spiral, also called SECI (Socialization, Externalization, Combination and Internalization) model, defined by Nonaka and Takeuchi. Knowledge passes through several transformations, which improve and spread knowledge inside the organization. (Bjørnson et al, 2008) (Rus et al, 2001).

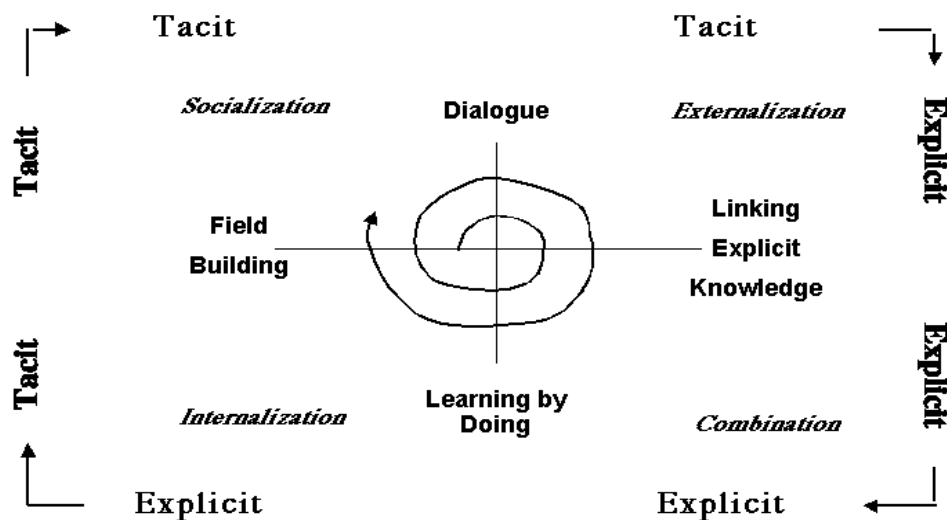


Figure 2.5 - Knowledge spiral, from (Nonaka et al, 1995)

2.4.5 Experience Factory Organization – EFO

The Experience Factory Organization aims to improve software development projects performance (cost, quality and schedule) by improving the overall experience after each project. This concept has recognized that the task of managing the experience of projects is not simple. With evident constrictions, like tight deadlines, technological challenges and concerns about the quality and efficiency, most of the projects don't have time or resources to see their tasks accomplished. So the EFO addresses this problem by assigning this task to a different entity. Thus, two distinct organizations take place: the Project Organization and the Experience Factory.

Project Organization handles packaged experience to deliver software products while Experience Factory gives support with tailored experience. Each one has different tasks, processes and requirements.

The Experience Factory handles all the artifacts generated in the process of software development (experiences, lessons learned, etc.) and generates repositories of experience.

A company implementing the EFO has to assume that learning from previous experience is the best way to solve problems. Creating a sub-organization to analyze and to store experience is not enough. They need to adapt the way how they do the work in order to incorporate the experience search and use, promoting the integration between the Project Organization and the Experience Factory. In Figure 2.6 is possible to see that integration through the information flow. The objective of an EFO isn't just the packaging and the experience storage. It isn't limited to information gathering. The most added value is on the analysis and synthesis of the experiences creating new knowledge (Lawton, 2001; Rus et al, 2002).

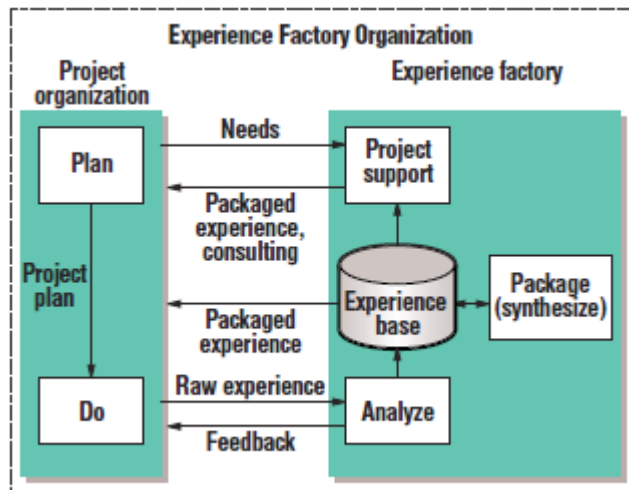


Figure 2.6 - Flow of information through the Experience Factory Organization, from (Rus et al, 2002)

The EFO relies on the fact that knowledge and experiences can be made explicit in order to be stored. Experience Engine is a variation of the EFO, that relies on tacit expertise instead of

explicit knowledge or experiences. To accomplish the concept two roles are needed. First, the Experience Broker, that links the people who have a problem with the one who have the necessary experience to solve it. Second, the Experience Communicator, that is an expert on specific topics. The Experience Broker make the connection between the problem owner and the Experience Communicator. The problem should not be solved by the Experience Communicator, he has to teach the problem owner how to solve it. Experience Engine was implemented at Ericsson Software Technology AB and SD&M AG (Johansson et al, 1999; Brössler, 1999; Rus et al, 2001).

2.5 Implementing Knowledge Management

To implement knowledge management in organizations, it is fundamental to be aware of the main challenges involved. Some of the most important obstacles are described below (Rus et al, 2001; Lawton, 2001):

- **Technology issues**

Sometimes to get the level of knowledge sharing planned, the integration of all systems and tools inside the organization becomes very hard. Software technology is always part of Knowledge Management, so is not possible to avoid this issue. Security is another technology issue. It is important to share knowledge, but not with wrong people.

- **Organizational issues**

A common mistake is to address all efforts into technology and to forget the methodology. It's important to plan the KM implementation. KM failures reveal that 50-60 percent of KM implementation failed due to a bad methodology or the lack of it.

- **Individual issues**

There is no time to create/document knowledge. Employees may not be comfortable on sharing what makes them valuable to the company. They may also not trust in knowledge created by someone else. Then no reuse happens.

- **Lack of standards**

Sometimes, in some organizations, the same concept has different meanings. This misunderstanding increases the resistance on sharing information.

Nowadays, storage and sharing knowledge is easier than in the past due to the technology available. So, the problem falls in Organizational Culture. This obstacle is considered the most difficult to surpass. Trying to implement a KM initiative is very challenging when employees don't have a culture that is pointing in the same direction as the initiative. A sharing culture

Knowledge Management

needs to be encouraged, otherwise, employees won't be willing to share their knowledge (Agresti, 2000; Rus et al, 2001; Rus et al, 2002).

Noting this obstacle, some cultural issues and solution were discussed and applied (Rus et al, 2001; Rus et al, 2002):

- **People and knowledge sharing**

The knowledge possessed by employees in organizations is what makes them valuable. They are paid for what they do with their knowledge, so it's quite natural to see some hesitation when they are asked to share it. The expert status is another reason that might lead some people not to share the knowledge. By sharing it, they risk to lose the expert status and the dependence of the organization on them. Lessons learned are another issue, because they relate directly with things that went wrong. Instead of interpreting them as a way to prevent failures from happening again, employees can connect lessons learned as a sign of incompetence that will be seen by management. At last, some software engineers are reluctant to reuse things made by someone else.

- **Sharing culture**

The most important factor to build a sharing culture is trust. Employees have to trust on managers and vice versa. The feeling that shared information can be harmful or misinterpreted has to be avoided. Trust is a good way to achieve it. Improvement comes from growing with lessons learned. But to achieve that, companies must build an environment/culture that allows it. Inside this environment, mistakes are seen as an opportunity to learn and aren't hidden. Experiences are collected and shared to help the company and not to evaluate people. Feedback is an essential complement to this environment, making knowledge sharing and experience collection more effective. People can be rewarded in order to share knowledge and to help colleagues.

- **Reward systems**

A rewarding system is a way organizations may choose to show to their employees the importance and the value of a knowledge sharing culture. The rewards include not only the sharing but also the reuse of shared materials. One example of a reward system, used by Xerox, is the creation of a "Hall of Fame" to rank employees contributions. At Hewlett Packard, they gave out free airline miles and Lotus Notes licenses for the readers of the knowledge base and also to the submissioners. Infosys created a new coin, "knowledge currency units", which could be exchanged for cash.

- **The evangelist**

In Hewlett Packard experience on the implementation of a knowledge management initiative, it has been identified the need for an evangelist or champion that could encourage employees to participate. With Hewlett Packard initiative they have concluded that the rewarding system wasn't sufficient. Without this role the initiative has failed.

- **Sharing experience with customers**

Companies besides learning from its own experience should also take advantage of external sources like customers and suppliers. The knowledge bases provided by software vendors are good examples. Those knowledge bases are usually public, allowing software engineers to search for useful knowledge.

- **Industry-wide knowledge**

Expert groups or committees in software industries have defined some patterns and standards dedicated to software development. The objective is the improvement of knowledge and expertise of software organizations. A good example of the vast knowledge initiatives in software engineering is the SWEBOK - Software Engineering Body of Knowledge. It represents the knowledge that a software engineer needs to know on the daily work. The ISO15504 also known as SPICE (Software Process Improvement and Capability determination), is another example, it's a framework for the assessment of processes.

2.6 Knowledge Management Systems – KMS

Referring the encyclopedia of knowledge management, a Knowledge Management System (KMS) is “a comprehensive ICT (Information and communication technologies) platform for collaboration and knowledge sharing with advanced knowledge services built on top that are contextualized and integrated on the basis of a shared ontology, and personalized for participants networked in communities. KMSs foster the implementation of KM instruments in support of knowledge processes targeted at increasing organizational effectiveness” (Maier et al, 2006).

A literature review about ICT supporting KM activities shows several terms related to KMS: knowledge warehouse, KM software, KM suite, KM (support) system, and KM technology, as well as, learning-management platform, learning-management portal, learning-management suite, learning-management system or organizational memory system. Despite all these terms, KMS and KM tools are the terms commonly used in literature and in the market. The main objective of KMS is to bring the knowledge previously acquired to the activities to be carried on or upcoming. KMS represent the technological component of KM initiatives. Their scope can be enterprise specific or cross organizational. The type of KM initiative will define

Knowledge Management

the type of KMS. These initiatives may include various types of contents of the knowledge life cycle like ideas, experiences, lessons learned, approved knowledge products, procedures, best practices or patents. Knowledge life cycle includes knowledge creation, organization, storage, retrieval, transfer, refinement and packaging, (re)use, revision, and feedback. KMSs improve the processes and tasks of the knowledge life cycle. The definition of KMS refers a “comprehensive ICT platform” which means that it supports user administration, messaging, conferencing and the sharing of knowledge. It's a platform that handles the knowledge processes and works as an integrating base system where KM applications are built on. (Maier et al, 2006).

Alavi and Leidner have written a simpler and more practical definition. They consider KMS as “IT-based systems developed to support and enhance the organizational processes of knowledge creation, storage/retrieval, transfer and application” (Alavi et al, 2001).

- **Proprietary Solutions**

Proprietary solutions can go from ready to use application to customized solution. They usually provide extreme advanced features like robust searching capabilities, sophisticated document storage and management, collaboration and discussion forums, advanced integration and document versioning. Although those solutions are extremely advanced, they cannot change Organizational Culture that is the main challenge when implementing a KM initiative. A special attention is needed to the integration between the KMS and the employees because it can lead to efficiency loss or even a tool rejection. The worst mistakes happen when organizations implement the KM initiative as a technology implementation (Gonzalez-Reinhart, 2005).

- **Open Source Solutions**

Some companies prefer to adopt open source solutions in a way to get the same benefits, avoiding acquisition price and paying for upgrades or extra support. These costs, over several years, are a good reason for rejecting proprietary solutions, but aren't the only reasons. Open source proponents promote their solutions referring that they have better performance due to the “numerous eyes” they have inspecting the code. Also they use less time fixing bugs. The easier customization is another factor to chose an open source solution over the proprietary. Besides all those advantages, they also have some disadvantages that make some companies hesitate. The solutions may lead to unpredictable costs and lack of support. The nonexistent vendor support is the main reason why companies don't implement open source solutions (Gonzalez-Reinhart, 2005).

2.6.1 Knowledge management software

In this section several knowledge management software categories will be discussed based on their contribution to the SECI model, developed by Nonaka and Takeuchi. The SECI model has been chosen because it is the most cited reference in KM area and the most influential work (Carvalho, 2006).

Intranet-based systems

Intranet-based systems provide a passive communication because information is pulled into the system. Intranets are good for systematize and store together explicit knowledge. The efficiency of intranets is dependent on organizational culture and strategies of information management. They need to be included in the routine of employees work. Despite all the features that an intranet may provide, they often are used just to store and to access corporate documents. Intranets are useful to achieve externalization, combination and internalization. The externalization is done when employee document their experiences and submit their knowledge to the intranet. Internalization is achieved by publishing lessons learned and best practices to the intranet which will then be read by employees. The combination is promoted by unifying different knowledge sources.

Content management systems - CMS

Paperwork is still used and has an important role in the company documentation. So, organizing paper documents is important in order not to lose that information. CMSs are useful to handle this organization. Paper documents are submitted into the CMS by a process of scanning and then analyzed by an optical character recognition (OCR) software. This process transforms the images into readable text by the CMS. The tools will next provide features like cataloging, metadata, searching, versioning and indexing. Only explicit knowledge is manipulated with this tool, promoting the combination process. The main objective of this tool is to turn available any kind of document (fax, e-mail, reports, videos, etc.).

Groupware

Collaboration between team members that are dispersed geographically is fundamental to organizations. Groupware tools allow collaboration in an informal way through synchronous (chat) and asynchronous (e-mail) communication styles. It supports communities of practice, where an expert somewhere in the world may help solving problems from the otherside of the globe. Groupware systems achieve externalization processes through collaboration, exchanging of informal messages and also from discussions. The introduction of instant messaging and videoconferencing has improved the quality of employees interactions and therefore, tacit knowledge exchange. Those features promote the process of socialization and internalization.

Workflow

Workflow systems give support to business processes. They track the information flow from people, places and tasks in processes. The information is structured and follow some predefined and ordered steps. Explicit knowledge is codified from the knowledge generated in business processes supporting the externalization. Workflow systems have three concepts associated to them. First, the concept of roles, played by people with specific skills, executing specific tasks. Second, the concept of rules, that comprises the way information needs to be processed. At last, the concept of routes, that can be explained as process steps that the knowledge flow must follow. Theses systems are also used for business process re-engineering (BPR) and process simulation. They allow a dynamic analysis of business processes. Workflow systems can be integrated in CMSs and Groupware tools.

Artificial intelligence-based systems

Expert systems, case-based reasoning (CBR) systems and neural networks are tools that take advantage of artificial intelligence. Expert systems consist in the observation of experts work and in the mapping of their knowledge to derivation rules. CBR systems provide the learning process through a set of narratives (cases) related to a problem. If a problem occurs, users may search in the case database for solutions to related problem. To match the cases and problems stated, there are matching patterns algorithms. These systems are used with good acceptance and success in help desk and call-center environments. With these systems, the externalization of experiences is facilitated through the narratives or cases. In the same way, users of the system internalize the knowledge that is embedded in the system. Neural networks, use statistical methods to relate problems and solutions. It is a very effective and intelligent system, because each time a new input is inserted in the system, everything is reformatted in order to improve relationships and consequently, to improve the final result. This promotes the combination process.

Business intelligence - BI

Business intelligence systems aim to analyze a huge amount of raw data, trying to extract from it useful information to the business. These systems include a front-end system (a tool to analyze a comprehensive set of data like data mining, query, reporting and OLAP (online analytical processing)), a back-end system, data warehouses and data marts. Information generated in business processes is stored in database management systems (DBMSs) which will then be the source of data to BI systems. Some filters are applied to the information extracted from the DBMS and then stored into data warehouses. These tasks represent the back-end features. After them, the front-end may identify hidden patterns in data warehouses. The users are now able to make their queries and reports. BI systems perform tasks such as sorting, categorizing, information structuring and reconfiguration: they also generate new information. The referred tasks lead to the combination process.

Knowledge map systems

Knowledge maps are used to store information about which knowledge is owned by each employee instead of storing the knowledge itself. It's like yellow pages about employees and their expertise. This systems have an advanced search engine to locate the best expert for a specific need. Knowledge maps are useful to manage human resources, because they provide maps about what knowledge/competencies employees have and the one that is lacking. By making the search of experts easier, the probability of the problem owner being supported is higher. This link between employees facilitates personal meeting, promoting tacit knowledge and experiences exchange, which leads to socialization.

Innovation support tools

Innovation is defined as the application of new ideas to products or services. Then, innovation happens when new patents are created, existing products are upgraded and new products are created. The tools that support innovation have to create new knowledge while products are being designed. There are some features they must include: technical database to store patents, articles and projects; graphic simulation to facilitate internalization and combination tools, to allow the test of unusual possibilities in products design. By combining explicit knowledge to create or update products or patents, these tools support the combination process.

Competitive intelligence tools - CI

Competitive intelligence tools have a cycle with five steps:

1. Planning and direction - definition of criteria that will guide the information gathering.
2. Collection of published information.
3. Primary source collection - information acquired from people instead of published sources.
4. Analysis and production - understanding collected data to create useful information.
5. Report and inform - delivering of processed information, in a legible structure, to the relevant recipients.

The main support of CI tools is concentrated in steps two and five. On step two, the tool will collect information from internet searches and corporate intranets. On step five, the system will deliver the reports, by mail for example, given the preferences of the users. The other steps are essentially constituted by human tasks that can not be greatly improved by the tool. Summarizing, the information gathered is processed (filtered and categorized) and then provided to someone who will use it. This process promotes the combination of information.

Knowledge Management

Knowledge portals

Knowledge portals were created to integrate several sources of information with unique interface to the users. Its main objective is to provide access to available information already stored somewhere and not to be a separate source of information. The features that are usually present in knowledge management portals are: enterprise taxonomies, classification of information, search engine and links (internal, external and information sources). End-users have the possibility to personalize their work by associating it with communities, interests, tasks, etc. Portals can also provide instant messaging, so users can see who is on-line and connect each other to get some help. Knowledge portals are evolving so much that they are integrating the features of the KM tools referred before. With this integration, knowledge portals achieve externalization, combination, internalization and socialization, all the processes of the SECI model.

In Table 2.6 are summarized the knowledge management software categories and their contribution to the SECI model.

<i>Table 2.6 - Categories of knowledge management software: summary table, from (Carvalho, 2006)</i>			
<i>Category</i>	<i>Dominating Knowledge Conversion Processes</i>	<i>Origin of Concepts</i>	<i>Examples</i>
Intranet-Based Systems	Externalization, Combination, Internalization	Computer Networks (Web Technology)	Apache HTTP Server
Content Management Systems	Combination	Information Science	Excalibur Retrieval Ware and File Net
Groupware	Socialization, Externalization, Internalization	CSCW (Computer-Supported Cooperative Work)	Lotus Family (Notes, Sametime) and MS Suite (Exchange, Outlook, Messenger)
Workflow	Externalization	Organization and Methods	ARIS Toolset (IDS Scheer)
Artificial Intelligence-Based Systems	Externalization, Combination, Internalization	Artificial Intelligence	Neugents (Computer Associates)
Business Intelligence	Combination	Database Management	Business Objects and Oracle 10g BI
Knowledge Maps	Socialization	Information Science and Human Resource Management	Gingo (Trivium) and Lotus Discovery Server
Innovation Support Tools	Combination, Internalization	Innovation and Technology Management	Goldfire Innovator (Invention Machines)
Competitive Intelligence Tools	Combination	Strategic Management and Information Science	Knowledge Works (Cipher Systems) and Vigipro (CRIQ/CGI)
Knowledge Portals	Socialization, Externalization, Combination, Internalization	Computer Networks and Information Science	Hummingbird and Plumtree

2.6.2 Knowledge management systems' architectures

Architectures are reference models for possible implementations of information systems. Two main architectures were defined for knowledge management systems: a centralistic version and a peer-to-peer (p2p) version. The two architectures can be used as a reference that helps the organization designing a specific KMS combining tools and systems already implemented.

The solutions most implemented in organizations are centralistic client-server solutions. In Figure 2.7 is represented an example of a centralistic solution. It promotes a central knowledge management server that joins all the knowledge shared and provides a set of services to users and upward layers. (Maier, 2006; Maier, 2006b)

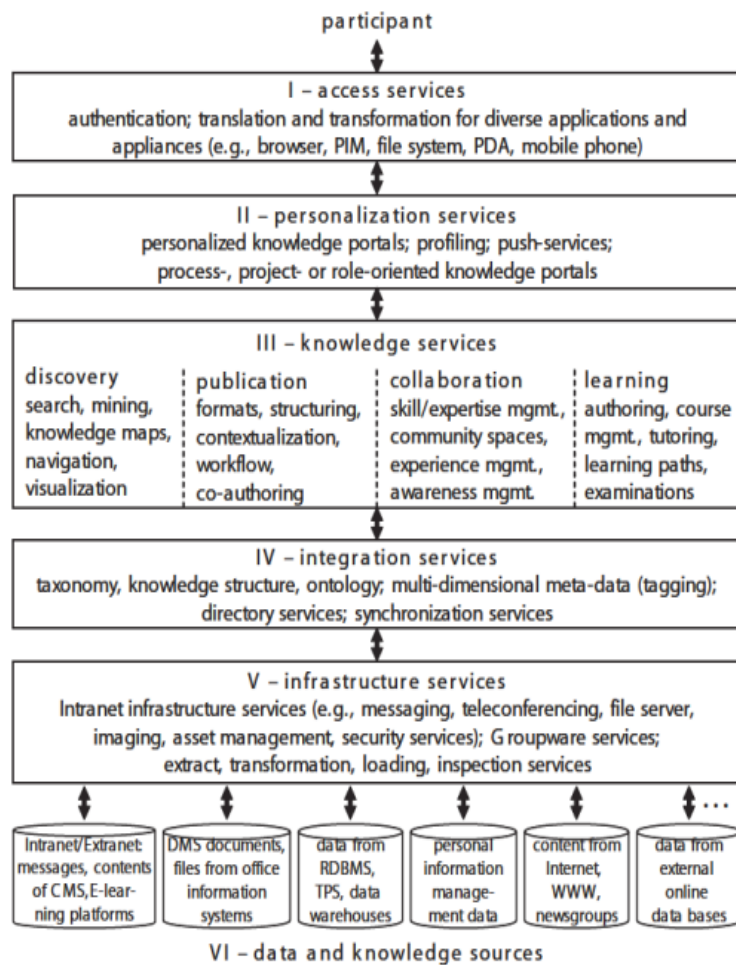


Figure 2.7 - Architecture of a centralized KMS, from (Maier, 2006)

Data and knowledge sources layer is composed by internal and external sources with structured and semi-structured information and knowledge.

Knowledge Management

Infrastructure services layer may support communication and information management functionality. It provides tools to extract and manipulate information and knowledge from the layer data and knowledge sources.

Integration services layer allows to categorize, analyze and link information and knowledge from several sources. It also supports synchronization services that provides the ability to work off-line, ie, exporting something to make some changes and then reintegrate it in the organizational knowledge base.

Knowledge services layer includes four activities related to knowledge:

- Discovery – search, retrieval and visualization of knowledge and experts.
- Publication – structuring, contextualization and publication of knowledge.
- Collaboration – creation, sharing and application of knowledge in a community environment
- Learning – tools that support courses management, learning paths and examinations. Also supports double loop learning with reflections on learning and knowledge processes.

Personalization services layer gives access to subject-matter portals for specific roles or communities. Several pages can be automatically generated using interest profiles or to serve a specific role.

Access services layer handle the content in order to display it in all applications and for all appliances needed. This layer also assures that only the right person accesses the right knowledge.

The peer-to-peer architectures came to solve some of the centralized version issues. They aim to reduce design, implementation and maintenance costs, as well as integrate personal knowledge workspace with the shared knowledge workspace. P2p also tries to expand the focus that usually stays at organizational internal knowledge and sometimes, knowledge processes cross organizational boundaries. This type of architecture can vary from pure p2p to hybrid architectures. The hybrid architectures are the ones that need more centralized support for the operation of the KMS. The main difference of p2p systems, comparing to centralized systems, focuses on the possibility to build personal knowledge bases locally without a shared space in a server and still being able to share the knowledge. Figure 2.8 represents an architecture of a server and peer system.

Both architectures, p2p and centralized, own the same layers but with differences inside each layer.

Infrastructure services manage knowledge source and personal data. They also support the exchanging of data between peers without neglecting the security of personal knowledge bases.

Knowledge Management

Integration services create a personal ontology with personal knowledge base. They distinguish private work spaces (only accessible by its owner), public work spaces (knowledge published via internet) and protected work spaces (accessible to a group of peers previously defined).

Knowledge services work similar to the centralized architecture.

Personalization services use individual profiles.

Access services work similar to the centralized architecture.

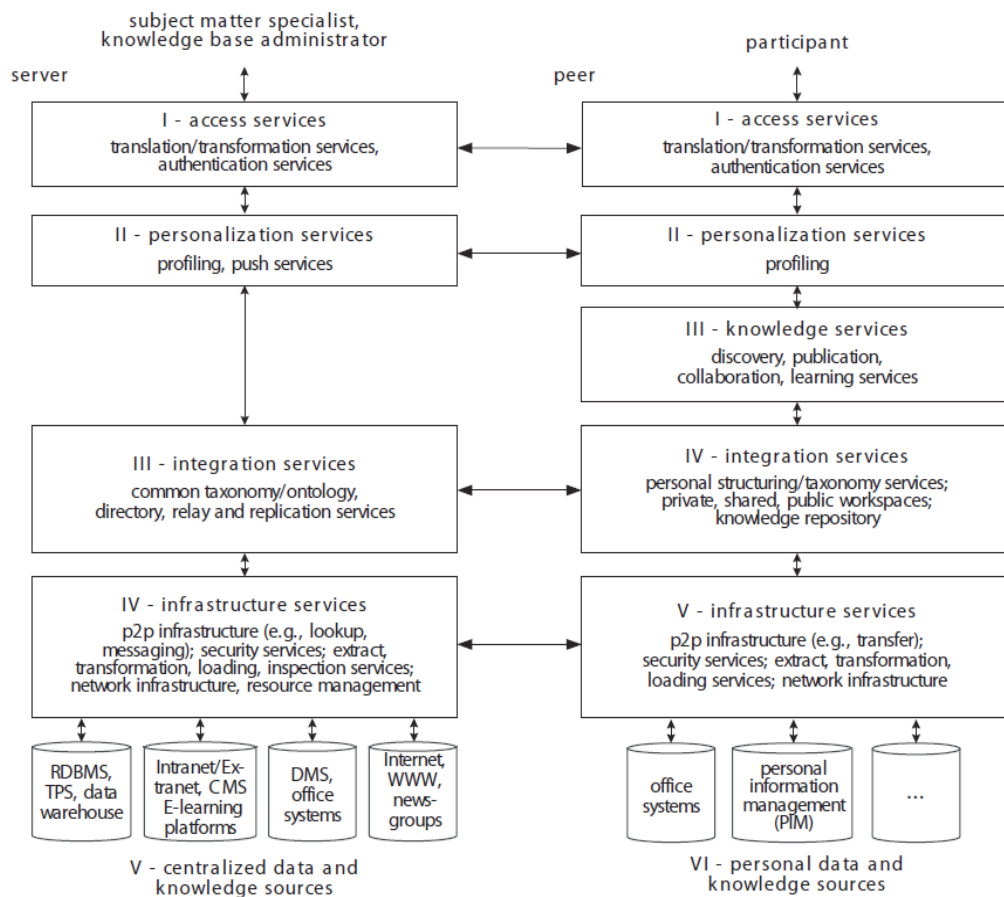


Figure 2.8 - Architecture of server and peer, from (Maier, 2006)

In Table 2.7 is summarized the relation between the type of architecture and the KM initiatives. Centralized KMSs are related to a codification strategy that is centrally managed in order to handle all kind of information. Considering p2p KMSs, the strategy is different. It is called personalization and can involve members of several institutions. This strategy has a decentralized management system that focuses on exchanging individual knowledge requiring a trustful culture (Mayer, 2006).

Knowledge Management

Table 2.7 - Type of KMS and type of KM initiative, from (Mayer, 2006)

<i>Characteristic</i>	<i>Centralized KMS</i>	<i>P2P KMS</i>
Strategy	Codification	Personalization
Organizational design	Central	Decentralized
Content	Primarily lessons learned, (approved) knowledge products, and secured knowledge, but also ideas, experiences, and individual contents	Individual contents, ideas, and results of group sessions and experiences
Organizational culture	Both types of culture (restrictive or loose user privileges)	Open, trustful culture

2.6.3 Wikis

The origin of the wiki word is Hawaiian (wikiwiki) and stands for “too-fast” or informal. The Wiki was created by Ward Cunningham in 1995 following the idea of a web page where is possible to have a cooperative and open work. He has chosen this name as a substitute to the word “quick” and to avoid using “quick-web” (Junior et al, 2009; Cunningham, 2009).

Ward Cunningham defined a wiki as a “free expansion collection of web pages connected in a hypertext system to store and to modify information - a database, where each page can be easily edited by any user with a browser” (Leuf et al, 2001). The main purpose of a wiki is the possibility to modify any original text. Thus, new knowledge is created and added to the existing one. In open wikis, everyone can edit or add new pages and in restricted wikis only people with a valid login can add new content. All users are allowed to modify any content without asking permission to the authors. At the end the pages are built in a collaborative way. Simplifying the concept, a wiki is as web-page that can be edited from a internet browser enhancing the publishing and sharing of knowledge on the web. The need for knowledge helped the driving force behind wikis and led to knowledge sharing. This need brings us to the creation of databases that can be accessed and edited by a group of people interested in a specific area (Junior et al, 2009).

Several applications for wikis can be enumerated:

- Exchanging of ideas, development of glossaries, dictionaries, textbooks, manuals, repositories, topics, meetings.
- Creation of structures of shared and collaborative knowledge that can create learning communities.
- Creation of a work schedule or the development of a project.
- Avoiding the use of telephone and e-mail, because users can search information before asking about it.
- Tracking system, providing the information about who changed what.

Knowledge Management

Wikis can be proprietary solutions like Confluence or open source solutions like TWiki which is not just a wiki but also a document or project management system (Confluence, 2010; TWiki, 2010).

With the spread and growth of wikis, they soon have become used in various environments, such as education (teaching and distance learning), business (employees share knowledge) and engineering (tool for project management). Wikis are used as a tool to improve people communication, meeting tracking, collaboration work and knowledge bases (Junior et al, 2009; Grace, 2009).

Table 2.8 list some types of wikis that exist today.

<i>Type</i>	<i>Description</i>
Personal wiki	User keeps it as a form of concept map or journal for an idea
Semantic wiki	Knowledge is described in a formal term which allows for machine-processing like a semantic web
Corporate wiki	Is mostly used internally in a corporate context contrary to public wiki on internet
Structured wiki	Combines benefits of sharing and collaboration of a plain wiki with structured elements of a database by allowing the structuring of information when needed
Peer-to-peer wiki	Wiki sites are shared between peers on a server-less system. It is stored on computers of the users and provides less security features

Wikis provide a conversational KM solution as a method for organizational knowledge creation. They are good solutions for business due to its easy implementation without a big investment. They also work well with geographically distributed teams. As seen in Table 2.9 wikis are able to be better than conventional conversational tools and techniques (Gonzalez-Reinhart, 2005; Wagner, 2004).

<i>Technology</i>	<i>Communication</i>	<i>Knowledge Repository</i>	<i>Knowledge Catalog</i>
E-mail	1-to-1, 1-to-many, person-to-person	Local e-mail archives possible	Local index possible
Static and DB backed web pages	1-to-many, approaching many-to-many, "dialog" between web pages through hyperlinks	Local archives	Local index possible, web rings create larger catalog
Discussion forum	Many-to-many in web based forums, repeated 1-to-many in list servers	Central repository if web based, local if list server	Central index if web based
Internet chat	1-to-1, many-to-many	Frequently none, transient communication	None

Knowledge Management

Table 2.9 - Conversational technology overview, from (Wagner, 2004)

<i>Technology</i>	<i>Communication</i>	<i>Knowledge Repository</i>	<i>Knowledge Catalog</i>
Video / audio streaming	1-to-many	Central host or decentralized streamers	None, streams not indexed.
Video / audio conference	1-to-1, 1-to-many	Local repository if content is recorded	None, content typically not indexed.
GDSS	Many-to-many	Available, but GDSS sessions often treated as one-off.	Typically none, but possible
Web Log	1-to-many, can approach many-to-many (similar to web pages)	Local repository within each weblog. "Metablogs" now emerging	Yes, local index, metablog may provide larger catalog
Wiki	Many-to-many	Yes, current knowledge and history ("temporal database")	Yes

2.6.3.1 Wiki Design Principles

As seen before, wikis are a script driven website where users may manipulate the pages content. Thus, users can learn and contribute with new information, making the site grow. This freedom that a wiki can provide is regulated in an informal way through principles defined by Ward Cunningham, Figure 2.9 (Gonzalez-Reinhart, 2005; Cunningham, 2010).


It is important to have in mind these principles to achieve a successful implementation of a wiki and to generate the community environment. Some principles, like the *open* principle, make some people concerned. Even if, nowadays, experience shows that people are not ill-intentioned and follow the principles, the power of being allowed to add, edit or delete any part of the wiki may be dangerous. However, this principle allows a constant interaction in the system by adding and correcting the content. Thus, knowledge sharing is promoted (Gonzalez-Reinhart, 2005).

Wikis are being incorporated into business environments avoiding the need for "countless conference calls, meeting and e-mails back-and-forth to resolve issues and understand requirements" (Gonzalez-Reinhart, 2005). Companies are using wikis to design, edit and manage projects and their teams. Users may view and edit wiki content through a collaborative work in accordance with the *observable* principle. The application of this tools into employees daily work enables a faster work competition and becomes part of the work process. This collaborative work can break the organization boundaries and achieve costumers and partners. With that external collaboration it's possible to identify new business opportunities and to reduce personal interactions with the customer. Less phone and e-mail interactions lead to time saving and consequently costs reductions (Gonzalez-Reinhart, 2005; Grace, 2009).

Considering the *organic* principle, it is possible to handle with static systems problem, because it recommends a site with a structure and contents flexible enough to be customized and to support evolution. That characteristic foster knowledge creation. Noting that employees only

Knowledge Management

have one way of interaction, through explicit written words, they may naturally document their tacit knowledge. Given this, the codification of tacit knowledge dispersed in the team turns explicit in the wiki pages. This constant communication improves trust between members.



Wiki Design Principles

Wiki has turned out to be much more than I'd imagined! That is not to say that I didn't imagine a lot. These are the design principles I sought to satisfy with the first release of Wiki. -- [WardCunningham](#)

Note that this page is only a reconstruction from memory of intentions I held at the beginning. Additional principles, like server robustness, have been forced upon me.

- **Simple** - easier to use than abuse. A wiki that reinvents HTML markup (`[[b]bold[/b]]`, for example) has lost the path!
- **Open** - Should a page be found to be incomplete or poorly organized, any reader can edit it as they see fit.
- **Incremental** - Pages can cite other pages, including pages that have not been written yet.
- **Organic** - The structure and text content of the site are open to editing and evolution.
- **Mundane** - A small number of (irregular) text conventions will provide access to the most useful page markup.
- **Universal** - The mechanisms of editing and organizing are the same as those of writing, so that any writer is automatically an editor and organizer.
- **Overt** - The formatted (and printed) output will suggest the input required to reproduce it.
- **Unified** - Page names will be drawn from a flat space so that no additional context is required to interpret them.
- **Precise** - Pages will be titled with sufficient precision to avoid most name clashes, typically by forming noun phrases.
- **Tolerant** - Interpretable (even if undesirable) behavior is preferred to error messages.
- **Observable** - Activity within the site can be watched and reviewed by any other visitor to the site.
- **Convergent** - Duplication can be discouraged or removed by finding and citing similar or related content.

There are many Wiki authors and implementers. Here are some additional principles that guide them, but were not of primary concern to me.

- **Trust** - This is the most important thing in a wiki. Trust the people, trust the process, enable trust-building. Everyone controls and checks the content. Wiki relies on the assumption that most readers have good intentions. *But see: [AssumeGoodFaithLimitations](#)*
- **Fun** - Everybody can contribute; nobody has to.
- **Sharing** - of information, knowledge, experience, ideas, views...

Comments:

- **Interaction** - This enables guest interaction.
- **Collaboration** - We believe that this could make a good collaboration tool, both synchronously and asynchronously.
- **Platforms** - We like the cross-platform implications.
- **Social Networks** - Its power for supporting collaboration is great.
- **It is FUN** - It is very easy and fun.

See: [WikiWikiHyperCard](#), [WikiHistory](#).

Figure 2.9 – Wiki Design Principles, from (Cunningham, 2010)

Trust is a key word in the wiki world, that relies on the faith of users believing that no one is malicious. However, some mechanisms are provided to prevent destructive acts and to restore the original content. This is done using “histories” that all pages have, allowing roll-back procedures and the ability to revert to previous versions. For business environments, it's required a better control of the manipulated information. To address that requirement, is possible to associate a registration of users allowing to trace everything that is made in the wiki. This feature allow companies to manage and reward contributors and also to encourage less active members. The wiki environment leads to the constructivist learning theory, where “trust enables an individual to express knowledge in order to construct it, and influence helps to refine knowledge” (Gonzalez-Reinhart, 2005).

Knowledge Management

The knowledge that persist in a wiki is considered objective, because if the *open* principle is contemplated, everyone can delete the information they disagree with. Following this idea, the existing information is essential for the relevant knowledge and unnecessary or wrong information is deleted. Encourage people to participate and to contribute is one of the biggest challenges in KM initiatives. It's extremely important to have that participation in order to achieve the advantages of wikis and therefore KM benefits. Wikis are based in hyperlinks that may be easily created and link to non-existent pages encouraging users to add the missing information. This feature represents the *incremental* principle. Search for prestige and community recognition are the motivations that may lead people to add the missing knowledge and contribute to the system (Gonzalez-Reinhart, 2005).

Tay Pei Lyn Grace has made an analysis of three case studies about adoption and usage of wikis in organizations. That analysis allowed her to identify more benefits than the ones identified before. One significant benefit with this adoption in organizations is the time saved. The training needed is substantially less comparing to common KMS. This is related to another benefit: the ease of use that reduce the training costs. Travels to attend meetings and for collaboration work were reduced saving time. An interesting point was the preference of users to use wikis instead of e-mail for several activities. Using e-mail to exchange documents lead to loose track of versions. Using a wiki to that activity, users always have access to the last version of the document and when it achieves the final version, it can be locked for modifications. This benefit reduces the number of e-mails exchanged and increases productivity. Wikis, through the collaboration benefit, give voice to users inside the organization, which helps to establish a trusting culture (Grace, 2009).

2.6.3.2 Wikis challenges

The same study by Tay Pei Lyn Grace, mentioned above, also addresses the challenges that a wiki implementation may face. The freedom and flexibility that wikis provide jeopardize security. This means that someone has to check the security and integrity of the content. Access control is another issue that has to be well implemented because some industries have sensible information about clients or other subjects that have to be controlled. When the topic is security, companies often see proprietary solutions more secure than open source.

In wikis systems there are several mark-up languages used, which may difficult data migration when implementing a wiki. Organizations must be careful when selecting and evaluating the wikis to adopt. They should test them in small and controlled environments in order to check architectural integrity and to prevent possible issues to appear in big implementations.

Users sometimes have doubts categorizing information. They define the structure and the categorization of the information in a subjective way. This behavior results in information stored in the wrong place, making its retrieval more difficult. Defining in predefined structure shared by the company will be useful to avoid this problem (Grace, 2009).

3 Problem: Software Knowledge Management

Different meanings for knowledge can be found in the literature. The aim of this work focuses in the software engineering knowledge context. This thesis is focused on knowledge management for software engineering, where knowledge is intensive, specific and volatile. Effective KM is essential due to extremely competitive industry, characterized by high-yield performance, stringent quality standards and short lead time. Usually, employees have the knowledge needed to the organization's core business, and often the knowledge goes with them when they leave the company. Adding to this, individual knowledge is difficult to get and to distribute throughout the organization. The whole process of development creates and manipulates information that needs to be formalized, taking it from documents, conversations, meetings, tests, etc.. A good management of this knowledge is crucial for the efficiency and success of organizations that develop and support software. The process to create, locate, disseminate and use of knowledge is not institutionalized or doesn't exist at all in companies. The willingness to use existing knowledge in a company is the main motivation to start a KM initiative (Wei et al, 2002; Rus et al, 2002).

Some problems that organizations are facing today may be solved implementing a KM solution. It's a fact that the main objectives in software development companies include the decreasing of time spent on the development, also the reduction of costs and the increasing of quality. It is known that development teams almost never take advantage from previous projects experiences, repeating mistakes. Introducing a process to manage knowledge and previous experiences lead companies to avoid those mistakes and their related rework. With each new project, individual and team experience increases. Organizations could benefit even more from individual experiences if they were able to share it. In the software development process, everyone involved has to make constant decisions. Those decisions are made based on individual knowledge or informal conversations, which works in small environments. As the

Problem: Software Knowledge Management

company grows, a process of sharing information is needed in order to allow informed decisions.

New technologies emergence is good to improve efficiency in software development, but their introduction may be problematic to projects. Developers take time to be proficient and they usually “learn by doing”, which can lead to some unexpected problems. A solution to quickly acquire the knowledge about new technologies and spread it through the organization is fundamental to address this issue. Beyond the inherent knowledge in software development companies, it's also needed to handle and to be aware about the existing knowledge on the field where software is going to be developed.

Policies, practices and culture are topics that companies have and need to spread. This information has to be documented so employees can access it whenever necessary. A formal representation of knowledge is needed, as well as its share, making everyone able to access it. Despite being encouraged to formalize the knowledge, the informal exchange of information must not be neglected because it promotes a knowledge-sharing culture.

Even if the necessary knowledge exists, mistakes still happen on software development. Knowing who knows what, makes it possible to locate an expert to solve a specific problem. It also provides information about the knowledge that exists inside the organization and the one that is lacking. This is useful to manage human resources identifying training needs and mapping the knowledge that disappear when an employee leaves the company (Rus et al, 2002).

Software development is a complex process and the generated knowledge is hard to manage. That's here where this dissertation focuses on: the needs and features necessary in software development in order to achieve a successful software knowledge management. A list of software knowledge management necessities are defined below. They are based on literature reviews made in chapter 2 and from problems identified by the company I work in. The focus is not to find the necessities of a specific software development model like waterfall, iterative, agile methods, or any other model. The aim is to take the common parts among them and improve the way how the knowledge is managed. All models have to deal with requirements, implementation, tests, bugs and planning. All those activities have precious knowledge that needs to be managed.

It's important to note that these necessities are not all the requirements to develop a tool which supports a particular model of development. Those details will not be addressed. The focus will be on the identification of specific needs to manage knowledge and to enhance their reuse.

Necessities for software knowledge management:

N1 – To handle individual knowledge. Each user must have a personal page or, optionally, a blog with personal data, experiences, ideas, skills, etc..

Problem: Software Knowledge Management

N2 – It must be possible to add, create, search and view information and knowledge whenever it is needed.

N3 – Documents may follow a template, but should not be dependent on it. The template can evolve. Templates may exist in empty and filled version.

N4 – All the documents created must have an unique identifier, a version control and an associated author for each version. Documents should preferably be created and edited in the system. Those that are not, must be managed and controlled by the system. It must be possible to identify the contribution of each author in each document.

N5 – All the documents must be exported to an editable and non editable format. It allows users to share information with people who don't have access to the system.

N6 – Several users may edit the same document what promotes collaboration between users. It is necessary to control concurrent access to the same information. It can be made, for example, by a lock system.

N7 – Tracking users actions permits to analyze users contributions. It allows to identify their specialties.

N8 – Knowing who knows what in order to help in resources allocation, experts identification and analyze of training needs. It allows to manage skills.

N9 – People with expert knowledge must be accessible to all employees for advice or knowledge exchange.

N10 – Concept of roles, which means people with certain skills, who execute specific tasks, having specific access rights. The roles must support information visualization by customers.

N11 – Internal and external links, in order to improve the navigation and the relation between the content.

N12 – Policies, practices and culture repositories. Everyone must have an easy access to this information. Here are included organizational processes, quality management process, conventions of communication and image, etc..

N13 – Communities of practice. This necessity allows employees to create a group of people interested in a specific domain. It aims to develop a subject or to solve problems about that domain.

N14 – External sources of information. External sources may be costumers or suppliers. These sources must allow shared experiences, such as common problems found by all costumers of a supplier. They may be used to gather new knowledge.

N15 – Track communication with external stakeholders. Using e-mail sometimes information becomes spread and hard to find or view.

Problem: Software Knowledge Management

N16 – Support innovation processes. A section where users can suggest new ideas or topics for improvement is an asset to the organization evolution. In both cases, it should be allowed to relate them with one project, point of improvement, etc..

N17 – Spreading information between employees through social networks improve the knowledge assimilation when new subjects are presented. So, employees must be notified about news or other kind of information through a social platform.

N18 – Support user administration and manipulation of existing knowledge.

N19 – Must handle the knowledge process and work as an integrating base system where KM applications are built on. Which means that it new applications may be added to the system to support new necessities that may appear with the business evolution.

N20 – Apply a lessons learned process to projects, which include postmortem reviews and retrospectives, in order to exchange experiences across projects.

N21 – Acquire knowledge about new technologies. With the natural evolution of technologies the “learn by doing“ must be avoided. Updated repositories for consultation and training programs must be provided.

N22 – Acquire domain knowledge. Acquire knowledge from customers, vendors or other sources about the field where the software fits.

N23 – Acquire knowledge about information spread through several sources. This information may be stored, combined and provided for visualization when needed. If the information cannot be acquired, it must exist instructions on how to access it.

N24 – It must be possible to connect several kinds of information. As example, it must be possible to connect risk with projects or requirements with development items. The connections must be able to be seen in a traceability matrix. Similarly, bugs reported by executing tests may be related to test cases and test cases to development items.

N25 – Projects must have resources allocated. This information show where are the resources working. The time needed for each employee must be defined in order to know the availability of each one. And eventually, to assign them to other projects. It allows to manage skill and show which skills are busy and which ones are available.

N26 – Creation of work-products packages. This packages usually called baselines are generated by specific versions of all work-products in a well defined phase of a project. This is very useful when several versions of the same product are spread through the customers. This need is also applied outside the project scope.

N27 – Workflows defining knowledge processes are recommended. Users are guided through the steps that they need to perform. The system must implement the workflow leading users to follow it.

N28 – If some information is managed or created by an external tool, the information has to be acquired in order to centralize it in the system.

Problem: Software Knowledge Management

N29 – Software development repositories. Here are included code conventions, Code library for reuse, development processes and models, etc..

N30 – Acquire knowledge about the software development life cycle (SDLC). This knowledge is created in several tools and spread through them. This information must be acquired, stored, combined and provided for visualization. This recognition of existing information on other platforms would improve the efficiency in knowledge acquisition and subsequent manipulation. The knowledge to be acquired will depend on the development model and on the needs of the company. It could be source files, code documentation, burn down chart, requirements, bugs, documents, etc..

4 Software Knowledge Management using Wikis

The necessities identified in chapter 3 might be implemented through features described in this chapter. The features provide the necessary functionalities for successful software knowledge management. Although, not all the features required for a software development process are included. The focus is not the process of software development but its knowledge management. However, the features described here may already be part of some process for software development or may be added to an existing one.

4.1 Features

The features that meet the needs described in the previous chapter are explained here. Some prototypes will be illustrated, but only for a visualization purpose and not a real implementation.

F1 – Login system

Description: The system must allow the creation of users. Each user must have access to the system through a username and password. With this feature, it is possible to track users contributions and then, to identify specialties of each one.

Needs covered: N1, N2, N4, N7, N8, N10, N13, N17, N18

Software Knowledge Management using Wikis

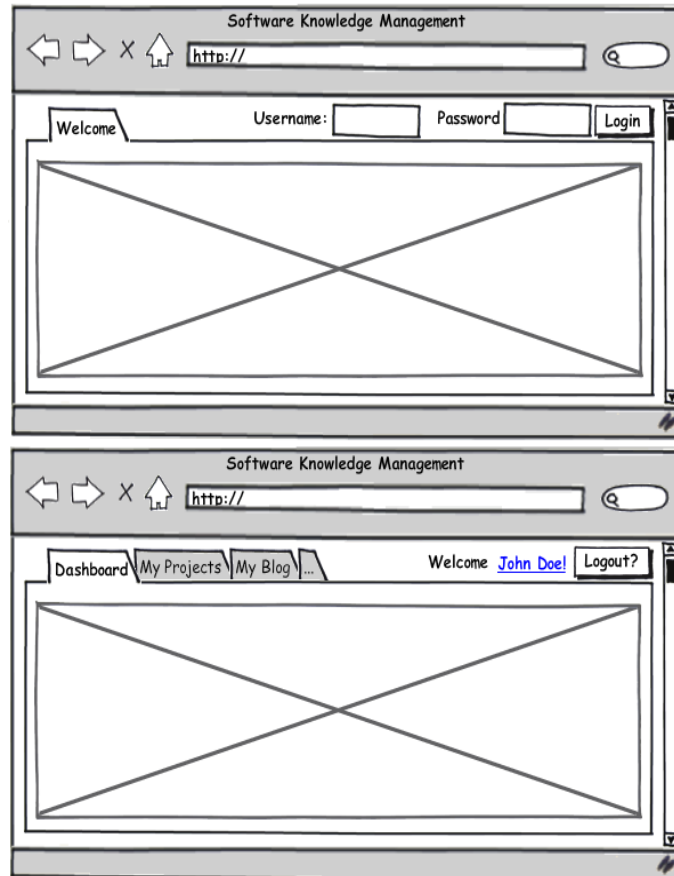


Figure 4.1 – Prototype for the login feature

F2 – Users Profile

Description: Each user of the system has a personal profile with the relevant personal information. Skills and expertise are mandatory information. Associated to the profile, users may have the option to create a blog where they express their experiences or ideas. Blogs are explained in a separated feature.

Needs covered: N1, N8, N25

F3 – Support Roles or groups of users

Description: Each user has an account that allows him to access the system. Once in the system, users have a predefined role associated to them. It allows them to have access to some parts of the system/content and with certain permission as defined in each role. This concept is used to know what kind of activities users perform. It adapts the information to them. It also must support a role that give the needed visibility to costumers. This feature can also be used in workflows, explained in another feature.

Needs covered: N10, N18, N25, N27

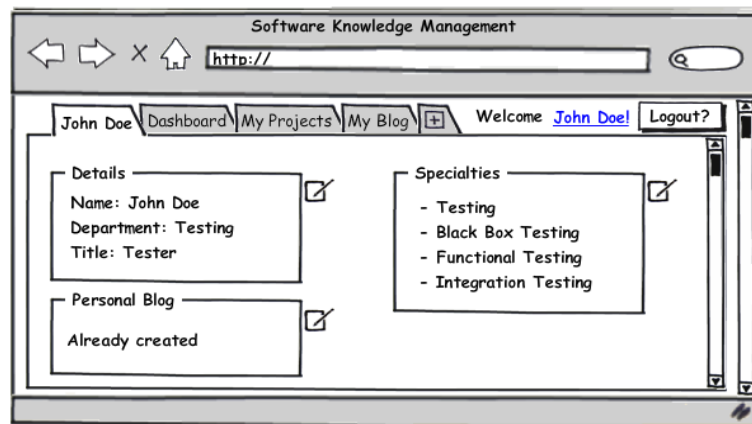


Figure 4.2 - Prototype of the user profile

F4 – Personal Blogs

Description: The blog will consist on a web page where users can add content creating articles inside the page through the rich text editor feature. These articles will be organized into categories defined by the authors. Categories provide an effective search. Other users might have the opportunity to leave comments in each article. Comments may be used to ask for some explanation or to help in the article subject. The author of the blog is notified if some comment is created. These features allow and promote experience sharing. It's a way how employees may express their experiences (tacit knowledge) and codify them into explicit written knowledge.

Needs covered: N1, N2, N9, N20

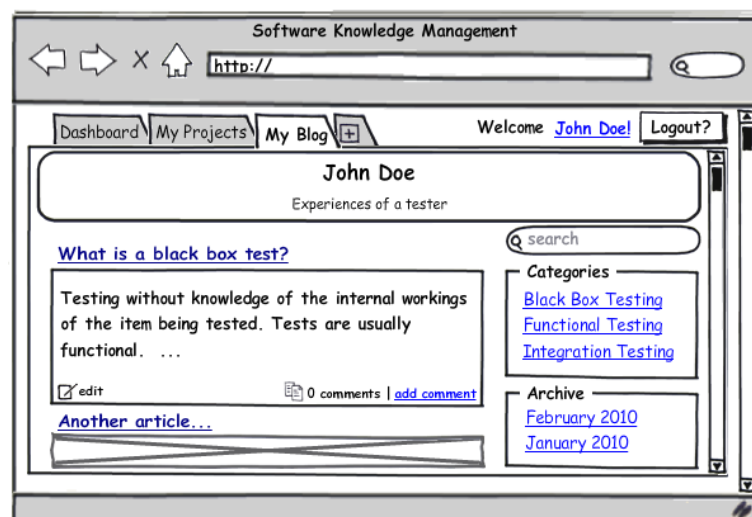


Figure 4.3 - Prototype for the blog feature

F5 – Rich text editor

Description: The rich text editor allow users to create content through a WYSIWYG (What You See Is What You Get) interface. This feature provides an editor that supports several types of media (ex: texts, links, images and videos). A good example of this feature is the TinyMCE ^(TM) - A free javascript WYSIWYG Editor (TinyMCE, 2010).

Needs covered: N2, N6, N11, N12, N13, N20

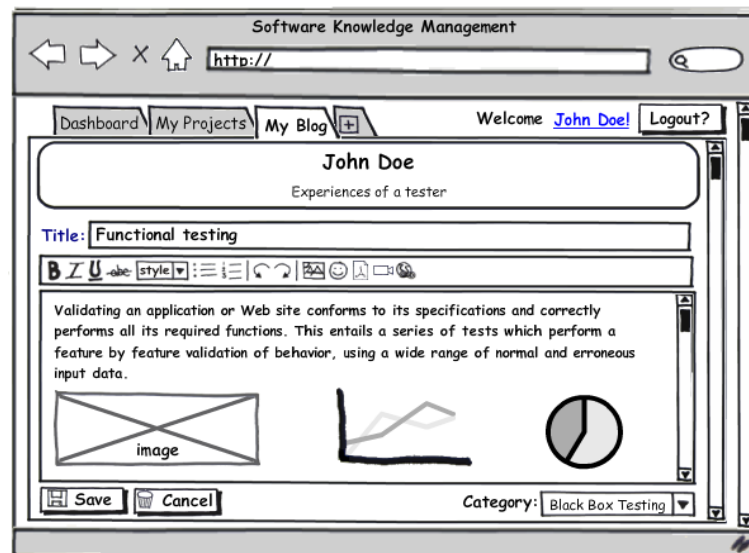


Figure 4.4 - Prototype for the rich text feature while creating a new article in a blog

F6 – Create/Edit Content

Description: Content pages or documents may be created or edited through the rich text editor interface. When creating a new page, it is possible to select a template that will structure the page. This structure makes easier the creation of a new content. Even selecting a template, the user mustn't be obliged to follow it completely. All the content created must be assigned to a specific area of the system, like a project area, a repository etc.. To edit content pages or documents, users have to belong to a role with rights to edit them. The same happens to the creation process. Each time something is edited or created, these activities are recorded, tracking the author and their contributions. All contents produced must be enabled to be rearranged. It means that a user with appropriate rights must be able to move the content location to the place where it is better organized. Concurrent access to the same information is controlled, preventing simultaneous changes to the same content.

Needs covered: N2, N3, N4, N6, N11, N12, N13, N18, N20

F7 – Templates

Description: Documents and content pages may follow some predefined templates. The system should allow creation and evolution of templates. This creation is similar to the creation of page content. Templates may be empty or fully filled.

Needs covered: N2, N3

F8 – Version Control

Description: All documents and content pages must be under a version control system. This allows the system to know the authors and contributors of the content. It must be possible to analyze the changes in the history of each page or document. The difference between each version is also a feature that has to be present, as well as the rollback to a previous version. For documents that aren't created inside the system, in the moment they are upload/inserted, they will be controlled by this version control system. If they are text documents, the same features will be available. If they're not, the version control system will only control the version and authors of each version. Each change to a document or page content must have a brief comment about the reason for changing what was changed.

Needs covered: N4, N6, N7, N23, N24, N25, N26, N28, N30

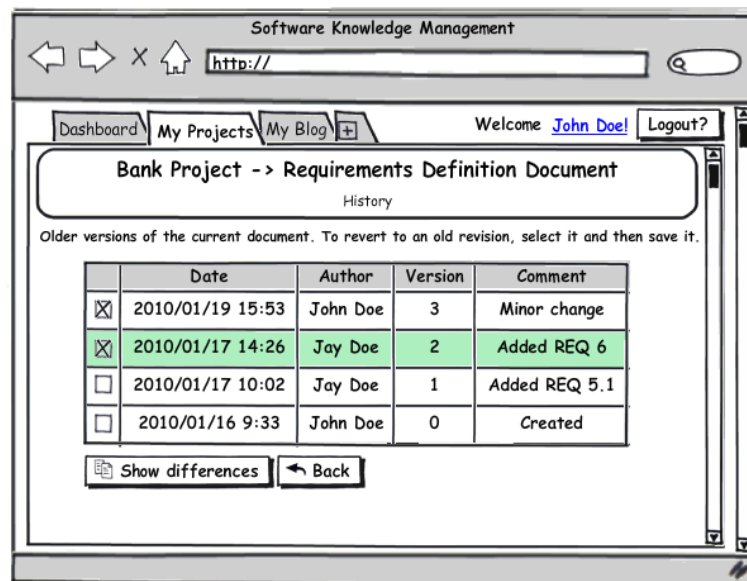


Figure 4.5 - Prototype for the version control feature

F9 – Track users contributions

Description: This feature provides the information about contributions of each user. This information is acquired through an analysis of information stored by the version control. With

Software Knowledge Management using Wikis

that information, it's possible to see if the contributions of a user are essentially in a specific field making him an expert.

Needs covered: N1, N4, N7, N8

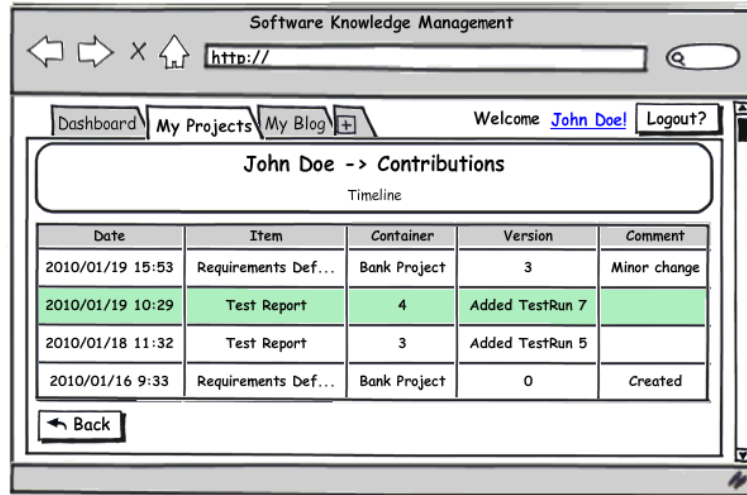


Figure 4.6 - Prototype of the contributions of a user

F10 – Repositories

Description: These repositories allow to organize information/documents about a specific area. They are sections of the system where knowledge about a specific field is stored and available for consultation by users. They must support files of any type of media and content creation through the interface. Some repositories may be used for reuse purpose, having a special structure. Reuse repositories allow the selection of several items to extract them as a package.

Needs covered: N2, N12, N21, N22, N29

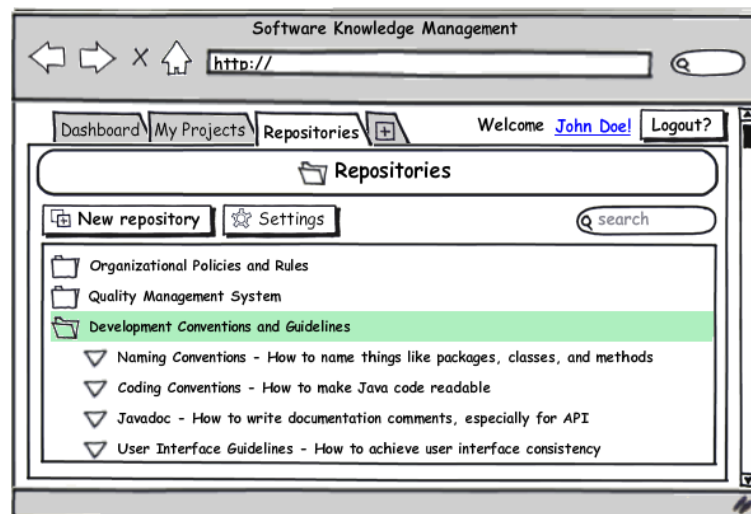


Figure 4.7 - Prototype of the repository feature

F11 – Communities of practice

Description: The system must have a section to manage several communities of practice. Each community must have a forum that allows discussion between members that belong to the community. Members may start new discussion or participate in discussions already opened. They also may decide to follow specific discussions in order to be notified each time that some activities happen in the discussion being followed. The discussion forum also have to support files.

Needs covered: N2, N6, N9, N13, N16

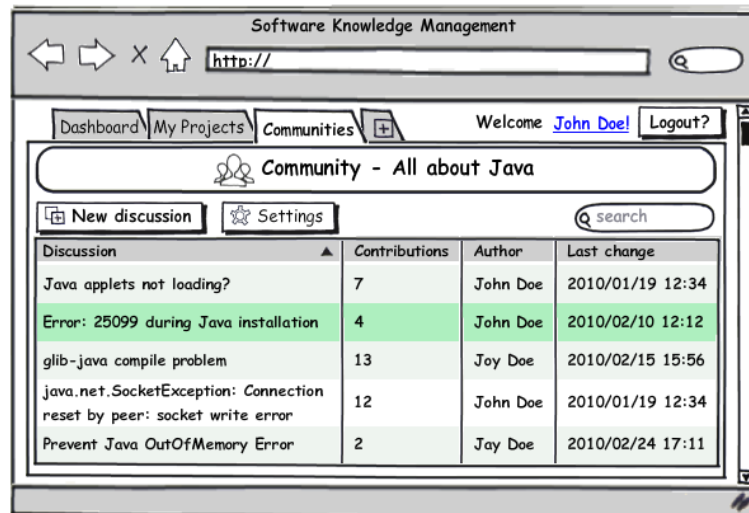


Figure 4.8 - Prototype of one community of practice

F12 – External sources database

Description: A list of external sources must be maintained in the system. Each external source has a description explaining its aim and a connection/link to it. If there is the possibility to import the source for the system and to keep it updated, this action is recommended. All the instructions to access the source must be described. Some sources may require a login and that information must be present in the external source sheet. These sources may come from customers or suppliers repositories or even from training services provided by an external company. These sources may support all kind of information.

Needs covered: N2, N11, N14, N19, N21, N22, N23, N30

F13 – Search

Description: The search feature must have three kinds of search methods. A global search will look for everything in the system, like documents, pages, users, external sources, repositories, communities of practice, blogs, etc.. This global search must provide filters in order to refine the search. This filter includes the selection of information types that the user

wants to find in the results. This global search will also be applied in specific environments, instead of a global search, ie, it will be applied for a specific project, a blog or other section of the system. The second kind of search is a document search. This version helps to find "that document" that we are seeking for. Documents, when created in the system or uploaded to it, are indexed allowing searching them by types, customized parameter or full text. The last kind of search is an expert search. Within this search, users may find colleagues with the expertise needed. This expertise may be identified by the personal profiles of each user, and by the contributions made in documents.

Needs covered: N1, N2, N9

F14 – Workflows

Description: This feature supports the implementation of processes in the system. The system must allow the creation and management of workflows. They represent a sequence of steps performed by employees in some documents, page contents and other items. Each step of workflows has an associated role or specific user that may execute the step. This way, it is possible to implement a review process or any other, by defining the steps and who performs them.

Needs covered: N6, N27

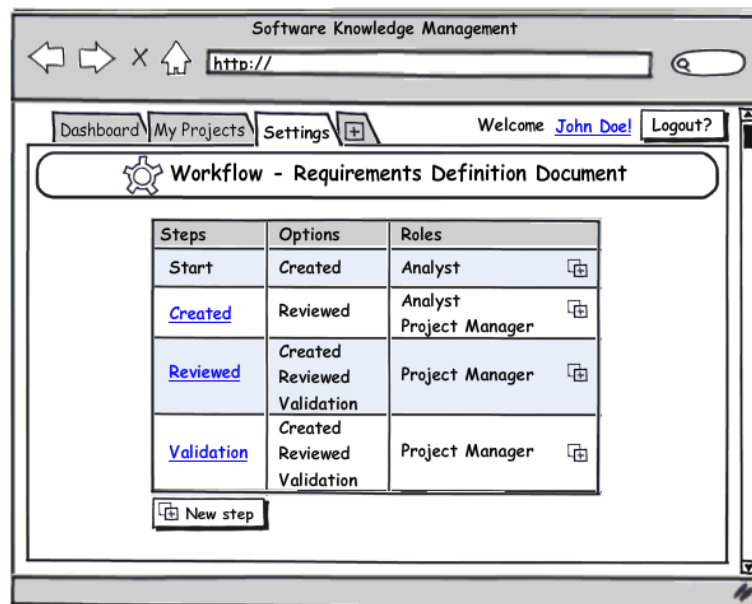


Figure 4.9 - Prototype of a workflow definition

F15 – Skills management

Description: This feature allows the listing of all users and associated skills or expertise. That information provides an overview about the company competences, helping in resources

allocation for projects. The information comes from users profiles and from the analyzed information acquired with users contributions. It must provide views by skills and by users, permitting to know which users have a specific skill or what skills have a specific user. This feature also support the identification of training needs.

Needs covered: N8, N9, N25

F16 – Track stakeholders communication

Description: To track communications made by stakeholders, which is mostly done by e-mail, the system must provide a way to store that information. This information must be stored in the section where the communication belongs, for example, a specific project. That conversation is tracked by creating a specific e-mail for each section where communications are supposed to be sent in. Then, each time users communicate by e-mail, they just need to add the e-mail address of the related section in the “cc” field. Then, the system will organize and store the conversations by the subject field in the respective section.

Needs covered: N2, N14, N15

F17 – Micro blogging

Description: This feature is used to notify users about news, updates or any other information that a company needs to publish to them. Each user will have a window in the system homepage where he can read the notifications. Each message must have a small size, for example, at most 200 characters. Besides receiving the notifications, users also may send new ones. So, everyone can follow other users in order to receive their messages. This allows each user to be aware about what's happening within the company or with a colleague in particular. The same may be implemented for activities or projects. As example, for projects, managers become responsible to administrate notifications. Some can be automatic, like documents upload. This micro blogging feature is similar to the Twitter service (Twitter, 2010).

Needs covered: N2, N6, N9, N17

F18 – Dashboard

Description: The system homepage for each user must include a dashboard. This dashboard is an interface where users may add predefined widgets (feature F19). Those widgets may be added and removed as needed. That interface make users aware of the information related to their needs.

Needs covered: N2, N17, N19

F19 – Widgets

Description: Widgets are small applications that are created to be used inside the system's dashboard. The system must provide a wizard to build simple widgets. A simple widget can be, for example, a window listing the last 5 activities of a project. A manual must be provided on how to build a widget.

Needs covered: N2, N17, N19

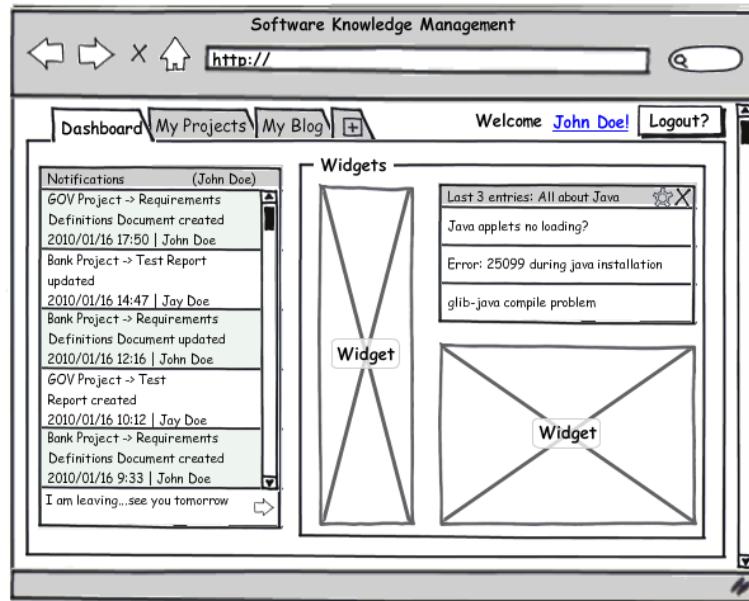


Figure 4.10 - Prototype of a dashboard with microblogging and widgets

F20 – Plug-ins

Description: This feature implies a support by the system to integrate with connectors. Each connector will connect the system with external tools in order to centralize all the knowledge spread by several tools. Development tools are examples of tools that are relevant to integrate in the system through a connector. They are able to retrieve information about the development process. Test tools are another example. They may store information about tests and bugs.

Needs covered: N2, N11, N14, N19, N21, N22, N23, N28, N30

F21 – Tagging

Description: One “tag” is a word that may describe/classify a document or any other type of content. When creating a new knowledge item (document, page content, etc..) the user may chose several words (tags) to describe the item. Then, is possible to classify content with key-words. The author writes the tags that better represents the content, but he also can chose tags

from a list that represents the most used. This feature makes the indexing of content more effective and improves the search.

Needs covered: N2

F22 – Project sections

Description: Assuming that all the software development activities are managed through projects, this section must exist in the system. All the tasks performed and knowledge produced inside the scope of a specific project must be managed and stored in its project section. Inside each project section, other subsections may be created to support and structure the activities performed in the project. Some subsections may be predefined, like requirements, tests, development or planning.

Needs covered: N2, N20, N25, N26

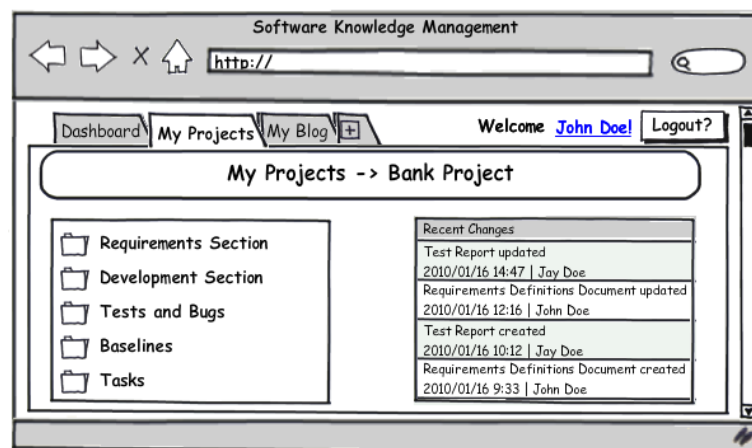


Figure 4.11 - Prototype of a project section

F23 – Baselines

Description: The baseline feature provides the ability to make a “snapshot” of the current status of artifacts, work-products or deliverables of the project. That means that with the creation of the baseline, a package is generated with all the items, in their latest versions, of one project. This feature may also be applied to a subsection of a project. Each baseline must have a description about why it was created.

Needs covered: N26

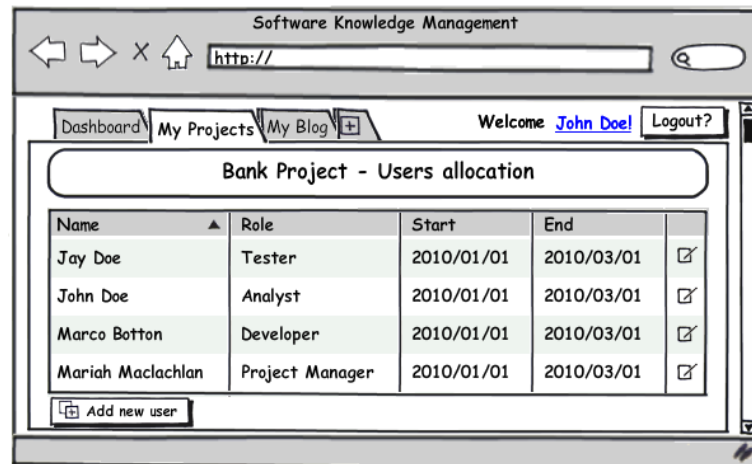
F24 – Users allocation

Description: Within each project, the project manager must be able to allocate the users needed for the project. For each user he specifies the time that they will contribute to the project. In this way the organization knows that those users are not available for other projects.

Software Knowledge Management using Wikis

It is necessary one view by project, where is possible to see which users are allocated and for how long: The view by user is also necessary, where is possible to see the allocation and availability of each user. These features provide information about which skills are been used and which ones are available.

Needs covered: N8, N25



Name	Role	Start	End	
Jay Doe	Tester	2010/01/01	2010/03/01	<input checked="" type="checkbox"/>
John Doe	Analyst	2010/01/01	2010/03/01	<input checked="" type="checkbox"/>
Marco Botton	Developer	2010/01/01	2010/03/01	<input checked="" type="checkbox"/>
Mariah Maclachlan	Project Manager	2010/01/01	2010/03/01	<input checked="" type="checkbox"/>

Figure 4.12 - Prototype of the user allocation feature

F25 – Traceability matrix

Description: This matrix supposes relationships in order to create the traceability matrix. This means that it must be possible to relate several knowledge items with each others. As example, if exists a item called requirement it must be possible to related it with a development item, showing the requirements that are being met. The same happens for any kind of items. This is represented in a table format correlating any knowledge item in a many-to-many relationship. The identifiers of one item are placed in the left column and the identifiers for the other item are placed across the top row. So whenever exists a relation between the identifiers of the left column and the identifiers of the top row, a mark is placed in the intersecting cell. This marks indicates the relation between the two items.

Needs covered: N16, N24, N25

F26 – Knowledge exportation

Description: This feature implements a mechanism to export knowledge items. The mechanism will allow users to export documents or content pages to editable and non editable formats.

Needs covered: N5

F27 – Administration

Description: This feature provides a supervision of all the system. It's supported by defining a role with access to everything, ie, an administrator. It also allows to manage users, like assigning roles to them. It customizes the system, creating roles, new templates, new document types, etc..

Needs covered: N5, N18

In Table 4.1 are mapped the features described and the addressed needs. With the features defined all the needs identified in chapter 3 are covered.

Table 4.1 - Traceability matrix between needs and features

		Needs																															
		N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30	#	
Features	F1	x	x		x			x	x		x			x					x													9	
	F2	x							x																		x					3	
	F3										x									x													4
	F4	x	x							x												x											4
	F5		x				x					x	x	x								x											6
	F6		x	x	x		x					x	x	x							x												9
	F7		x	x																													2
	F8				x		x	x	x															x	x	x	x	x					9
	F9	x			x		x		x	x																							4
	F10		x											x									x										5
	F11		x				x				x				x				x														5
	F12		x										x			x							x	x	x								8
	F13	x									x																						3
	F14							x																									2
	F15									x	x																	x					3
	F16		x													x	x																3
	F17		x					x			x									x													4
	F18		x																	x													3
	F19		x																	x													3
	F20		x									x				x						x	x	x									9
	F21		x																														1
	F22		x																									x	x				4
	F23																																1
	F24									x																		x					2
	F25																			x							x	x					3
	F26						x																										1
	F27						x																										2
#	5	16	2	4	2	6	3	5	5	2	4	3	4	4	3	1	2	4	4	4	4	3	3	3	2	7	3	2	2	1	3	-	

4.2 Wikis and implemented features

Because of its effectiveness, flexibility and simplicity, wikis are widely used by software engineers for the collaborative creation of information and all its management. Although they are already very useful, a better support for software knowledge management can be achieved.

This section aims to identify the most popular wikis in order to make an analysis of the features already provided by them. The features that will be considered are those identified in the section 4.1. To accomplish the objective will be used an online platform for the comparison of wikis, the WikiMatrix. WikiMatrix is a website where it is possible to compare wikis through one matrix. It shows the characteristics and features of each wiki making the comparison easier. Nowadays, there is a big list of options with different applications. It becomes difficult to know which one might best serve our needs. The site has all the relevant information about 123 wikis, and allows to easily find the differences between each one of them. The wiki database includes wikis hosted by third parties or to be installed and also open-source or proprietary (WikiMatrix, 2010).

In Table 4.2 are listed the 25 most popular wikis. This list is provided by WikiMatrix. The objective is to find what are the features, from the section 4.1, that usually are supported in wikis. Each wiki will be analyzed feature by feature, but only with the information present in WikiMatrix and in the site of the respective wiki. No test or verification is made to check the respective features. From the list of 25 most popular wikis, the TiddlyWiki won't be analyzed, because it's purpose is for individual usage.

Table 4.3 summarizes the analysis made feature by feature in each wiki. As expected it's readily apparent that much of the features are widely implemented by most wikis. Looking at implemented features, in 19 wikis are found at least 14 ones, which represents half of them. It means that more than half of wikis implement more than half of the features. On the other hand, there are 9 features that are only implemented by 5 or less wikis and 2 features are not implemented. From the list and the analysis there are several options from where is possible to start building a Knowledge Management System.

Confluence and Foswiki are the wikis that support more features, 21. The main difference between these two wikis concerning the implemented features is that, while Foswiki has a Skills Management feature, Confluence has the Micro Blogging. The fact that Foswiki is free and open source is another characteristic that distinguishes them, because Confluence is a proprietary solution. Besides the analyzed features, the Foswiki supports other useful features like "macros". Macros allow to dynamically compose pages, for example to include other pages, or show a search result embedded in a page. It also supports simultaneous page editing. Different users may do simultaneous edits of the same topic, and then Foswiki merges the different changes automatically. If there is a conflict that cannot be merged automatically, it inserts a "change mark" into the text to highlight conflicts. These conflicts happen only when

Software Knowledge Management using Wikis

one user edits the same part of a topic as someone else. Looking at Confluence, it also has some useful extra features. It has some integrations implemented out of the box. It has a Microsoft Office (<http://office.microsoft.com>) support where users can import, view or edit content. Users who don't have Office installed, can view and interact with the content using Confluence's native Office viewer. Microsoft SharePoint (<http://sharepoint.microsoft.com>) is another integration feature supported and with which is possible to embed SharePoint lists inside a wiki page. This integration also provides a combined search between the two platforms. It also supports an integration with JIRA (<http://www.atlassian.com/software/jira/>) which is a software used for bug tracking, issue tracking, and project management. This integration permits to centralize the information and combine it in Confluence.

Table 4.2 - 25 most popular wikis (WikiMatrix, 2010)

#	Name	URL
W1	DokuWiki	http://www.dokuwiki.org/
W2	MediaWiki	http://www.mediawiki.org/
W3	TWiki	http://twiki.org/
W4	TikiWiki CMS-Groupware	http://tikiwiki.org/
W5	PhpWiki	http://phpwiki.sourceforge.net/
W6	PmWiki	http://www.pmwiki.org/
W7	Confluence	http://www.atlassian.com/software/confluence
W8	MindTouch	http://www.mindtouch.com/
W9	MoinMoin	http://moinmo.in/
W10	XWiki	http://www.xwiki.org/
W11	Foswiki	http://foswiki.org/
W12	JSPWiki	http://www.jspwiki.org/
W13	bitweaver	http://www.bitweaver.org/
W14	WackoWiki	http://wackowiki.org/
W15	TiddlyWiki	http://www.tiddlywiki.com/
W16	Wikispaces	http://www.wikispaces.com/
W17	BusinessWiki	http://onbusinesswiki.com/
W18	WikkaWiki	http://wikkawiki.org/
W19	MojoMojo	http://mojomojo.org/
W20	ScrewTurn Wiki	http://www.screwturn.eu/
W21	PBwiki	http://pbworks.com/
W22	MoniWiki	http://moniwiki.kldp.net/
W23	PukiWiki	http://pukiwiki.sourceforge.jp/
W24	Daisy	http://www.daisycms.org/
W25	Midgard Wiki	http://www.midgard-project.org/documentation/net-nemein-wiki/

Table 4.3 - Wikis and implemented features

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W16	W17	W18	W19	W20	W21	W22	W23	W24	W25	#	
F1 – Login system	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F2 – Users Profile	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	12
F3 – Support Roles or groups of users	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	22
F4 – Personal Blogs	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	13
F5 – Rich text editor	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	23
F6 – Create/Edit Content	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F7 – Templates	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗	17
F8 – Version Control	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F9 – Track users contributions	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	3
F10 – Repositories	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F11 – Communities of practice	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	23
F12 – External sources databases	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F13 – Search	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	24
F14 – Workflows	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	4
F15 – Skills management	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	1
F16 – Track stakeholders communication	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	1
F17 – Microblogging	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	2
F18 – Dashboard	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	6
F19 – Widgets	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	5
F20 – Plug-ins	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	22
F21 – Tagging	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	14
F22 – Project sections	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	23
F23 – Baselines	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0
F24 – Users allocation	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	1
F25 – Traceability matrix	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0
F26 – Knowledge exportation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	18
F27 – Administration	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	22
#	17	14	17	19	15	15	21	17	15	19	21	15	16	11	15	16	15	15	12	13	20	15	13	14	11	

5 Conclusions and Future work

5.1 Goals achieved

The objectives of this dissertation were focused on the following topics:

- Necessities for software knowledge management.
- Features that cover all the necessities identified.
- An analysis of the most popular wikis relating their features and the ones defined in this dissertation

With the literature review it was possible to understand the knowledge management concepts and its implication in organizations. Some of the issues that software organizations have on knowledge management are similar to those that other organizations face. Although, some issues are very specific in the software context.

The research about wikis has supported the benefits that they bring to organizations. As seen, wikis provide a conversational KM solution as a method for knowledge creation. They are attractive due to its easy implementation, low cost installation and because they enhance collaboration between teams, even if geographically distributed. Wikis are easy to use because wiki pages can be created, edited and linked only using a web browser. To create new content, the template support makes it easier to create and update pages keeping the consistence of the content. Wikis keep track on every single version of each page preventing accidental or malicious actions. Users may contribute gradually to the pages building contents in a collaboratively way. The conversational method of interaction is very useful to codify the knowledge, making it explicit. Wikis are also good platforms to be used as a portal that accesses other tools.

Conclusions and Future work

The challenges and advices found in literature made possible the creation of a list of necessities for knowledge management in software engineering. After the list of necessities the next step was to define the functionalities needed to get addressed all necessities.

To achieve the last topic, the 25 most popular wikis were identified and analyzed. Each feature defined in the list was verified on each wiki to know if they implement them. At the end, one matrix as been built comparing the features that each wiki implements.

It was possible to conclude that more than a half of wikis implement more than an half of features. This conclusion meets the idea defined in the chapter 1.1, Context, that wikis by themselves, with the features provided by default, already implement a considerable part of the functionalities needed.

5.2 Future work

Not necessarily a future work, but a continued work is to keep the attention on the evolution of wiki platforms and how they can support knowledge management for software development companies.

The continuation of this work can be done by finding a case study to implement the features defined. The case study implies an organization with a specific software development model with a process where the features will be inserted. After the case study selection, a wiki has to be selected in order to support the implementation of the features. Then the benefits are measured and some conclusions can be made.

Even without a case study, this work can be continued by selecting a type of software development and exploring it. After defining a typical process that implements the model, it should be made a selection of the wiki that best supports the process. Then, the features and needs identified in this dissertation could be integrated in the wiki.

References

- Agresti, W.; (2000) Knowledge management, advances in computers, Academic Press, 53, 171-283
- Alavi, M.; Leidner, D. E.; (2001) Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues, MIS Quartely, 25(1), 107-136
- Argyris, C.; (1991) Teaching Smart People How to Learn, Harvard Business Review, The New York Times Special Features/Syndication, 4(2), 4-15
- Atherton, J. S.; (2009) Learning and Teaching: Experiential Learning, last modified on February 4, 2010, and viewed on February 4, 2010 - <http://www.learningandteaching.info/learning/experience.htm>
- Bjørnson, F. O.; Dingsøy, T.; (2008) Knowledge management in software engineering: Systematic review of studied concepts, findings and research methods used, Information and Software Technology, 50(11), 1055-1068
- Brössler, P.; (1999) Knowledge management at a software engineering company - an experience report, Workshop on Learning Software Organizations, LSO'99, 163-170
- Carvalho, R. B.; Ferreira, M. A. T.; (2006) Knowledge Management Software, Encyclopedia of knowledge management, David Schwartz Editor, Bar-Ilan University, 410-418
- Confluence; (2010) Enterprise Collaboration and Wiki Software – Confluence, last modified on February, 2010, and viewed on January 12, 2010 - <http://www.atlassian.com/software/confluence/>
- Cunningham, W.; (2009) Wiki History, last modified on October 22, 2009, and viewed on January 16, 2010 - <http://c2.com/cgi/wiki?WikiHistory>
- Cunningham, W.; (2010) Wiki Design Principles, last modified on February 3, 2010, and viewed on February 10, 2010 - <http://c2.com/cgi/wiki?WikiDesignPrinciples>
- Davenport, T. H.; Prusak, L.; (1998) Working knowledge: how organizations manage what they know, Harvard Business School Press, 1st Edition

Conclusions and Future work

- Dingsøy, T.; Bjørnson, F. O.; Shull, F.; (2009) What do we know about Knowledge Management? Practical implications for Software Engineering, IEEE Software, 26(3), 100-103
- Earl, M.; (2001) Knowledge management strategies: towards a taxonomy, J. Management Information Systems, 18(1), 215-233
- Gonzalez-Reinhart, J.; (2005) Wiki and the Wiki Way: Beyond a Knowledge Management Solution, Information Systems Research Center, 1-22
- Grace, T. P. L.; (2009) Wikis as a knowledge management tool, Journal of knowledge management, 13(4), 67-74
- Johanson, C.; Hall, P. C. M.; (1999) Talk to Paula and Peter - They are experienced, Workshop on Learning Software Organizations, LSO'99, 171-185
- Junior, J. B.; Coutinho, C. P.; (2009) Collaborative writing tools in engineering education: challenges for knowledge management and sharing, V International Conference on Multimedia and Information and Communication Technologies in Education, 2:1070-1074
- KMCI; (2007) The new KM - What is it?, Knowledge Management Consortium International, last modified on December 4, 2007, and viewed on December 6, 2009 - http://www.kmci.org/the_new_knowledgement.html
- Lawton, G.; (2001) Knowledge management: ready for prime time?, IEEE Computer, 34(2), 12-14
- Leuf, B. & Cunningham, W.; (2001) The wiki way: Quick collaboration on the web, Addison-Wesley Ed. Boston
- Maier, R.; Hädrich, T.; (2006) Knowledge Management Systems, Encyclopedia of knowledge management, David Schwartz Editor, Bar-Ilan University
- Maier, R.; Hädrich, T.; (2006b) Centralized Versus Peer-to-Peer Knowledge Management Systems, Knowledge and Process Management, 13(1), 47-61
- Nonaka, I.; (1994) A dynamic theory of organizational knowledge creation, Organization Science, 5(1), 14-37
- Nonaka, I.; Takeuchi, H.; (1995) The knowledge-creating company: how Japanese companies create the dynamics of innovation, Oxford University Press
- Polyani, M.; (1967) The tacit dimension, Routledge & Kegan Paul
- ProvenModels; (2010) ProvenModels - Double loop learning - Chris Argyris, Donald Alan Schön., last modified on February 5, 2010, and viewed on February 5, 2010 - <http://www.provenmodels.com/5/double-loop-learning/>
- Rus, I.; Lindvall M.; Sinha S.S.; (2001) Knowledge Management in Software Engineering A State-of-the-Art-Report, Data & Analysis Center for Software (DACs)

Conclusions and Future work

- Rus, I.; Lindvall M.; (2002) Knowledge Management in Software Engineering, IEEE Software, 19(3), 26-38
- Senge, P. M.; (1990) The fifth discipline: the art and practice of the learning organization, Currency Doubleday
- Soanes, C.; Stevenson, A.; (2005) Oxford Dictionary of English, Oxford University Press, Publication date: 11 August 2005
- TinyMCE; (2010) TinyMCE, last modified on January, 2010, and viewed on January 29, 2010 - <http://tinymce.moxiecode.com/index.php>
- Twiki; (2010) TWiki - the Open Source Enterprise Wiki and Web 2.0 Application Platform, last modified on January, 2010, and viewed on January 12, 2010 - <http://twiki.org/>
- Twitter; (2010) Twitter, last modified on February 2010, and viewed on February 15, 2010 - <http://twitter.com/>
- Wagner, C.; (2004) Wiki: A technology for conversational knowledge management and group collaboration, Communication of the Association for Information Systems, 13, 265-289
- Wei, C. P.; Hu, P. J.-H.; Chen, H. H.; (2002) Design and evaluation of a Knowledge Management System, IEEE Software, 19(3), 56-59
- Wenger, E.; (2006) Communities of practice, a brief introduction, last modified on June 2006, and viewed on February 5, 2010 - <http://www.ewenger.com/theory/index.htm>
- Wiig, K. M.; (1999) Comprehensive Knowledge Management, Working Paper Revision 2, Knowledge Research Institute, Inc. - http://www.knowledgeresearch.com/krii/downloads/comprehensive_km.pdf
- WikiMatrix; (2010) WikiMatrix - Compare them all, last modified on February 2010, and viewed on February 15, 2010 - <http://www.wikimatrix.org/>