UNIVERSITY OF CALGARY

Inter-Team Learning for Agile Software Processes

by

Thomas Chau

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

MARCH, 2005

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Inter-Team Learning for Agile Software Processes" submitted by Thomas Chau in partial fulfilment of the requirements of the degree of Master of Science.

Supervisor, Dr. Frank Oliver Maurer, Department of Computer Science

Dr. Günther Ruhe, Department of Computer Science

Dr. Mike W. Chiasson, External Examiner, Haskayne School of Business

March 2, 2005

Date

Abstract

Software development is a collaborative and knowledge-intensive process demanding expertise in the customers' business domain and mastery over the technologies required to satisfy the customers' business needs. These demands are common among all software development organizations but are particularly more acute for those that practice agile software methods and contain multiple inter-dependent teams or teams that use similar technologies and processes. To address the lack of support for inter-team learning in agile methods, this work proposes an integrated approach that combines the Communities of Practice concept and an augmented Experience Factory framework. To help support this integrated inter-team learning approach, this work presents a proof-of-concept web-based collaborative environment, MASE. Initial results from two case studies conducted in academic and industrial settings provide insights and evidences for the use of MASE for inter-team learning purposes.

Acknowledgements

This thesis would not have been possible without the support and help of many hearts and minds over the past two and a half years.

First and foremost, I would like to thank my supervisor, Frank Maurer, for his expertise and patience. He has provided me with invaluable insights and motivation throughout the development of this work.

I would also like to thank my fellow research group colleagues. In particular, Lawrence Liu and Harpreet Bajwa for kick-starting the implementation of the process model component of MASE; Carmen Zannier for always being so cheerful during my extended stay at our shared workspace; Grigori Melnik for bringing endless enthusiasm and curiosity and for his sincere feedback on my publications and presentations. I especially want to thank Kris Read for the countless hours he spent on coding this research, for his witty comments, and for making pair-programming such a great experience.

Much appreciation goes towards the students and the empolis employees who have participated in the case studies and have provided invaluable feedback to this work.

Much gratitude also goes towards Joe Poon and Joanna Ha and all my friends for their listening ears and encouragement when I had lost the faith during the course of this work.

Most important of all, I would like to thank my parents, my brother, and especially Maggie for always being there, for their understanding, and for their unwavering support throughout all these years.

Dedication

To my parents, Andrew and Miranda, my brother, Augustine, and Maggie

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

In this thesis, I address the issue of knowledge management—specifically, inter-team learning—in software organizations that practice agile software development methods. I begin this chapter with a discussion on the need for knowledge management in software companies. Next, I analyze agile software processes from the knowledge management perspective and expose the inherent limitations in agile processes that motivate this research. Finally, I present the specific goals of this thesis. I conclude the chapter with a structural overview of this thesis.

1.1 Knowledge Management in Software Engineering

Software development is a collaborative and knowledge-intensive process. It demands expertise in the customers' business domain and mastery over the technologies required to satisfy the customers' business needs. These demands and the complexities in knowledge sharing are exacerbated by environmental factors such as:

1. customers' requirements change frequently and unpredictably as a result of turbulent business environments;
2. new technologies emerge constantly;
3. software teams are often under pressure to deliver products in less time;
4. most of the knowledge in software development is tacit, residing in employees' brains [Rus et al., 2001];
5. high staff turnover rate due to downsizing and outsourcing [Tiwana, 2000]; and,
6. the need for distance collaboration among distributed teams.

The first three environmental factors, in particular, are known as the "home grounds" for using agile software development methods [Boehm and Turner, 2003]. Therefore, agile method practitioners have a pressing need for knowledge management support in their organizations.

Knowledge management, informally, can be defined as a set of strategies that *capture*, *organize*, and *share* both tacit and explicit knowledge of employees so that others in the organization may make use of and *learn* from the knowledge to be more effective and productive [Alavi and Leidner, 1999]. Based on this definition, *knowledge sharing* is seen as a core part of, but not equivalent to, *knowledge management*. Likewise, *knowledge management* is closely related to but not equivalent to *learning*. Whereas *knowledge management* focuses on what can be done with knowledge, *learning* focuses on the impact on individuals when they make use of knowledge. In this thesis, I will use the terms knowledge management and knowledge sharing interchangeably since facilitating knowledge sharing is at the core of knowledge management.

1.2 Knowledge Management in Agile Methods

In agile methods, such as eXtreme Programming [Beck, 1999], Scrum [Schwaber et al., 2001], and Crystal [Cockburn, 2001], knowledge sharing is facilitated by several practices: on-site customer, pair programming and pair rotation, daily stand-up meetings, retrospectives, and cross-functional teams.

The on-site customer practice involves having the customer representatives working as part of the software development team [Beck, 1999]. It is intended to foster learning between the customers and the development team. The on-site customers are responsible for producing acceptance tests for the software and prioritizing system features from the customers' perspective. The significance of this practice is that the on-site customers are to be co-located with the developers. The availability of the customer representatives at the developers' workplace facilitates close interaction between both sides. On one hand, developers can discuss the requirement details with the customers when they have questions about the system features. On the other hand, quickly developed software can be demonstrated to the on-site customers allowing them to directly discover any mismatch between their requirements and the developed software. This direct feedback loop allows knowledge on the system requirements to be acquired and clarified faster than the typical scenario where the customers and the developers

exchange their understandings primarily via requirements specifications and design documents.

Pair programming involves two members of the development team working in front of a single computer designing, coding, and testing the software together [Beck, 1999]. The collaborative nature of the practice makes it a very social process characterized by informal and spontaneous communications. As the pair work together, they share various kinds of knowledge—some explicit but mostly tacit. This includes task-related knowledge, contextual knowledge, and social resources. Task-related knowledge can include knowledge pertaining to the system requirements, coding convention, design practices, technology knowledge and tool usage tricks. Contextual knowledge is background information by which facts are interpreted and used. For instance, knowing from past experiences or "war stories" when to or when not to use a particular design pattern in different coding scenarios. Examples of social resources include personal contacts and referrals. These various types of knowledge are often not documented for many reasons, such as developers being overburdened with other tasks or they deem what they know to be irrelevant or of no interest to others. Such knowledge is often only uncovered via informal and casual conversation [Fitzpatrick, 2001]. Therefore, the social nature of pair programming made it a great facilitator for eliciting and sharing tacit knowledge. To ensure knowledge shared among a pair is disseminated to the entire team, pairs are recommended to rotate from time to time. The social nature of pair programming also helps to create and strengthen networks of personal relationships within a team, and nurture an environment of trust, reciprocity, shared norms and values [Williams et al., 2000, 2002]. These are critical to sustain an ongoing culture of knowledge sharing.

While pair programming sessions facilitate communication within a pair, daily stand-up meetings—Scrum meetings—are intended for facilitating communication within the entire team [Schwaber et al., 2001]. A typical daily stand-up meeting usually last no more than half an hour. During the meeting, team members report their work progress since the last meeting; state their goals for the day; and voice problems related to their

tasks or suggestions to their colleagues' tasks. Such meetings provide visibility of one's work to the rest of the team; raise everyone's awareness of who has worked on or is knowledgeable about specific parts of the system; and encourage communications among team members who may not talk to each other regularly. Team members know whom to contact when they work on parts of the system that they are unfamiliar with.

To reduce potential long communication chains and the ensuing communication errors between the various roles—e.g., customers, business analysts, developers, and testers—who are involved in software development, agile methods practitioners advocate the use of cross-functional teams over role-based teams [Cockburn, 2001]. A role-based team contains only members of the same functional role and information exchange with other functional teams are primarily document-based. In contrast, a cross-functional team brings together individuals of all defined roles; this can help reduce the communication errors between the various roles [Melnik and Maurer, 2004] resulting in more effective knowledge sharing and collaboration which in turn lead to reduced product development time [Haas et al., 2000; Fredrick, 2003].

To foster continuous learning and improvement of the development process, agile methods practitioners advocate the use of retrospectives [Kerievsky, 2001]. Retrospectives are in essence post-mortem reviews on what happened during development except that they are conducted not only at the end of a project but also during the project. Retrospectives facilitate the identification of any success factors and obstacles of the current management and development process. In cases where team members experience setbacks with the current process, such as lengthy stand-up meetings, retrospectives provide the opportunity for these issues to be raised, discussed, and dealt with during the project and not only at the end of project.

1.3 Research Motivation

A common trait among the aforementioned knowledge sharing practices in agile processes is the dominant role social interactions play in them. While the social nature of the practices made them great at tapping tacit knowledge and in fostering the creation and

reinforcement of the relationship networks within a team, there are two inherent limitations in these practices.

Firstly, all the aforementioned practices, in their original forms, rely on face-to-face communication which restricts the use of them to co-located and small teams, usually with less than twenty people [Beck, 1999]. Unfortunately, for many reasons, it is sometimes impossible to co-locate an entire team and sole reliance on informal knowledge sharing will present challenges.

Secondly, the above practices only facilitate the sharing of knowledge within a team but not across multiple teams within an organization [Kähkönen, 2004; Doran, 2004]. This inter-team learning (also known as organizational learning in management science literature) issue can have a significant impact on the competency, productivity, and product quality of those software companies with multiple inter-dependent teams or teams using similar technologies and processes. This is because there may exist in such a company multiple teams that face common problems or have overlapping interests in specific knowledge areas. The need for organizational learning in such companies is supported by findings from studies which indicate that software development teams are often prone to either repeat past mistakes [Basili et al, 2001] or duplicate effort in "re-inventing the wheel" as they are not aware of experiences acquired or successful solutions adopted from other teams within the organization [Brössler, 1999]. A common attempt to address this inter-team learning issue in software companies that practice agile methods is to transfer workers from one work team to another [Simons, 2004]. However, this is challenging due to cost and is slow due to time constraints.

1.4 Thesis Problems

In software engineering, previous research on knowledge management include those that discuss novel inter-team learning strategies encompassing both organizational processes and tool support; those that propose models and frameworks for comparing these strategies; as well as those that report on experiences and effectiveness of applying these strategies. Most of these works are conducted before agile methods were being broadly

discussed; as such there is relatively little research that investigates inter-team learning in the context of agile methods. Likewise, there is little research done that focuses on inter-team learning in the agile methods community. Hence, the following problems represent knowledge that prior research has not yet addressed and are examined in this thesis:

1. It is unknown to the research community what constitutes the appropriate inter-team learning approach for software organizations that practice agile methods;
2. It is not known what the requirements are for a knowledge sharing tool that is to facilitate inter-team learning in an agile software development organizational setting;
3. It is not known if existing tool support for inter-team learning is appropriate for agile methods practitioners as there has been no prior work in investigating the appropriateness of existing inter-team learning approaches for software organizations that practice agile methods; and,
4. Shall there be a need for a novel knowledge sharing tool, it is not known if such a tool will be useful. After such a tool is designed and implemented, it may then be both possible and desirous to evaluate the extent to which the tool is successful in supporting inter-team learning in software organizations.

As agile methods are increasingly adopted in large companies [Aleksiuk and Wiebe, 2003; Cockburn and Highsmith, 2001; Palmer and Felsing, 2002; Poole and Huisman, 2001; Rumpe and Schröder, 2002; Schwaber et al., 2001; Stapleton, 1997], there is a need for agile methods practitioners to address the lack of support for organizational learning in the processes. By complementing agile methods with support for organizational learning, the use of agile methods can be extended to a larger context.

1.5 Thesis Goals

In this thesis, I will address the aforementioned problems with the following goals:

1. I will investigate inter-team learning approaches that are appropriate for software organizations practicing agile methods by analyzing existing inter-team learning

approaches found in the software engineering communities and in other disciplines;

2. I will outline a set of criteria for the design of an inter-team learning tool tailored for agile method practitioners by analyzing the technical implications of the inter-team learning approach that is appropriate for agile methods practitioners;
3. I will design a proof-of-concept tool that agile methods practitioners can use to facilitate inter-team learning in their companies; and,
4. I will perform an empirical evaluation on the effectiveness of the proof-of-concept tool in practice.

This research focuses on a sub-area overlapping the fields of *knowledge management* and *software engineering*. The study of knowledge management looks generally at issues, processes, and tools for enhancing the use of knowledge [Rus et al., 2001, pp. 18] whereas the study of software engineering concerns issues, processes, and tools regarding software development [Abran et al, 2004]. Within the area of knowledge management, *inter-team learning* represents a subset of works that looks specifically at knowledge management issues in an organizational setting. Some research in inter-team learning focuses on software organizations and these works represent a sub-area that overlaps with the software engineering discipline. Hence, this sub-area is also known as *inter-team learning for software engineering*. My research concentrates on the investigation of inter-team learning issues in software organizations that practice agile processes, and the associated issues in tool support. Since agile software processes and inter-team learning are within the broader research disciplines of software engineering and knowledge management respectively, the outcome of this research will provide a better understanding to these two respective fields. Figure 1.1 summarizes the context and scope of my research.

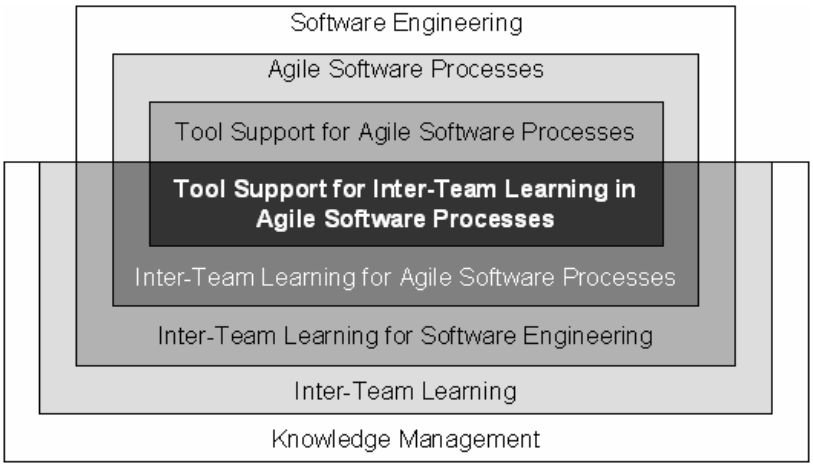


Figure 1.1: The research context and scope of this work

1.6 Structure of Dissertation

The remainder of this thesis is organized as follows: Chapter 2 presents an overview on various approaches to organizational learning. Chapter 3 examines available tool support for knowledge sharing in terms of how well they support these various approaches and how well they address the essential requirements for tools that facilitate knowledge sharing and organizational learning in agile software organizations. In Chapter 4, I present the design of the proof-of-concept tool, MASE. Following that, I discuss in Chapter 5 a qualitative analysis of MASE. Following that, I discuss in Chapter 6 an empirical evaluation of MASE. Chapter 7 concludes this work and suggests possible future work in the area of organizational learning for agile software organizations.

Chapter 2. Inter-Team Learning Approaches

In this chapter, I review existing research on organizational learning. First, I discuss the Experience Factory framework that originates from the software engineering community. I describe how organizational learning is intended to be achieved in this framework, its potential benefits, and its usage in practice. Based on the analysis of these case studies, I present related work in the organizational science discipline: the codification and personalization approaches, particularly the Communities of Practice concept. For each of these approaches, I examine their intended inter-team learning dynamics, potential benefits, limitations, and their compatibilities with agile methods. Drawing on my analysis, I propose an approach to inter-team learning for agile software organizations.

2.1 The Software Engineering Perspective

In software engineering, a key reported work to address the need for inter-team learning in software organizations is Basili's Experience Factory framework [Basili et al., 1994]. In this framework, a software organization consists of one organizational unit called the experience factory and many project organizations—each of which is responsible for delivering software products to their customers.

At the beginning and during the execution of a project, a project organization provides the project's characteristics to the experience factory such as information about its customers, the desired products, and competencies of the project team, etc. Using this information as well as data stored in the experience base, the experience factory provides to the project team packaged experiences that are tailored from similar projects in the past. Examples of such packaged experiences include characteristics about the past projects, tools, reusable product components, process models, resource models, and quality models, etc.

Similarly, during the project execution and at the end of a project, the experience factory obtains feedback from the project organization. These feedbacks may include lessons learned, data about the development process, resource utilization, quality records, and actual performance of the models that are provided by the experience factory and are used by the project. The experience factory analyzes this project-specific data in two phases. In the first phase, the experience factory draws conclusions about the project-specific experiences and feeds them back to the project organization. In the second phase, it analyzes the project-specific experiences in terms of their applicability to other ongoing and future projects, generalizes them as much as possible, and stores them in the experience base to facilitate planning and support for future projects.

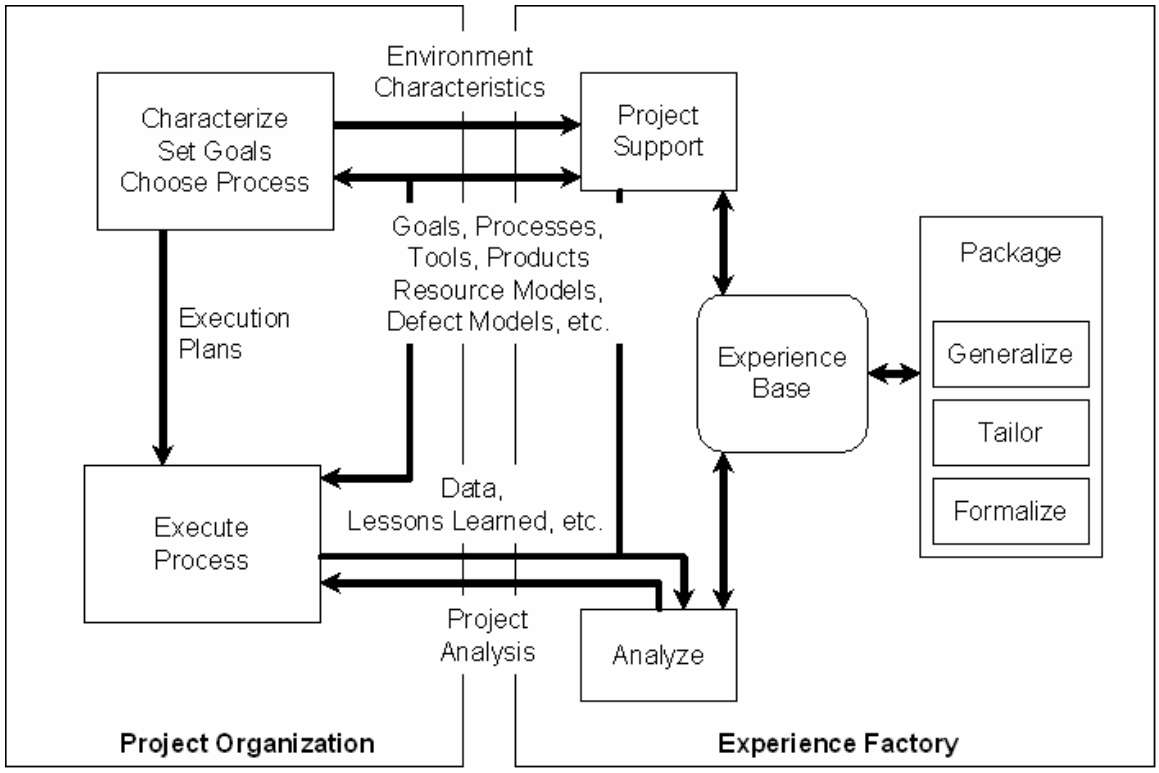


Figure 2.1: The Experience Factory framework (adapted from [Basili et al., 1994])

2.1.1 Potential Benefits

In principle, the potential benefits of the Experience Factory approach are manifold.

First of all, by establishing the experience factory as a separate organizational unit that focuses only on the logistics for facilitating inter-team learning, the framework helps

project teams in an organization to learn from their own experiences without burdening project teams with the effort to create reusable experiences packages in addition to their regular product development effort.

Second, its two-phase analysis of project-specific experiences provides opportunities for process improvement at both the project and the organizational levels.

Third, standardization of knowledge and practices that are deemed relevant for the entire organization is made easier since the tasks of collecting, analyzing, generalizing, storing, and disseminating knowledge are all centralized in one organizational unit.

2.1.2 Potential Drawbacks

On the other hand, there are also potential drawbacks to the Experience Factory approach. The experience factory unit resembles conventional knowledge-oriented structures such as corporate universities, centres of excellence, and process groups [Rus and Lindvall, 2002]. A drawback of these organizational forms is that, in practice, members of such groups are usually specialists and are not the practitioners who would put the knowledge to use, and thus may be perceived by the line employees as living in an ivory tower and have their recommendations be devalued or dismissed [Houdek et al., 1998].

2.1.3 Case Studies

In practice, the Experience Factory framework can be implemented in various ways. This is because the framework only describes what knowledge management tasks need to be done but not how these tasks—experience elicitation, analysis, generalization, formalization, packaging, customization, and dissemination—are to be done. In fact, Basili defines the Experience Factory's role is to transfer "*experiences by building **informal** and **formal** or schematized, and product-sized models and measures of various software processes, products, and other forms of knowledge via **people**, **documents**, and **automated support***" [Basili et al., 1994, emphasis added]. Therefore, in practice, different organizations adopt and adapt the Experience Factory framework according to their business needs and organizational constraints.

For instance, at NASA [Basili et al., 1992; Liebowitz, 2002], employees' experiences and expertise are elicited both formally via controlled processes like exit interviews and metrics collection initiatives as well as informally via special interest groups. Similarly, two approaches of opposing nature are used for experience packaging and dissemination. On one hand, employees' tacit expertise is made explicit into documents (e.g., narrative case studies, videos, lessons-learned CDs, etc.) or statistical models (e.g., cost and defect models, process models, etc.), and employees are referred to these documents when they need to access particular expertise for their tasks. On the other hand, NASA also uses a human-oriented approach to facilitate sharing of tacit experience among employees via person-to-person contacts. This is achieved by putting in place processes like inter-team knowledge sharing forums as well as tools that provide information about employees' competency areas and their contact information.

Compared to NASA, Ericsson's implementation of the Experience Factory framework places a much stronger emphasis on "*verbally communicating valuable experience and de-emphasizes the practice of collecting lots of data*" [Johansson et al., 1999]. Ericsson achieves this by nurturing social networks of "experience brokers"—employees who mediate referrals to other employees with the expertise needed by the knowledge seekers. From the perspective of knowledge management tasks, it is through these person-to-person contacts instead of through tools like sophisticated knowledge repositories that the majority of expertise is captured, analyzed, generalized, customized, and disseminated.

This human-oriented approach is also adopted at sd&m, a large software firm in Germany [Brössler, 1999]. The difference between the Experience Factory implementations at Ericsson and sd&m is that at Ericsson, all the knowledge management tasks are perform by the experience factory staff regardless of the domains of the requested knowledge or experience. In contrast, at sd&m, each experience factory staff is only responsible for one specific knowledge domain and s/he is responsible for maintaining an experience base related to his/her assigned knowledge area.

Within the research division at DaimlerChrysler [Houdek et al., 1998; Wieser et al., 1999], its Experience Factory implementation involves the use of various contrasting approaches, similar to that in NASA. In particular, DaimlerChrysler prefers to rely on controlled and formal processes like focused interviews and metrics collection initiatives to elicit potential reusable experiences (qualitative and quantitative) from its employees. In terms of experience packaging, DaimlerChrysler has experimented with both structuring the experience packages into some predefined formats as well as allowing submitted experience packages to evolve organically into structures that the end-users agree on. Unlike NASA, however, DaimlerChrysler relies mostly on person-to-person contacts such as meetings and trainings for disseminating experience packages as it observes that "*not all valuable pieces of experiences will ever be documented; a large part will remain in the heads of people*".

2.2 The Organizational Science Perspective

These various ways of implementing the Experience Factory or any other inter-team learning strategies in general, can be classified by organizational science researchers [Hansen et al., 1999] into two broad approaches: the codification approach and the personalization approach.

2.2.1 The Codification Approach: Its Benefits & Drawbacks

The codification approach views knowledge as an object which can be collected, stored, transmitted, reused, and created [Alavi and Leidner, 1999]. As such, it believes that effective management of knowledge is to be achieved by organizational processes that transform employees' tacit experiences into explicit and structured forms such as models or documents, and by tools that connect employees to these documents. This approach has the following benefits:

1. **Reduce the likelihood of knowledge loss**. Most of the knowledge used in software development is tacit [Rus et al., 2001] and it is unreliable to depend solely on human memory for storing the vast amount of knowledge relevant for

software development. By capturing the tacit knowledge of the employees into explicit forms, the risks of knowledge loss due to attrition can be mitigated.

2. **Enable collaboration in a space- and time-independent manner**. Provided that the externalized knowledge captures all the information needed by the knowledge seekers, the end-users can access the externalized knowledge at any place and at any time regardless of the availability of the expertise providers. This is especially advantageous for distributed software development efforts.
3. **Facilitate efficient search and retrieval**. If the externalized knowledge is stored in a highly structured manner according to particular schema or ontology, the search and retrieval of such externalized knowledge can be made very efficient with the support of artificial intelligence retrieval techniques such as case-based reasoning and similarity-based retrieval [Davenport and Prusak, 1998; Henninger, 2003].

Despite the above benefits, there are also challenges in adopting the codification approach:

1. **Not all types of knowledge can be made explicit.** As mentioned above, most of the knowledge needed for software development is tacit but capturing this tacit knowledge and converting it into explicit forms is not a trivial task. Researchers in the social psychology and behavioural science fields discover several limitations when humans try to elicit their expertise [Hinds and Pfeffer, 2003]. First, from the cognitive perspective, humans tend to store their expertise mentally in an abstract fashion and forget the mundane details. Hence, people often have difficulty in converting their expertise from their abstract mental representations to a level of representation that is detailed enough for novices to understand and apply. Even if the experts can elicit their expertise in a way that novices can understand and apply, that probably demands great time and effort on the part of the experts. The second limitation relates to the fact that even if the experts have the capability in eliciting their expertise, they may be reluctant to do so for reasons such as the organizational culture is not conducive to knowledge sharing. Due to these cognitive and motivational limitations on the parts of

humans to explicate their expertise, knowledge elicitation cannot happen or happen efficiently.

2. **Externalized knowledge is incomplete.** A corollary of the fact that not all types of knowledge can be made explicit efficiently, if at all, is that what is being captured in explicit forms is often incomplete [Hildreth, 2004]. One particular type of knowledge that is often omitted in explicit forms is contextual knowledge; experiences and context-related experiences by which decisions are made and explicit knowledge is interpreted, used, and communicated [Orr, 1996; Brown and Duguid, 1998]. In software development, one example of contextual knowledge is knowledge from past experiences or "war stories" of when to or when not to use a particular software design pattern in different coding scenarios. This type of knowledge is often not documented, mostly because people deem what they know presently or from the past to be irrelevant or of no interests to others; and such knowledge is often only revealed via informal and casual interaction [Fitzpatrick, 2001]. The importance of contextual knowledge, the difficulty in capturing it in explicit form, and the effects of its omission from externalized knowledge are illustrated in Hewlett-Packard's experiences [Brown and Duguid, 1998] and others [Hansen et al., 1999]. At HP, the company spent much effort in identifying "best practices" throughout the organization and captured them into explicit forms; only to discover that a "best practice" for teams at one site did not turn out to be the "best practice" for teams at another site.
3. **Explicit knowledge easily becomes outdated.** Even if all the knowledge that is needed by the intended recipients can be completely captured, there still exists the risk of the externalized knowledge quickly becoming out of date; the main reason being that humans learn continuously and generate new knowledge much faster than it can be made explicit. While it may be possible to codify this updated knowledge, the effort of continuously updating the out-of-date knowledge may outweigh the potential benefits. This challenge is especially evident in software engineering where there is a need to deliver the latest knowledge to the developers to support their daily work due to increasingly short product life cycles

and frequent changes in software development technologies [Feldmann and Rus, 2003]. In fact, in a recent case study on how software engineers use documentation, close to 70% of the subjects indicated that knowledge embedded in various types of documentation is always outdated compared to the current state of a software system [Lethbridge et al., 2003].

2.2.2 The Personalization Approach: Its Benefits & Drawbacks

In comparison to the codification approach, the personalization approach views knowledge as deeply embedded in work practices and social relationships; that knowledge cannot be easily collected, transmitted, reused, and created without social interactions between the knowledge providers and seekers. Hence, the personalization approach is characterized by processes and tools that promote knowledge sharing via person-to-person contacts; connecting employees to one another rather than to knowledge in their explicit forms.

This human-centric approach to learning is exemplified by the Communities of Practice framework [Lave and Wenger, 1991]. In their work, Lave and Wenger view an organization not only as one consisting of multiple teams that are defined by an official organizational structure; but also as a constellation of communities of practice. Based on their studies, Lave and Wenger come to define a community of practice (CoP) as a group of people, who have a shared interest in a common knowledge *domain*; who advance their knowledge in that domain via interactions with members of the same community and via joint development of a shared repertoire of resources that becomes the community's *practice* [Wenger et al., 2002].

In practice, a CoP can take many forms. It may be small or big, with membership ranging from only a few to a thousand. It may last for a short time or a very long time. Its members may be co-located or distributed, as in the case of the community of open source software developers who design the Linux operating system [Linux]. In terms of its membership composition, a CoP may be homogenous, in which members are of the same role, or heterogeneous, in which members share a common interest in a knowledge domain and not much else. Its membership boundary may stay within an organizational

unit, or span across organizational units, or even span across organizations. As for community structure and activity, a CoP can be highly informal, in which there is no hierarchy and the community has no specific deliverable, or highly formal, in which there are defined roles and defined routines. With respect to status, a CoP may be unrecognized or institutionalized in an organization's official hierarchy.

Although any given CoPs may vary on the aforementioned dimensions, all CoPs share in common in terms of the nature of their learning dynamics. Within a CoP, members gain knowledge on their domain primarily by interacting with each other. These interactions are generally very social-oriented and they exist in various forms: collaborative activities, spontaneous discussions, and informal gatherings. During these interactions among community members, knowledge of various kinds are *elicited*, *shared*, and *evolved*. Typical examples include: knowledge about the problem domain, attempted solutions, and the practice itself; contextual knowledge such as stories, conventions, and lingo within the community; as well as social resources like peer-recognized experts within the community and relevant people to contact. As one is expressing his/her experiences or knowledge amidst these interactions, those personal experiences or knowledge can become collective knowledge and vice versa. It must be noted though that this transformation of personal knowledge to collective knowledge, and vice versa, relies heavily on the community members' participation in these social interactions. It is also by the members' participation in these interactions that a CoP generates new knowledge and mitigates the risk of knowledge loss due to the departure of established community members.

While all CoPs rely on their members' participation in these social interactions for knowledge sharing and creation, Wenger also observes from his studies of effective communities of practice, that knowledge sharing via these process of social participation are in reality, and need to be, complemented with reification processes [Wenger, 1998; Hildreth, 2004]. The reification processes are essentially processes through which community members explicate knowledge of their own as well as their collective knowledge in various forms which becomes part of the community's practice. Explicated

knowledge may take the forms of concepts, stories, procedures, tools, and artefacts. The significance of the reification processes is that through the explicated knowledge, community members have a common baseline knowledge on their problem domain and a common ground for communication, coordination, and collaboration. The need to explicate knowledge is similar to the codification approach. Yet unlike it, Wenger discovers that this reification process is seldom performed in isolation but is often performed together with the social participation process. In fact, Wenger argues that this is necessary because knowledge consists of a tacit and an explicit aspect and that explicated knowledge can only capture the explicit aspect. Hence, Wenger deems the community members' participation in the social interactions among each other to be critical for recovering the tacit aspect of knowledge and for negotiating the true meaning of the explicated knowledge.

For facilitating knowledge sharing and creation across different CoPs within an organization, these two processes of social participation and reification play the same important roles. In particular, Wenger suggests that learning across two different CoPs can be enhanced by the participation of those who are members of both communities. As these members participate in their respective communities, these members can shed new insights on that community's existing collective knowledge using their knowledge of the other community.

In principle and in practice, the concept of CoP has the following benefits and capabilities:

1. **A relatively more effective organizational structure for managing knowledge**. In an organization, people are often grouped together based on their functional roles, or the projects they are working on, or both (matrix organizations). While it is true that a lot of learning happens in these forms of organizations, these social structures are not optimal for addressing knowledge-related issues because the knowledge generated and experience gained can be limited in scope and are easily lost. With functional-based organization, functional departments are focused on their own functional area (customer service, design, testing, etc.), so their

knowledge often remains localized and limited in scope. In comparison, members of a CoP are bound by a shared interest in a common knowledge domain. Members may span across multiple functional organizations; knowledge generated or experience gained within such CoPs are not restricted to the vision of one single functional unit. With project-based and matrix organizations, the primary priority of project teams are on product delivery; learning is usually sacrificed for product delivery when the two goals compete against each other. In addition, project teams are temporary, so their knowledge is largely lost when they disband. In contrast, a CoP is formed by people who have a common desire to advance their practice, thus learning is the primary objective of a CoP. The informal nature of CoPs also provides members with the liberty of setting the community's objective and agenda themselves. Furthermore, members of a CoP are not bound by the projects that they are working on; hence, a CoP is more likely to have a longer lifespan than a typical project team, especially in dynamic business environments where projects come and go. Since a CoP is formed and run by practitioners, the knowledge and experience generated within a CoP is also less likely to be belittled or dismissed by fellow practitioners. Based on these various dimensions such as its lifespan, its learning-focused philosophy, and its membership composition, a CoP can be a relatively more effect organizational structure for managing knowledge.

2. **Promote synergies across business units**. Since members of a CoP are bound by a shared interest in a common knowledge domain and not by their functional roles. Therefore, a CoP has the potential of linking staff that are interested in similar knowledge domains but dispersed across organizational units, as seen in DaimlerChrysler [Haas et al., 2000]. A CoP can also help connect and coordinate isolated initiatives addressing similar knowledge areas; thereby reducing the amount of duplicated effort [Brown and Duguid, 1998].
3. **Better in managing tacit knowledge**. Previously, it is pointed out that not all types of knowledge can be efficiently captured in explicit forms, if at all, and that these types of tacit knowledge such as contextual knowledge are often only

revealed via person-to-person interactions. Since person-to-person interactions are the predominant form of interaction within a CoP, a CoP is more capable than codification-based approaches for sharing, and evolving tacit knowledge.

4. **Means for reinforcing relationship network among employees**. A CoP usually builds on existing social networks among those who are passionate about a common knowledge domain [Wenger et al, 2002]. As these CoP members interact with one another, they not only exchange their insights and experiences about the CoP's domain, but also information about themselves [Fitzpatrick, 2001]. It is exchanges of these kinds of social knowledge that help to further nurture a sense of trust, reciprocity, shared norm and values among community members. In an organizational setting, the development of these social capitals is critical for nurturing an environment of knowledge sharing, and the social interactive nature of CoP helps to facilitate that.
5. **Increase motivation for knowledge sharing**. In a CoP, all members join the community voluntarily and they all share common goals. As such, they are more likely to motivate themselves in constantly evolving and enriching their community's collective knowledge and practice. Within an organization, it must be noted that such self-motivation can be facilitated or extinguished easily depending on an organization's culture and policy [Wenger et al., 2002].
6. **Increase retention of talent**. Successful inter-team learning hinges on an organization's ability to retain its staff [Demarco and Lister, 1999]. In organizational science literature, it is recognized that involvement in professional relationships and being able to develop strong social ties with co-workers are highly regarded by employees for deciding to remain in an organization [Cappelli, 2000; Dessler, 1999]. Communities of Practice can help achieve the first goal in the case of those CoPs whose knowledge domains overlap with their members' professions. Such a CoP can help its members develop professionally besides reinforcing the social ties among them. By participating in the interactions in the community, members of such CoPs are kept abreast of ongoing developments in their field and they can benchmark their expertise against that of colleagues in

other organizational units or even in other organizations. By providing employees the opportunity to be involved in professional relationships with colleagues and to develop strong ties with them, CoPs can help an organization in retaining its staff.

Despite the above benefits, the concept of Communities of Practice also has the following drawbacks and potential pitfalls:

1. **Difficult to facilitate**. The difficulty in facilitating CoPs stems primarily from their informal nature. Since the activities of a CoP can be highly informal or spontaneous, it can be hard to be aware of its presence outside the core group of the CoP. The fact that membership in a CoP is voluntary also means that it is tough to define the boundary of the community, which makes identifying the community difficult. For these reasons, CoPs can be difficult to support. Support for CoPs is also complicated by the fact that the "*growth of a community of practice depends primarily on the voluntary engagement of their members and on the emergence of internal leadership*" yet without "*intentional cultivation, the communities that do develop will depend on the spare time of members, and participation is more likely to be spotty, especially when resources **(e.g., time)** is lean*" [Wenger et al, 2002, emphasis added].
2. **Insular from dissimilar CoPs**. Among the many problems confronting organizations, there are those that require solutions which are not confined to any one practice, or even to a single organization. This necessitates knowledge sharing across CoPs who may be dissimilar in terms of their domains, communities, and practices. Dissimilar domain implies different interests and perspectives; dissimilar community entails members unfamiliar with each other which make interactions difficult due to the lack of trust; dissimilar practice implies different concepts, lingo, experiences, and standards of performance. All these dissimilarities make knowledge sharing across dissimilar CoPs difficult [Brown and Duguid, 1998; Wenger et al., 2002].
3. **Experts overloaded with recurring information needs**. In a CoP, members learn mainly by being situated at the practice and by participating in interactions with fellow community members. For newcomers to get up to speed with the

community's baseline knowledge, it is important for them to have access to the more established community members. Yet without explicating enough of the insights the community has developed, these more established members of the community may be overwhelmed by recurring information needs which will prevent them from focusing on advancing the community's existing knowledge.

4. **Insular from similar CoPs**. It is not uncommon that a knowledge domain attracts the interests of more than one CoP. However if a CoP is too passionate about this particular domain to the extent that it vies for exclusive ownership of the domain, the arrogance of the CoP will prevent it from communicating, collaborating, and learning with other CoPs that are also interested in the same knowledge domain. This phenomenon is very common in the information technology industry where different groups of companies often propose different standards for designing software systems even though the standards are intended for the same purpose, as in the case of voice-enabled software [Lyman, 2004].
5. **Insular from new or peripheral CoP members**. While the social interactive nature of a CoP helps to develop and strengthen the relationship among its members, overly tight bonds within the CoP's core group or the CoP in general may stifle new ideas or constructive criticisms causing the community's knowledge and practice to remain stagnant. Overly tight relationships within a CoP may also hinder participation from peripheral or new CoP members. Participation from peripheral or new CoP members, however, is vital for the continual innovation of a CoP's practice as these members can bring new insights to a CoP's existing collective knowledge. It remains unknown how much cohesion within a CoP is required for the healthy development of a CoP and how much cohesion is "too much".
6. **Incompatible CoP goals with organizational needs**. Since a CoP is often unrecognized within, or even invisible to, the organization and the fact that CoP members can set the community's agenda themselves, it is very possible that a CoP pursue its own agenda with little regard for what the organization really needs in terms of expertise and capability.

2.3 Appropriateness with Agile Culture

From the above discussion, it is clear that various strategies (e.g., the Experience Factory, the codification approach, and the Communities of Practice) are available for organizations to select from for facilitating inter-team learning; each strategy having its benefits and drawbacks. The implication of this for an organization is that its choice of inter-team learning strategy cannot be made thoughtlessly. Emphasizing an approach that is incompatible with an organization's competitive strategy and culture not only will not bring benefits for the organization; it may, in fact, be detrimental to the organization as suggested by the case studies reported in [Hansen et al., 1999].

2.3.1 Does the Experience Factory fit with the Agile Culture?

In the case of the Experience Factory, it is incompatible with the culture of a software organization that practices agile methods in the following respect. According to the description of the framework, the experience factory unit and the various project teams in the organization form a hierarchical relationship. In this hierarchical relationship, the flow of knowledge that is deemed to be relevant for more than one project team must channel through the experience factory unit. While this centralized nature of the experience factory unit is good for standardizing knowledge, in doing so, the experience factory unit also risks being the bottleneck in the sharing of knowledge [Neus, 2001]. This risk is particularly exacerbated in an environment where there is an urgent need for time-critical and highly relevant knowledge; a typical environment that agile software organizations often find themselves in [Boehm and Turner, 2003]. Hence, if the Experience Factory framework is to be adopted in an agile software organization, adaptation of the framework needs to be made.

2.3.2 Does the Codification Approach fit with an Agile Culture?

While there are various strands of agile methods, they all share in common in their embrace of the values and principles found in the Agile Manifesto [Agile, 2001]. For those organizations that practice agile methods, these values and principles are reflected in their organizational cultures as well [Robinson and Sharp, 2004]. One such principle is

the belief that knowledge is socially constructed, collectively held, and can be efficiently tapped via face-to-face interactions. On the other hand, agile methods practitioners do not reject documentation entirely as a knowledge sharing medium. In fact, agile methods practitioners advocate the use of "lean and mean" documentation when there is a need and it is clear that documentation satisfies such a need the best [Jeffries, 2001].

In contrast, it appears that the codification approach, with its view of knowledge as object and emphasis on vigorously explicating knowledge, is in direct conflict with the agile culture, particularly the notion of "lean and mean" documentation. This is especially the case for agile software organizations that are situated in those "sweet spot" environments ideal for agile methods [Boehm and Turner, 2003]. Under such settings, strict use of the codification approach is going to undermine the effectiveness of agile methods. On the other hand, not all agile software organizations are situated in those ideal "sweet spot" environments, as in the case of distributed development [Kircher, 2001]. In those cases, it is desirable for those agile software organizations to integrate the codification approach with the practices of agile methods in a non-invasive and lightweight manner [Greening, 2001]. To that end, those agile software organizations often use basic tools such as static project web sites, Wikis [Leuf and Cunningham, 2001], and newsgroup for codifying their knowledge and experiences [Kerievsky, 2001; Simons, 2004]. These tools and the role of tool in supporting inter-team learning will be examined in details later in Chapter 3.

2.3.3 Does Communities of Practice fit with the Agile Culture?

With Communities of Practice, many of its ideas are aligned with the values and principles advocated in the Agile Manifesto. First, both see tacit knowledge embedded in each individual employee and those embedded in the relationship network among employees not as a liability but as an asset. Second, both approaches encourage self-organizing processes. Third, both emphasize the use of person-to-person interactions as the primary learning mechanism. In addition, the CoP theory also suggests the occasional use of the codification approach as a complementary measure.

2.4 The Need for an Integrated Approach

From the above analysis, it is apparent that the self-organizing Communities of Practice framework is more in harmony with the agile culture than the Experience Factory framework and the codification approach. In spite of this, Communities of Practice alone may not be sufficient for supporting inter-team learning due to its drawbacks and potential pitfalls, as discussed in the previous section. However, *within a company setting*, which is the focus of this thesis, some of these drawbacks and potential pitfalls may be overcome with the integration of the CoP approach and an augmented Experience Factory framework.

For instance, it is known that CoPs are vastly different from typical business units in an organization and therefore cannot be managed as such. This is due to the informal nature of CoPs and the fact that the functioning of CoPs depends heavily on social factors. Yet, it is also acknowledged that the long term success of CoPs in a company setting requires intentional facilitation and support from the organization. With its mandate to facilitate knowledge sharing within an organization, the experience factory unit in the Experience Factory framework fits naturally into this *supporting* role needed by CoPs. However, for the experience factory unit to assume this *supporting* role, it requires adjustment to the traditional roles and responsibilities that the experience factory unit takes on.

First, the experience factory unit needs to take an active role in identifying CoPs within the organization. While CoPs can be difficult to identify due to their informal nature and unofficial status, it is common in an organization that a CoP forms around a particular job or functional role; the job being the domain and/or the practice [Orr, 1996; Haas et al., 2000; Kähkönen, 2004]. The fact that CoPs may form around similar roles or jobs in a company provides a starting point for identifying potential CoPs. Through identifying and formally recognizing a CoP, the skill area represented by the practice of a CoP is made visible. If this skill area overlaps with the business core competency, an organization can be sensitive to ensure it maintains and nurtures this critical skill base. Without this formal recognition, it is likely that both core and peripheral members of that

CoP, who may be widely scattered across business units, disappear quietly and not cause an alarm.

Second, it is critical for the experience factory unit to serve a *supporting* as opposed to a *controlling* role. This implies both authorizing and delegating to CoPs the tasks of experience elicitation, analysis, generalization, formalization, packaging, customization, and dissemination. This can be achieved by coaching CoP coordinators in performing these knowledge management tasks together with their community members. In contrast, in typical implementations of the Experience Factory framework, the work of these knowledge management tasks are all centralized in and controlled by the experience factory unit. By delegating these tasks to each individual CoPs, the experience factory unit is less likely to become the bottleneck in the free-flow of valuable experience within the organization. Furthermore and more importantly, the delegation of knowledge management tasks to CoPs and their members may help reduce the "ivory tower" obstacle in experience sharing across multiple teams in an organization. This is because the spread expertise are both originated from and disseminated by the CoP members, who practice the skill and put the knowledge to regular use. Their recommendations are less likely to be devalued or dismissed by fellow practitioners. In comparison, in typical Experience Factory implementations, the spread knowledge and experience are disseminated by, and sometimes originated from, the experience factory unit members. These experience factory unit members are usually not the practitioners who would put the knowledge to actual use, and thus may be perceived by the line employees as living in an ivory tower and have their recommendations—relevant or irrelevant; useful or useless—be devalued or dismissed [Houdek et al., 1998].

Third, as a result of distributing the work of the knowledge management tasks to CoPs, the experience factory unit needs to take an active role in initiating knowledge sharing occasions and activities both within and, more importantly, across CoPs. The initialization of knowledge sharing activities across CoPs is not only important for standardizing practice and potential reusable experience, it is also important for advancing the practice of the CoPs as well.

2.4.1 Viability of an Integrated Approach

As the problem of organizational learning is only starting to receive attention in the agile methods community recently [Henninger and Maurer, 2002; Doran, 2004; Kähkönen, 2004], the above idea of integrating the CoP and Experience Factory frameworks in an agile software organization may appear novel at first for agile methods researchers and practitioners. In actuality, the application of an integrated approach to inter-team learning is not new, especially in the organizational science and knowledge management communities.

In the organizational science field, Hansen and his partners have observed in their studies of knowledge-intensive industries such as consultancies, health care providers, and IT companies, that both the codification and the personalization approaches are often used in parallel [Hansen et al., 1999]. In other words, inter-team learning strategies in practice often lie somewhere on a continuum from the codification approach at one end to the personalization approach at the other end. Despite the contrasting nature between these two approaches, they are found to serve different purposes. In their observations, Hansen et al. have discovered the use of the codification approach to help increase knowledge re-use achieving a leverage effect and that this benefit is best realized in environments where the business requirements are relatively stable or when standardized products are developed. With respect to the personalization approach, Hansen et al. have found the use of it to help generate new knowledge achieving an innovation effect and that this benefit is best realized in environments where highly customized solutions are needed to fulfill unique business requirements.

Similar discoveries are also reported in Trittmann's study of inter-team learning practices at various software organizations [Trittmann, 2001]. Similar to Hansen's work, Trittmann also suggests that it is essential to combine both codification and personalization approaches to facilitate inter-team learning, especially in software organizations, for the following arguments.

First, Trittmann argues that there exist various types of expertise considered important in software development. Examples include domain expertise (knowledge about the business domain), process expertise (knowledge about the development methodologies, design principles, etc.), and technical expertise (knowledge about particular technologies required for particular tasks).

Second, the choice of using the codification or the personalization approach should be considered for each type of expertise. Consequently, it is possible and practical to use the codification approach for one type of expertise while using the personalization approach for another, depending on the nature of the knowledge domain. For instance, when a development team is to implement a new and ambiguous system feature, the development team may need to acquire new knowledge about the business domain and perhaps about new technologies needed for this particular system feature. At the same time, the development team may also leverage process expertise such as object-oriented analysis and design, software inspections, etc.

While the combined benefits of using both codification and personalization approaches for facilitating inter-team learning are promising, Trittmann and Hansen et al. also suggest that the two approaches need not be pursued to the same degree in an organization's inter-team learning strategy. In fact, the extent to which particular approach is to be pursued more should match the culture and the primary needs of an organization. In the case of those software organizations that practice agile methods, given the humanistic people-centric nature of the development process, their inter-team learning strategies should place a greater emphasis on the personalization approach and compliment that with the codification approach. Conversely, if the primary inter-team learning objective of a software organization is to leverage existing experience and expertise, its inter-team learning strategy should place a greater emphasis on the codification approach.

2.5 Summary

In this chapter, I have provided an overview on the different approaches the software engineering and organizational science communities take to address the challenge of inter-team learning in an organization. In software engineering, the predominant approach to address inter-team learning is the Experience Factory framework. This framework does not prescribe how the concept is to be implemented. The flexibility of the framework is illustrated in the various Experience Factory implementations as reported in NASA, Ericsson, and DaimlerChrysler. In the organization science discipline, these various ways of implementing the Experience Factory framework are generalized and described in terms of being the codification approach, the personalization approach with the concept of Communities of Practice (CoP) examined more in detail, or a combination of both.

With the Experience Factory framework, it has the benefits of facilitating inter-team learning without placing overhead on existing project teams; providing opportunities of process improvement at both the project and organizational level; and making the standardization of potentially reusable experience easier. Yet, it can clash with the organizational culture in an agile software development environment due to its centralized nature and primarily non-practitioner membership.

With the codification approach, it has the benefits of mitigating the risks of knowledge loss; enabling collaboration across time and space; and facilitating efficient knowledge search and retrieval. Yet, there are drawbacks associated with the approach and they stem from the fact that not all types of knowledge can be explicated efficiently and that explicated knowledge is often incomplete and easily becomes outdated. By itself, the approach's emphasis on explicit knowledge makes it incompatible with the agile organizational culture.

With the personalization approach or the CoP framework, it has the benefits of being a more effective organizational structure than other organizational forms such as functional departments or project teams for managing knowledge; promoting synergies

across official business units; eliciting and sharing tacit knowledge better; providing implicit means for reinforcing relationship networks among members; increasing members' intrinsic motivation for knowledge sharing; and increasing the likelihood of retaining talent. Yet, a CoP also has the drawbacks of being difficult to support due to its often informal nature. It can become insular from other CoPs (similar or dissimilar in nature) or from new or peripheral CoP members. Overemphasis on person-to-person interaction may also overload experts in a CoP with recurring information needs. Finally, the goals of CoP may be incompatible with organization knowledge needs. These drawbacks make the sole use of the CoP approach inadequate for inter-team learning purposes in an agile software organization, despite the many commonalities it shares with the agile values and principles.

Based on case studies reported in both software engineering and organizational science literature, it is found that an integrated approach to inter-team learning is not only feasible but also beneficial and necessary. In light of this, it is suggested that the integration of the CoP approach and an augmented Experience Factory framework be used for facilitating inter-team learning in an agile software organization. Having examined the different approaches to inter-team learning, particularly the organizational and cultural factors in play, I will investigate in the next chapter existing tool support for knowledge sharing in terms of their capability in supporting the aforementioned approach.

Chapter 3. Tool Support Criteria & Assessment

To help support and facilitate knowledge sharing and the inter-team learning approaches mentioned in the previous chapter, various technologies and tools have been developed. The potential benefits of technology in knowledge management are manifold. First, it helps extend the reach and enhance the speed of knowledge transfer. Second, it enables the knowledge of an individual or a group to be extracted and structured, and then used by other members of the organization without the need for synchronous communication. Third, it also helps in the codification of knowledge and reduces the workload of experts. It even helps in its generation occasionally.

Having said that, it is also important to note that support and facilitation for knowledge sharing does not always have to be technological. In fact, even the most advanced technological applications fail unless the tools can be easily blend in the practitioners' job and social practices [Huysman and de Wit, 2003]. This phenomenon is also termed the "*IT productivity paradox*" where "*the increased use of IT does not result in expected increases in productivity*". While there are many factors contributing to the "*IT productivity paradox*", one of the most cited factors is "*mismatches between information technologies and their organizational settings…Without a match between the culture of an organization and the cultural assumption embedded in an IT innovation, a costly implementation failure is likely to occur*" [Ruppel and Harrington, 2001].

Given the agile software development culture, it is suggested at the end of the previous chapter (§2.4) that an integration of the Communities of Practice concept and an augmented Experience Factory framework would be an appropriate approach for facilitating inter-team learning. In this chapter, I first outline the specific requirements for tool support that aims to meet this goal. I then present a critical evaluation on existing knowledge sharing tools in terms of their abilities to support the proposed integrated

approach. The resulting critical evaluation will provide the motivation for the proposed proof-of-concept tool, MA6SE, details of which is elaborated in Chapter 4.

3.1 Tool Support Criteria

To help facilitate inter-team learning in a software organization, and to support the Experience Factory framework in particular, prior works by [Birk and Kröchel, 1999; Basili et al, 2001; Schneider and von Hunnius, 2003] found it essential for the ideal knowledge sharing tool to provide the following capabilities (in no specific order of importance):

1. **Support the entire lifecycle of reusable artefacts**. This entails the facilities for the *creation* and *retrieval* of artefacts. In addition, the tool should facilitate easy *reuse* of the artefacts and allow users to provide *feedback* on the artefacts.
2. **Precise representation of reusable artefacts**. To facilitate better reuse of artefacts, the tool should allow but not force users to precisely characterize artefacts in a formal representation. The tool should also allow users to adjust the granularity (to generalize or specialize) of reusable artefacts.
3. **Analysis of the experience base**. This entails the capability to analyze the *structure*, the *content*, and the *usage* of the experience base. Understanding the underlying structure of the content in the experience base and its usage can provide insights to questions such as which parts of the experience base is under-used or actively used. These insights can help experience base administrators to improve the tool so that artefacts in it can be browsed or retrieved more easily.
4. **Support different types of artefacts**. The tool should not restrict the users in explicating whatever they deem necessary to only a limited set of formats.
5. **Intuitive characterization of reusable artefacts**. For easier retrieval and more convenient maintenance of artefacts, users should be able to characterize the content and the application context of the artefacts in terms that they are familiar with.

6. **Links between reusable artefacts**. Different artefacts may contain information or experience that may be related. To enhance knowledge discovery, the tool should make it easy for users to establish explicit links between related artefacts.
7. **User guidance for artefacts retrieval**. Retrieving artefacts can be an expensive task in that users need to analyze the experience documented in the artefacts and assess its relevancy with their tasks at hand before they can reuse the artefact. The tool should provide guidance for users to retrieve artefacts that may be potentially relevant for their tasks at hand.

These suggested requirements are based on experiences in providing and using knowledge sharing tools to support the Experience Factory framework in software organizations that do not practice agile methods. Yet, these requirements apply equally well for knowledge sharing tools that are to be used in an agile software development setting. Under such a setting, however, the aforementioned requirements are insufficient; an inter-team learning approach in an agile software development environment also demands support for the Communities of Practice concept but such demands are not addressed by the aforementioned requirements. To address such demands, the ideal knowledge sharing tool that is to be used in an agile software development setting should also provide the following additional capabilities (in no specific order of importance), based on previous work by [Wenger, 2001; Wenger et al., 2002, pp. 197]:

8. **Presence awareness**. While a CoP will benefit from interaction and collaboration in a co-located setting, it is also desirable to extend these interactions and collaboration to a distributed setting. In such case, to reduce the difference between distributed and co-located interactions, the tool should provide rich awareness information among distributed CoP members. This entails reminding the presence of a community to its members and enhancing the ability of community members to be aware of each other's interest or activities.
9. **CoP rhythm generation and detection**. A rhythm of events is vital to the longevity of a CoP (physical or virtual alike). On one hand, the tool should allow users to specify both routine and ad-hoc events so members can participate in them synchronously or asynchronously. On the other hand, the tool should

provide support for promoting awareness of community activities to community members.

10. **Variety of interaction**. Interactions among CoP members are critical for the development of the CoP's shared practice. The fact that humans naturally, even unconsciously, collaborate in various styles and that knowledge creation and sharing is inherent in daily collaboration demand the tool to provide equivalent support for the various collaborative work styles [Greenburg and Roseman, 2003].
11. **Ease of Participation**. This concerns the efficiency in getting involved in a CoP's activities. If it is difficult to get involved in a CoP's activities, one is less likely to participate in that CoP. To encourage participation in a CoP, the tool needs to make it easy for potential and new CoP members to get involved.
12. **Allow for short-term return**. The attraction of potential CoP members depends partly on a CoP's ability to provide immediate value to them. The tool can facilitate this by providing quick answers to questions as well as access to artefacts and experts.
13. **Allow for long-term return**. It is insufficient for a CoP to provide only short-term value. This is because members of a CoP, especially the core members, often have a long-term interest in the CoP's domain and practice. This long term interest is often manifested in the development of a shared repertoire of resources. As such, the tool needs to provide mechanisms for CoP members to maintain and refine their shared repertoire of resources.
14. **Integration with external information source**. Knowledge creation in a CoP often takes place when members belonging to different CoP interact with one another. This type of interactions can help generate rhythm for a CoP. Therefore, the tool should allow members of a CoP to be kept abreast of activities in external CoPs, or even interact with members of other CoPs.
15. **Foster belonging and relationship**. Success of a CoP hinges on close interaction among its members and close interaction is influenced by the level of knowledge and trust CoP members have in each other. To help CoP members acquire

information about and develop trust with each other, the tool should provide mechanisms for users to develop their personal profiles that reflect their identities.

16. **Managing CoP boundaries**. Knowing the boundary of a CoP involves knowing whether one is a member as well as how often and how intense one participates in the CoP's activities. This knowledge is useful for effective communication among CoP members. As such, the tool needs to be able to detect one's membership in a CoP as well as how often and how intense one participates in a CoP. As a CoP grows, it is also common for it to spawn sub-communities or special interest groups. Therefore, the tool should also provide this capability as well.
17. **Evolve with community's needs**. As a human organization, a CoP evolves over time; going through various stages [Wenger et al., 2002]. As a CoP evolves, members should not need to use a different tool or learn a new one. The technical implication of this is that the tool needs to be extensible and be flexible in its configuration.
18. **Support active community building**. Anecdotal evidences indicate that there often exists a core group of members within a CoP that assumes an active role in cultivating the CoP. As such, the tool should offer various administrative facilities for monitoring and configuring the use and effectiveness of the community space.
19. **Identification of potential CoPs**. In an organization, it is argued that it is common that there already exist informal networks consisting of people who have the ability and passion to develop their competencies [Wenger and Snyder, 2000]. Therefore, the challenge is to identify these networks and help them to develop into CoPs.

These 19 requirements mentioned above are drawn upon existing accepted criteria for knowledge sharing tools that support the Experience Factory framework as well as those that support the Communities of Practice concept. Together, these criteria present the essential functionalities that should be addressed by any tools that aim to facilitate inter-team learning in an agile software development environment.

3.2 Existing Tool Support for Knowledge Sharing

Understanding the specific criteria mentioned above provides a means to critically evaluate existing knowledge sharing tools for facilitating inter-team learning in an agile software development setting. From these assessments, knowledge sharing tool developers can understand and leverage the strengths of existing tools to devise more effective tool support for knowledge sharing that is fit for the occasion.

In general, existing tool support for knowledge sharing can be classified under the following categories [Wenger, 2001; Lindvall et al., 2002]:

- Document Management and Experience Base;
- Collaboration Services;
- Knowledge Discovery; and
- Portals.

This section analyzes each of these four categories of tools in turn. For each of them, I will discuss the primary goal of the tools in the category, the knowledge sharing approaches they support, the common underlying technologies, a list of common features, and an overview of specific tools that represent the category.

Other types of tools that are important for knowledge management include those that are intended for customer relationship management, intellectual property management, and online training (e-Learning management). These three types of tools tackle interesting problems in knowledge management but the nature of these problems lies outside of the scope of this work. Hence they will not be addressed here.

3.2.1 Document Management and Experience Base

Document management systems represent the majority of existing tool support for knowledge management [Davenport and Prusak, 1998; Wenger, 2001]. These tools are based on the assumption that users regularly produce documents (in structured or unstructured forms) as part of their work practices; document being the work product and/or a communication medium. Based on this assumption, it is argued that documents

contain much valuable product and process knowledge; henceforth, management of documents being the primary knowledge management concern.

Common features provided by these tools such as Xerox DocuShare [DocuShare] and [LiveLink] include storage of documents (in any formats); classification of documents (by keywords or hierarchical structures, etc.); manual document search and retrieval (by text, or similarity measures, or ontology, etc.); subscription and notification of updates; version control; and access control mechanisms. In general, these types of tools lack facilities for providing feedback on documents and making explicit links among documents that may be related. These two facilities however are regarded as important, in software engineering literature that focuses on knowledge management, for the retrieval, maintenance, and continual refinement of the documented experiences [Birk and Kröchel, 1999].

Experience Bases are similar to document management system in terms of features and intent. Some experience base systems like BORE [Henninger, 2003] provide additional capabilities for providing feedback on and making explicit links among documented experiences. The primary goal of most experience bases is to serve as an implementation of the Experience Factory framework. They are the focus of most of the software engineering literature on knowledge management [Dingsøyr, 2000]. In the case of BORE, it provides a repository that contains experience collected from projects across the entire organization. These experiences are organized as cases, which are used to generate pre-defined tasks for similar projects in the future. A case is similar to a project task in nature. The generated set of tasks serves as a “best practice” guide. Project team members can diverge from the generated plan and not perform the suggested tasks if they deem the tasks to be inappropriate for the project situation at that time. In such cases, team members can submit their experiences and details of the tailored tasks in a structured format to the repository. A dedicated team of people is recommended to maintain the integrity of the cases stored in the repository. BORE’s explicit support for the reuse and sharing of process knowledge facilitates learning based on prescribed process.

Wiki [Leuf and Cunningham, 2001] is another type of experience base. With Wiki, users can share their experiences and knowledge on web pages (wiki pages). Unlike typical web sites though, any users can create, update, and organize the wiki pages in real-time using only their web browser. Many Wiki-based experience bases also provide features such as classification of pages; subscription and notification of updates; and version control. In comparison with the tools mentioned above, information in Wiki can only be annotated in an unstructured format whereas information in other tools such as BORE is annotated in a mostly structured fashion.

Both document management systems and experience bases share in common in their lack of support for person-to-person communication and collaboration; therefore they are not sufficient to support Communities of Practice. Most of them use the document metaphor in their tool design. They rely on the use of and support only the codification approach. Consequently, the advantages and disadvantages of the codification approach apply to this category of tools as well. With its emphasis on managing the contents embedded in documents, this category of tools may still be useful. However, it is clearly not sufficient in itself as an adequate knowledge sharing solution in an agile software development environment due to its document-centric metaphor and its lack of support for person-to-person interactions.

3.2.2 Collaboration Services

The primary goal of tools in this category is to enhance collaboration and communication, especially in a distributed environment. A majority of tools in this category supports only the personalization approach; hence the benefits and drawbacks said of the personalization approach apply equally well to this type of tools. Tools in this category range from *project management spaces* such as [VersionOne] to *community-oriented software* like [Communispace] to *asynchronous discussion software* such as e-mail and newsgroup to *real-time interactive applications* like instant messaging and audio/video conferencing solutions.

With *project management software*, typical features provided include definition of tasks and team membership, assignment of tasks to members, and monitoring of task progress. In the case of [VersionOne], it is a web-based project management tool. The web-based nature of the tool allows it to be used by team members working at the same place or at different locations. It provides each team member with a private web page which serves as his/her own information portal. However, VersionOne predefines all the content in one's personal information portal showing only tasks that are assigned to the team member. In fact, all information content in both tools can only be created and browsed in a structured way. Team members cannot control the formality of the content nor can they specify their own search query for retrieving information.

In the case of [Communispace], it is a web-based tool with explicit support for community-oriented activity such as brainstorming in addition to typical facilities such as asynchronous discussion, real-time chat and profiling of users. In terms of support for knowledge artefacts, it only allows users to categorize their contribution to a limited set of ten categories.

Some tool like Lotus QuickPlace [QuickPlace] also provides capabilities for threaded discussion, polling and voting, and document management. However, the document management facilities are usually not integrated with the project management facilities. For example, even when it is clear that a document is needed or is produced as part of performing a project task, there is no explicit way for users to associate the resulting documents with that particular task. This can hinder efficient knowledge retrieval when another user is performing a similar task in the future and he/she is trying to look up knowledge that was needed or generated for this task. Another common drawback of these project coordination tools is that they often impose a rigid and universal structure for content navigation and typical users are not allowed to modify the navigational structure. This lack of flexibility may hinder free form and spontaneous knowledge sharing which is critical for efficient communication in an agile software development environment.

With *asynchronous discussion software* like email and newsgroup, typical features provided include presence awareness, threaded discussions, feedback and rating on postings, rudimentary document management facilities, and statistical analysis of community activities. These features can also be found in typical *community-oriented applications*, which provide additional capabilities for instant messaging, polling and voting, defining event calendar, and more sophisticated user profiles.

With *real-time interactive applications* like instant messaging, typical features provided include presence awareness, audio/video streaming, co-authorship of artefacts via shared whiteboard and/or other applications, and recording of conversations. Some tools like Astound Conference Center [Astound] even allow for moderated conversations.

From an information sharing perspective, real-time collaboration tools like audio/video conferencing and application sharing facilitate the informal and spontaneous social interactions necessary for sharing tacit knowledge. However, one drawback of such synchronous collaboration tools is that their usage is limited only to team members working at the same time. Assuming normal work hours, this is difficult for teams with members working in different time zones. In contrast, tools like e-mail and newsgroups only support asynchronous communication and collaboration. While they are great for discussion purposes, it is difficult to retrieve and organize information embedded in them. This is because information on a particular knowledge topic is often scattered across multiple discussion threads. In addition, such tools do not allow users to evolve ("refactor") information content even if the information is outdated. This inhibits efficient knowledge discovery.

In terms of applicability in an agile software development environment, this set of tools is valuable due to its support for person-to-person communication and collaboration but it is inadequate for an agile organization in itself. This is because most of these tools, except for project spaces, lack project management facilities and such facilities are important given the project-centric nature of an agile software development environment. With project spaces, they are also inadequate by themselves despite their support for project management facilities. This is because the project management facilities are often

not integrated with document management facilities and are pre-structured without the ability to adapt to the needs of a specific CoP.

3.2.3 Knowledge Discovery

The primary goal of tools in this category is to provide insights and generate new knowledge based on existing data found in documents, conversation dialogues, or any types of items found in one or more experience bases. Archetypical tools in this category include data visualization tools, data mining applications, and expert systems. A majority of these tools use sophisticated statistical analysis, complex simulation models, and artificial intelligence technologies to identify concepts (typically for textual data), patterns (typically for numerical data), and relationships in the underlying data and generate insights on it.

In general, this type of tools is often used in conjunction with other tools such as experience bases and collaborative tools. As such, they are by themselves insufficient for knowledge sharing and inter-team learning purposes. In addition, the effectiveness of these tools hinges on the quality of the underlying data which in turn relies on large data size. Large data set, however, can usually only be collected over a long time. Therefore, this type of tools is particularly not suited for handling technical knowledge, which changes frequently in an agile software development environment. In contrast, this type of tools may be valuable in providing insights on domain and process experiences, such as decision support for conducting software inspections [Ruhe, 2002].

3.2.4 Portals

This category of tools retrieves for a given user information from various sources (responsible tasks, affiliated projects and CoPs, etc.) and displays them in a coherent manner. Its primary goal is to provide information that is relevant for a user based on his/her personal preferences and role in the organization or CoPs, such that a user can spend less time in retrieving needed knowledge. This type of tools is based on the assumption that users are members of multiple projects, or participate in multiple CoPs. Typical features provided by this type of tools include user-customizable information

sources and presentation; integration with document and project management facilities; subscription and notification of information items. While most of these tools like [LiveLink] provide communication capabilities, they are mostly in the form of asynchronous discussion mechanisms. This is useful for supporting communication but is insufficient due to the lack of support for real-time collaboration.

3.3 Tool Support Assessment

In this section, I evaluate the aforementioned tools against the 19 tool support criteria stated in section 3.1. A summary of this assessment is presented in Table 4.1. In this table, a (✓) mark indicates that a tool provides explicit and comprehensive support for a specific criterion. If a tool does not appear to provide explicit support for a specific criterion and offers no alternative means for supporting it, it is marked with a (✗). If a tool appears to provide elementary but not adequate support for a specific criterion, it is marked with a (✓/✗).

In terms of support for different types of artefacts, all tools but Wiki and VersionOne allow users to store artefacts of arbitrary types in the repositories. With Wiki, wiki page and image are the only types of artefact that users can store in the repository. With VersionOne, users can only store information in the form of an electronic story card, which is essentially a web page input form with pre-defined input fields.

As for support for the intuitive characterization of artefacts, all tools but Communispace and Astound Conference Centre allow users to define their own categories for classifying their artefacts.

In terms of support for the entire lifecycle of reusable artefacts, all tools let users create, update, and browse artefacts. However, only BORE and LiveLink provide the facilities for users to reuse artefacts and give feedbacks on them. With BORE, these features are discussed previously in section 3.2.1. With LiveLink, it achieves them by allowing users to export an artefact (or a collection of them) into a template (or a collection of templates). Subsequently, when other users create a new artefact, they have

the option to tailor the new artefact based on the template. For collecting feedbacks on the templates, LiveLink allows users to rate and comment on the templates.

As for support for both informal and formal representation of artefacts, all tools allow users characterize artefacts informally but only BORE and LiveLink also allow users to characterize artefacts in a more formal manner. BORE achieves this by organizing potentially reusable artefacts into a process model whereas LiveLink achieves this via a user-defined ontology.

With respect to support for cross-references among artefacts, only BORE, Wiki, VersionOne, and email/newsgroup provide this capability. With Wiki, it not only allows users to manually define links among artefacts, it can also automatically generate cross-references among potentially related artefacts.

In terms of facility for guiding users to retrieve potentially relevant artefacts, Wiki, VersionOne, and Communispace provide only limited support. In the case of Wiki and Communispace, they allow users to create a personal portal in the repository. Within these personal portals, users can include whatever information content they find relevant but this requires manual effort. With VersionOne, it automatically provides users with their own portals that show only all the tasks that the users are working on. This is inadequate as users may also find relevant information in tasks that they are not working on. In comparison, DocuShare, BORE, and LiveLink provide better mechanisms for guiding users to retrieve potentially relevant artefacts. In the case of BORE, for each task that the users are responsible for, it automatically provides users with information from similar tasks in the past. For DocuShare and LiveLink, they not only provide users with their own portals but they also provide a workflow engine that lets users define how information (document) is to be automatically delivered to the appropriate users.

With respect to support of various collaborative work styles, Communispace and LiveLink are the only tools that support both asynchronous and synchronous work as well as the use of both structured and unstructured information. In the case of Wiki, VersionOne, and email/newsgroup, they only support asynchronous work. With

DocuShare, BORE and QuickPlace, while they support the use of structured and unstructured information, they do not facilitate real-time collaboration. As for Astound Conference Centre, while it supports both asynchronous and real-time collaborative work, it has no provision for the use of structured information.

As for support for online presence awareness, none of the tools except Communispace provide this capability. In terms of providing quick answers to questions and access to artefacts and/or experts, nearly all tools achieve this goal by providing search capability or opportunities for real-time communication with other users; BORE being the only exception. With respect to the ability for members to maintain and refine their shared repertoire of resources, all tools except Astound Conference Center provide a centralized repository.

In terms of capabilities for generating routine and ad-hoc events and making users aware of these events, most of the tools either provide full or no support. In the case of Wiki, it provides limited support by allowing users to see a listing of all the wiki pages that have been updated lately.

As for integration with external information sources, DocuShare, Communispace, QuickPlace, and LiveLink provide only limited support as users can only insert hyperlink to external web sites. Email and newsgroup applications extend this capability by also allowing user to embed the content of external web pages.

In terms of support for fostering better knowledge and trust among community members, most tools maintain a public profile for each user but they differ in the amount of information that is published and the level of control the end-user has in customizing his/her personal profiles. In the case of DocuShare, a user profile shows only limited information about the user and the user cannot customize the content in the profile. With Wiki, users can create their own portals and have total control over the content in them. While it keeps track of additional user-related information, such as pages that the users contribute to and accessed the most, such information is hidden from the end-users.

With respect to supporting sub-communities, only DocuShare, newsgroup, and LiveLink provide such facilities. Other tools do not allow users to define more than one community within the tool. In terms of the potential of being an extendible and configurable tool, DocuShare, QuickPlace, email/newsgroup applications, and LiveLink all provide a set of application programming interfaces (APIs) that third-party software developers can use to extend the tools' existing functionalities.

With respect to facilities for analyzing tool usage and identifying potential CoPs, LiveLink is the only tool that provides both. With DocuShare, BORE, email/newsgroup applications, and Wiki, their supports for these two capabilities are limited; they capture tool usage information but such information is often concealed from the end-users which hinder self-organized community building. As for Communispace, while it does not support more than one community, it allows users to see for themselves how the tool is being used in general and by specific users.

In summary, among all the tools examined above, there does not exists one that provide full support for all of the 19 criteria that are essential for any knowledge sharing tools that are to be used in an agile software development setting.

Criterion	Tool								
	DocuShare	BORE	Wiki	VersionOne	Communispace	QuickPlace	e-mail / newsgroup	Astound Conference Center	LiveLink
Support different types of artefacts	✓	✓	✓/×	✓/×	✓	✓	✓	✓	✓
Intuitive characterization of reusable artefacts	✓	✓	✓	✓	✓/×	✓	✓	✓/×	✓
Support for creating, editing, retrieving, and reusing artefacts	✓/×	✓	✓/×	✓/×	✓/×	✓/×	✓/×	✓/×	✓
Support for providing feedback on reusable artefacts	×	✓	×	×	×	×	×	×	✓
Precise representation of reusable artefacts	×	✓	×	×	×	×	×	×	✓
Links between reusable artefacts	×	✓/×	✓	✓/×	×	×	✓/×	×	×
User guidance for artefacts retrieval	✓	✓	✓/×	✓/×	✓/×	×	×	×	✓
Variety of interactions	✓/×	✓/×	×	×	✓	✓/×	×	✓/×	✓
Presences awareness	×	×	×	×	✓	×	×	×	×
Ease of participation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Allow for short-term return by providing quick answers to questions as well as access to artefacts and/or experts	✓	×	✓	✓	✓	✓	✓	✓	✓
Allow for long-term return by providing mechanisms for CoP members to maintain and refine their shared repertoire of resources	✓	✓	✓	✓	✓	✓	✓	×	✓
Rhythm generation and detection	✓	×	✓/×	×	✓	✓	✓	×	✓
Integration with external information sources	✓/×	×	×	×	✓/×	✓/×	✓	×	✓/×
Foster belonging and relationship	✓/×	×	✓/×	×	✓	✓	✓/×	×	✓
Managing CoP boundaries	✓	×	×	×	×	×	✓	×	✓
Extendible and configurable	✓	×	×	×	×	✓	✓	×	✓
Identification of potential CoP	✓/×	×	✓/×	×	✓/×	×	✓/×	×	✓
Analysis of experience base / support active community building	✓/×	✓/×	✓/×	×	✓	×	✓/×	×	✓

Table 3.1: Assessment of existing tool support for knowledge sharing

3.4 Summary

In this chapter, I have elicited the requirements for tool support that aims to facilitate knowledge sharing, and particularly inter-team learning, in an agile software development environment. These requirements are drawn upon existing accepted criteria for tools that support the Experience Factory framework as well as those for tools that support the Communities of Practice. Based on these criteria, a tool that supports knowledge sharing and inter-team learning in an agile software development environment should provide the following capabilities:

1. Support different types of artefacts;
2. Intuitive characterization of reusable artefacts;
3. Support the entire lifecycle of reusable artefacts;
4. Precise representation of reusable artefacts;
5. Analysis of the experience base;
6. Links between reusable artefacts;
7. User guidance for artefacts retrieval;
8. Variety of interaction;
9. Presence awareness;
10. Ease of participation;
11. Allow for short-term return;
12. Allow for long-term return;
13. Rhythm generation and detection;
14. Integration with external information sources;
15. Foster belonging and relationship;
16. Managing CoP boundaries;
17. Extensible and configurable;
18. Support active community building; and,
19. Identification of potential CoPs.

Based on these criteria, I evaluate four categories of existing tool support for knowledge sharing can be classified. With document management systems and experience bases, a

majority of them support only the codification approach to knowledge sharing. While these tools are still useful, they are by themselves insufficient for knowledge sharing purposes in an agile software development environment due to their lack of support for person-to-person interaction. With collaborative tools, they support the personalization approach (not necessarily the CoP framework though). While some also support the codification approach, the support is incomplete. Tools in this category generally lack flexibility in terms of the structure for navigating to information or lack convenient mechanism for evolving ("refactoring") information content. These drawbacks make the sole use of this type of tool inadequate for knowledge sharing and inter-team purposes in an agile software development environment. With knowledge discovery tools, they are insufficient by themselves as they are often and need to be used in conjunction with other types of knowledge management tool support. With portals, they provide capabilities for seamlessly integrating various types of knowledge sharing tools. However, they fall short of being the complete tool support for knowledge sharing and inter-team learning due to their general lack of support for real-time collaboration.

From the previous chapter, it is indicated that inter-team learning in an agile software organizations demands the integration of both the CoP concept and an augmented Experience Factory framework. Yet, the analysis of existing tool support for knowledge sharing support presented in this chapter reveals a gap among existing tool support in meeting this need. These tools either provide support for only one approach or they provide inadequate support for an integrated approach. In the next chapter, I will present the design and implementation of a novel tool support that aims to (1) facilitate knowledge sharing and inter-team learning by (2) providing flexible support for an integrated approach, and that is (3) appropriate for use in an agile software development environment.

Chapter 4. MASE: Proof-of-Concept

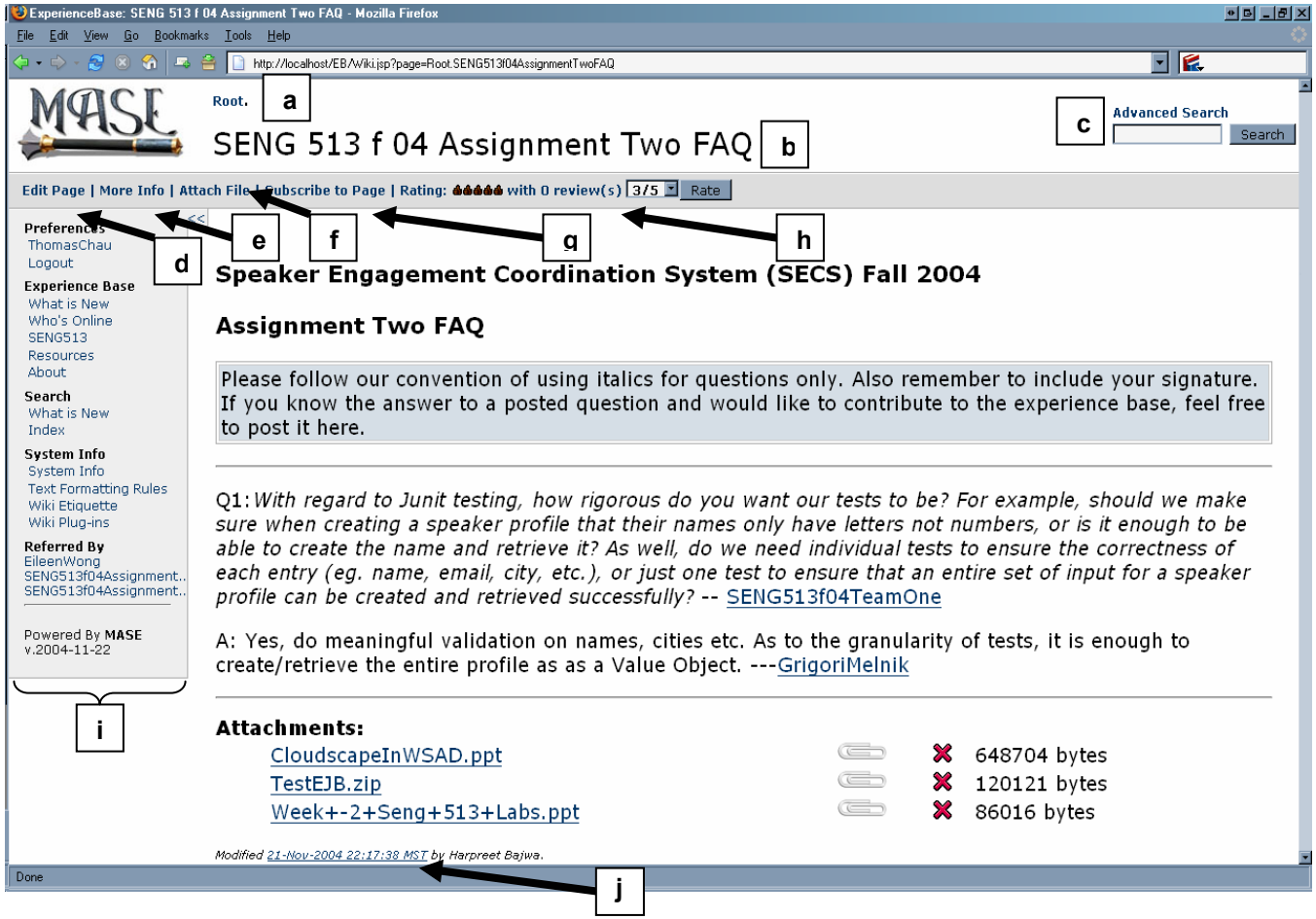
The previous chapter elicits several criteria that are deemed necessary for tools that aim to support both the Communities of Practice concept and the Experience Factory framework in an agile software development environment. Based on these criteria, I present MASE, a proof-of-concept tool that attempts to more fully support the integrated approach stated above. First, I describe the functionalities of MASE. The primary objective of the section is to elaborate on how MASE can satisfy each of the criteria stated in the previous chapter; it is not intended to be an exhaustive listing of all of MASE's features. I then examine the technical details underlying the functioning of MASE. I close by a walkthrough of how the tool is envisioned to be used in a typical agile software development environment.

4.1 Tool Functionality

MASE is a web-based collaborative environment where information content is embedded in a collection of wiki pages. Wiki pages are in essence web pages and their differences will be discussed later in §4.2. With MASE, users can *access*, *create*, *update* the page content, and even *organize* the structure of these pages in *real-time* by just using their existing web browsers. This is in contrast to typical web sites where the content or the structure is predefined, and contribution (hence maintenance) of content is only restricted to a few people (webmasters). Part of MASE's behaviour is analogous to a real world whiteboard where information on it can be *easily* and *quickly* edited, thus inviting people to collaborate on it by authoring information that they deem is important for others.

4.1.1 Artefacts Creation, Editing, Linkage, and Retrieval

In MASE, all information content is presented in wiki pages. A wiki page is in fact the primary form of artefact in MASE. To *access* or *view* any information content in MASE, users simply enter in their web browsers the URL of the wiki page that contains the information that they need. Figure 4.1 illustrates a wiki page in MASE.



(a) Page location in the hierarchy

(b) Page title

(c) Search facility

(d) Edit page control

(e) View a page's history and properties

(f) Attach files to a page

(g) Subscribe and receive notification of page update

(h) Peer review a page

(i) Navigation menu

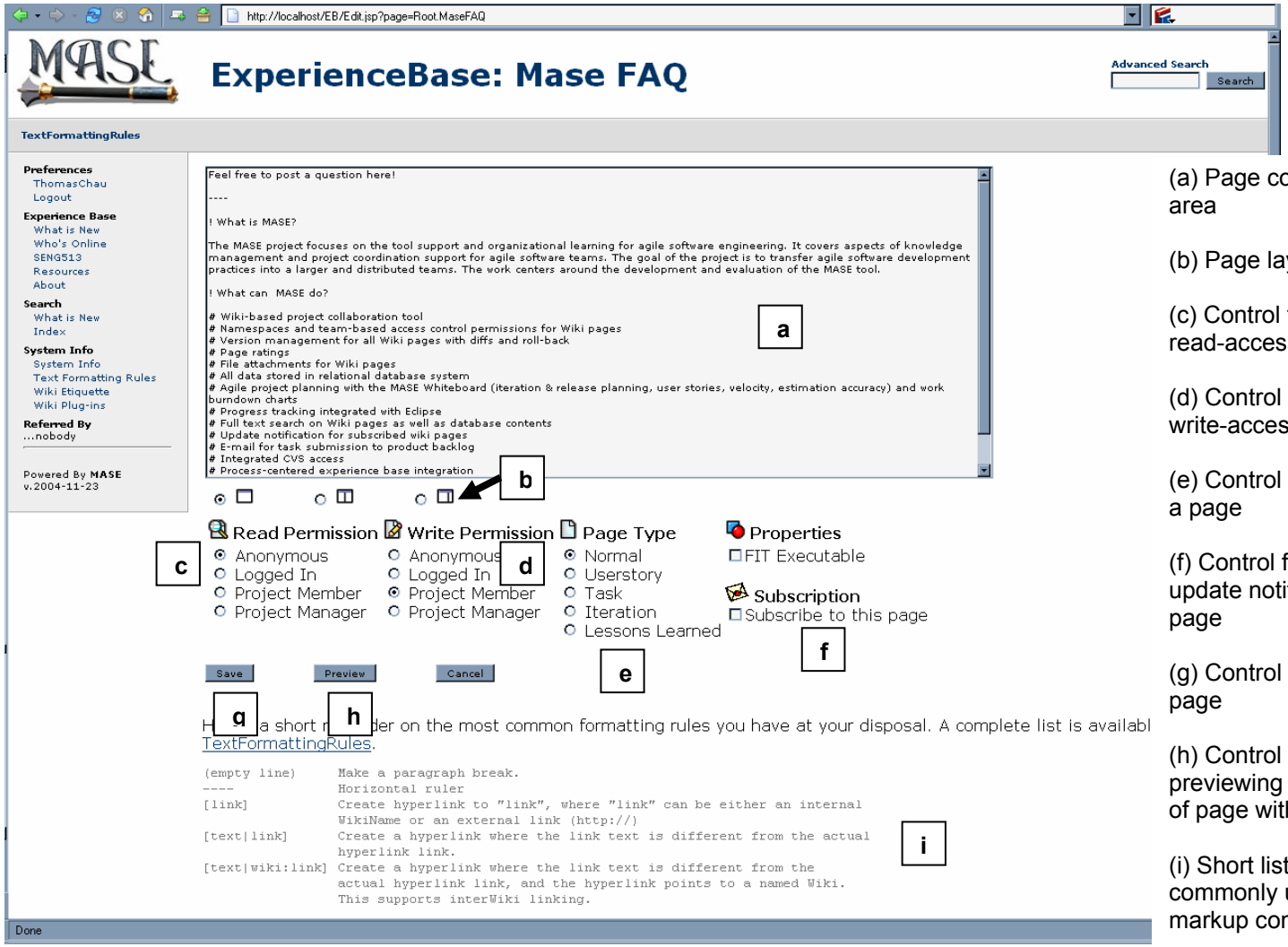
(j) Last modification information

Figure 4.1: A typical wiki page in MASE

In the above example, it can be seen that the URL of the page consists primarily of the name of the page. Creating a new wiki page in MASE follows the same fashion as browsing a page; users simply enter the name of the wiki page in the web browser. Upon receipt of the users' request, MASE immediately checks if there exists a wiki page with the specified name. If the specified page does not exist, MASE will prompt the users to add content to the new page (Figure 4.2).

Figure 4.2: MASE automatically prompts for users to contribute content when browsing a page yet to exist

Upon clicking on the "create it" hyperlink (Figure 4.2), MASE presents the users with the edit interface (Figure 4.3). To edit the content of a wiki page in MASE, users simply enter their contribution in the text area (Figure 4.3a) using the Wiki markup language. The Wiki markup language is much simpler than HTML, the language for authoring content in web pages. In fact, a list of all Wiki markup commands including examples fits easily onto a single page. When users are satisfied with their added content, they can click on the "Save" button (Figure 4.3g). The updated content is immediately visible to other users when they browse the page.



(a) Page content editing area

(b) Page layout control

(c) Control for adjusting read-access

(d) Control for adjusting write-access

(e) Control for classifying a page

(f) Control for receiving update notification of a page

(g) Control for saving a page

(h) Control for previewing the rendering of page without saving it

(i) Short list of most commonly used wiki markup commands

Figure 4.3: Interface for editing a wiki page in MASE

To define a link from a wiki page X to another wiki page Y, users can browse to wiki page X and click on the "Edit Page" hyperlink (e.g., Figure 4.1d). Then in the editing text area (Figure 4.4, left), users simply need to type in the name (not the URL although that is allowed) of wiki page Y by capitalizing each word in the page name and join them together (Figure 4.4, left). If the wiki page to link to does not exist yet, MASE automatically appends a "?" character behind the name of that page and shows it as a hyperlink; inviting other users to contribute content to that page (Figure 4.4, right).

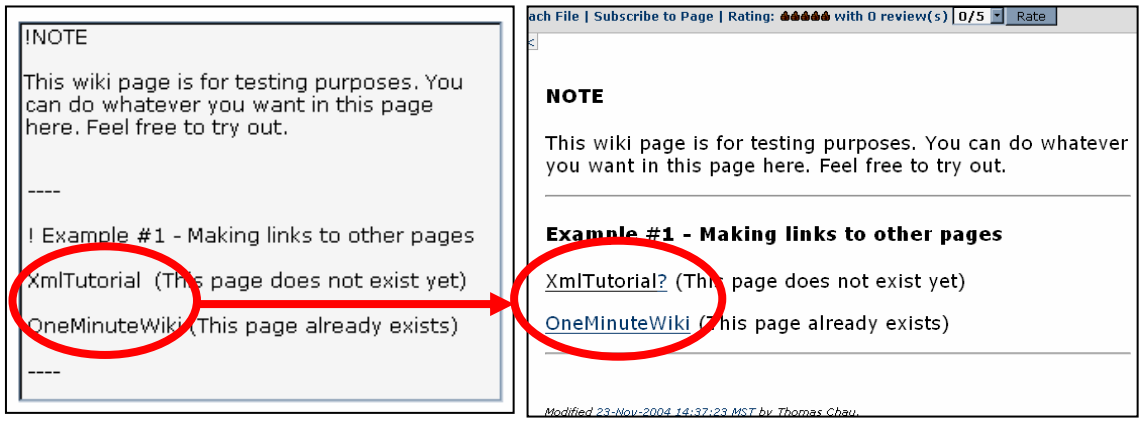


Figure 4.4: Linking wiki pages in MASE

The use of the Wiki markup language in MASE allows users to annotate information in an *unstructured* and informal manner. In cases where the use of the Wiki markup language for annotating information is insufficient, MASE allows users to attach files of any formats to a wiki page (Figure 4.1f).

MASE also supports the use of *structured* information via the concept of plug-ins. In general, a plug-in presents a form that either (1) allows users to submit data following a specific schema or (2) performs some logic and then displays the resulting information in a structured fashion. For instance, MASE contains a *ReferringPagesPlugin* that lists all wiki pages containing links to a given wiki page. This plug-in and, in fact, any other plug-ins can be placed anywhere in any wiki pages by any users. To include a plug-in in a wiki page, users simply need to type in the name of that plug-in at the desired location inside the content of that wiki page (e.g., Figure 4.5, bottom).

To enhance the retrieval of information (unstructured or structured alike), MASE adopts a two-prong approach. First, MASE attempts to make site navigation easier by allowing users to customize the navigational structure of their MASE web site according to their specific needs. This is achieved by making the site navigation menu as a typical wiki page (Figure 4.5). In addition, MASE also shows the currently viewed wiki page in context of other wiki pages by displaying a list of other wiki pages that refers to it (Figure 4.5, lower left). Second, MASE provides full-text indexing and searching capabilities on any unstructured and structured content (Figure 4.1a).

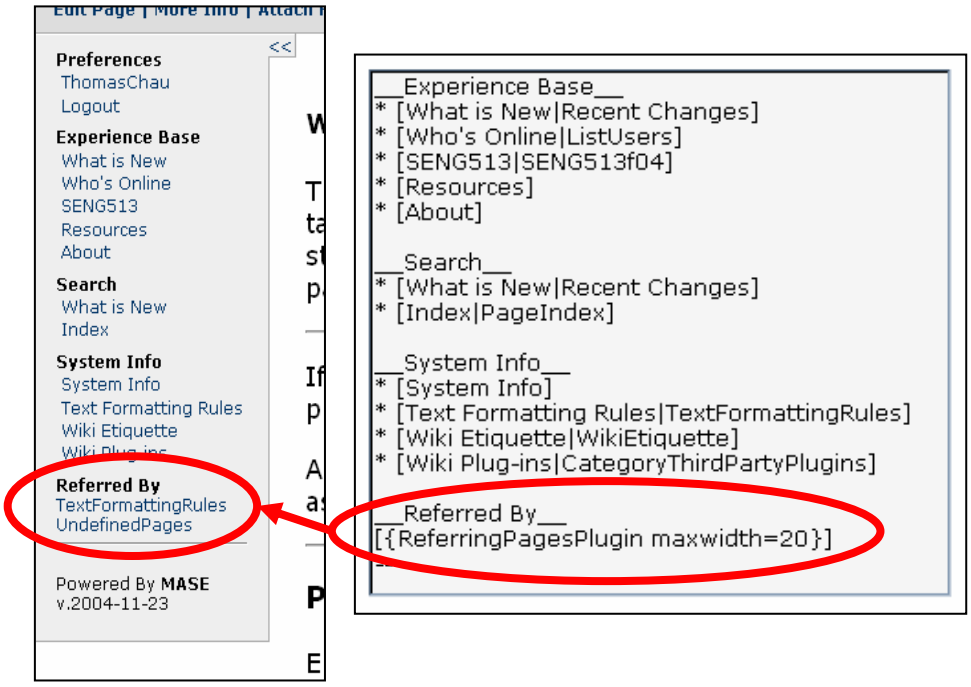


Figure 4.5: The navigation menu in MASE is just a typical wiki page, thus allowing users to customize the site navigational structure according to site content

4.1.2 Artefacts Characterization, Reuse, and User Guidance

For characterizing a wiki page, MASE by default allows users to classify a wiki page according to the following types: *Normal*, *Task*, *User Story*, *Iteration*, and *Lessons Learned* (see Figure 4.3e). The types of *Task*, *User Story*, and *Iteration* are particularly chosen because they are commonly used concepts in an agile software development environment. To classify a wiki page X into a category not mentioned above, users simply need to embed a link to a wiki page with the desired category's name (e.g.,

Category Z). When others browse to the page "Category Z" and if the page contains the *ReferringPagesPlugin*, they can see all wiki pages that are grouped under the category, "Category Z". This capability is elaborated further in the following figure.

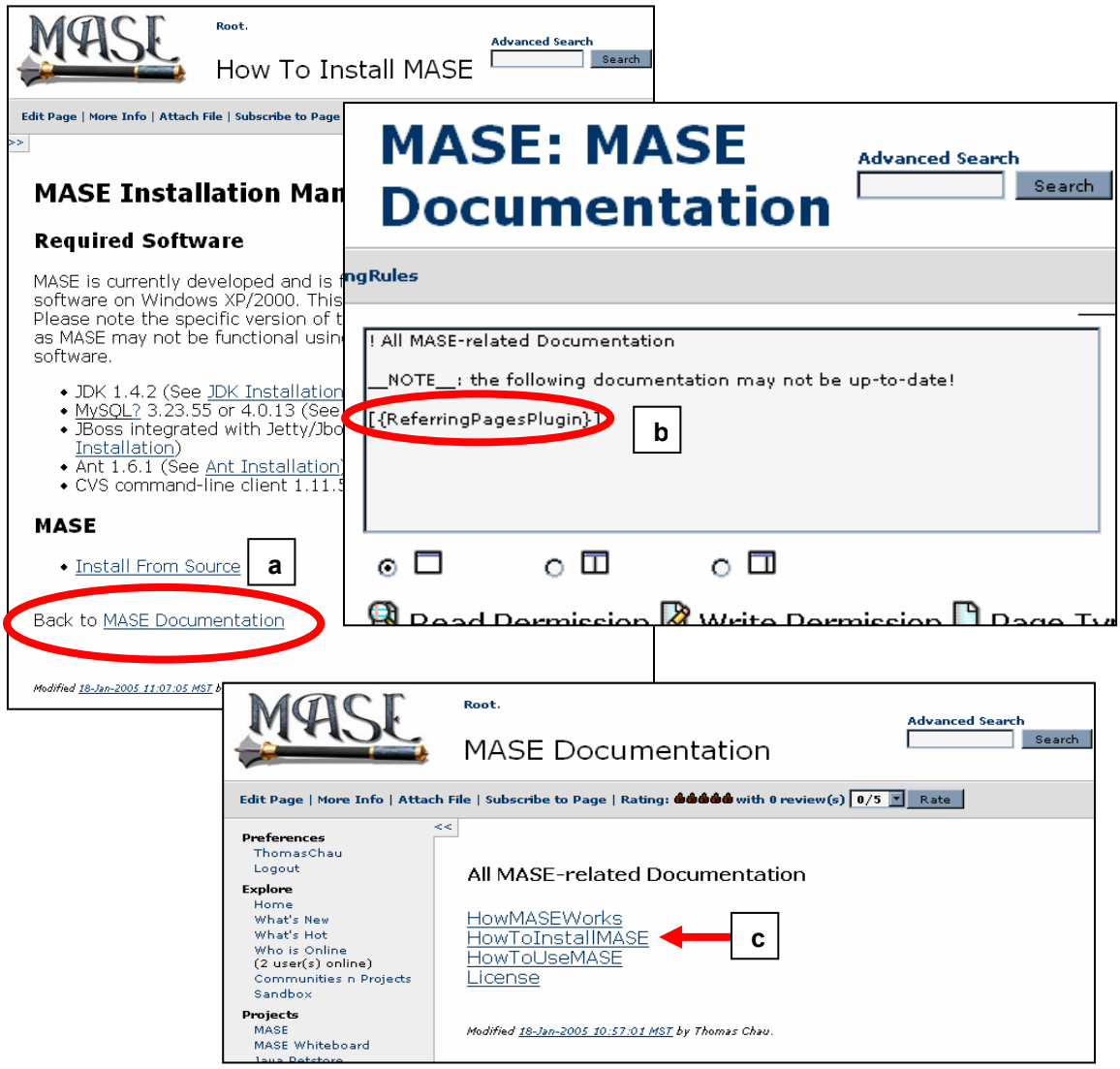


Figure 4.6: To classify the page "How To Install MASE" into the category "MASE Documentation", users simply need to: (a) embed in the page "How To Install MASE" a back reference that links to "MASE Documentation", and (b) include the "*ReferringPagesPlugin*" in the page "MASE Documentation". The latter step allows users who browse to the page "MASE Documentation" to see (c) a list of pages that are under the category, "MASE Documentation".

To better organize potentially reusable artefacts, especially those that contain task-related knowledge, MASE allows users to organize wiki pages into a *process model*. A *process model* is a hierarchical structure consisting of commonly used tasks (*process*

types). Users can decompose a *process type* into finer details in terms of the various possible ways to accomplish the commonly used task (*process type*). Each of these possible variations is referred to as *decompositions* and each of them can be further broken down into one or many *process types*. These *process types* and *decompositions* can be defined and updated using plug-ins provided by MASE. For each *process type* or *decomposition*, there is also a wiki page associated with it so users can annotate unstructured information about them as they see fit. In §4.3, I will describe a common MASE usage scenario that illustrates the potential benefits offered by having a *process model*.

To retrieve artefacts that they are only interested in, users can subscribe to any wiki pages that they are interested in (e.g., Figure 4.1g). When these pages are modified, MASE automatically notifies the users of the changes via e-mail. Furthermore, MASE by default provides each user with a wiki page that can serve as a personal portal for that user. Like any other wiki pages, any content can be stored in these personal wiki pages. This allows users to have complete control over the structure and the type of information that they deem relevant to themselves only. MASE also provides other mechanisms for guiding users to artefacts that may be potentially relevant to them and they are described later in §4.3.

4.1.3 Variety of Interaction

As a web-based collaborative environment, MASE by default provides support for co-located and distributed work. To support asynchronous collaborative work, MASE allows users to store the state of any wiki pages at any time (Figure 4.3g). Furthermore, MASE provides a version control facility by tracking all changes made to all wiki pages and any files attached to them. As illustrated in Figure 4.1, MASE users can use a wiki page as a discussion forum to interact in an asynchronous fashion. Users can also rate their perceived usefulness of any wiki pages (Figure 4.1h).

To support real-time communication and collaboration, MASE offers a plug-in that integrates with Microsoft NetMeeting and another plug-in (the *ListUserPlugin*) that

provides awareness information by displaying all users who are currently using MASE (Figure 4.7a). Whenever a user connects to MASE, the IP address of that user's computer is recorded. By viewing that online user's personal profile (Figure 4.7b), one can start a NetMeeting session with that user. These two plug-ins make it possible for users to be aware of each other's presence and facilitate informal and spontaneous interaction, which is critical for sharing tacit knowledge.

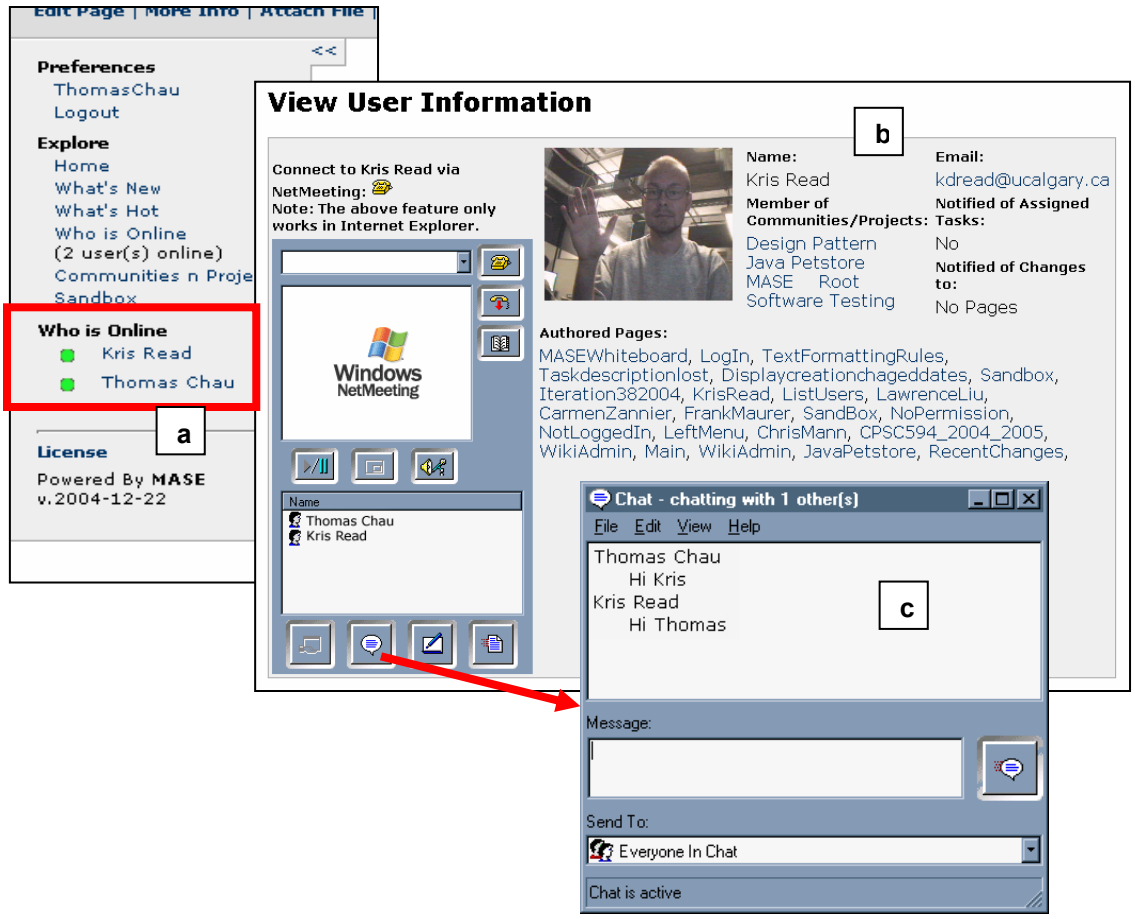


Figure 4.7: (a) The *ListUsersPlugin* (embedded in the navigation menu) allows users to be aware of each other's presence. By viewing an online user's personal profile (b), users can start a NetMeeting session with that user (c).

While the aforementioned facilities allow users to interact with each other (asynchronously or in real-time), it must be emphasized that these interactions often only occur as users have an increased level of knowledge and trust with each other. To help users acquire information about each other and hence develop trust, the aforementioned *ListUserPlugin* provides a listing of all MASE users which can serve as a member

directory. In addition, users can take advantage of their personal wiki pages to describe their areas of expertise and interest. Alternatively, users can retrieve a list of all wiki pages and attachments a particular MASE user has contributed in order to infer the areas of interest of that particular user (Figure 4.7b).

4.1.4 Ease of Participation

Aside from motivational factors, the intense use of the aforementioned interaction facilities in MASE also hinges on how easy it is for users to access those facilities and participate in the ensuing interactions. With respect to access, MASE by default grants all users the same level of access; any users can browse, create, update the content of, and organize the structure of wiki pages. As for participation, MASE attempts to make writing (providing information) as easy as reading (accessing information) via the use of the simple Wiki markup language. As illustrated in Figures 4.3 and 4.5, information content on any wiki page are nearly free-formatted text; what users enter when they edit the page closely resembles with what others see when the page is browsed.

However, these measures alone are insufficient for facilitating efficient participation. This is because one's participation in a CoP's activities or collaborative work may interfere with one's daily work. To mitigate this risk, MASE makes participation and collaborative work on a CoP's artefacts as seamless as possible by integrating CoP-related facilities together with facilities that support an agile project member's common project tasks. This integration is described later in §4.3.

4.1.5 Managing CoP Boundaries

One central element that bonds members of a CoP together is their shared interest in a common knowledge domain. Having said that, some members of a CoP may also be interested in sub-areas within the larger common knowledge domain, thus forming sub-communities. Consequently, there can be multiple levels and types of participation within a CoP. To facilitate this, MASE allows users to define groups of users (Figure 4.8). It also allows users to organize wiki pages into a hierarchy structure. Pages pertaining to a particular CoP or a subgroup of a CoP can be grouped together (Figure 4.9a). If

necessary, users can also grant different levels of read- and write-access to a particular collection of pages (Figure 4.9a). However as suggested in the previous section, MASE by default grant read- and write-access of all pages to all users in order to facilitate free, spontaneous and informal collaboration among different users, even if they belong to different CoPs.

For connection to external CoPs, MASE also offers a plug-in that users can use to embed inside a wiki page the content of another web page on the Internet, thus allowing them to keep abreast of news from these external information sources (Figure 4.9b).

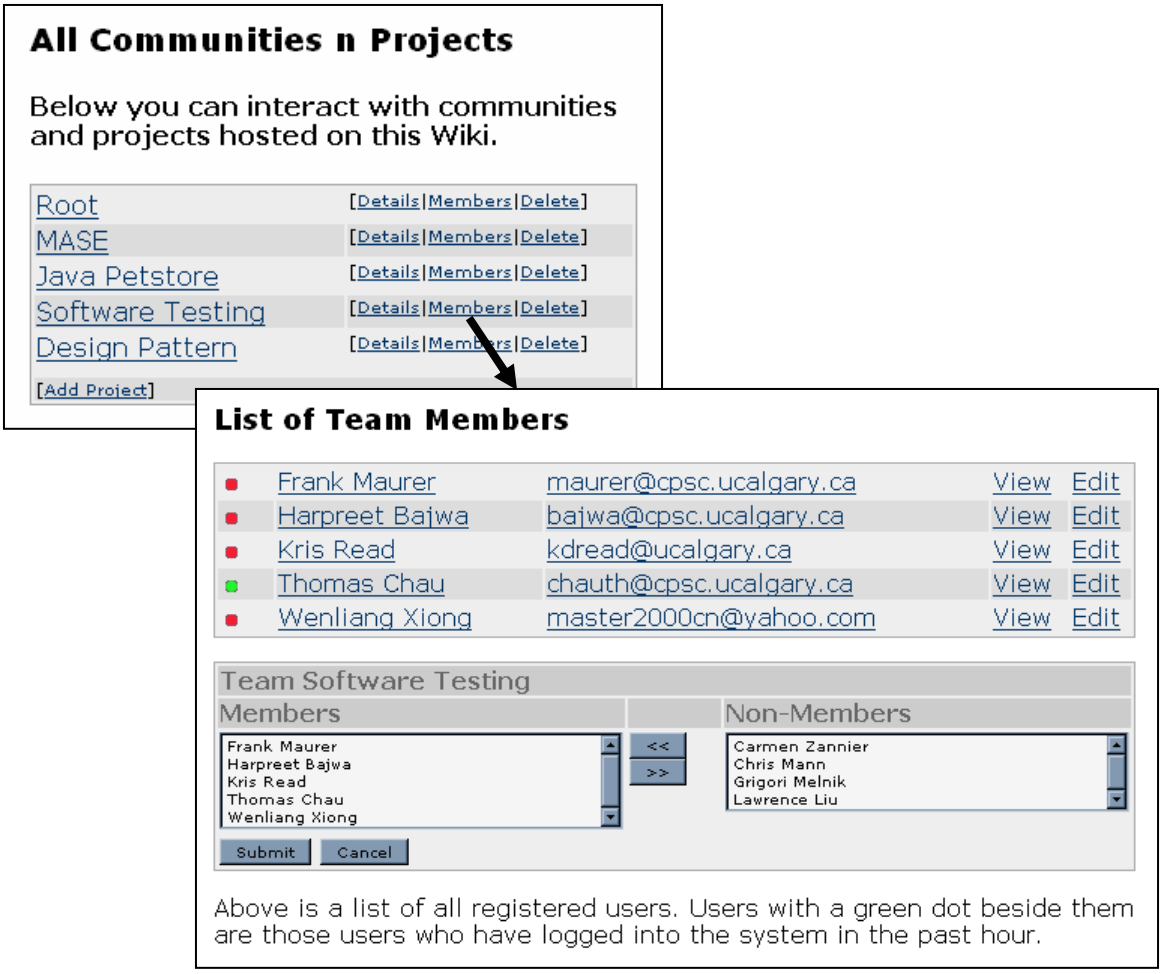


Figure 4.8: Creating and defining membership of sub-communities in MASE

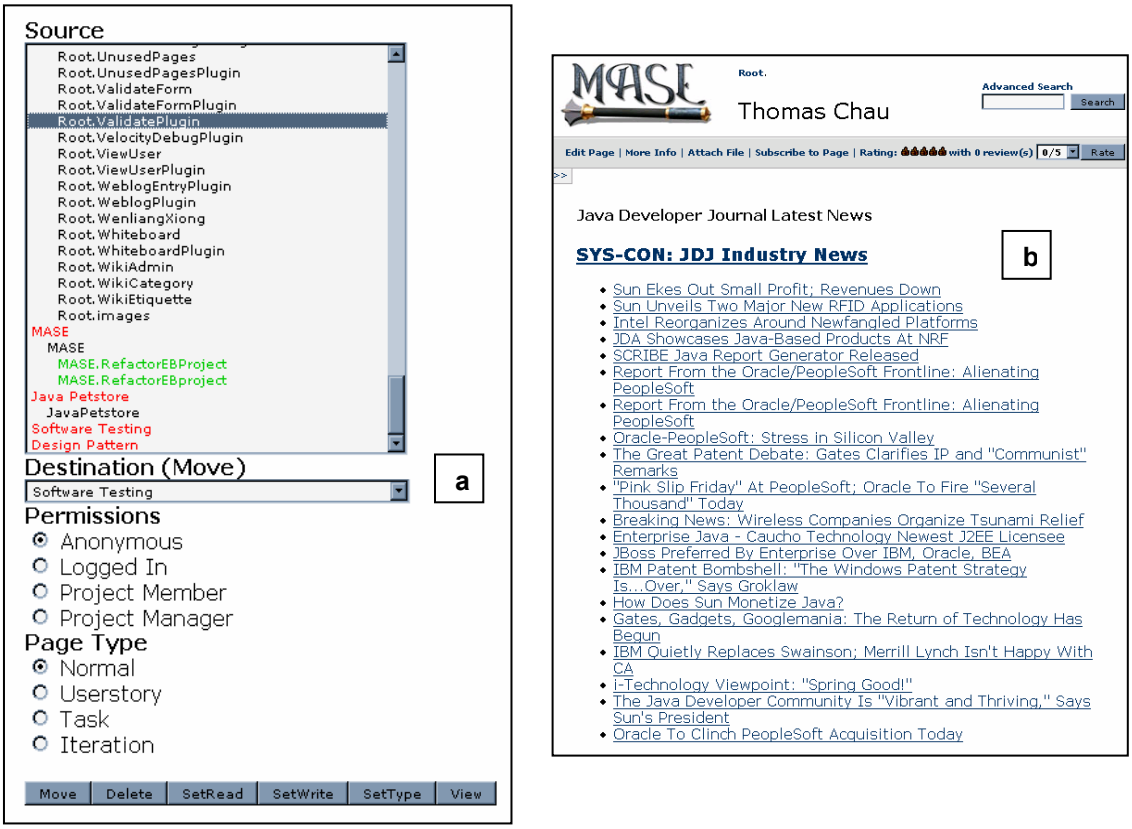


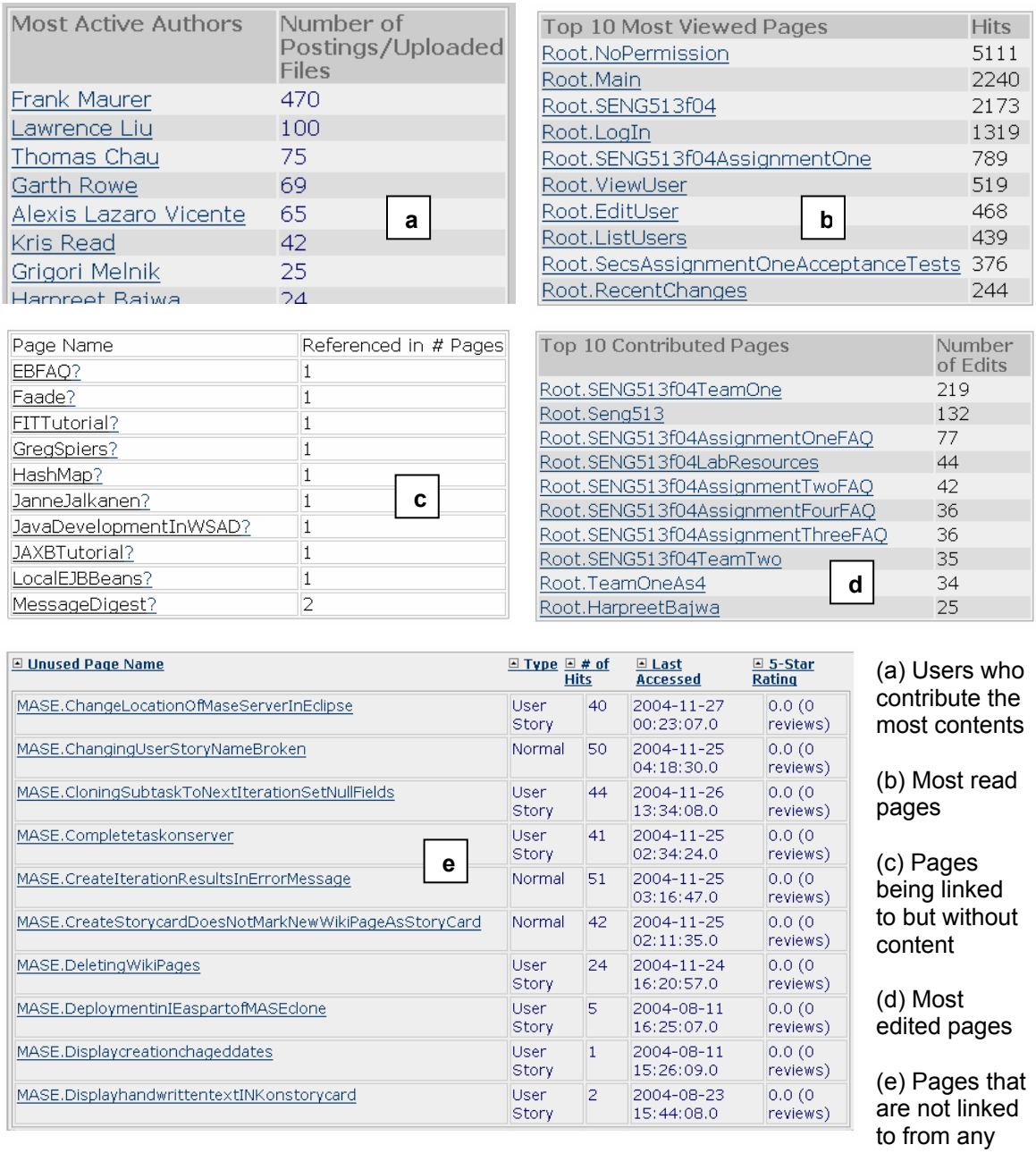
Figure 4.9: (a) A MASE plug-in that allows users to organize wiki pages into hierarchy; (b) Another MASE plug-in that lets users to embed external information sources (e.g., articles from an online magazine like the Java Developer Journal) into any wiki pages

4.1.6 Monitor Usage

To better understand how the tool is used, MASE automatically collects the following data: (1) the wiki page or attachment being accessed; (2) the type of access, if it is a browse, or an edit, or a search, or a download; (3) the time of access; and, (4) the user accessing the resource. In cases where users interact with each other using the NetMeeting plug-in, MASE tracks the time and participants of the corresponding NetMeeting interaction.

These statistics allows users to be aware of ongoing collaborative work on artefacts and activities in MASE (Figure 4.10). Using these statistics, those responsible for maintaining the repository can concentrate their effort on those parts of the repository that are used the most. Furthermore, these statistics provide insights into potential CoPs. Potential CoPs may be identified as: (1) those users who *frequently* browse and/or

contribute content to a given wiki page, or (2) those users who *frequently* engage in NetMeeting interaction with each other. To be flexible, the tool purposely leaves the definition of "frequent" up to the discretion of those members (mostly likely those in the experience factory unit) who are responsible for supporting CoPs in the organization.



Most Active Authors	Number of Postings/Uploaded Files
Frank Maurer	470
Lawrence Liu	100
Thomas Chau	75
Garth Rowe	69
Alexis Lazaro Vicente	65
Kris Read	42
Grigori Melnik	25
Harpreet Bajwa	24

a

Top 10 Most Viewed Pages	Hits
Root.NoPermission	5111
Root.Main	2240
Root.SENG513f04	2173
Root.LogIn	1319
Root.SENG513f04AssignmentOne	789
Root.ViewUser	519
Root.EditUser	468
Root.ListUsers	439
Root.SecsAssignmentOneAcceptanceTests	376
Root.RecentChanges	244

b

Page Name	Referenced in # Pages
EBFAQ?	1
Faade?	1
FITTutorial?	1
GregSpiers?	1
HashMap?	1
JanneJalkanen?	1
JavaDevelopmentInWSAD?	1
JAXBTutorial?	1
LocalEJBBeans?	1
MessageDigest?	2

c

Top 10 Contributed Pages	Number of Edits
Root.SENG513f04TeamOne	219
Root.Seng513	132
Root.SENG513f04AssignmentOneFAQ	77
Root.SENG513f04LabResources	44
Root.SENG513f04AssignmentTwoFAQ	42
Root.SENG513f04AssignmentFourFAQ	36
Root.SENG513f04AssignmentThreeFAQ	36
Root.SENG513f04TeamTwo	35
Root.TeamOneAs4	34
Root.HarpreetBajwa	25

d

Unused Page Name	Type	# of Hits	Last Accessed	5-Star Rating
MASE.ChangeLocationOfMaseServerInEclipse	User Story	40	2004-11-27 00:23:07.0	0.0 (0 reviews)
MASE.ChangingUserStoryNameBroken	Normal	50	2004-11-25 04:18:30.0	0.0 (0 reviews)
MASE.CloningSubtaskToNextIterationSetNullFields	User Story	44	2004-11-26 13:34:08.0	0.0 (0 reviews)
MASE.Completetaskonserver	User Story	41	2004-11-25 02:34:24.0	0.0 (0 reviews)
MASE.CreateIterationResultsInErrorMessage	Normal	51	2004-11-25 03:16:47.0	0.0 (0 reviews)
MASE.CreateStorycardDoesNotMarkNewWikiPageAsStoryCard	Normal	42	2004-11-25 02:11:35.0	0.0 (0 reviews)
MASE.DeletingWikiPages	User Story	24	2004-11-24 16:20:57.0	0.0 (0 reviews)
MASE.DeploymentinIEaspartofMASEclone	User Story	5	2004-08-11 16:25:07.0	0.0 (0 reviews)
MASE.Displaycreationchageddates	User Story	1	2004-08-11 15:26:09.0	0.0 (0 reviews)
MASE.Displayhandwrittentext INKonstorycard	User Story	2	2004-08-23 15:44:08.0	0.0 (0 reviews)

e

(a) Users who contribute the most contents

(b) Most read pages

(c) Pages being linked to but without content

(d) Most edited pages

(e) Pages that are not linked to from any where

Figure 4.10: These plug-ins compute tool usage statistics to allow users and administrators to gauge the rhythm of ongoing activities and collaborative work happening online

4.1.7 Extendible and Configurable

From the users' perspective, it is more convenient for them if the tools that they are accustomed to can evolve with their needs. This demands the underlying tool to be extendible and configurable in order to address requirements that are unforeseen when the tool is first developed. To facilitate this, MASE allows tool developers to extend its existing functionalities by programming additional functionalities, packaging them as a self-contained component (plug-in), and incorporate them into MASE without the need to modify the underlying MASE source code (e.g., Figures 4.7-4.10). MASE also allows tool developers to customize its look-and-feel according to their communities' needs. This customization can also be done without modifying the MASE source code.

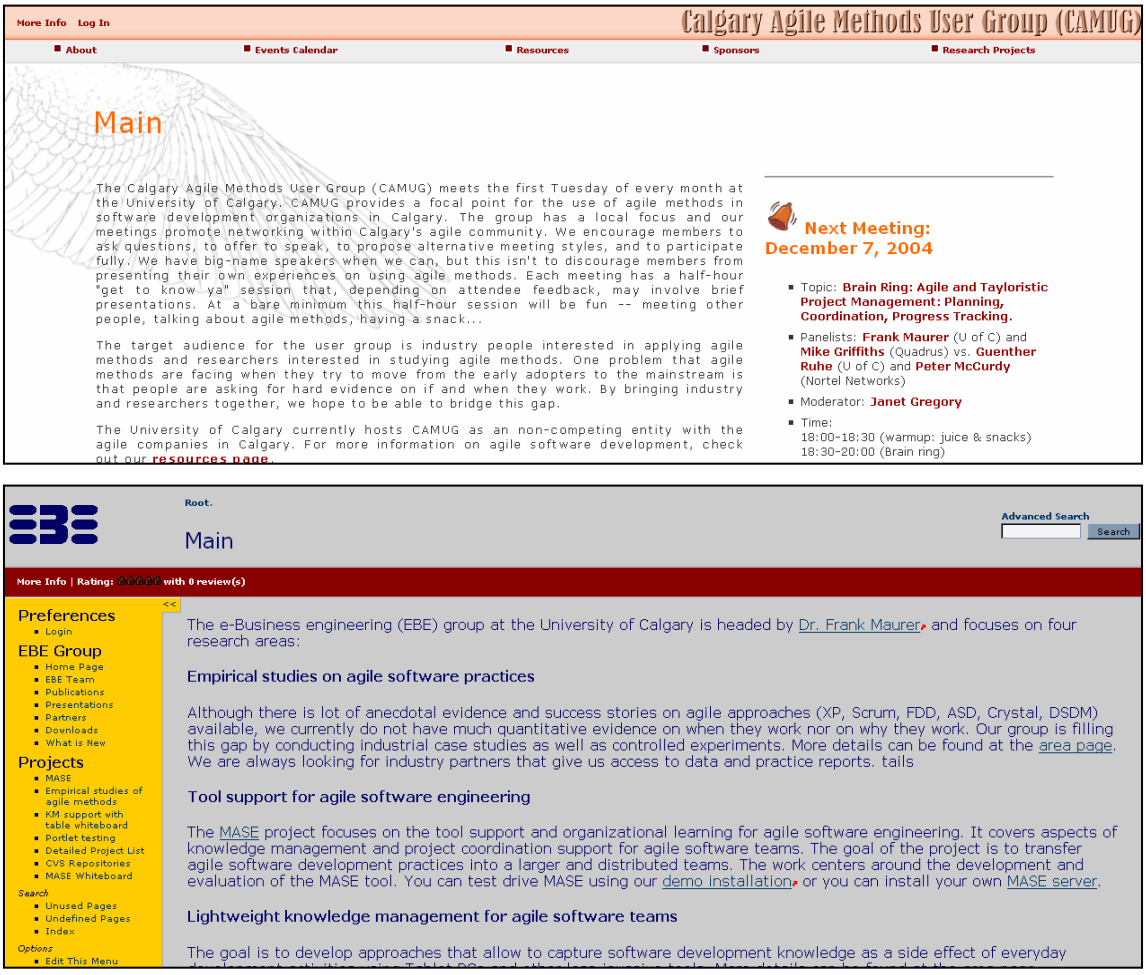


Figure 4.11: Despite their drastic differences in appearance, they are two copies of MASE running on the same source code

4.2 Implementation Details

As a web-based application, MASE is built atop a client-server architecture and is designed as a server-side application where all user requests are executed in a central server. The following figure illustrates a typical execution sequence where MASE accepts a user's request for a specific wiki page containing a plug-in and returns the page content as seen in a web browser (numbers match those in Figure 4.12).

1. A user uses a web browser to submit a request for a specific wiki page.
2. Upon receipt of the user request, the Request Dispatcher component diagnoses the type of the request and logs usage data for the requested page. Since this is a read-access for a wiki page, the Request Dispatcher delegates the request to the Wiki Engine component for further processing. If the incoming request is a write-access to some structured content, the Request Dispatcher will delegate the request to the Data Persistence Service component as illustrated by the dotted arrow.
3. The Wiki Engine component asks the Data Persistence Service component to retrieve the requested page from the underlying data storage mechanism (e.g., file system, database, etc.).
4. The Data Persistence Service component retrieves the specified page from the underlying data storage mechanism.
5. At this point, the content of the retrieved wiki page is formatted in Wiki markup language as that is the format into which content of a wiki page is stored. The Data Persistence Service component returns the retrieved page to the Wiki Syntax Parser component for converting the retrieved page into a format understandable to web browsers.
6. The Wiki Syntax Parser deciphers the content of the wiki page and translates it from the Wiki markup language to HTML, a format understandable to all web browsers. For each plug-in that the Wiki Syntax Parser detects in the wiki page, it will execute the plug-in by invoking the plug-in's callback method as defined in an application programming interface (API) which all plug-ins must adhere to.

7. Upon invocation, a plug-in performs its logic, and if necessary, retrieves structured data from the underlying data storage, and passes any data that needs to be displayed to its plug-in template. A plug-in template corresponds exactly to one plug-in.
8. The plug-in template accepts any input data and displays them in HTML.
9. The plug-in returns the generated HTML output back to the Wiki Syntax Parser.
10. The Wiki Syntax parser incorporates the returned HTML output into the translated content and continues translating the remaining page content.
11. When all the content of the specified wiki page is translated, the Wiki Syntax Parser passes the resulting HTML output to a collection of layout templates. These templates contain placeholders for page content and the layout can be modified (independent of other components) according to specific requirements of a community as shown in Figure 4.11. Upon completion, the laid out page content is displayed in the user's web browser.

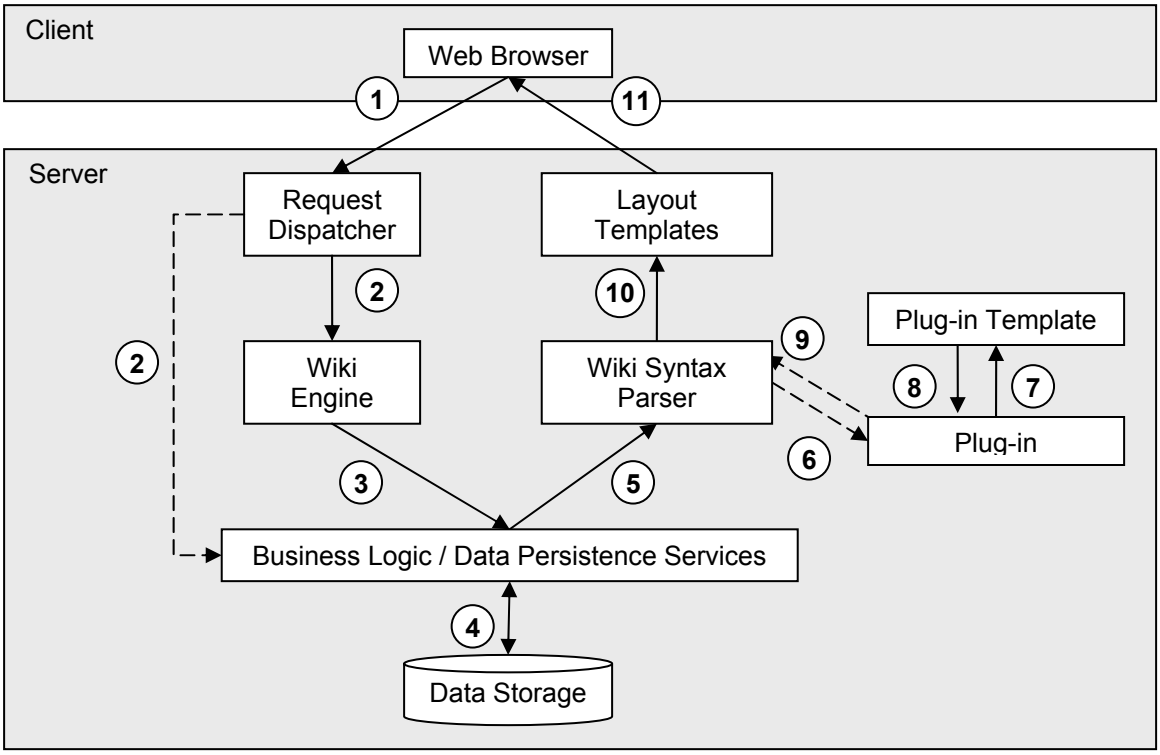


Figure 4.12: An execution view of MASE

With respect to system architecture, MASE is designed based on the Java 2 Enterprise Edition (J2EE) application platform [J2EE]. The J2EE platform is chosen for various reasons. First, the J2EE platform provides a standard set of application programming interfaces (APIs) for developing distributed application and this standard is widely accepted and used by the software development industry. The resulting application can be deployed and executed in different J2EE-compliant servers without modification of the application source code. This gives an organization the flexibility to switch among different J2EE-compliant servers offered by different server vendors while still using the same application. Second and more importantly, the J2EE platform offers several reusable components that address common infrastructure service requirements critical for building distributed applications. Leveraging these reusable components helps reduces tool developers' time for designing distributed application and allows tool developers to concentrate on solving the application requirements. One such technology is the Enterprise Java Bean (EJB) technology of the J2EE platform. It is used to simplify the implementation of the Business Logic Service and Data Persistence Service components in MASE. This technology allows components to interact with each other over a network and it provides automatic mapping of contents in a relational database to constructs (e.g., Session Beans and Entity Beans in Figure 4.13) in the Java programming language. The application developer does not need to write code that deals with connecting to a database according to database-specific protocols and parse the text or bytes returned from the database into primitive data types or constructs in Java.

The use of the J2EE application platform implies the use of Java as the programming language and the use of the N-tier system architecture. The Java programming language [Java] has the benefits of being object-oriented and portable. By object-oriented, it is meant that the language allows for constructs that naturally map to real-world objects; that functionalities can be partitioned into independent modules allowing code to be easily reused. By portable, applications written in Java can be executed in various operating systems without modifications to the application source code. With the N-tier system architecture, application functionalities are partitioned into

coarse-grained units called tiers. The figure below illustrates the N-tier system architecture of MASE.

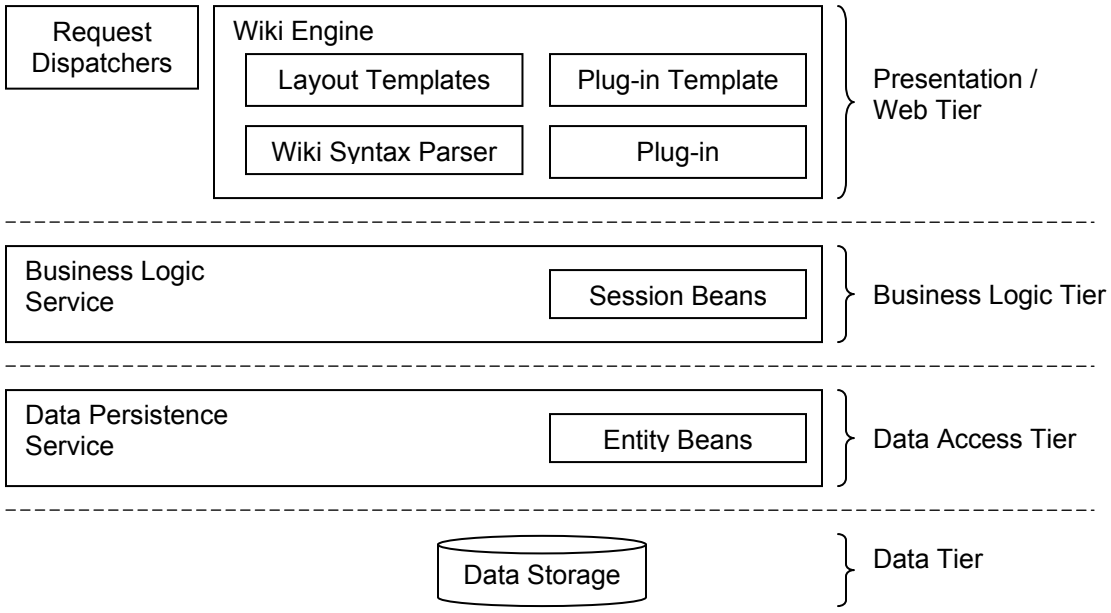
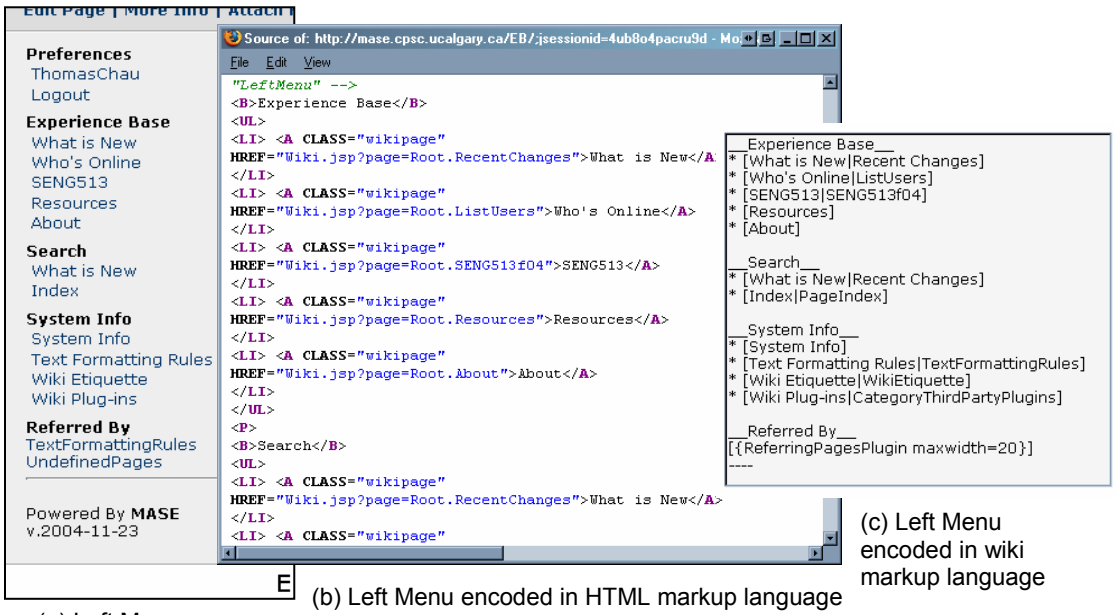


Figure 4.13: The N-tier system architecture of MASE

As mentioned above, the Business Logic and Data Persistence Service components are implemented using the EJB technology of the J2EE platform. For the Wiki Engine component, it is developed based on JSPWiki [JSPWiki], a specific implementation of the Wiki concept [Leuf and Cunningham, 2001].

The Wiki concept aims to facilitate quick collaboration by enabling any users to access, create, update, and organize web pages in real-time using only a web browser. These web pages are referred to as wiki pages. Wiki pages differ from web pages in that content in a wiki page is nearly free-formatted text whereas in a typical web page content is often embedded among HTML markup elements. Figure 4.14 contrasts the difference in the encoding of the same content in a typical web page and in a wiki page. The wiki concept is not only well accepted by agile methods practitioners [Kerievsky, 2001] but it is also used by practitioners in other fields, such as publishers and educators [Leuf and Cunningham, 2001].



(a) Left Menu

(b) Left Menu encoded in HTML markup language

(c) Left Menu encoded in wiki markup language

Figure 4.14: Content encoding of the navigation menu in a typical web page and in a wiki page

The JSPWiki implementation of the Wiki concept is chosen particularly as the Wiki Engine component in MASE for a number of reasons. First, it is implemented in Java. Second, it is by itself a J2EE-compliant web-tier component. Third, its system architecture provides a plug-in API which allows tool developers to easily add new functionalities with little or no modification to other parts of JSPWiki. More importantly, plug-ins are part of the JSPWiki's wiki syntax and they can be easily used by end-users without much configuration effort by the tool administrators. Plug-ins can be included on any wiki pages simply by referencing the names of the plug-ins in the wiki pages (e.g., Figure 4.5, bottom). Fourth, its Wiki markup language is a superset to that of C2 [Leuf and Cunningham, 2001], the popular and original implementation of the Wiki web concept. This reduces the learning time for users who are already familiar with the use of the original Wiki web. In addition, cross-referencing among wiki pages can be done with minimal effort by following the wiki naming conventions (see §4.1.1). Fifth, it provides version control mechanism for wiki pages. Sixth, its system architecture separates the logic for presentation layout and that for serving content. This allows for customizable layout without modification to the JSPWiki source code.

With respect to its limitations, JSPWiki only supports asynchronous collaboration. For supporting different types of artefacts, it only allows users to annotate information in an unstructured manner. For information retrieval, it does not have support for full-text search. It also does not have facilities for monitoring the usage of the tool. For these reasons, MASE is developed as an extension of JSPWiki in order to overcome these limitations while leveraging the benefits JSPWiki provides. For the sake of clarity, I defer the discussion of the high-level data structures used in MASE to Appendix C.1.

4.3 Example Walkthrough

The following is an illustration of a typical usage of MASE in an agile software development environment.

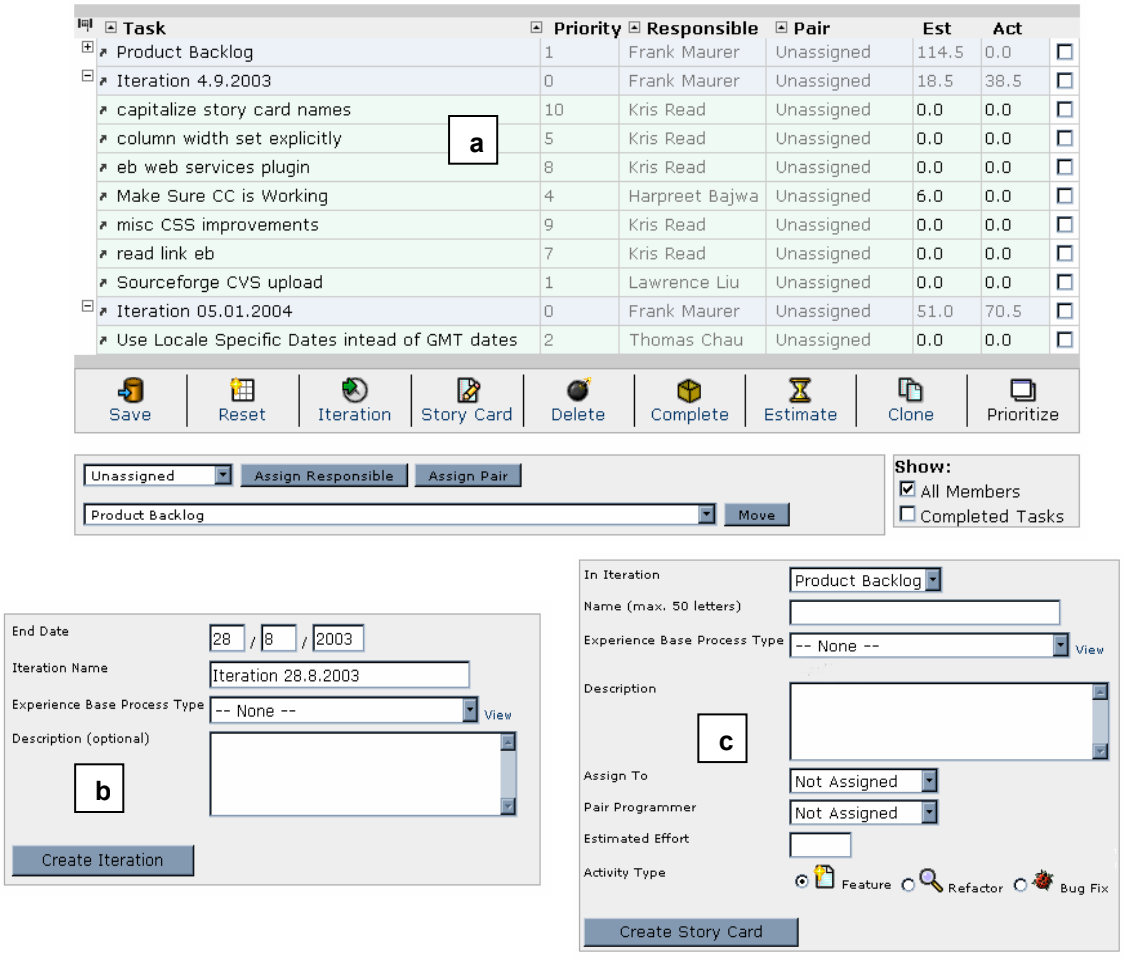


Figure 4.15: MASE plug-ins for supporting agile project management practices: (a) the *Whiteboard* plug-in; (b) the *CreationIteration* plug-in; and, (c) the *CreateStoryCard* plug-in

First, at a project planning meeting discussing the features to be done for the next iteration, team members can go to a wiki page in MASE containing the Whiteboard plug-in (Figure 4.15a). Through the plug-in, Frank (the customer) and Thomas (the project manager) can create iterations (Figure 4.15b) and user stories (Figure 4.15c) as structured data to be stored in the database.

Upon creation of an iteration, Kris and Lawrence (members of the development team) can submit the amount of their available times for that iteration, and MASE will compute a suggested size for that iteration using the developers' estimate accuracy from the previous iteration (Figure. 4.16). This suggested iteration size serves as a guide for Frank (the customer) and Thomas (the project manager), so they can prioritize user stories by allocating them to other iterations or back to the product backlog (Figure 4.15a).

Team Status for Iteration 24.8.2004	Available Hours	Initial Estimates	Initial Accuracy	Current Estimates	Current Accuracy	Actual Effort	Velocity
Frank Maurer	0.0	6.0	100.0%	6.0	100.0%	0.0	?
Kris Read	20.0	6.0	100.0%	6.0	100.0%	0.0	0.0%
Lawrence Liu	40.0	33.0	100.0%	33.0	100.0%	8.0	20.0%
Thomas Chau	20.0	0.0	?	0.0	?	0.0	0.0%
	Set Hours						
Totals	**Available Hours**	**Initial Estimates**	**Initial Accuracy**	**Current Estimates**	**Current Accuracy**	**Actual Effort**	**Velocity**
	80.0	45.0	100.0%	45.0	100.0%	8.0	10.0%
Goals							
Previous Velocity	67.9%						
Suggested Iteration Size	54 hours						
Accuracy Adjusted Size	44 hours						

Figure 4.16: The *TeamMetrics* plug-in within MASE that compute a suggested iteration size for a given iteration using the developers' estimate accuracy from the previous iteration.

Details of a user story or a task can be retrieved via plug-ins and stored in wiki pages. As a result, developers are free to annotate additional information, in free-formatted text, about the specific tasks that they are working on if they find the default set of input fields provided by the plug-in inadequate (Figure 4.17).

In addition, in cases where a particular user story involves a commonly performed task, team members can associate that user story with a process type in the process

model, a hierarchical data structure consisting of commonly performed tasks known as process types (see §4.1.2). When MASE retrieves a wiki page that contains structured content pertaining to a user story and if that user story is associated with a process type in the process model, MASE automatically retrieves the wiki page containing information about the associated process type and displays it side-by-side with the wiki page containing the user story-related information. For instance, the lower right corner of Figure 4.17 shows that Lawrence is working on the task, "Refactor EB Project" and that this task is already associated with the "Front Controller J2EE Refactoring" process type in the process model. Every time when Lawrence visits this page to access information about his "Refactor EB Project" task, he automatically sees information about others' experiences on tasks of similar kind. These experiences may have been provided by others (in this case, Carmen) whom Lawrence may know but are from other teams in the organization. Knowing that Carmen and he both share common interest in the topic of refactoring, Lawrence can contact Carmen, exchange their individual experiences and expertise, and collaborate together.

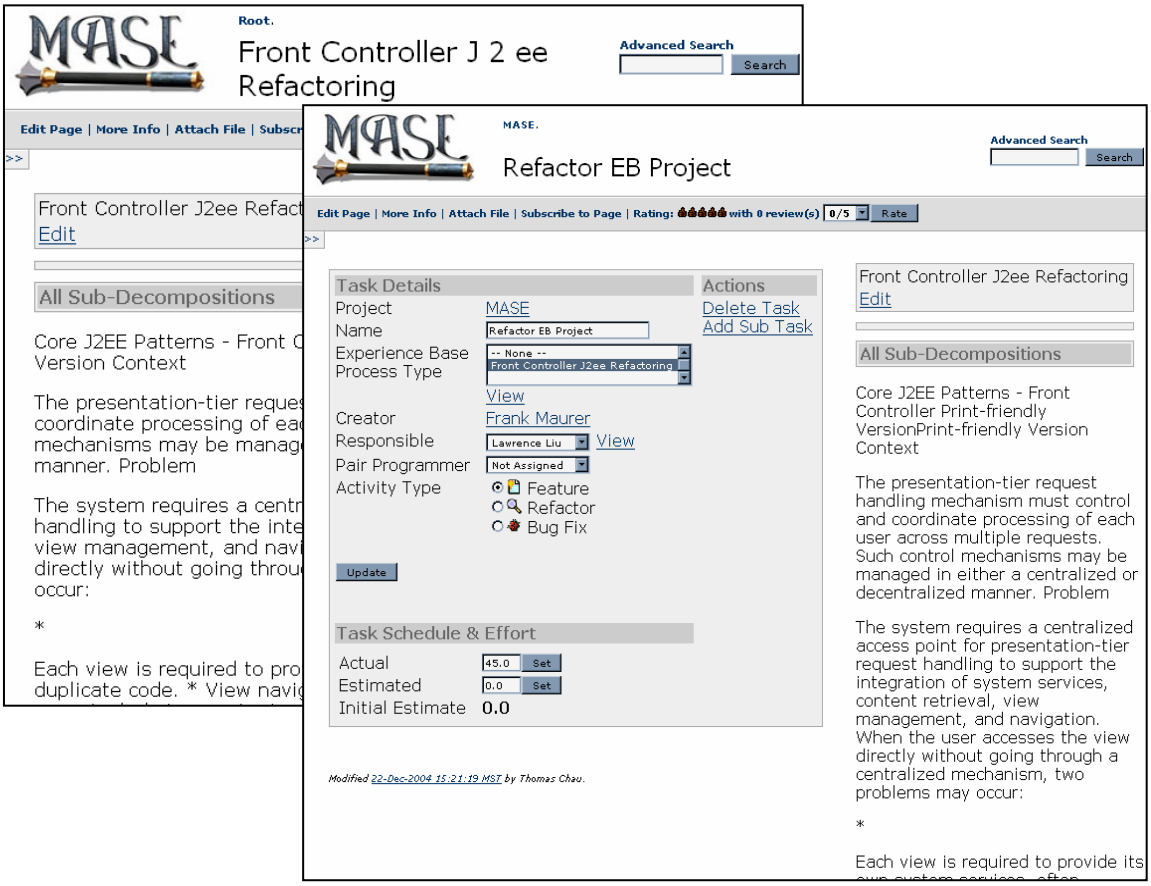


Figure 4.17: MASE delivers on demand potential reusable knowledge content from commonly performed tasks in the process model to workspace

4.4 Summary

In this chapter, I have described the design and capabilities of MASE in supporting the integrated approach of an augmented Experience Factory framework and the Communities of Practice concept for facilitating inter-team learning in an agile multi-team software development environment.

MASE is a web-based collaborative environment. It follows the Wiki concept and is designed as a J2EE application. With MASE, users can create, edit, cross-reference, and retrieve artefacts of various types, such as wiki pages, iterations, tasks, user stories, and generic files from a repository. For information retrieval and organization, MASE provides portals for individual users and allows for subscription and notification of artefacts updates. Users can also use MASE to organize artefacts, especially those that

contain task-related knowledge, into a hierarchical process model containing information about commonly preformed tasks. The example walkthrough illustrates that information in a process model can be reused and automatically be retrieved as agile methods practitioners look up information about their work tasks.

MASE also accommodates various styles of collaborative work. As a web-based tool, it supports co-located and distributed work. It allows users to save the state of any wiki pages, thus supporting asynchronous interaction. It also provides plug-ins for affording online awareness and integrating with Microsoft NetMeeting, thereby supporting real-time interaction. Its default permission model of allowing any users to create and edit any content at any time as well as its use of the simple Wiki markup language for content encoding allows for ease of participation for both knowledge seekers and providers.

MASE also tracks tool usage statistics for analyzing the content and usage of the tool. Using these statistics, users and administrators alike can infer areas of interest for users, gauge the rhythm of online collaborative activities, identify potential CoPs, and concentrate their repository maintenance effort. To facilitate different levels of participation within a CoP, MASE allows users to define their own groups and organize the associated artefacts accordingly in a hierarchical structure. To facilitate knowledge sharing across different CoPs, MASE provides means for integration with external information sources.

Due to its use of a plug-in architecture, MASE allows tool developers to extend its existing functionalities. The separation of layout presentation logic and content management within the system architecture also allows the user interface of MASE to be easily configurable.

Through the example walkthrough, I have demonstrated how MASE can be used in both the planning practices and day-to-day tasks in an agile software development environment.

In the next chapter, I will discuss the extent to which MASE satisfies the 19 criteria that are essential for any knowledge sharing tools that are to be used in an agile multi-team software development environment.

Chapter 5. MASE: Qualitative Analysis

One of the objectives of this work is to investigate and, if necessary, design the appropriate tool support for facilitating inter-team learning in an agile software development setting. As discussed at the end of chapter 3, there is indeed a need for a novel tool support to address our research problem of how to facilitate inter-team learning in an agile multi-team software organizational setting. This proof-of-concept tool, MASE, is discussed in details in the previous chapter. In order to explore and evaluate the effectiveness of MASE, I present in this chapter a qualitative analysis of MASE using the 19 criteria elicited at the beginning of chapter 3. In the next chapter, I will present an empirical evaluation of MASE in both academic and industrial settings.

In the following table is a summary of the 19 criteria and the extent to which MASE satisfies them.

Criterion	MASE
Support different types of artefacts	✓
Intuitive characterization of reusable artefacts	✓
Support for creating, editing, retrieving, and reusing artefacts	✓
Support for providing feedback on reusable artefacts	✓
Precise representation of reusable artefacts	✓
Links between reusable artefacts	✓
User guidance for artefacts retrieval	✓
Variety of interaction	✓
Presence awareness	✓
Ease of participation	✓
Allow for short-term return by providing quick answers to questions as well as access to artefacts and/or experts	✓
Allow for long-term return by providing mechanism for CoP members to maintain and refine their shared repertoire of resources	✓
Rhythm generation and detection	✓
Integration with external information sources	✓
Foster belonging and relationship	✓
Managing CoP boundaries	✓
Extendible and configurable	✓
Identification of potential CoP	✓
Analysis of experience base / support active community building	✓

Table 5.1: Assessment of MASE by the 19 tool support criteria

In terms of support for different types of artefacts, MASE lets users annotate information in wiki pages or in any other types of files that can be attached to any wiki pages (see Figure 4.1f). For facilitating the intuitive characterization of artefacts, MASE allows agile team members to conveniently classify their artefacts into predefined types that are commonly used in agile software processes. Alternatively, users can also classify their artefacts into arbitrary types other than the ones listed above (see §4.1.2).

With respect to support for creating, editing, and retrieving artefacts, MASE users can perform all such manipulations on the artefacts via a web browser (see Figures 4.1, 4.2). As for support for reusing artefacts, MASE users can organize potentially reusable artefacts into hierarchical process models, content of which is also annotated in wiki pages (see §4.1.2, §4.3). Users can subsequently provide feedback on them by giving a rating and/or posting comments on those wiki pages (see Figures 4.1h, 4.2). Furthermore, users can decompose (or consolidate) contents of the process models into finer (or coarser) granularity at any time. Users can also store the contents of the process models and any content in MASE in a representation that is unstructured, or structured, or a combination of both.

With respect to support for cross-references among artefacts, MASE not only allows users to manually define links among artefacts, it can also automatically generate cross-references among potentially related artefacts (see Figure 4.4). In terms of facilities for guiding users to retrieve artefacts that are potentially relevant to them, MASE, users can subscribe to any artefacts that they are interested in. For task-related information in particular, knowledge content in the process models can automatically be retrieved as agile methods practitioners look up information about their work tasks (see §4.3).

With respect to supporting various styles of collaborative work, MASE supports co-located and distributed work as it is a web-based tool. It allows users to save the state of any wiki pages, thus supporting asynchronous interaction. It also provides plug-ins for affording online awareness and integrating with Microsoft NetMeeting, thereby supporting real-time interaction. These capabilities and the provision of a full-text search

engine provide opportunities for users to acquire quick answers to questions and access to artefacts and/or experts.

MASE's default permission model of allowing any users to create and edit any content at any time as well as its use of the simple Wiki markup language for content encoding allows for ease of participation for both knowledge seekers and providers. With respect to the ability for members to maintain and refine their shared repertoire of resources, MASE provides a centralized repository.

In terms of facilities for raising users' awareness of collaborative work other than their own, MASE allows users to see a quick list of recently updated artefacts. For project-related activities in particular, users can choose to be notified of new tasks. As for integration with external information sources, MASE not only allows users to embed hyperlinks to external web sites but also the content of external web sites within wiki pages (see Figure 4.9b).

In terms of support for fostering better knowledge and trust among community members, MASE automatically provides each user with a personal portal and maintains for each user a public profile. The personal profile contains both automatically generated information and user-defined content (see Figure 4.7b). With respect to supporting sub-communities, MASE can be used by multiple CoP. To facilitate different levels of participation within a CoP, MASE allows users to define their own sub-communities and organize the associated artefacts accordingly in a hierarchical structure (see Figure 4.9a).

To allow itself to be extendible, MASE's system architecture provides a set of APIs that third-party software developers can use to develop novel plug-ins to extend MASE's existing functionalities. The separation of layout presentation logic and content management within the system architecture also allows the user interface of MASE to be easily configurable (see Figure 4.11).

MASE also tracks tool usage statistics for analyzing the content and usage of the tool (see Figure 4.10). Using these statistics, users and administrators alike can infer areas

of interest for users, gauge the rhythm of online collaborative activities, identify potential CoPs, and concentrate their repository maintenance effort.

5.1 Limitations

In spite of the aforementioned capabilities, there are also limitations to MASE, particularly in terms of how it achieves some of the aforementioned capabilities. For instance, to support real-time interaction, MASE provides a plug-in that integrates with Microsoft NetMeeting. While this can help facilitate informal and spontaneous interaction and collaboration, this support is quite limited as it only supports one-to-one communication and not a one-to-many communication.

5.2 Summary

In comparison to the existing knowledge sharing tools examined in Chapter 3, the above qualitative analysis indicates that MASE is capable of addressing all of the 19 criteria that are essential for any tools that aim to facilitate inter-team learning in a multi-team agile software organizational setting. Nonetheless, it is also desirable to investigate the effectiveness of MASE in practice. To this end, I will discuss in the next chapter two case studies on MASE and their respective findings.

Chapter 6. MASE: Empirical Evaluation

In the preceding chapter, I have described the design and implementation of MASE, a proof-of-concept tool for facilitating knowledge sharing and inter-team learning in an agile software development environment. In order to provide evidence for supporting the use of a tool like MASE, I present in this chapter details of two case studies that were conducted to empirically evaluate MASE.

First, I introduce the goals of these two case studies. Then, I discuss the first and second case studies conducted in an academic environment and in an industry setting respectively. For each case study, I detail its environmental context, its methodology including a description of the study participants, the apparatus, the data collection and analysis strategies, the results, and an interpretation of the results. I close by inspecting the limitations of the two case studies.

6.1 Objectives

The objectives of the two case studies reported here are both exploratory and descriptive in nature with the following research goals in mind:

1. to understand how developers use an informal knowledge sharing tool like MASE for sharing their experiences and learning;
2. to elicit the challenges, enabling factors, and perceived benefits of using an informal knowledge sharing tool like MASE for knowledge sharing and inter-team learning; and,
3. to gather feedback on MASE's usability and functionalities

For the first goal in particular, the specific research questions that I attempt to address via the case studies are:

1. How do users collaborate with each other in MASE, given the tool's flexibility in supporting real-time and asynchronous work?

2. What types of knowledge (e.g., task-related, team-related, personal, social, etc.) are being shared on MASE?
3. What are the characteristics of the users with respect to the information that they use in MASE?
4. How do users use the information content in MASE, given the tool's flexibility in supporting the use of structured and unstructured information?
5. To what extent do users self-organize among themselves in maintaining the information content in MASE given the open-edit nature of the tool?

6.2 Case Study in an Academic Setting

Prior to commencement of this first case study, I obtained ethics approval from the University of Calgary for experimentation involving human participants. The letter of approval from the ethics committee at the University of Calgary can be found in Appendix B.3.

6.2.1 Environmental Context & Participants

Participants of this study include a total of 42 senior undergraduate students of which 25 are enrolled in the course, *Web-Based System*, at the University of Calgary (UC) and the remaining 17 enrolled in a similar course, *Internet Software Techniques*, at the Southern Alberta Institute of Technology (SAIT). In particular, the UC course is a 4^{th} year course students of which were either in their second last or last year of their 4-year computer science program. The 25 UC students were divided into 6 teams each consists of 4 to 5 members. Likewise, the 17 SAIT students were divided into 6 teams each consists of 3 to 4 members.

Both courses lasted 4 months and they exposed students to the latest technologies and practices in building Web-based enterprise systems. The content of both courses were offered by the same instructor in approximately the same time period. Over the 4-month time frame, all teams were required to complete 6 comprehensive programming assignments involving the construction of a document review system. Each assignment was completed in a short iteration, usually 2 to 3 weeks long. Teams were encouraged to

share their individual learning and work on their assignments following agile development practices such as short iteration and pair programming. In terms of the participants' knowledge or past experience with Web technologies, responses to the post-study questionnaire indicate that the mean amount of experience is between 1 to 12 months.

6.2.2 Apparatus

In the first week of the study, all teams received an hour of training on using MASE. When each assignment was released, dedicated knowledge brokers (the class instructor and teaching assistants) posted in MASE information that they deemed as important for all teams to know for completing their assignments. All teams from both UC and SAIT had read- and write-access to MASE.

6.2.3 Data Collection and Analysis Strategies

For data collection, MASE logged all usage of the tool by any participants as discussed in §4.1.6. In addition, each participant was asked to voluntarily complete a questionnaire (Appendix B.2) at the end of the 4-month observation.

For data analysis, I constructed a database with information from the MASE system logs and from responses to the questionnaires. I inspected the logs manually and applied a simple categorization of the users into: *student* and *dedicated knowledge brokers*; of tool access into: *read*, *write*, *search*, and *synchronous interactions*; and of the content in each wiki page into: *problem understanding*, *instrumental*, *projective*, *social*, and *content navigation aid*.

The categorization scheme of the wiki page content is based on work done in the information science field [Taylor, 1991]. In his work, Taylor defines *problem understanding* as using information to increase comprehension of a problem; *instrumental* as using information to follow guidelines or procedures; *projective* as using information to make forecasts or scenarios; and, *social* as using information to develop relationship that can be personal and/or political in nature. Unlike the other four categories, the category of *content navigation aid* is specific to MASE. This is because

MASE allows users to collaboratively author content freely; as such, some wiki pages may be authored specifically to guide users to find particular types of content.

6.2.4 Results

To answer the first research question of how software developers make use of MASE to collaborate with each other, I analyzed the various ways that users accessed information in MASE. The result (Table 6.1) suggests that MASE is primarily used for asynchronous collaboration as indicated by the low percentage of use of MASE's NetMeeting integration feature. The result also indicates that users use MASE primarily for retrieving information content. In comparison, there was very little use of the tool to collaboratively author information content. With respect to information retrieval patterns, the high number of read requests compared to search requests indicates that the search engine is seldom used.

Types of Access	# of Use	% of Use
Read	32,978	95.52
Write	1,376	3.99
Search	152	0.44
Synchronous interactions via NetMeeting	17	0.05

Table 6.1: In an academic setting, how do users use MASE to collaborate with each other?

To answer the question of what types of knowledge can be found in a knowledge repository in a software development setting, I analyzed and categorized all the wiki page content in MASE. The analysis reveals that users make use of wiki pages in MASE to:

- understand the business rationales underlying the programming tasks;
- discuss, troubleshoot, and seek help from peers on technical problems that arose while performing the programming tasks;
- guide users to find potentially relevant pages by topic;
- share knowledge on common design patterns for solving programming tasks;
- store information about themselves, their interests, and their availabilities; and,
- schedule programming tasks and track their progress.

When the aforementioned usages of the wiki page were analyzed using Taylor's categorization scheme, those pages used for "*understanding the business rationales*

underlying the programming tasks" could be categorized as "problem understanding". For pages used for "*guiding users to find potentially relevant pages by topic*", they could be classified as "content navigation aid". For pages used for "*scheduling programming tasks and progress tracking*", they could be classified as "projective".

Not all pages can be classified precisely under one category, though. For instance, those pages used for "*storing personal contact and availability information*", they could be classified as "social" and "expertise location".

Those used for "*discussing, troubleshooting, and seeking help from peers on technical problems*" could be categorized as "problem understanding" because users commonly described on those pages the specific problems that they were having troubles with. Such pages could also be classified as "instrumental" because it was common that they contain detailed steps that a user can follow for solving the technical problems he/she is facing. These pages could also be classified as "expertise location" because users generally provided an explicit link to their personal page or their team's page when they made their postings. Other users could, and some actually did, followed those links to obtain the contact information of the team or user who made the posting.

The result of this content analysis is summarized in Table 6.2. The result also indicates that the various types of information content are not used to the same degree. For instance, although only 36% and 13% of the wiki pages in MASE were used for problem understanding and content navigation aid purposes, these two categories of pages were accessed the most at 72% and 64% of the time respectively. In fact, Figure 6.1 shows that 80% of all the read-accesses to information content in MASE were concentrated to less than 20% of the pages.

Types of Content	% of Pages	% of Read Access	% of Write Access	% of Overall Access
Problem Understanding	36	70	2	72
Content Navigation Aid	13	64	1	64
Instrumental	45	47	2	49
Expertise Location	25	25	2	27

Social	28	19	2	21
Projective	2	1	0	1

Table 6.2: In an academic setting, what types of knowledge can be found in a software development knowledge repository?

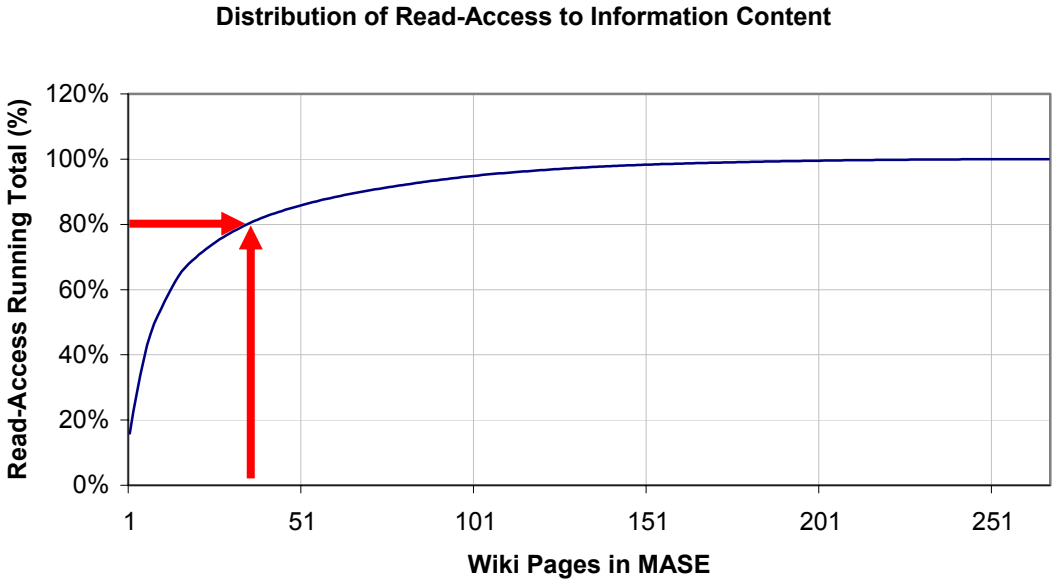


Figure 6.1: In an academic setting, how are read-accesses distributed in a software development knowledge repository?

Of the 10 most visited wiki pages in MASE, 7 of them were pages not bundled with initial MASE installation but they were created by users spontaneously to share knowledge unique to their tasks. Of the other 3 pages, the "ListUsers" wiki page lists contact information for other MASE users, and provides the means for users to interact with one another via email or NetMeeting; the "RecentChanges" wiki page lists all pages that have been updated over a certain period of time as specified by users; and the "Main" wiki page serves as the home page of MASE.

With respect to content contribution, over the course of the case study, MASE users created nearly 3 times (208/72) as many new wiki pages as the default wiki pages that were bundled in MASE when the case study began. Nonetheless, the trend depicted in Figure 6.1 is also observed for distribution of write-access to information content; about 80% of the content in MASE is contributed by about 20% of the users. Yet when the distribution of write-access to information content is analyzed in terms of user

characteristics, the result (Table 6.3) reveals that particular type of information content are contributed more by particular type of users. For instance, the authoring of content for problem understanding and instrumental purposes were almost evenly distributed among the student developers and the dedicated knowledge brokers. In contrast, close to 80% of content for social and expert finding purposes were authored by student developers whereas dedicated knowledge brokers authored 98% of wiki pages for assisting content navigation. In terms of collaborative authoring of information content in MASE, about 8% of the pages (23/280) were contributed by 2 or more authors.

Type of Content	Student Developers Contribution (%)	Dedicated Knowledge Brokers Contribution (%)
Problem Understanding	41	59
Instrumental	45	55
Projective	100	0
Social	82	18
Expertise Location	76	24
Content Navigation Aid	2	98

Table 6.3: In an academic setting, how is write-access to different types of content in a software development knowledge repository distributed across different types of user?

To answer the question of to what extent users would formalize or structure the information content in a software development knowledge repository, the analysis in Table 6.4 shows that an overwhelming amount (92%) of information was annotated by the users in either purely unstructured form or in a hybrid form that combines both unstructured and structured forms.

Formalization of Content	% of Pages
Unstructured	75
Both	17
Structured	8

Table 6.4: Users' preference for formalization of repository content in an academic setting

To answer the question of whether there exists any self-organized maintenance of information content in the knowledge repository given the open-edit nature of MASE, I analyzed the distribution of write-access to MASE by the user role. Of the 61 registered users in MASE, only 38 of them contributed content. Furthermore, the analysis (Figure 6.2) shows that close to 80% of the content are contributed only by approximately 20% (12 out of 38) of the authors. In fact, the analysis in Table 6.5 shows that close to 50% of

the content in MASE are contributed by dedicated knowledge brokers who are responsible for structuring and updating the repository content.

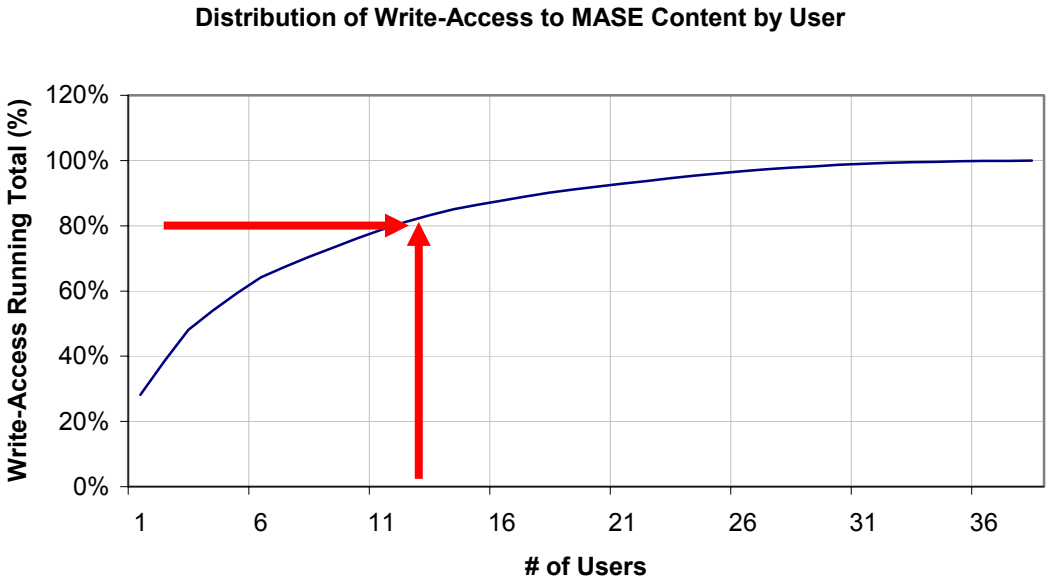


Figure 6.2: In an academic setting, do users self-organize among themselves in maintaining the content in a software development knowledge repository?

User	Role	Write-Access Running Total (%)
7	Dedicated Knowledge Broker	28
1	Dedicated Knowledge Broker	38
4	Dedicated Knowledge Broker	48
13	Student Developer	54
9	Student Developer	59
28	Student Developer	64
19	Student Developer	67
31	Student Developer	70
11	Student Developer	73
25	Student Developer	76
29	Student Developer	79
14	Student Developer	81

Table 6.5: In an academic setting, close to 50% of the content are contributed by dedicated knowledge brokers who are responsible for maintaining the repository.

To understand the challenges, the enabling factors, and the perceived benefits of using an informal knowledge sharing tool like MASE for knowledge sharing and inter-team learning, all 42 student developers at both UC and SAIT were asked to voluntarily complete a questionnaire (Appendix B.2) at the end of the study. The response rate of the questionnaire is 71% with 30 respondents.

When asked of the means they had used to collaborate with peers outside of their teams (Table 6.6), over 60% of the respondents reported MASE and instant messaging tools. Nearly half of the respondents opted for face-to-face conversations as well.

Collaboration Means	Yes (n=30)
MASE	63%
Chat tools	60%
Face-to-face conversations	43%
Phone	23%
e-mail	17%

Table 6.6: Breakdown of collaborative means in an academic setting

Respondents were also asked what motivated them and what made them hesitant in *contributing* and *looking up* content in MASE. For *contributing* content, *motivating factors* cited include: when the respondent's team had exhausted all other resources and still could not solve their problems on their own; when the respondents saw that others had also posted question; and, when the respondent knew the answers to the questions posted by others. Common *obstacles* to contribute content include: potential long response time and the respondent did not know the subject under discussion.

For *looking up* content in MASE, *motivating factors* cited include: desire to know how other teams had approached the problems that the respondent had encountered; and, when the respondent's team had exhausted all other resources and still could not solve their problems. Common *obstacles* to look up content include: the respondent could readily talk to other people about the problem face-to-face; the respondent preferred to solve the problems that they have encountered on their own; and the respondent did not think others would have the answers to their problems.

When asked of their perceived usefulness of MASE for organizing and sharing one's experience or learning with others (Table 6.7), over half of the respondents found MASE to be helpful or very helpful with another one-third found MASE to be average.

Perceived Usefulness	Respondents (n=29)
Not at all to Very Little	14%
Average	34%
Helpful to Very Helpful	52%

Table 6.7: Perceived usefulness of MASE in an academic setting

When asked of their willingness to use MASE for knowledge sharing, two-thirds (67%) of the 27 respondents stated that they would.

6.2.5 Interpretation

This case study sets out to explore and describe how software developers would use an informal collaborative knowledge repository such as MASE. The insight to this high-level question can be found by looking at the answers to the specific research questions:

Question 1: How do users collaborate with each other in MASE, given the tool's flexibility in supporting real-time and asynchronous work?

Observations from the case study indicate that MASE is primarily used for asynchronous collaboration in spite of its support for synchronous collaboration. The little use of the tool's NetMeeting integration feature is explained by the fact that most student developers were often co-located at the same computer lab working on their programming tasks; they preferred to talk to each other in person when they encountered difficulties in their tasks, as indicated by their response to the survey questions. In addition, most student developers probably had been using on a regular basis some type of chat tools other than NetMeeting prior to the case study; they probably preferred much more to continue using their favourite chat tools instead. While MASE was used very little for actual synchronous collaboration, the users made lots of attempts to engage in potential real-time collaboration. This is supported by the fact that the "ListUser" wiki page was listed among the top 10 most accessed pages as it showed the intense use of the online awareness feature of MASE. MASE is also found to be used mostly for retrieving information content. There was comparatively very little use of the tool for

collaboratively authoring content. This may be explained by the answers to research question 6.

Question 2: What types of knowledge (e.g., task-related, team-related, personal, social, etc.) can be found in an informal knowledge repository like MASE?

Despite the informal nature of the tool, observations from the case study show that users did not exploit the open-edit policy of the tool to store any arbitrarily kind of content. With the exception for using information for projective purposes, much of the content in MASE reflects Taylor's description of information use accurately [Taylor, 1991].

Question 3: What are the characteristics of the users with respect to the information that they use in MASE?

Despite the high number of read-access to pages that served as content navigational aid, results from the observation indicate that the task of maintaining content on these pages was often left to the dedicated knowledge brokers. This may be because these pages were perceived by the regular users (the student developers) to be belonged to users with higher authority (the dedicated knowledge brokers); or just that the information annotated on those pages was accurate and regular users saw no need to update them. In comparison, regular users authored most of the pages that were used for social and expert finding purposes. In fact, regular users use their personal wiki pages *actively* for accessing resources they frequently used. In general, the content of most of the personal wiki pages is very similar to one another. The active use of these personal wiki pages suggests the value in providing users with their personal information space in knowledge sharing tools.

Question 4: How do users use the information content in MASE, given the tool's flexibility in supporting the use of structured and unstructured information?

Observations from the case study indicate that an overwhelming 94% of the information content was annotated in either unstructured or in a hybrid form that consists

of both structured and unstructured form. This shows that there is a greater need for unstructured than structured knowledge and provides clear evidence that it is insufficient for knowledge sharing tools to support only structured information content; there needs to be built-in support for both structured and unstructured content.

Question 5: Do users self-organize among themselves in maintaining the information content in MASE given the open-edit nature of the tool?

Results from the observations reveal that nearly half of the content in MASE was contributed by the dedicated knowledge brokers. In particular, these dedicated knowledge brokers were responsible for maintaining most of the pages that served as content navigational aid as mentioned in the answers to question 3. These two facts suggest the lack of self-organizing processes among the regular users in maintaining the repository content. This finding is not surprising given the academic setting of this case study; student developers often expect information to be given to them and they are often reluctant to seek and share their learning with one another.

Question 6: What are the challenges and enabling factors of using an informal knowledge sharing tool like MASE for knowledge sharing and inter-team learning?

Based on the participants' feedback from the post-study questionnaire, one common challenge to share one's learning with other teams in MASE is the lack of trust in other teams that they would possess the right expertise. Another challenge is reluctance to re-use known solutions. The common motivating factors for sharing one's learning with other teams are knowledge of the subject under discussion and a desire to help others. With respect to the issue of lack of trust among teams, possible explanations include the lack of relationship among the different teams prior to the course.

The above findings all reflect issues that are more social and cultural than technical in nature. These findings reaffirm the anecdote that social and cultural issues must be addressed first before the benefits of a technical solution can be fully realized.

6.3 Case Study in an Industrial Setting

To acquire more insights into how an informal knowledge repository such as MASE is used and to validate the observations found in the previous case study, the previous case study is replicated in an industrial setting.

6.3.1 Environmental Context

The software company involved in this case study is empolis (Germany), which specializes in producing software for knowledge management support. As such, its domain expertise gives empolis unusually keen insight into its own inter-team learning requirements. Empolis consists of employees who are distributed across four offices in Germany, as well as offices in the UK, Poland, Hungary and Dubai. This distribution of labour and resources presents unique challenges in how knowledge is represented and shared.

Within the empolis organization, several small teams work on three historically grown product lines of software that share some commonalities. As such, these teams operate relatively independently of one another on different but often related projects and core products. For example, one team may be working on a new product release, while another team works on customizing the previous release for a customer. Yet another team may be engaged in research and new product development. Depending on the project, teams consists of varying stakeholders—customer-specific project teams include marketing personnel, while new development teams consists mostly of programmers.

Initially, there was no way for these teams to share experiences in a consistent and recognized way. Occasions for face-to-face interactions had been set up as much as possible but the benefits of these efforts were limited due to the distributed team environment. To overcome this constraint, some teams had taken matters into their own hands and used personal web sites, web logs, and even a wiki for representing artefacts and information among members of single teams. Knowledge contained in these ad-hoc knowledge repositories include best practices, lessons learned, technical data, artefacts used and created by the teams, and even some personal information shared among more

social developers. These existing mechanisms were not applied across project or team boundaries and were not recognized between empolis offices in different cities and countries. In addition, the user base of these repositories often was entirely made up of software developers, and did not largely include other stakeholders or employees. Teams with better knowledge management tools were recognized at empolis to produce better results. Aware of these practices and their effects, the upper management at empolis deemed a more widely adopted inter-team learning solution as a necessity.

Empolis' interest in MASE was chiefly in its lightweight, flexible approach to the Experience Factory concept, and in its ability to support the personalization strategy that is lacking in other knowledge management tools which support only the codification strategy. Even empolis' own knowledge management tools focused on this approach. Goals at empolis included both gaining from the benefits of knowledge management in practice as well as gathering experience with such lightweight tools to help with future development of their own knowledge management solutions.

6.3.2 Participants and Apparatus

With respect to the demographics of MASE users at empolis, a total of 80 employees from 8 teams self-registered for MASE accounts. About 10% of the total registered users are managers and technical writers with the remaining 90% of users being developers.

Design and installation of the MASE system at empolis was largely driven by a specific vision communicated by the CEO, based on specific examples he had observed and elicited from existing teams and some of his own ideas. The primary objective was to create a company-wide wiki for internal use, allowing teams to self-organize new knowledge after some resources and time were expended entering existing important information into the system. Bottom-up support was initially varied, with some teams and individuals being highly enthusiastic and participatory, and others unsure of the value that any top-down system would bring them. In order to advertise the MASE presence, the empolis CEO offered a reward to users who contributed content to MASE in the early

days. At the time of this writing, no more explicit incentives were used to motivate contribution to MASE.

After the initial installation of MASE and while it was being used, we kept a wiki page of bugs and feature requests from wiki users.

One critical feature added to MASE at the request of empolis was a private messaging system to allow users to communicate privately within MASE rather than using email, as well as to receive in-browser notification of wiki page changes. The private messaging mechanism works very similar to the text-chat facilities in online chat software such as MSN Messenger. This feature was implemented as the NetMeeting plug-in in MASE only works on computer running the Windows operating system. Although the private messaging feature was fairly simple to implement, informal feedback from users indicate that this feature was critical to making many of the users feel as though they were communicating and collaborating rather than simply generating documents.

Another major feature requested by the CEO was a way to examine the users' participation level in MASE. This was in order to see how quickly MASE was being adopted, from what offices or projects, and with what kinds of data.

Another feature added at empolis was paper-based reporting. By default, MASE is not set up to generate printable reports of wiki contents, project tasks or other information. Even though the history of all wiki pages was kept internally, a feature was devised at their request to make it possible to periodically print off large reports of all wiki content.

Because empolis is a German company, some effort was required to ensure that the wiki pages and the plug-ins were compatible with international character sets. Although much of the business at empolis is done in English, spontaneous and interpersonal communication is almost always done in German, which resulted in an interesting mix of both languages depending on the type and purpose of the wiki content.

6.3.3 Data Collection and Analysis Strategies

A partnership between empolis and the University of Calgary allowed a researcher (not the author) to work on-site at an empolis business office over two three-week periods, installing and customizing MASE to meet their knowledge management needs (e.g., configuring MASE to store content in a file system instead of in a source code configuration management server). Installation of MASE was done at a single office in Kaiserslautern, Germany, however the presence of an organization-wide intranet means that teams and individuals from all empolis offices have access to MASE.

Data for this case study has been collected over a three-month period, starting at the end of May 2004 and ending in mid August 2004. The observations come mainly from static analysis of logs generated by MASE and from informal feedback given by the empolis representatives. All the contents in the wiki pages are analyzed in a similar manner as described in §6.2.3.

6.3.4 Results

To answer the research question of how project teams at empolis make use of MASE to collaborate with each other, I analyzed the various ways that users accessed information in MASE. The result (Table 6.8) suggests that MASE is primarily used for asynchronous collaboration. It also indicates that project teams at empolis use MASE primarily for retrieving information content and not as much for collaboratively authoring information content. With respect to information retrieval patterns, the result indicates that users prefer to browse to the desired content rather than using the search engine.

Types of Access	# of Use	% of Use
Read	10,400	80.88
Write	1,757	13.66
Search	509	3.96
Synchronous Interaction	193	1.50

Table 6.8: In an industrial setting, how do users use MASE to collaborate with each other?

To answer the question of what types of knowledge can be found in a knowledge repository in a software development setting, I analyzed and categorized all the wiki page content in MASE. The analysis reveals that users make use of wiki pages in MASE to:

- exchange ideas for solution development;
- record decisions made and their rationales during meetings;
- following instructions from answers to frequently asked questions;
- share social information such as the internal dart throwing competitions
- identify experts in particular parts of the products or development processes;
- self-coordinate themselves to collaborate on project tasks; and,
- track progresses on project tasks.

These results are summarized in Table 6.9. It shows that nearly 50% of the use of MASE was on exchanging ideas on domain and technical problems. In fact, Figure 6.3 shows that 80% of all the read-accesses to information content in MASE were concentrated to just over 20% of the pages (170 out of 815 pages) and about 25% of the read-accesses were to the top 10 most visited pages.

Types of Content	% of Pages	% of Read Access	% of Write Access	% of Overall Access
Problem Understanding	46	40	8	49
Content Navigation Aid	16	39	3	42
Instrumental	43	30	6	36
Social	28	32	2	35
Projective	29	18	3	21
Expertise Location	18	19	1	20

Table 6.9: In an industrial setting, what types of knowledge can be found in a software development knowledge repository?

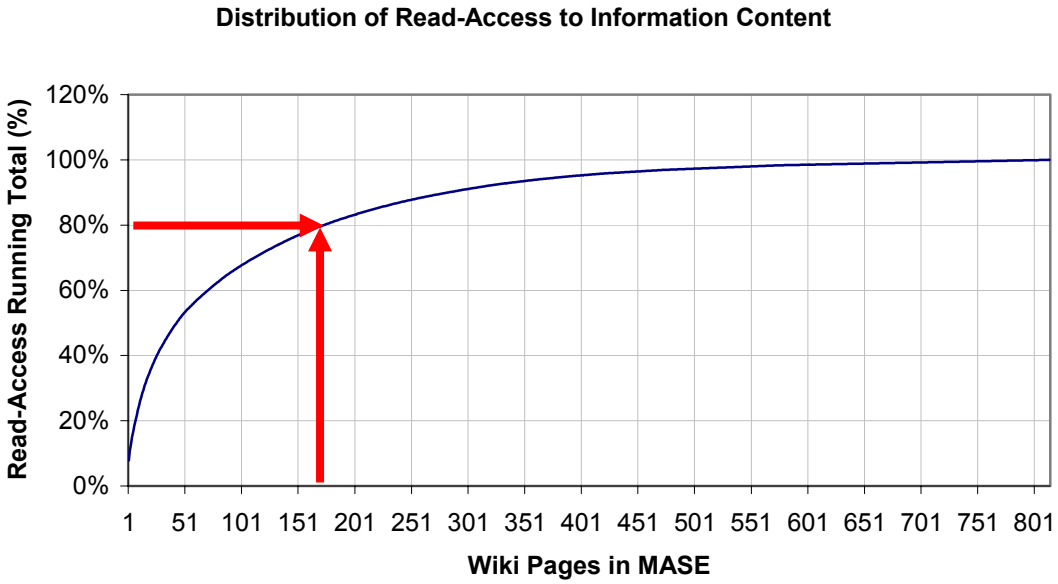


Figure 6.3: In an industrial setting, how are read-accesses distributed in a software development knowledge repository?

Of these 10 most visited wiki pages, 8 of them were pages not bundled with the initial MASE installation but were created by project teams at empolis spontaneously to share knowledge unique to their project or team. Of the other 2 pages, the "RecentChanges" wiki page lists all pages that have been updated over a certain period of time as specified by users; the "ListUsers" wiki page lists contact information for other MASE users, and provides the means to contact and interactively message those users, a feature detailed in §6.3.2.

To answer the question of to what extent users would formalize or structured the information content in a software development knowledge repository, the analysis in Table 6.10 shows that over half of the content is annotated in purely unstructured form with the remaining half annotated in either semi-structured or totally structured format. In particular, 80% of the structured content was generated by 6 of the 34 plug-ins and these 6 plug-ins were used to:

Formalization of Content	% of Pages
Unstructured	58
Both	23
Structured	19

Table 6.10: Users' preference for formalization of repository content in an industrial setting

- aid navigation by finding the context of the currently viewed page in relation to other pages;
- embed or retrieve content from other online resources such as the Java API;
- rate wiki pages on the quality of the posted content;
- receive e-mail notifications of changes made to any subscribed wiki pages; and,
- list the number of times a particular wiki page has been read.

With respect to content contribution, employees at empolis created about 12 times (761/54) as much new wiki pages as the default wiki pages that were bundled in MASE when the case study began. As for collaborative authoring of information content in MASE, about 7% of the pages (60/815) were contributed by 2 or more authors.

To answer the question of whether there exists any self-organized maintenance of information content in the knowledge repository given the open-edit nature of MASE, I analyzed the distribution of write-access to MASE. Of the 80 registered users, 49 of them contributed content. In total, 75% of contributions were by the top 10 users, of which 8 were developers and 2 were technical writers, and 55% were by the top 5 users who were all developers. While managers account for 10% of the user base, they were noticeably absent from the aforementioned list of top contributors.

The analysis also reveals that nearly half (47%) of the edits were to a subset of 20 pages, and roughly over one-third (34%) of edits were to a subset of 10 pages. This is similar to the trend depicted in Figure 6.3. In addition, only about 15% of pages were created but never accessed again.

When asked of the challenges, the enabling factors, and the perceived benefits of using an informal knowledge sharing like MASE for knowledge sharing and inter-team learning, the users indicated that the largest motivating factors were (in decreasing importance):

- presence of required information;
- contributing information to MASE is very easy;
- desire to help others; and,

- encouragement from management.

In particular based on informal feedback from the empolis users, many empolis users found that with MASE, "*contributing is very easy, nothing before did the job that easy*".

With respect to users' perceived benefits of MASE, analysis of MASE usage on a weekly basis provides some insights to this question. As shown in Figure 6.4, employees at empolis make use of MASE consistently every week. The large increase in overall access in the 4th week coincided with the completion of the final customization of MASE and the bootstrapping of content into the repository. The sudden drop in overall access in the 11th week coincided with the German school holiday when many of the empolis employees were not at work. Our analysis also reveals that about 80% of all read-access were made by the top 20 users, of which 14 were developers and 6 were either managers or technical writers, and close to 60% of all read-accesses were by the top 10 users of which 7 where developers.

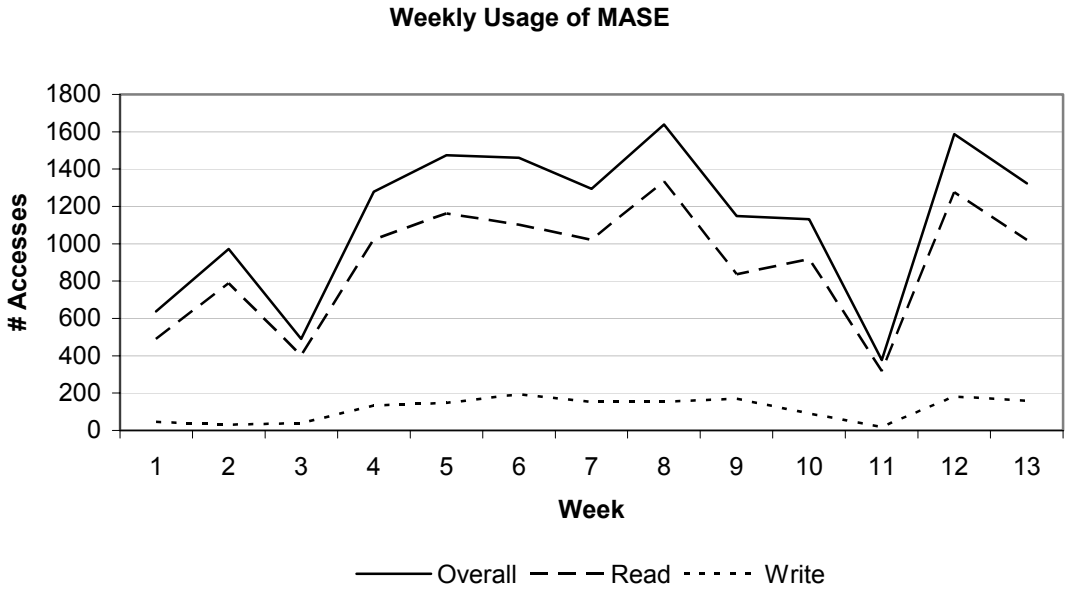


Figure 6.4: Weekly usage of MASE in an industrial setting

6.3.5 Interpretation

Based on the observations and analysis of MASE usage, the following interpretations are made with respect to the research objectives and questions stated in §6.1.

First, observations from the case study indicate that MASE is primarily used for asynchronous collaboration. Having said that, empolis staff's urgent demand for the private messaging system and the intense use of features such as the "ListUser" plug-in in the "ListUser" wiki page provide strong evidence that it is critical for knowledge sharing tools to support not just the codification strategy but also the personalization strategy by providing means for easy communication and collaboration. This reinforces the views in the Computer Supported Cooperative Work (CSCW) [Greenburg and Roseman, 2003] and learning theory [Wenger et al., 2002; Hildreth, 2004] literature that tools which integrate codification mechanisms together with collaboration and communication features are critical for enhancing knowledge sharing and creation.

Second, observations from the case study shows that much of the content in MASE reflects Taylor's description of information use accurately [Taylor, 1991]. Unlike the findings from the first case study, there is no one particular type of information content that is under-utilized in empolis' use of the MASE repository.

Third, given MASE's flexibility in support for both structured and unstructured information, observations from the case study indicate that roughly over 80% of the information content was annotated in either unstructured or in a hybrid form that consists of both structured and unstructured form. Similar to the finding from the first case study, the finding from this case study demonstrates that there is a greater need for unstructured than structured knowledge. It provides clear evidence that it is insufficient for knowledge sharing tools to support only structured information content; there needs to be built-in support for both structured and unstructured content.

Fourth, despite the fact that less than a handful of users had contributed the vast majority of content, it should be highlighted that none of the top 10 contributors in MASE are management staff. Most of them are developers with a few of them being technical writers. In a sense, they formed a self-organized experience factory unit while still working on their normal development tasks. This self-organized process of running the experience factory may have the benefit of avoiding the ivory tower syndrome that is sometimes attributed to centralized experience factory units and process groups ("process

police"). Members of such a typical centralized experience factory units are often not involved in the day-to-day software development tasks and the knowledge content that these members put in an experience repository is often perceived as out-of-touch with reality and therefore not used. This leads some users of existing experience bases to raise concerns about the usefulness of such experience bases—sometimes polemically calling them "write-only memories" or "experience cemeteries" [Melnik and Holz, 2004]. This was certainly not the case with the use of MASE at empolis given that only 15% of wiki pages are created and not access again; and that MASE remained in active use to this date after one of our researchers left empolis.

Fifth, with respect to the perceived advantages of using MASE for knowledge sharing, many empolis users reported an unparallel ease in contributing information using MASE. This may be attributed to the simplicity of the wiki markup language and we believe it is this simplicity that facilitates efficient collaboration between information contributors and readers. This is because information content on any wiki pages are nearly free-formatted text. In typical project web sites, information content is often embedded among HTML markup elements and this can make it time consuming for updating content. While one can argue that even the wiki markup language may be too complex for typical computer users and that a WYSIWYG (What-You-See-Is-What-You-Get) web page editor should be used instead, there are two counterarguments. First, software developers are no ordinary typical computer users and they are unlikely to have problems learning and using the wiki markup syntax, as supported by the empolis staff's feedback. Second, the use of WYSIWYG web page editors requires the users to use another tool. In contrast, updating information content in wiki pages is nearly as easy as reading them. In addition, end-users can simply use the web browser to update information. They do not have to learn another separate tool just for editing information on a web-based knowledge repository. At empolis, one user even stated that MASE are great to "*get people involved and make them having fun while writing*".

On the other hand, the ease of contribution in MASE has resulted in a proliferation of wiki pages. In fact, one user even stated that "*retrieving becomes harder*

every day, because of the growing amount of content and the lack of structure". To address this concern, a future challenge is to provide users a way to categorize wiki pages with as much ease as that for making updates to wiki pages.

6.4 Limitations

While the two case studies detailed in this chapter provide insights into how multiple software development teams use an informal knowledge repository, MASE in particular, there also exist the following threats to the validity of these two case studies.

First, the two case studies are conducted within a short time period during which the tool's long term effectiveness (or ineffectiveness) for facilitating inter-team learning cannot be accurately observed.

Second, in both case studies, the analysis of the content in the MASE repository is conducted by the same investigator, whose judgement may be highly subjective.

Third, the number of case studies conducted is low. It is insufficient for revealing accurately the tool's effectiveness. Furthermore, one of the case studies involves student developers in an academic setting which may not accurately reflect how a knowledge sharing tool is used in an industrial setting. While the other case study is conducted in an industrial setting, the derived findings cannot be generalized as those are findings from just one case study alone.

Forth, the positive feedbacks from the participants in the industrial case study are collected informally and hence may not be representative of other participants who have not provided feedback.

6.5 Summary

In this chapter, I have described the methodologies and results for two case studies of MASE. The initial results provide evidences for the need of knowledge sharing tools to incorporate not only repository technologies but also those that facilitate communication and collaboration among people. The results also provide evidences that there is a greater

need for unstructured than structured knowledge content, and hence the need of knowledge sharing tools to support not only structured but more importantly unstructured information content. The results also reveal that an informal knowledge sharing tool such as MASE is used for sharing content for problem understanding, instrumental, projective, social, expertise location, and content navigation purposes. In the industrial case study, we also observe self-organized maintenance of the repository content among the ordinary repository users as a result of the open-edit nature of MASE. In both case studies, feedbacks from the end-users provide encouraging support for the continuing use of MASE for knowledge sharing purposes.

Chapter 7. Conclusion & Future Work

In this chapter, I conclude my thesis by summarizing the research contributions that I have made. First, I revisit the research problems that motivate this work. I then describe my research contributions by outlining how I have addressed each of the research problems. I close by suggesting areas of potential future work.

7.1 Thesis Problems

In Chapter 1, I outlined four research problems with respect to the issue of inter-team learning in agile software development methods. These problems represent knowledge that previous research has not yet addressed, including:

1. **The appropriate inter-team learning approach for software organizations that practice agile methods**. In software engineering, prior research on knowledge management include those that discuss novel inter-team learning strategies encompassing both organizational processes and tool support; those that propose models and frameworks for comparing these strategies; and those that report on experiences and effectiveness of applying these strategies. Most of these works are conducted before agile methods began to gain wide attention; as such there is relatively little research that investigates inter-team learning in the context of agile methods. Likewise, there is little research done that focuses on inter-team learning in the agile methods community.
2. **The requirements for knowledge sharing tools that aim to facilitate inter-team learning in an agile software development organizational setting**. Since it is not known what constitutes the appropriate inter-team learning approach in an agile software organization, we also do not know what are required of tools that are to facilitate knowledge sharing and inter-team learning in a software organization that practices agile methods.

3. **If existing tool support for inter-team learning is appropriate for agile methods practitioners**. Understanding the tool criteria for supporting an inter-team learning approach that fits an agile software development setting provides us with a basis for evaluating existing tool support and the need for novel tool support.
4. **Shall there be a need for a novel knowledge sharing tool support, if such a tool will be useful**. In case a novel tool is needed and after such a tool is designed and implemented, it is desirable to evaluate the extent to which the tool is successful in supporting inter-team learning in software organizations.

7.2 Thesis Contributions

This thesis addresses the aforementioned problems with the following research contributions:

1. proposal of an inter-team learning approach that integrate the Communities of Practice concept and an augmented Experience Factory framework for use in an multi-team software organization that practice agile methods (address problem 1);
2. identification of nineteen criteria essential for tools that support the proposed integrated inter-team learning approach (address problem 2);
3. a critical evaluation of existing knowledge sharing tool support using the identified tool criteria (address problem 3);
4. the design and implementation of MASE, a tool that supports the integrated approach needed for facilitating inter-team learning in an agile software development environment (address problem 3); and,
5. an empirical evaluation on how an informal knowledge repository such as MASE is used by software development teams and its effectiveness for facilitating inter-team leaning (address problem 4).

These contributions are elaborated below.

I addressed the 1st thesis problem by proposing an inter-team learning approach for use in software organizations that practice agile methods. To this end, I

analyzed existing inter-team learning approaches, specifically the Experience Factory framework, the codification and personalization strategies, and the Communities of Practice (CoP) concept, in the context of agile methods.

My analysis found that the Experience Factory framework has the benefits of facilitating inter-team learning without placing much overhead on existing project teams; providing opportunities of process improvement at both the project and organizational level; and making the standardization of potentially reusable experience easier. Yet, it can clash with the organizational culture in an agile software development environment due to its centralized nature and primarily non-practitioner membership.

With the codification approach, it has the benefits of mitigating the risks of knowledge loss; enabling collaboration across time and space; and facilitating efficient knowledge search and retrieval. However, not all types of knowledge can be explicated efficiently and that explicated knowledge is often incomplete and easily becomes outdated. Its emphasis on explicit knowledge makes it incompatible with the agile organizational culture.

With the personalization approach and the CoP framework, they have the benefits of being a more effective organizational structure for managing knowledge; promoting synergies across official business units; eliciting and sharing tacit knowledge better; providing implicit means for reinforcing relationship networks among members; increasing members' intrinsic motivation for knowledge sharing; and increasing the likelihood of retaining talent. Yet, a CoP can be difficult to support due to its informal nature. It can become insular from other CoPs, or from new or peripheral CoP members. Overemphasis on person-to-person interaction may also overload experts in a CoP with recurring information needs. The goals of CoP can also be incompatible with organization knowledge needs.

As seen, there are advantages to each of the studied approaches but the use of any one of them alone are either in direct conflict with the core values of the agile software development culture or is inadequate for inter-team learning purposes. In light of this, I

studied the feasibility of an integrated approach and found that an integrated approach consisting of both the codification and personalization strategies (not CoP) to inter-team learning is not only feasible but also beneficial and necessary, as reported in industrial case studies in both software engineering and organizational science literature. In light of this, I propose an inter-team learning approach that integrate the CoP approach and an augmented Experience Factory framework be used for facilitating inter-team learning in an agile software organization. The novelty of this proposal is that it is an application of existing approaches in a novel situation; the novel situation being a multi-team software organization that practices agile methods.

I addressed the 2nd thesis problem by identifying the requirements for knowledge sharing tools that support inter-team learning in an agile software development organizational setting. The proposed inter-team learning approach of integrating the Experience Factory and the CoP concepts have significant implications on the design of knowledge sharing tools. These implications are manifested in a total of nineteen tool criteria. Based on these requirements, **I evaluated existing knowledge sharing tools which addressed the 3rd thesis problem**. The evaluation revealed a gap among existing tool support in meeting these requirements. Existing tools such as document management systems or experience repositories provides support for only the repository aspect of the requirements and neglect those that address the needs of a CoP. Other tools like email and instant messaging provide rich support for communication and collaboration but they lack in support for the repository aspect of the nineteen requirements.

As a result, **I followed up on the 3rd thesis problem by designing and implementing MASE, a tool that supports the integrated approach needed for facilitating inter-team learning in an agile software development environment**. It is an extension of the Wiki web and portal concepts with additional flexible support for various styles of collaborative work—co-located and distributed; asynchronous and real-time; unstructured and structured information content—thereby addressing typical needs demanded by a CoP. MASE also supports the structuring of potentially reusable

information content into process models and delivers them automatically to an agile project team member's workspace.

I addressed the 4th thesis problem by providing empirical evidences on how an informal knowledge sharing tool such as MASE is used by software teams via two case studies performed in academic and industrial settings respectively. The initial results provide evidences for the need of knowledge sharing tools (1) to incorporate not only repository technologies but also those that facilitate communication and collaboration among people, and (2) to support not only structured but also unstructured information content. It also reveals that an informal knowledge sharing tool such as MASE is used for sharing content for problem understanding, instrumental, projective, social, expertise location, and content navigation purposes. In the industrial case study, we also observe self-organized maintenance of the repository content among the ordinary repository users as a result of the open-edit nature of MASE. In both case studies, feedbacks from the end-users provide encouraging support for the continuing use of MASE for knowledge sharing purposes.

7.3 Future Work

The work presented in this thesis represents only a small step in addressing the issue of inter-team learning in an agile software development environment; a research area where there still exists lots of room for future work.

In terms of tool support, the tool described in this work is only a proof-of-concept. It will be beneficial to enhance the performance and the usability of the tool. To facilitate better collaboration among users, richer support for real-time collaboration is also desirable.

As illustrated in the industrial case study, when many minds collaborate together in a Wiki repository, content albeit useful will be put in the wrong place. This problem will deteriorate as users contribute more and more content into the Wiki repository over time. One area of future work is tool support for analysing Wiki page content and their interrelationship and providing knowledge refactoring support.

In terms of empirical evaluation, there are limitations to the two case studies presented in this thesis. They are limited in their observation timeframe and a longitudinal study will be beneficial for more accurately determining the long term effectiveness of the proposed approach and of MASE in facilitating inter-team learning.

While a substantial amount of data has been collected in both case studies, data such as the number of read- and write-accesses are not direct indicators of the amount of learning individual users may derive as a result of using MASE. In light of this, it is desirable to perform a direct observational study of the use of the proposed approach and of MASE in order to access their learning impacts on software teams.

Given information content that are similar in nature, a study that compares and contrasts the use of MASE against another knowledge sharing tool can also provide insights into the values of an informal knowledge repository like MASE.

Lastly, this work proposes an inter-team learning approach that is tailored for multi-team software organizations that practice agile methods or those software companies with organizational cultures similar to agile organizations. Yet, the software organization studied in this empirical work does not fully satisfy the aforementioned criteria. To validate the proposed approach, it is desirable to conduct an empirical study on software companies that fit the aforementioned criteria.

Appendix A. References

Abran, A., Moore, J., Bourque, P., Dupuis, R. (2004), Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE Press, pp. 1

Agile, (2001), Agile Manifesto, http://www.agilemanifesto.org, last accessed, November 7, 2004

Alavi, M., Leidner, D. (1999), **Knowledge Management Systems: Issues, Challenges, and Benefits**, *Communications for the Association for Information Systems 1(7)*

Aleksiuk, D. and Wiebe, R. (2003, March), **Description of Agile Initiation Service, and Experiences Introducing Agile at TransCanada**, *Presentation at the Calgary Agile Methods User Group*, University of Calgary, Calgary, Canada

Astound, **Astound Conference Center**, http://www.milori.com/ttools/webmeet/astound.asp, last accessed, November 15, 2004

Basili, V., Caldiera, G., McGarry, F., Pajerski, R., Page, G., Waligora, S. (1992), **The Software Engineering Laboratory – An Operational Software Experience Factory**, *Proceedings of the 14th International Conference on Software Engineering (ICSE)*, IEEE Press, pp. 370-381

Basili, V., Caldiera, G., and Rombach, H. (1994), **Experience Factory**, J. Marciniak (Ed.), *Encyclopedia of Software Engineering vol. 1*, pp. 469-476. John Wiley Sons.

Basili, V., Lindvall, M., and Costa, P. (2001), **Implementing the Experience Factory Concepts as a Set of Experience Bases**, *Proceedings of the 13th International Conference on Software Engineering & Knowledge Engineering (SEKE)*, pp. 102-109

Beck K. (1999), **Extreme Programming Explained: Embrace Change**, *Addison Wesley*.

Birk, A. and Kröchel, F. (1999), **A Knowledge Management Lifecycle for Experience Packages on Software Engineering Technologies**, *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE '99),* Kaiserslautern, Germany, Springer

Boehm, B. and Turner, R. (2003), **Rebalancing Your Organization's Agility and Discipline**, *Proceedings of XP/Agile Universe 2003*, New Orleans, US, Springer, pp. 1-8

Brössler, P. (1999), **Knowledge Management at a Software Engineering Company – An Experience Report**, *Proceedings of the International Workshop on Learning Software Organizations (LSO)*, Kaiserslautern, Germany, Springer

Brown, J.S. and Duguid, P. (1998). **Organizing Knowledge**, *California Management Review 40(3)*, pp. 90-111

Capelli, P. (2000), **A Market-Driven Approach to Retaining Talent**, *Harvard Business Review 78(1)*, HBS Press, pp. 103-111

Cockburn, A. (2001), **Agile Software Development**, *Addison Wesley*, Reading, MA.

Cockburn, A. and Highsmith, J. (2001), **Agile Software Development, The People Factor**, *IEEE Computer 34(11)*, IEEE Press, pp. 131-133

Cohen, B. and Prusak, L. (2001), **Good Company: How Social Capital Makes Organizations Work**, *HBS Press*, pp. 19

Communispace, http://www.communispace.com, last accessed, November 15, 2004

Constant, D., Sproull, L., and Kiesler, S. (1996), **The Kindness of Strangers: The Usefulness of Electronic Weak Ties for Technical Advice**, *Organizational Science 7(2)*, pp. 119-135

Davenport, T. and Prusak, L. (1998), **Working Knowledge**, *HBS Press*, pp. 123-144.

Demarco, T. and Lister, T. (1999), **Peopleware**, *Dorset House Publishing Company*

Dessler, G. (1999), **How to Earn Your Employees' Commitment**, *Academy of Management Executive 13(2)*, pp. 58-67

Dingsøyr, T. (2000), **An Evaluation of Research on Experience Factory**, *Proceedings of the 2nd International Workshop on Learning Software Organizations (LSO)*, Oulu, Finland

Dingsøyr, T and Røyrvik, E. (2003), **An Empirical Study of an Informal Knowledge Repository in a Medium-sized Software Consulting Company**, *Proceedings of the 23rd International Conference on Software Engineering 2003 (ICSE)*, IEEE Press

DocuShare, **Xerox DocuShare**, http://docushare.xerox.com, last accessed, November 15, 2004

Doran, H. (2004), **XP: Help or Hindrance to Knowledge Management?**, *Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in*

Software Engineering (XP 2004), Garmisch-Partenkirchen, Germany, Springer, pp. 215-218

Feldmann, R. and Rus, I. (2003), **Panel: When Knowledge and Experience Repositories Grow, New Challenge Arise**, *Proceedings of the 5th International Workshop on Learning Software Organizations (LSO)*, Luzern, Germany, pp. 111-112

Fitzpatrick, G. (2001), **Emergent Expertise Sharing in a New Community**, M.S. Ackerman, V. Pipek & V. Wulf (Ed.): *Sharing Expertise: Beyond Knowledge Management*, MIT Press, Cambridge, MA

Fredrick, C. (2003), **Extreme Programming: Growing a Team Horizontally**, *Proceedings of XP/Agile Universe 2003*, New Orleans, US, Springer, pp. 9-17

Greenburg, S. and Roseman, M. (2003), **Using a Room Metaphor to Ease Transitions in Groupware**, M.S. Ackerman, V. Pipek & V. Wulf (Ed.), *Sharing Expertise: Beyond Knowledge Management*, MIT Press, Cambridge, MA

Greening, J. (2001), **Launching Extreme Programming at a Process-Intensive Company**, *IEEE Software 18(6)*, IEEE Press, pp. 27-33

Haas, R., Aulbur, W., and Thakar, S. (2000), **Enabling Communities of Practice at EADS Airbus**, M.S. Ackerman, V. Pipek & V. Wulf (Ed.), *Sharing Expertise: Beyond Knowledge Management*, MIT Press, Cambridge, MA

Hansen, M.T. (1999), **The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organizational Units**, *Administrative Science Quarterly 44(1)*, pp. 82-111

Hansen, M.T., Nohira, N., and Tierney, T. (1999), **What's Your Strategy for Managing Knowledge?**, *Harvard Business Review 77(2)*, HBS Press, pp. 106-116

Henninger, S. (2003), **Tool Support for Experience-Based Methodologies**, S. Henninger & F. Maurer (Ed.), *Advances in Learning Software Organizations – Proceedings of the 4th International Workshop on Learning Software Organizations*, Springer, pp. 44-59

Henninger, S. and Maurer, F. (2002), **Advances in Learning Software Organizations**, *Springer*

Hildreth, P. (2004), **Going Virtual: Distributed Communities of Practice**, *IDEA Group Pusblishing*.

Hinds, P. and Pfeffer, J. (2003), **Why Organizations Don't Know What They Know**, M.S. Ackerman, V. Pipek & V. Wulf (Ed.), *Sharing Expertise: Beyond Knowledge Management*, MIT Press, Cambridge, MA

Houdek, F., Schneider, K., and Wieser, E. (1998), **Establishing Experience Factories at Daimler-Benz An Experience Report**, *Proceedings of the 20th International Conference on Software Engineering (ICSE)*, Kyoto, Japan, IEEE Press, pp. 443-447

Huysman, M. and de Wit, D. (2003), **A Critical Evaluation of Knowledge Management Practices**, M.S. Ackerman, V. Pipek & V. Wulf (Ed.), *Sharing Expertise: Beyond Knowledge Management*, MIT Press, Cambridge, MA

J2EE, **Java 2 Enterprise Edition**, http://java.sun.com/j2ee, last accessed November 30, 2004

Java, http://java.sun.com, last accessed, November 30, 2004

Jeffries, R. (2001), **Essential XP: Documentation**, http://www.xprogramming.com/xpmag/expDocumentationInXP.htm, last accessed, November 7, 2004

Johansson, C., Hall, P. and Coquard, M. (1999), **Talk to Paula and Peter – They are Experienced**, *Proceedings of the International Workshop on Learning Software Organizations (LSO)*, Kaiserslautern, Germany, Springer

JSPWiki, http://www.jspwiki.org, last accessed, November 30, 2004

Kähkönen, T. (2004), **Agile Methods for Large Organizations – Building Communities of Practice**, *Proceedings of Agile Development Conference 2004*, Salt Lake City, US

Kavanaugh, A., Reese, D., Carroll, J., and Rosson, M. (2003), **Weak Ties in Networked Communities**, M. Huysman, E. Wenger, & V. Wulf (Ed.), *Communities and Technologies*, Kluwer Academic Publishers, Netherlands, pp. 265-286

Kerievsky, J. (2001), **Continuous Learning**, *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering (XP 2001)*, Cagliari, Italy

Kircher, M. (2001), **eXtreme Programming in Open-Source and Distributed Environments**, *Presentation at JAOO Conference 2001*, Århus, Denmark

Lave, J. and Wenger, E. (1991), **Situated Learning. Legitimate Peripheral Participation**, *Cambridge University Press*

Lethbridge, T., Singer, J., and Forward, A. (2003), **How Software Engineers Use Documentations: The State of Practice**, *IEEE Software 20(6)*, IEEE Press, pp. 35-39

Leuf, B. and Cunningham, W. (2001), **The Wiki Way – Quick Collaboration on the Web**, *Addison Wesley*

Liebowitz, J. (1999), **Knowledge Management Handbook**, *CRC Press*, Boca Raton, FL

Liebowitz, J. (2002), **A Look at NASA Goddard Space Flight Center's Knowledge Management Initiatives**, *IEEE Software 19(3)*, IEEE Press, pp. 40-42

Linux, http://www.linux.org, last accessed, November 3, 2004

Lindvall, M., Rus, I., and Sinha S. (2002), **Technology Support for Knowledge Management**, *Proceedings of the 4th International Workshop on Learning Software Organizations (LSO)*, Chicago, US, Springer

LiveLink, **OpenText LiveLink**, http://www.opentext.com, last accessed, November 16, 2004

Lyman, J. (2004), **Competing Voice Standards vie for Developers' Attention**, *IT Manager's Journal April 2004*

Melnik G. and Holz, H. (2004), **Advances in Learning Software Organizations, 6th International Workshop, LSO 2004**, *Lecture Notes in Computer Science, vol. 3096*, Springer

Melnik G. and Maurer, F. (2004), **Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing**, *Proceedings of the Agile Development Conference 2004*, Salt Lake City, US, IEEE Press

McDermott, R. (1999), **Nurturing Three Dimensional Communities of Practice: How to get the most out of human networks**, *Knowledge Management Review 2(5)*, pp. 22-25

Neus, A. (2001), **Managing Information Quality in Virtual Communities of Practice**, E. Pierce & R. Katz-Haas (Ed.), *Proceedings of the 6th International Conference on Information Quality at MIT*, Sloan School of Management, Boston, US

Orr, J. (1996), **Talking about Machines – An Ethnography of a Modern Job**, *IRL Cornell Press*.

Palmer, S. and Felsing, J. (2002), **A Practice Guide to Feature-Driven Development**, *Prentice Hall*

Poole, C. and Huisman, J. (2001), **Using Extreme Programming in a Maintenance Environment**, *IEEE Software (18)6*, IEEE Press, pp. 42-50

QuickPlace, **Lotus Team Workspace**, http://www.lotus.com/quickplace, last accessed, November 16, 2004

Robinson, H., and Sharp, H. (2004), **The Characteristics of XP Teams**, *Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004)*, Garmisch-Partenkirchen, Germany, Springer, pp. 139-147

Royce, W. (1970), **Managing the Development of Large Software Systems: Concepts and Techniques**, *Proceedings of WESCON*

Ruhe, G. (2002), **Software Engineering Decision Support – A New Paradigm for Learning Software Organization**, *Proceedings for the 4th International Workshop on Learning Software Organization (LSO),* Chicago, US, Springer

Rumpe., B. and Schröder, A. (2002), **Quantitative Survey on Extreme Programming Projects**, *Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2002)*, Alghero, Italy

Ruppel, C. and Harrington, J. (2001), **Sharing Knowledge Through Intranets: A Study of Organizational Culture and Intranet Implementation**, *IEEE Transactions on Professional Communication 44(1)*, IEEE Press, pp. 37-52

Rus, I., Lindvall, M., and Sinha, S. (2001), **Knowledge Management in Software Engineering**, *Fraunhofer Center for Experimental Software Engineering Maryland*, The University of Maryland

Rus, I. and Lindvall, M. (2002), **Knowledge Management in Software Engineering**, *IEEE Software 19(3)*, IEEE Press, pp. 26-38

Schneider, K. and von Hunnius, J-P. (2003), **Effective Experience Repositories for Software Engineering**, *Proceedings of the 25th International Conference on Software Engineering (ICSE)*, IEEE Press

Schneider, K., von Hunnius, J-P., and Basili, V. (2002), **Experience in Implementing a Learning Software Organization**, *IEEE Software 19(3)*, IEEE Press, pp. 46-49

Simons, M. (2004), **Distributed Agile Development and the Death of Distance**, *Sourcing and Vendor Relationships Advisory Service Executive Report 5(4)*, Cutter Consortium

Stapleton, J. (1997), **DSDM Dynamic Systems Development Method: The Method in Practice**, *Addison Wesley*

Schwaber, K., Beedle, M., and Martin, R. (2001), **Agile Software Development with SCRUM**, *Prentice Hall*, Englewood Cliffs, NJ.

Taylor, R. (1991), **Information Use Environments**, V. Dervin & M. Voight (Ed.), *Progress in Communication Sciences 10*, Ablex, Norwood, NJ

Tiwana, A. (2000), **The Knowledge Management Toolkit: Practical Techniques for Building a Knowledge Management System**, *Prentice Hall*.

Trittmann, R. (2001), **The Organic and Mechanistic Form of Managing Knowledge in Software Development**, *Proceedings of the 3rd International Workshop on Learning Software Organization (LSO 2001)*, Kaiserslautern, Germany, Springer, pp. 22-37

VersionOne, http://www.versionone.net, last accessed, November 15, 2004

Wenger, E. (1998), **Communities of Practice: Learning, Meaning, and Identity**, *Cambridge University Press*

Wenger, E. (2001), **Supporting Communities of Practice: A Survey of Community-Oriented Technologies**, http://www.ewenger.com/tech, last accessed, November 15, 2004

Wenger, E., McDermott, R., and Snyder, W. (2002), **Cultivating Communities of Practice**, *HBS Press*

Wenger, E. and Snyder, W. (2000), **Communities of Practice: The Organizational Frontier**, *Harvard Business Review 78(1)*, HBS Press, pp. 139-145.

Williams, L., Kessler, R., Cunningham, W., and Jeffries, R. (2000), **Strengthening the Case for Pair Programming**, *IEEE Software 17(4)*, IEEE Press, pp. 19-25

Williams, L. and Kessler, R. (2002), **Pair Programming Illuminated**, *Addison Wesley*

Wieser, E., Houdek, F., and Schneider, K. (1999), **Systematic Experience Transfer Three Case Studies from a Cognitive Point of View**, *Proceedings of the International Conference on Product Focused Software Process Improvement (PROFES)*, Oulu, Finland, VTT, pp. 323-344

Appendix B. Case Study Materials

B.1 Consent Form

Research Project Title:
Examining Knowledge Sharing with Tool Support for Agile Software Development Teams

Investigator(s):
Thomas Chau, Kris Read, Frank Maurer (chauth,readk,maurer)@cpsc.ucalgary.ca

This consent form, a copy of which has been given to you, is only part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, you should feel free to ask. Please take the time to read this carefully and to understand any accompanying information.

Description of Research:
The purpose of this research is to examine how multiple teams, using agile development practices, share their expertise with each other. This research investigates whether developers exhibit particular patterns in sharing expertise with each other and in using an experience base that is shared among all teams (i.e. use of personal dialogue/on-line chat vs. written documents, amount of retrieval vs. amount of update).

Procedure:
By checking on the "**I agree to have my usage of the MASE to be logged**" checkbox, the following information will be logged: (0) the date and time you access MASE; (1) the IP address of the computer you use to access MASE; (2) the web pages in MASE that you have browsed; (3) the web pages in MASE that you have updated; (4) number of attempts to contact other MASE users via the on-line chat plug-in in MASE; and (5) hyperlinks that you have clicked on inside MASE.

If you check on the "**I want to have my usage of the MASE to be logged anonymously**" check box, you can still use MASE but none of your actions while you are using MASE will be logged.

Likelihood of Discomforts:
There is no harm, discomfort, or risk associated with your participation in this research.

Confidentiality:
Participant anonymity will be strictly maintained. Reports and presentations will refer to participants using only an assigned number. No information that discloses your identity

will be released or published without your specific consent to disclosure. All the data collected will be stored in a password-protected computer and only be accessible to the investigator.

Primary Researcher(s):
Thomas Chau and Kris Read are M.Sc. students in the Department of Computer Science at the University of Calgary under the supervision of Dr. Frank Maurer. This study is conducted as part of their research and will be included in their theses.

By checking on the "**I agree to have my usage of the MASE to be logged**" checkbox, you indicate that you have understood to your satisfaction the information regarding participation in the research project and agree to participate as a subject. In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. You are free to withdraw from the study at any time. Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. If you have questions concerning matters related to this research, please contact any of the three investigators of this research.

If you have any questions or issues concerning this project that are not related to the specifics of the research, you may also contact the Research Services Office at 220-3782 and ask for Mrs. Patricia Evans.
[ ] I agree to have my usage of MASE to be logged
[ ] I want to have my usage of MASE to be logged anonymously

Participant's Signature Date

Investigator and/or Delegate's Signature Date

Witness' Signature Date

A copy of this consent form has been given to you to keep for your records and reference.

B.2 Post-Study Questionnaire

What were you knowledge/experience with Web technologies prior to taking this course?

a) none b) 1 – 12 months c) more than 12 months

What are the means that you used to collaborate with members OUTSIDE of your team? Collaboration includes sharing what you know and seeking information or help from others.

(Please circle all that apply AND indicate approximately how often you use them, %)
a) MASE c) NetMeeting plug-in in MASE d) Chat tools
e) Others (i.e. phone calls, face-to-face conversations, etc.). Please elaborate:

To what extent have you posted your questions, solutions, and recommendations on MASE?

a) Not At All b) Rarely c) Occasionally d) Often e) Very Often

What motivated you to do so?

What discouraged you from doing so?

To what extent have you gained knowledge from MASE postings made by others that is useful for your tasks?

a) Not At All b) Rarely c) Occasionally d) Often e) Very Often

What motivates you to seek information or help from members OUTSIDE of your team?

What discourages you from seeking information or help from members OUTSIDE of your team?

To what extent do you think a Wiki-based experience base, like MASE, is helpful for organizing information and sharing one's expertise/learning with others?

a) Not At All b) Very Little c) Average d) Helpful e) Very Helpful

Please elaborate:

What kinds of tools would you use in the future for sharing ideas and organizing experiences? [Choose all that apply] Please explain.

- Wiki-driven experience base (such as MASE)
- Mailing list/listserv
- Static Web sites
- Other, please specify:

B.3 Ethics Approval

UNIVERSITY OF CALGARY

MEMO

CONJOINT FACULTIES RESEARCH ETHICS BOARD
c/o Research Services
Room 602 Earth Science
Telephone: (403) 220-3782
Fax: (403) 289 0693
Email: plevans@ucalgary.ca
Thursday, January 22, 2004

To: **Kristopher D. Read**
Computer Science

From: Dr. Janice P. Dickin, Chair
Conjoint Faculties Research Ethics Board (CFREB)

Re: **Certification of Institutional Ethics Review:** Examining Tool Support for Agile Software Development Teams

The above named research protocol has been granted ethical approval by the Conjoint Faculties Research Ethics Board for the University of Calgary.

Enclosed are the original, and one copy, of a signed **Certification of Institutional Ethics Review**. Please make note of the conditions stated on the Certification. A copy has been sent to your supervisor as well as to the Chair of your Department/Faculty Research Ethics Committee. In the event the research is funded, you should notify the sponsor of the research and provide them with a copy for their records. The Conjoint Faculties Research Ethics Board will retain a copy of the clearance on your file.

Please note, an annual/progress/final report must be filed with the CFREB twelve months from the date on your ethics clearance. A form for this purpose has been created, and may be found on the "Ethics" website, http://www.ucalgary.ca/UofC/research/html/ethics/reports.html

In closing let me take this opportunity to wish you the best of luck in your research endeavor.

Sincerely,

Patricia Evans
Executive Secretary for:
Janice Dickin, Ph.D., LLB., Faculty of Communication and Culture and
Chair, Conjoint Faculties Research Ethics Board

Enclosures(2)
cc: Chair, Department/Faculty Research Ethics Committee
Supervisor: Dr. F. Maurer

UNIVERSITY OF CALGARY

CERTIFICATION OF INSTITUTIONAL ETHICS REVIEW

This is to certify that the Conjoint Faculties Research Ethics Board at the University of Calgary has examined the following research proposal and found the proposed research involving human subjects to be in accordance with University of Calgary Guidelines and the Tri-Council Policy Statement on *"Ethical Conduct in Research Using Human Subjects"*. This form and accompanying letter constitute the Certification of Institutional Ethics Review.

File no:	**CE101-3820**
Applicant(s):	**Kristopher D. Read** Thomas Chau
Department:	**Computer Science**
Project Title:	**Examining Tool Support for Agile Software Development Teams**
Sponsor (if applicable):	

Restrictions:

This Certification is subject to the following conditions:

1. Approval is granted only for the project and purposes described in the application.
2. Any modifications to the authorized protocol must be submitted to the Chair, Conjoint Faculties Research Ethics Board for approval.
3. A progress report must be submitted 12 months from the date of this Certification, and should provide the expected completion date for the project.
4. Written notification must be sent to the Board when the project is complete or terminated.

Janice Dickin — 21 January 2004

Janice Dickin, Ph.D, LEB, **Date:**
Chair
Conjoint Faculties Research Ethics Board

Distribution: (1) Applicant, (2) Supervisor (if applicable), (3) Chair, Department/Faculty Research Ethics Committee, (4) Sponsor, (5) Conjoint Faculties Research Ethics Board (6) Research Services.

B.4 Raw Data

As mentioned in §6.2.3 and §6.3.3, data for both the academic and industrial case studies are collected primarily via logs of MASE's usage. Due to the large volume of the logs, all the raw data for the two case studies are packaged in the CD-ROM accompanying this thesis. Also included in this CD-ROM are the responses to the post-study questionnaire from the participants in the academic case study.

Appendix C. More Information on MASE

C.1 High Level Data Structures

The following figure presents an overview of the high-level data structures (objects) in MASE.

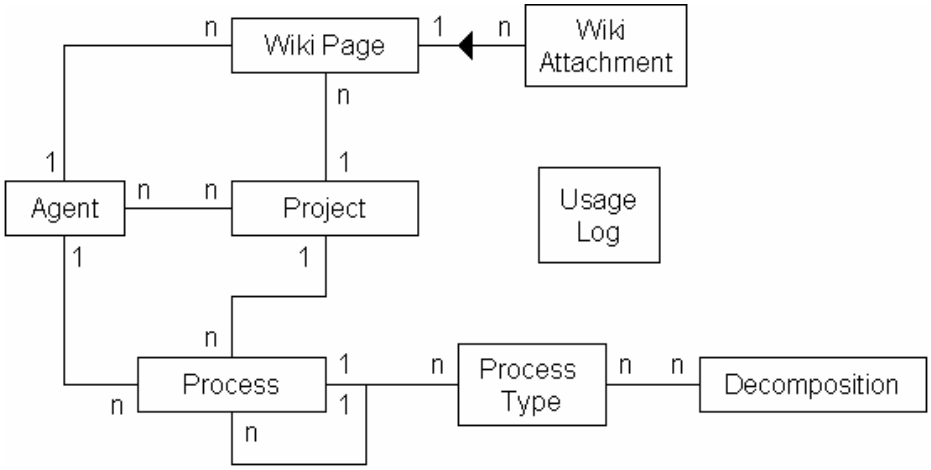


Figure C.1: High-level data structures in MASE

As its name suggest, the Wiki Page object stores all information related to a wiki page. The Wiki Page object is also a complex object in that it can be extended to a new type of object, in this case the Wiki Attachment object, with additional or refined properties. In other words, the same manipulations that can be done on a Wiki Page object can also be done on a Wiki Attachment object. For instance, an Agent object can author and subscribe to a Wiki Attachment object just as it can to a Wiki Page object; hence the 1-to-n relationship between the Agent and the Wiki Page objects. A Wiki Page object can contain many Wiki Attachment objects and it can also be associated with a Project (community). This is shown as the 1-to-n relationships between the Wiki Page and the Wiki Attachment objects and between the Project and the Wiki Page objects respectively.

Besides its relationship with the Agent object, a Project object can also be divided into many Processes objects; hence the 1-to-n relationship between the two types of object.

An Agent object is synonymous with a user in MASE. As such, an Agent object can be members of many Projects (community) and it can be responsible of many Processes. This is shown as the n-to-n relationship between the Agent and the Project objects and as the 1-to-n relationship between the Agent and the Process objects respectively.

In agile methods' terminology, a Process object can be a Release, or an Iteration, or an User Story, or a Task. A Process object can contain sub-processes, hence the reflexive 1-to-n relationship. A Process can also be associated with many Process Type objects in the process model. This is shown as the 1-to-n relationship between the two classes of objects.

A Process Type object represents a commonly performed task and it can be performed in many different ways, each of which is represented as the Decomposition object. A Decomposition object can be further broken down into many Process Types, hence the n-to-n relationship between the Process Type and the Decomposition objects.

The Usage Log object tracks all usage of the Wiki Page and Wiki Attachment objects.