

Integrated Process Support and Lightweight Knowledge Sharing for Agile Software Organizations

Thomas Chau, Frank Maurer
Department of Computer Science
University of Calgary
Calgary, Canada
{chauth,maurer}@cpsc.ucalgary.ca

Abstract

Distributed development is sometimes inevitable and must be dealt with when agile methods become more commonly used. The increased complexities in communication, collaboration, and coordination associated with distributed development coupled with the lack of support for organizational learning in agile practices present challenges to tool developers for agile teams. This paper describes a suite of integrated tools that (1) accommodates various collaboration styles; (2) provides process support for specific agile practices; and (3) facilitates organizational learning.

1. Introduction

In their original forms, agile software development practices, such as release and iteration planning, on-site customers, and pair programming, are to be used with co-located teams in order to maximize the high bandwidth afforded by face-to-face communication. Unfortunately, distributed software development is sometimes inevitable for reasons like: (1) insufficient office space to co-locate an entire team; (2) the team is too large to work in a single space; and (3) work practices such as telecommuting and offshore development efforts explicitly demand distributed collaborative work.

While there are benefits with distributed development, it comes at a cost of increased complexity in coordination, communication, and collaboration [1]. These increased complexities can be addressed via tool support to some extent. For such tools to be useful for agile teams, they should also provide support for specific agile practices and help overcome some potential drawback in agile methods.

One such potential drawback is the lack of explicit support for organizational learning [12]. Although the concept of learning is embedded in various agile software development practices, such as, on-site customers, pair

programming, pair rotation, daily Scrum meetings, and retrospectives, these practices only address knowledge sharing *within* a team. They do not address issues of knowledge sharing across team boundaries: in a large organization there may exist multiple teams that work on similar tasks, face common problems, or have overlapping interests in specific knowledge areas.

To address the above challenges, we present MASE and EB. Both are extensions of the Wiki web concept [3]. MASE is a process support environment providing support for specific agile practices. EB is an experience repository designed to facilitate knowledge sharing among agile teams in a software organization. Section 2 examines some existing tools used by distributed teams and agile practitioners, and identifies an important area of improvement for existing tools. Section 3 describes how this improvement is realized by the use of Wiki web as an application platform to build web-based collaborative tool like MASE and EB. Section 4 provides example scenarios of how MASE and EB can be used to support specific agile practices and to facilitate organizational learning. We conclude the paper with a summary of our contributions in section 5.

2. State of Practice

For communication and collaboration among distributed members, typical tools used include e-mail, newsgroups, and instant messengers. 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.

For project planning and coordination of dispersed members, the use of static project websites is common among distributed teams. In practice, the maintenance of such project websites is often authorized to a few individuals, if not one. This tight control over information authorship has the advantage of ensuring control over the quality of information made available to the entire team. However, the tight control can also limit potential opportunities for expertise sharing and collaborative knowledge creation among distributed team members. These opportunities are critical to agile teams’ success as developers are often confronted with short development and learning times as well as unpredictable and continuous changes in both technologies and customer needs.

The reliance on a few individuals to maintain a static project website can also create a bottleneck in keeping the website content current and relevant. Although this problem may be addressed by the use of content management systems like PhpNuke [2] which typically open up information authorship to the entire team, such systems can introduce other potential problems as well. They usually impose a rigid navigational and content structure (often based on a timeline and not on message contents) which does not facilitate free form knowledge sharing well. Furthermore, information retrieval may not be any easier than that in static project websites as users are usually not allowed to modify the site navigational structure. These drawbacks can also be found in project planning and coordination tools such as [8] and [9] that are tailored for agile teams. For a detailed comparison between our approaches and existing tools for agile teams, see [13].

While each of the above approaches has different benefits and drawbacks, a limitation common to all the above tools is that they restrict users to only one type of collaboration style. This forces software developers to use a myriad of non-integrated tools for their day-to-day collaborative work. As cited in CSCW [4] and software engineering tools construction literature [15], this is inconvenient and inflexible in practice given the fact that humans switch unconsciously “continually and effortlessly among different collaboration styles: across time, across place, and so on” [4]. This underlies the need for tool support to accommodate more than one particular style of collaborative work.

3. Wiki Web as an Application Platform

To support various styles of collaborative work, we used the Wiki web concept as the framework on which we design our proposed tools. Wiki technology enables any users to access, create, organize, and update any web pages in real-time using only a web browser. To edit these web pages (“wiki pages”), users apply the Wiki markup language which is much simpler than HTML: a list of all Wiki markup commands including examples fits easily onto a page.

The original Wiki implementation, however, suffers from similar drawback as other tools mentioned above: it only supports asynchronous collaboration and it only allows users to capture information in an unstructured format like text and graphics.

In order to address this inflexibility, we use JSPWiki [14] as the underlying framework for building our proposed tools. JSPWiki is an open-source Java-based implementation of the Wiki web concept and it is chosen for two primary reasons. First, its markup language is very similar to that of the original Wiki. This reduces the amount of learning time for users who are already familiar with the use of the original Wiki web. Second, its software architecture provides a plug-in API which allows tool developers to easily develop additional self-contained functionalities that address the focus on text and graphics in the original Wiki web implementation.

For instance, to support synchronous collaborative work, we have developed a plug-in that integrates with Microsoft NetMeeting and another that provides awareness information by displaying all users who are currently using the system. We have also extended JSPWiki such that every time when a user logs in, the tool tracks the IP address of that user’s computer. By viewing an online user’s personal profile, one can start a NetMeeting session with that user.

To support the use of structured information content, plug-ins can be developed (1) to present input forms that allow users to submit data following specific schemas or (2) to output tables that display information retrieved from databases in a structured fashion. Structured information contents are useful for reporting purposes, if needed by the team (while automatically generating this reports is difficult based on text-only information).

As a web-based collaborative tool, it is important to provide search facility besides providing carefully crafted information architecture to help users navigate easily to information they need. For this purpose, we have extended JSPWiki with full-text searching capabilities on unstructured and structured content without forcing users to learn two different querying languages.

To provide users complete control over the structure and the type of information that they deem relevant to themselves, we have extended JSPWiki such that when a

user first logs in, the tool provides that user with a default wiki page that serves as a personal portal for that user.

The benefits of using JSPWiki as the underlying framework for web-based collaborative work are manifold.

First, JSPWiki allows any plug-ins to be included on any wiki pages simply by referencing the names of the plug-ins in the wiki pages. This capability together with the fact that any content on any wiki pages can be modified by any one gives end-users the flexibility to design both the information content and the navigational structure of their web-based information repository.

Second, the wiki nature of JSPWiki helps minimize the amount of manual effort needed to cross-reference information resources stored in different wiki pages. This is because hyperlinks are automatically created from one wiki page to another when the name of a wiki page is mentioned in other wiki pages. This automatic hyperlink creation can reduce maintenance overhead often found in other forms of web-based information repository. However, it must be noted that this benefit is best realized if all users of the wiki pages follow the same taxonomy when contributing content.

Third, the simple JSPWiki markup language 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. This makes it time consuming to update content. Although one can argue this nuisance can be relaxed via the use of WYSIWYG web page editors, this 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 information repository.

Forth, the plug-in architecture of JSPWiki makes it easily extendible. This is demonstrated in the two different extensions, MASE and EB, that we have developed to meet our goals of supporting agile development practices and facilitating inter-team learning in agile software companies.

4. Sample Extensions

MASE and EB differs from each other in that they contains different set of plug-ins. MASE consists of a number of plug-ins that as a whole provides a process support environment tailored for agile development teams. EB contains a different set of plug-ins that as a whole provides an experience repository intended to facilitate knowledge sharing among multiple teams in a company. Despite the different sets of plug-ins they contain, both MASE and EB include a common of set of plug-ins provided by JSPWiki which supports the various styles of collaborative work mentioned above.

4.1. MASE: Support for Agile Practices

Figures 1 and 2 show a snapshot sequence from an example MASE usage scenario: from a wiki page containing the Whiteboard plug-in (Figure 1), project managers and customers can create iterations (Figure 2a) and user stories (Figure 2b) as structured data to be stored in the database.

Task	Priority	Responsible	Pair	Est	Act	
Product Backlog	1	Frank Maurer	Unassigned	114.5	0.0	<input type="checkbox"/>
Iteration 4.9.2003	0	Frank Maurer	Unassigned	18.5	38.5	<input type="checkbox"/>
capitalize story card names	10	Kris Read	Unassigned	0.0	0.0	<input type="checkbox"/>
column width set explicitly	5	Kris Read	Unassigned	0.0	0.0	<input type="checkbox"/>
eb web services plugin	8	Kris Read	Unassigned	0.0	0.0	<input type="checkbox"/>
Make Sure CC is Working	4	Harpreet Bajwa	Unassigned	6.0	0.0	<input type="checkbox"/>
misc CSS improvements	9	Kris Read	Unassigned	0.0	0.0	<input type="checkbox"/>
read link eb	7	Kris Read	Unassigned	0.0	0.0	<input type="checkbox"/>
Sourceforge CVS upload	1	Lawrence Liu	Unassigned	0.0	0.0	<input type="checkbox"/>
Iteration 05.01.2004	0	Frank Maurer	Unassigned	51.0	70.5	<input type="checkbox"/>
Use Locale Specific Dates instead of GMT dates	2	Thomas Chau	Unassigned	0.0	0.0	<input type="checkbox"/>

Save
Reset
Iteration
Story Card
Delete
Complete
Estimate
Clone
Prioritize

Unassigned
Assign Responsible
Assign Pair

Product Backlog
Move

Show:
 All Members
 Completed Tasks

Figure 1. The Whiteboard plug-in allows the customers and the development team to track their progress in the project by showing the estimated and actual effort expended at various granularities (iteration, user story). It is actually a giant input form allowing developers to update their progress.

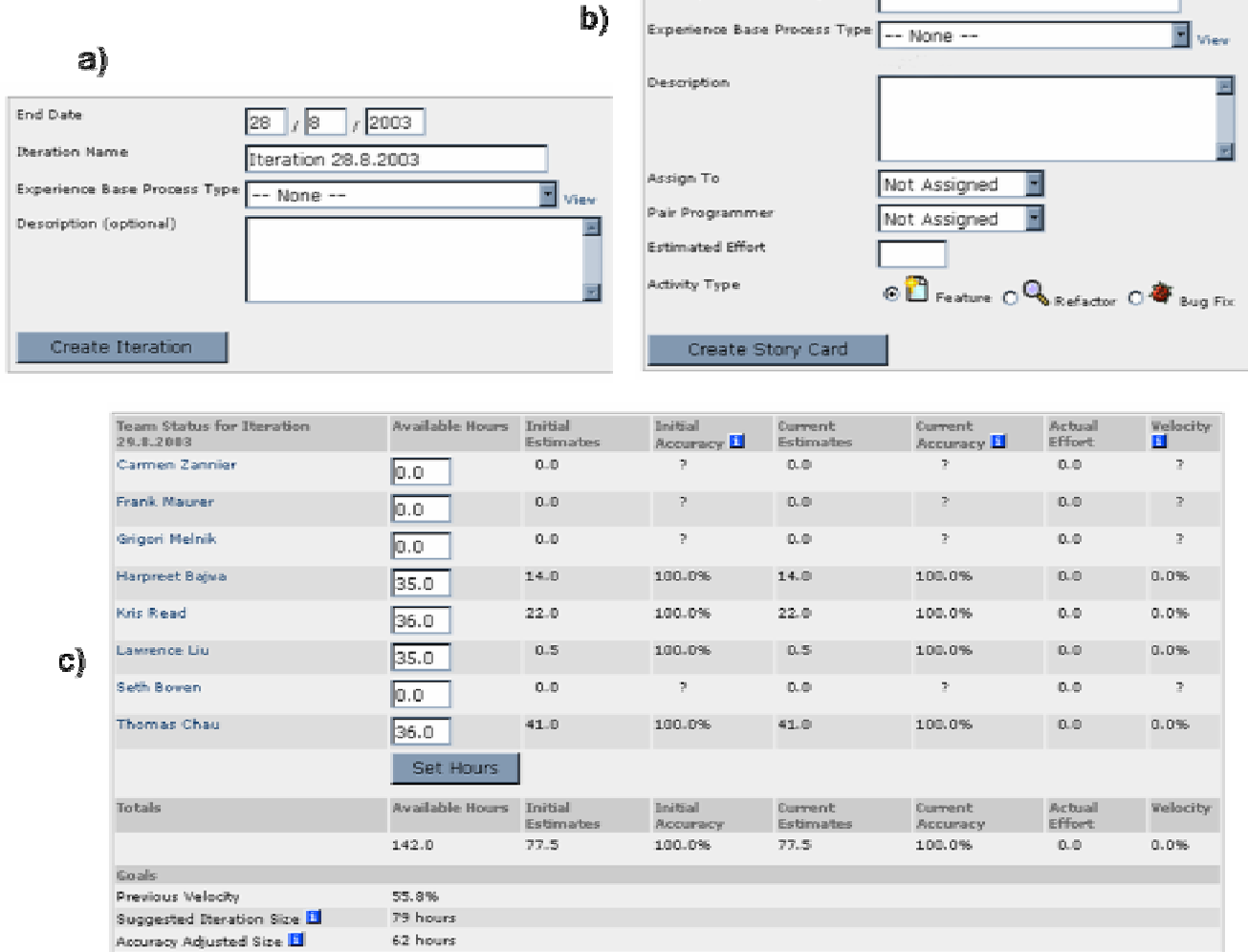


Figure 2. Using the buttons at the bottom of the Whiteboard plug-in (Figure 1), users can launch other plug-ins to (a) create new iteration and (b) create new user story. Figure c allows users to view effort metrics for a particular individual or for the entire team.

Upon creation of an iteration, each member 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' estimation accuracy from the previous iteration (Figure 2c). This suggested iteration size serves as a guide for the customers and project managers, so they can prioritize user stories by allocating them to other iterations or back to the product backlog (Figure 1).

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 if they find the default set of input fields provided by the plug-in inadequate.

Besides the support for agile project planning and tracking, developers can also make use of the integration with NetMeeting to perform distributed pair programming [5] by sharing their code editor; to collaborate on a design together using the shared whiteboard; to perform daily Scrum meetings [6] with distributed team members who are working at the same time using the video and audio conferencing and multi-user text-chat features of NetMeeting.

4.2. EB: Support for Organizational Learning

To facilitate organizational learning, EB provides two levels of support: one facilitates the dissemination of

organizational expertise to teams and another that facilitates knowledge sharing among multiple teams.

To facilitate the dissemination of expertise that is accepted organization-wide to teams, EB supports modelling of common tasks. Such process models can

contain detailed description about common tasks and their possible decompositions into finer-grained subtasks in a structured format. Users can also store any other information about the model as free-formatted text. This concept is illustrated in the top left window of figure 3.

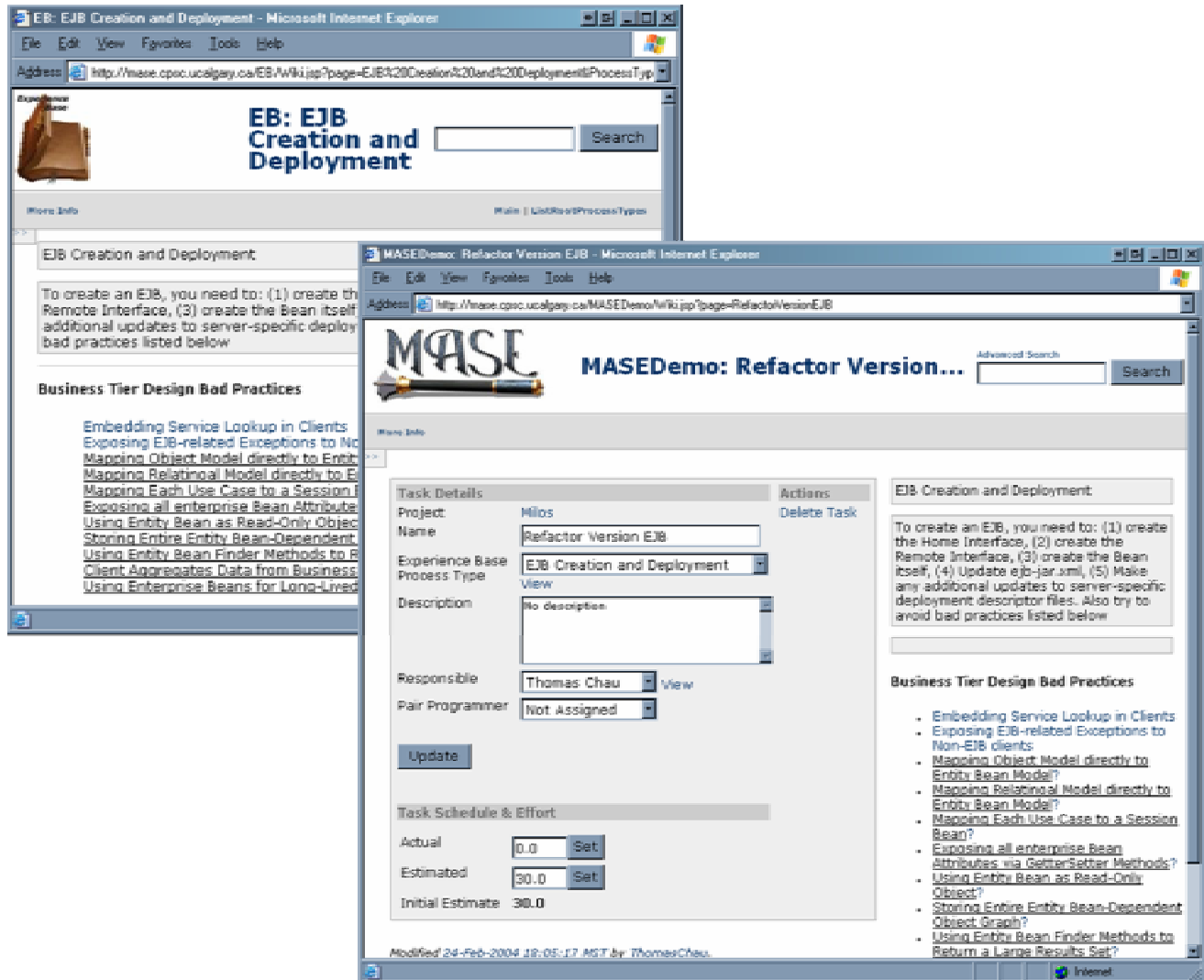


Figure 3. The top left window shows a process model for a common development task, EJB Creation and Deployment. It contains descriptions of the steps one should follow in performing tasks of similar kind. These are stored in a structured format (shown in grey). Additional information such as some common bad practices to avoid is stored as free-formatted text. The bottom right window shows that MASE automatically retrieves task-specific information from particular process model in EB. This integration between EB and MASE is made possible by the fact that EB exposes all its process models as web services and that MASE contains a plug-in that allows users to specify the address of EB’s web services.

To effectively disseminate organizational knowledge to operational teams, capturing information in process models is not enough. It is crucial to make the information easy to retrieve as well. To this end, EB attempts to increase the awareness for existing knowledge and reduce users’ time in looking up information by automatically delivering task-relevant information to

users’ project workspace. This concept is illustrated in the bottom right window of figure 3.

It shows that Thomas is working on the “Refactor Version EJB” task in the “Milos” project and this task is linked to the “EJB Creation and Deployment” process model in EB.

Every time when Thomas accesses information about the “Refactor Version EJB” task in MASE, he

automatically sees information about others' experiences on tasks of similar kind, in this case, EJB Creation and Deployment.

These experiences may have been provided by others whom Thomas may know but are from other teams in the organization. Since these experiences are annotated on wiki pages, the open-edit nature of wiki and EB's support for synchronous work allow Thomas and co-workers from other teams who share common information needs with him to establish contact, exchange their individual experience and expertise, and collaborate together.

5. Potential Limitations and Future Work

The Wiki-based application framework underlying our approach is based on the assumption that there is a need for collaborative knowledge creation in a lightweight manager in an information repository. Hence, our approach may not be appropriate for other types of projects or domains.

Immediate future work is to gather feedback on the practicality and usability of the tools. An observation study is currently being conducted but no analysis result is available.

6. Conclusion

Distributed development is sometimes inevitable and must be dealt with by agile methods practitioners. The increased complexities in communication, collaboration, and coordination associated with distributed development demand tools that can accommodate various styles of collaborative work. For such tools to be useful for agile teams, they also have to provide support for agile practices and overcome the challenge in organizational knowledge sharing.

This paper presents some extensions to an existing Wiki web implementation to provide flexible support for various collaborative work styles: co-located and distributed teamwork; asynchronous and synchronous work activities; and use of structured and unstructured information. Using this Wiki-web implementation as an application platform on which other web-based collaborative tools are built, we present a suite of tools that address the above concerns.

One such tool is MASE which provides process support for specific agile practices, such as release and iteration planning, distributed pair programming, and daily Scrum meetings. Another tool is EB which provides a shared experience base that facilitate organizational learning in an agile software company by disseminating organizational knowledge to project teams. It also facilitates knowledge sharing across different teams by providing process models on common tasks and

delivering task-relevant information to team members' project workspace on demand via its integration with MASE.

While Wiki web is not a new concept, the novelty of our approach is the extension of existing Wiki web implementation and integration with other CSCW tools to facilitate flexible collaborative work and adapting them for use by agile methods practitioners.

7. References

1. Ebert, C., De Neve, P. (2001), Surviving Global Software Development, *IEEE Software*, 18(2), 62-69.
2. PhpNuke <http://www.phpnuke.org> (Last Visited: March 5, 2004)
3. Cunningham, W. Leuf, B. (2001), *The Wiki Way Quick Collaboration on the Web*, Addison Wesley, Reading, MA.
4. Greenburg, S., Roseman, M. (1998), Using a Room Metaphor to Ease Transitions in Groupware, in M.S. Ackerman, P. Volkmar & W. Volker, eds, "Sharing Expertise: Beyond Knowledge Management", MIT Press, Cambridge MA.
5. Stotts, D., Williams, L., Nagappan, N., Baheti, P., Jen, D., Jackson, A. (2003), "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming", in F. Maurer, D. Wells, eds, *Proceedings of XP/Agile Universe 2003*, Springer, Berlin Heidelberg New York.
6. Beedle, M., Schwaber, K. (2001), *Agile Software Development with SCRUM*, Prentice Hall, Englewood Cliffs, NJ.
7. TWiki <http://www.twiki.org> (Last Visited: September 25, 2003)
8. VersionOne <http://www.versionone.net> (Last Visited: September 25, 2003)
9. Xplanner <http://www.xplanner.org> (Last Visited: September 25, 2003)
10. Henninger, S., Ivaturi, A., Nuli, K., Thirunavukkaras, A. (2002), "Supporting Adaptable Methodologies to Meet Evolving Project Needs", in D. Wells, L. Williams, eds, *Proceedings of XP/Agile Universe 2002*, Springer, Berlin Heidelberg New York.
11. Wenger, E. McDermott, R., Snyder, W. (2002), *Cultivating Communities of Practice*, Harvard Business Press, Boston, MA
12. Henninger, S. (2002), "Panel: Can Agile Methods and Learning Software Organizations Support Each Other", in S. Henninger, F. Maurer, eds, *Proceedings of the 4th International Workshop on Learning Software Organizations 2002*, Springer-Verlag, Berlin Heidelberg
13. Chau, T., Maurer, F. (2004), "Tool Support for Inter-Team Learning in Agile Software Organizations", in *Proceedings of the 6th International Workshop on Learning Software Organizations 2004*, Springer-Verlag (to appear)
14. JSPWiki <http://www.jspwiki.org> (Last Visited: April 14, 2004)
15. Gray, J., Liu, A., Scott, L. (2000), "Issues in Software Engineering Tool Constructions", in *Proceedings of 2nd International Symposium on Construction of Software Engineering Tools 2000*