Improving Requirements Management in Extreme Programming with Tool Support – an Improvement Attempt that Failed

Jukka Kääriäinen, Juha Koskela, Pekka Abrahamsson, Juha Takalo
VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90571 Oulu, Finland
{jukka.kaariainen; juha.koskela; pekka.abrahamsson; juha.takalo}@vtt.fi

Abstract

While Extreme programming (XP) relies on certain principles, it requires an extensive set of tools to enable an effective execution of its practices. In many companies, putting stories on the board may not be sufficient for managing rapidly changing requirements. The objective of this paper is to report the results from a study where a requirement management tool – the Storymanager – was developed to meet the needs of a XP project team. The tool was used in a case project where a mobile application for real markets was produced. The tool was dropped by the team only after two releases. The reasons of the process improvement failure are addressed in this paper. The principal results show that the tool was found to be too difficult to use and that it failed to provide as powerful a visual view as the paper-pen board method. The implications of these findings are addressed for both the practitioners and researchers in the field.

1. Introduction

Extreme programming (XP) is a well known agile software development method. While XP relies on certain principles, such as communication and simplicity, it also requires tools to enable an effective execution of its practices. In many companies, listing the stories on to the board is not sufficient for managing changing requirements. We made an attempt at finding a solution for managing user stories and tasks in electronic format as a part of the Eclipse tool integration framework, but there were no solutions available for this. Thus, we decided to develop a specific plug-in application for the Eclipse environment, which was called the Storymanager. One of the characteristic of methods and tools is that they need to be adapted to fit a certain company or project context [1]. Thus, these tools have to take into account the nature of the project, including the development methods used in the project.

The objective of this paper is to report the results from a study where a requirement management tool – the Storymanager - was developed to meet the needs of a fast moving XP project team. The aim of the tool was to minimize rework and automate the time consuming paper-pen practices, such as recording story and task items on the board and then separately on an excel sheet, or equivalent.

The paper is composed as follows. The background concepts of Extreme Programming and Requirements Management (RM) are first introduced. Then the related research is laid out regarding RM in XP context. This is followed by a detailed discussion on our solution for RM in XP environment. Then, research design is described. Finally, the results are presented and discussed. The paper is concluded with final remarks and the identification of future research needs.

2. Background

This section introduces the concepts of extreme programming and requirements management.

2.1. Extreme Programming

Extreme programming (XP) as a concept has emerged in the late 90's [2]. Along with XP, several agile methods have emerged (for an overview, see, e.g., [3]). XP addresses the issues of changing requirements and their cost by simplifying management tasks and documentation. XP uses an iterative and incremental software process performed in relatively short cycles.

Product development in the XP process starts with a "planning game." Planning game can be divided into "release planning" and "iteration planning" [4]. During the planning game, the customer writes user stories, which are estimated by the developers and then prioritized by the customer. After this, developers divide the stories into tasks and give an estimate for each task. The next step in the XP process is the actual development, during which the iterations are produced and released. Then finally, acceptance tests are used to validate the completion of stories.

2.2. Requirements Management

Requirements management (RM) can be seen as a parallel support process for other requirements engineering processes [5, 6]. It ensures that requirements are documented and that they are traceable during product development and that changes to them are properly handled.

Requirements identification is an essential pre-requisite for RM. It focuses on the assignment of an unique identifier for each requirement [5]. These identifications can be used to unambiguously refer to requirements during product development and management. Further, requirement attributes can be used for recording additional information about requirements [7].

Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in both forward and backward direction [8, 9]. Gotel [8] emphasizes the life cycle aspect of traceability. Requirements form the basis of design and implementation activities, and they should be traceable through the life-cycle of a product. Requirement traceability is needed, e.g., for verification and change impact analysis activities.

Requirements change management refers to the ability to manage changes to requirements [6]. It also ensures that similar information is collected for each proposed change and that overall judgments are made about the costs and benefits of a proposed change. Even if requirement specification is comprehensive, some changes can take place during development. This gives rise to the need for clearly defined practices that provide guidance for handling possible changes to requirements.

3. Related research

In this section, the related research is presented. The results of this review are used for building research lenses (an analysis framework), which will be used to analyze the results of this case project later in the paper.

Traditional XP relies on efficient communication, which is one of the basic values of the method [2, 10]. XP emphasizes communication, e.g. through practices, such as "Open Workspace", "Pair Programming" and "Planning Game" [2, 11]. Macias et al. [12] state that interactive communication between the developers, clients and managers in XP should be emphasized. Face-to-face communication is an efficient mechanism in realizing this. For an agile team to be successful, good communication mechanisms have been found to be critical [13].

The agile principles value working software over comprehensive documentation [3]. Beck [2] also emphasizes lightweight documentation in XP based development. Ambler [13] emphasizes the slogan "Travel light" in the context of documentation. Ambler states that it is useful to produce just enough documentation and to update it only when needed. This enables the team to be more effective in producing results that deliver more business value for the customer than traditional paper-driven methodologies.

Ease-of-use is an important aspect when developing tool support for XP development (and actually for any SW related work). For example, Lippert [14] identifies ease-of-use as a very important aspect for XP tool support during continuous integration. The tool should not slow down the product development or cause additional maneuvers during fast-paced development. O'Brian Holt [15] present some factors for assessing usability, including aspects such as: Is the system easy to learn to use? Is it possible to modify the system without reducing its usability? Is the system comfortable and satisfying to use? Nielsen [16] defines the different aspects of usability as follows: easy to learn, efficient to use, easy to remember, few errors, and subjectively pleasing.

Some authors have considered requirements management from an XP point of view. Breitman and Leite [17] support XP by using a scenario structure to organize information elicited through user stories. While they do not agree with Beck who maintains that implemented stories should be discarded, they highlight the traceability of stories. Nawrocki et al. [18] state that the main weaknesses of the XP approach to requirements management is the lack of requirements documentation. This causes problems especially when managing changes to requirements and maintaining traceability. Alike, Wagner [19] concludes that the lack of written, traceable requirements can make it difficult to maintain the developed software over time. On the other hand, Wagner [19] states that requirements baselining exists, in some form, in the XP process, because each iteration contains an agreed set of stories. From a change management viewpoint, the requirements management literature in fact proposes quite rigorous processes for managing requirement changes [20]. However, formal and cumbersome practices for change management do not fit the nature of XP. Therefore, lightweight and simple practices for managing changes in XP have turned out effective in practice [21].

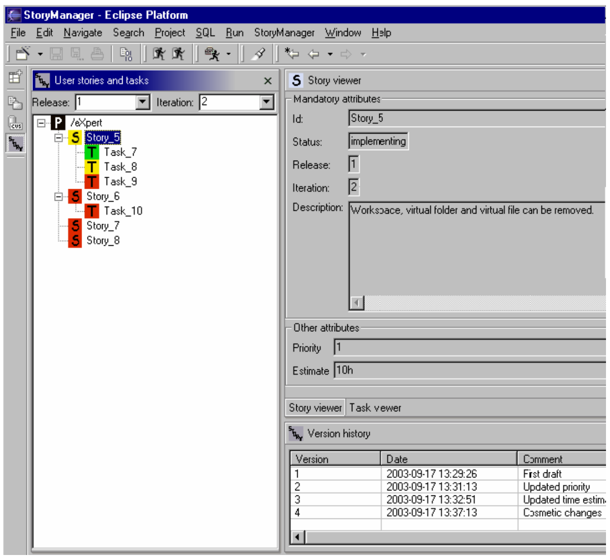
Several authors have addressed the tool support used for managing user stories and tasks. Internet-based tool support for distributed XP, called MILOS, has been introduced by Maurer and Martel [22]. This solution supports virtual software teams with communication, collaboration and coordination. The solution allows the user to write and manage stories and tasks in electronic format. Rees [23] has presented a tool called DotStories for managing user stories, claiming that the tool approaches an ideal solution for user story management. Rees also refers to spreadsheets and databases as further potential tools for managing user stories. Lippert et al. [24] claim that a computer cannot be used for the planning game. On the other hand, they further argue that a computer can be used just for writing stories and tasks and printing them out on paper. The XP process itself does not exclude the use of automated tools for storing user stories. Actually, tools and databases can provide a means for more effective information management [25] [5].

Integrated environment and data sharing enable the project team to focus on development work, while daily data management has been automated. This means that all project-related data is managed at a unified location and integrated tool support eases tedious tasks, such as information retrieval, distribution, consistency checking, archiving, etc. It has been stated in literature that management system integration is likely to improve the consistency and sharing of product-related information (e.g. [26] and [27]).

4. Tool support for the management of user stories: the Storymanager tool

Our solution to managing user stories and tasks is called Storymanager. The proposed solution was integrated in the Eclipse environment. Eclipse is a development environment and a tool integration framework found suitable for XP development. A detailed description of the proposed Storymanager solution has been published in [28] (Figure 1).

Our intention was to remain agile, no new solution or practice should jeopardize the fundamental idea of adaptable and lightweight processes. The basic intention of this study was to transfer manual XP requirements management practices into an electronic form and yet try to remain agile. A further aim was to integrate the solution into the Eclipse tool integration framework. An integrated tool framework enabled the project team to work with one channel throughout the whole development life cycle. During the planning game, the team was working through the Storymanager plug-in. Stories and tasks were stored into an MySQL relational database. During implementation and testing, the team was working, e.g., through a JDT (java development environment) plug-in and a Junit (testing framework) view. From an information management point of view, our approach included support for requirements management and configuration management. Requirements management was used in this environment for sharing and managing user stories and tasks, while configuration management (CVS) was used for managing and sharing code and other documentation.



Mandatory attributes

Id: Story_5

Status: implementing

Release: 1

Iteration: 2

Description: Workspace, virtual folder and virtual file can be removed.

Other attributes

Priority 1

Estimate 10h

Version	Date	Comment
1	2003-09-17 13:29:26	First draft
2	2003-09-17 13:31:13	Updated priority
3	2003-09-17 13:32:51	Updated time estim
4	2003-09-17 13:37:13	Cosmetic changes

Figure 1. Storymanager - the main view

The Storymanager allows a specification of story and task attributes according to the needs of the project. The program enables filling in story/task cards according to project-specific attributes. The AutoID facility of the program is used to automatically create unique identifiers (ID) for stories (and tasks). Certain attributes, however, are mandatory, e.g. status, description, release identifier, iteration identifier. Other attributes are user-defined and optional.

The program allows the user to modify stories, but in XP only the last story version is relevant. Thus, the application contains only the last updated version. According to basic XP principles, formal and bureaucratic change management activities are not considered appropriate. However, the application stores a version history of the item (story/task), which can be used for examining the history of the item. The attributes “Release” and “Iteration” contain information about the selection of items for specific releases and iterations. In fact, this corresponds with the requirements baselining facility indicating an agreed set of items for a specific release and iteration.

During iteration planning, a set of stories is selected for next iteration. This is illustrated using an iteration attribute. Then the tasks are defined for next iteration. A task can be assigned to a story (traceability between stories and tasks). In this case, the program stores the linking information in a MySQL Link-table. However, a task can also be created under a project root and allocated afterwards.

Certain supporting features are needed regardless of the phase of the project. These features include check out/in capabilities, views and reporting. The reporting features are used for printing stories and their respective tasks and putting them on the board when operating in an open workspace.

5. Research design

This section describes how the research and experiment settings were designed.

5.1. Research settings

In this chapter, the research settings used for developing and validating the solution designed for managing user stories are depicted. Application development and validation are based on two XP experiment projects (Figure 2) using the action research [29, 30] approach as the principal methodological driver. Avison [31] and Fowler & Swatman [32] have used the action research method to build information system development methods. Action research is done in cycles, each cycle consisting of planning, action, observation and reflection phases. After each cycle, there will be a revised plan for the next cycle as a result. We applied the action research approach while trying to improve the management of user stories and tasks in XP development.

A project called eXpert was used for developing a system for managing the research data and documents at VTT. The project used XP practices for developing the system. Detailed results of the eXpert project can be found in [33]. The project used a manual solution for managing user stories and tasks, as suggested in the XP literature. During the project, the need for electronic user story and task management emerged. After the project, the results were analyzed and an application was constructed to support a more automated, i.e. electronic, user story management. The requirements for electronic story management and the application itself were introduced in [28]. The validation and improvement planning of the application were carried out as part of the zOmbie-project, which was concerned with developing mobile application software in the Eclipse environment. The validation of the electronic user story management tool was carried out and observations and interviews were made during the project. After the project, the results were analyzed and improvement ideas were produced for future development.

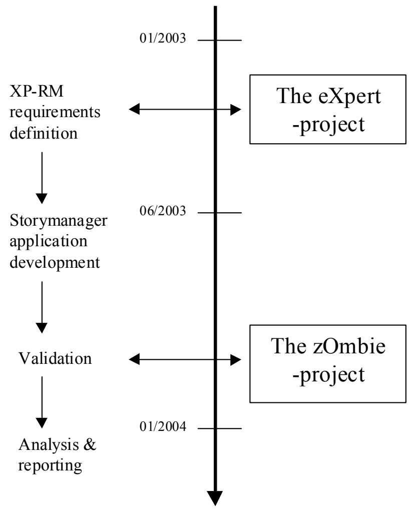


Figure 2. Storymanager development

An analysis framework was constructed to analyze and understand the results of the validation. The framework was based on the survey of related research and underlying XP concepts and requirements management, as presented in sections 2 and 3 of this paper. Our analysis framework reflects technical issues as well as those concerning the methodological aspects of XP. The technical issues focus on the definition of functionality that is needed for requirements management in an integrated XP development environment. The methodological aspects refer to the underlying nature of the XP method. Table 1 presents the analysis framework.

Table 1. Analysis framework

Perspective	Description	Key references
Communication	Does the solution allow open communication between developers and between developers and customers?	[2, 10, 11]
Documentation	Does the solution allow lightweight documentation?	[2], [3], [13]
Ease-of-use	Is the solution easy to use, so as to support fast-paced iterative development?	[14], [15], [16]
Functionality	Does the solution support functional needs for requirements management (identification, traceability and change management) and integrated development environment?	Requirements management : [5], [6], [7], [8], [9], [21], [17],[19], [24] Integrated development environment: [26], [27]

5.2. Experiment settings

The functionality of the application was validated and tested in an experimental XP project called zOmbie. In the zOmbie project, mobile application software was developed in the Eclipse environment. In this experiment, the Eclipse environment was complemented with Storymanager. The aim of the zOmbie project was to produce a real financial sector software product for real markets. The project was an engineering success. The product is now being marketed. The Eclipse environment was used successfully already in the previous XP case study of VTT [33]. In zOmbie case study, the Eclipse environment was used together with the following tools (Table 2).

Table 2. Tool environment in the zOmbie experiment

Tool	Version	Description
Eclipse	2.1	Tool integration framework
Storymanager	1.0.0	User story and task management
JDT (part of Eclipse package)	2.1	Java development environment
Junit	3.8.1	Testing framework
CVS	1.11.2	Version management

The verification was carried out in the XP experiment project developing mobile application software in Eclipse environment. The project team consisted of 5 developers and a project manager. The project worked according to XP practices, but some of the practices needed slight adaptation (e.g., test-first development in mobile application is challenging) according to the business needs.

The aim of Storymanager validation was to use the XP project to verify our solution for requirements management. The focus was to ensure that requirements management support was adequately considered in the integrated development environment and that the solution allowed the XP project to remain "agile" and "lightweight". The project team was allowed to systematically change any practices if they felt that these did not work. Thus the project group were trained and encouraged to "think according to XP values".

Quantitative and qualitative data were collected throughout the project. The project had nominated a person responsible for metrics, who was monitoring that data was collected systematically. The metrics and practices for gathering quantitative data were defined before the project start-up. Qualitative data was collected from the project team by using a specified comment template. The template contained questions about the applicability of the Storymanager solution. The comments were processed and analyzed using Post Mortem [34] sessions. The role of XP coach was extended in zOmbie. The coach was also acting as a Storymanager advisor, collecting experiences concerning its usage. Even after the project, comments and improvements were inquired from the project team and coach. The inquiry performed after the project contained the following questions, which were formulated based on the experience that the Storymanager was abandoned after two releases and manual story and task management was used during the rest of the project:

– Which were the advantages of Storymanager compared to manual story and task management?
– Which were the disadvantages of Storymanager compared to manual story and task management and why was the Storymanager abandoned?
– Give other comments and suggestions for the improvement of electronic management of user stories and tasks?

6. Results

Table 3 presents the basic quantitative data from the experiment. It provides basic information about the size and schedule of the project and helps the reader to understand the nature of the project where the Storymanager tool was used.

Table 3. Background information about the mobile application case project

Collected data	Release 1	Release 2	Release 3	Release 4	Release 5	Correction release	Total
Calendar time (weeks)	1	2	2	2	1	0.4	8.4
Total work effort (h)	115.3	238.9	273.2	255.6	123.7	66.8	1073.5
Planning day effort (h)	37.1	22.7	32.8	24.5	15.7	13.5	146.3
# User stories implemented	3	4	5	5	1	1	19
# Tasks implemented	11	25	18	18	10	10	92

The development team used Storymanager for storing, distributing and retrieving story and task information during the first two releases. Stories and tasks were created, modified and maintained in Storymanager and printed out from the system and put on the board. After the second release, the project team moved into manual story and task management, because of the visualization and usability problems of electronic story and task management. The project manager had some experience with manual management of user stories and tasks, which made it possible to move from electronic to manual management.

This section introduces the results of the interviews. First, the project team and coach were asked to voice their opinion about the advantages of the Storymanager for story and task management. A collection of answers from the zOmbie coach and team members are presented in the following:

"Easy to operate with stories and tasks when they are in electronic format. In manual format, story or task modification required rewriting the whole card ."
"Integration to Eclipse enables easy access to tool."
"Manual cards are sometimes lost, but when they are in electronic format, they are easily accessible."
"The use of the status attribute was easy and the color codes were practical."
"The customer can easily follow the implementation of stories from a remote office using Storymanager"
"Manual archiving is not needed after a project"

Then the project team and coach were asked about the disadvantages of Storymanager for story and task management and they were also asked to give the reasons for abandoning the tool. The answers received from the zOmbie team members and coach are listed in the following:

"Not clear. It is easier to see the Big Picture of the project (e.g. status) when the manual story and task cards are put on the board"
"The reports (layout of printed story/task cards) were not good. If they were better, it would be more useful."
"The tasks in Storymanager are in text format. However, tasks sometimes also contain other formats than just pure text (i.e. pictures, etc.)."
"It was difficult to move tasks between stories."
"The usability was not good."
"The AutoID functionality was confusing in Storymanager."
"More disadvantages than advantages."

The project team and coach were also asked to give further comments and suggestions for improvement. In the following, a summary of answers received from zOmbie team members and coach is presented:

"Support is needed for XP project management."
"A tool should be easy to use."
"It should be easy to see the Big Picture of the project when using the tool."
"The tool should allow moving tasks more easily between stories."
"The tool should allow linking tasks with application code."
"Printing of stories and tasks with bigger font."

Table 4 summarizes the advantages and disadvantages of the Storymanager. The reasons for moving from electronic management to manual management and their implications are analyzed in next section.

Table 4. Summary of the results gained from the Storymanager tool study

Perspective	**Results** ***("+" strengths," –" weaknesses/enhancements)***
Communication	+ The customer can easily follow the implementation of stories from a remote office using Storymanager + Color codes can be used to visualize the status of the story/task on computer screen – the Storymanager system or printed story/task reports do not make it easy to see the overall status ("big picture") of a project
Documentation	+ Manual stories or task cards need not be archived separately – Tasks can contain just text description – Reports are not clear (layout and font size)
Ease-to-use	+ It is easy to manage stories and tasks (modification) + Stories and tasks are easily accessible in electronic format – General usability of the tool is not good – Moving tasks between stories is difficult
Functionality	+ Integration to Eclipse enables easy access to tool and information + Tool provides reliable means to store, modify and retrieve information – AutoID provides "meaningless" code for a story or task – Support for XP project management should be added to the tool – There should be a possibility of linking tasks with application code

7. Discussion

This section analyses the results against the analysis framework defined in section 5.1. Agile methods, such as XP, aim towards efficient communication and lightweight documentation. Although some additional or modified XP practices were used in the VTT zOmbie-project, the basic development philosophy relied on open communication and lightweight documentation. As presented in related literature, [e.g. 5, 25, 35, 36], the adaptation of product information management should be made on the basis of the business context of a project. Thus, in this case, the nature of the XP method drives the adaptation of requirements management tool support. Table 5 summarizes the most important findings and their implications.

When considering the results in section 6, it can be noted that the developed solution for electronic management of user stories and tasks tackles mainly the same things as the manual one. The clear advantages of the electronic solution compared with manual management are related to the ability to reliably store, modify, distribute, retrieve and archive stories and tasks and the ability to operate in an integrated development environment where all development tools are available.

There are two fundamental differences between electronic and manual management. These are related to information visibility and usability. Open workspace allows the project team to use paper cards for stories and tasks and put them on the board. This allows the team to get an overview of stories and tasks just by having a look at the board, discussing the items and making modifications directly to the story and task cards. While this way of working is, in fact, highly effective and it emphasizes natural interaction between developers, it does require that the team members share an open workspace. Rees [23] states that one problem connected with using databases for managing user stories is that they provide just poor group visualization of all cards. Our experiment supports this claim. Of course, electronic management also allowed us to print the stories and tasks and then to put them on the board. This has been suggested by Lippert et al. [24], who first claim that a computer cannot be used for the planning game, and then further specify that a computer can be used just for writing stories and tasks and printing them out on paper. However, our experiment shows that not even this approach works, if the editing and maintaining of printed story and task cards takes up too much effort in fast-paced development. Furthermore, the format of printed cards should be highly distinct to be able to compete with manual ones. Thus, further research is needed to explore the possibilities for improving visualization in electronic management of user stories and tasks.

Table 5. Findings and their implications

Perspective	Findings	Implications
Communication	Electronic management of stories and tasks enables remote customers and other stakeholders to view the status of the project in real-time. Electronic management seems to be an obvious solution when operating in a distributed development environment but it can jeopardize natural interaction between developers and the visibility of information in open workspace.	Visualization of stories and tasks is a challenge and requires further research. Furthermore, electronic management of user stories and tasks seems to be effective if the customer is off-site and the project is distributed over several sites.
Documentation	Electronic task cards should be able to contain also other formats of information than text descriptions. The format of reports should be clear.	Possibilities of integrating a drawing or modeling tool into Storymanager and Eclipse should be considered.
Ease-of-use	Tool usability should be good in fast-paced development.	Application development using, e.g., a User-Centered Design approach to ensure that usability issues are considered.
Functionality	The tool should also provide additional value for daily routines, not just automated support for the old "pen and paper" practices.	Support for XP project management and linking between tasks and application code should be examined as a part of the Eclipse environment.

It also became apparent during the zOmbie-project that electronic management should also allow other formats of information than just pure text (e.g. pictures, models, etc.). During the project, the operation with models was an important part of the work because the product being developed was complicated. On the other hand, in the eXpert-project, graphical modeling was not an important aspect. Therefore, the product being developed itself seems to affect the requirements management needs of the project. This claim is supported, e.g., by Kotonya and Sommerville stating that the type of system under development affects the requirements management solution [6]. The use of modeling in XP has been considered by Beck [2], who argues that although modeling can be used during the XP process, the pictures should not be saved. We cannot, however, agree with this proposition. If models are made, they should be archived as any other information during the project for maintenance reasons. On the other hand, modeling support easily makes the Storymanager tool complex.

If a project is distributed over several sites, electronic management of stories and tasks seems quite an obvious solution. This has been demonstrated by Maurer and Martel [22], for example. However, one problem encountered with Storymanager had to do with its usability. In fast-paced development, the usability should be good. It is difficult to justify tool usage if the tool slows down the development and requires additional maneuvers. One way of tackling this kind of problem is to use User-Centered Design approaches during tool development, so as to ensure that usability issues are considered [37].

The basic problem in the Storymanager requirements management solution is that even though it does address the requirements management issues of product development, it also fights against the values of the XP method. The solution reduces simplicity and open communication between team members when operating in an open workspace during fast-paced product development.

The developed automatic solution focused mainly on the automation of manual story and task management activities. However, Tolvanen [1] states that, in the long run, the promise of tools does not lie just in the automated support of old “pen and paper” methods. Our findings support this claim. Consequently, the aim of supporting manual operations without providing extensive features or significant additional value for developers proved too narrow. Some of the comments gained from the zOmbie project team addressed this issue. For example, some enhancement ideas were voiced concerning providing support for project management using story and task efforts and related automatic calculation, and also concerning automated traceability between tasks and code.

8. Conclusions and future research

This paper presents the results from a study where a requirements management tool called Storymanager was developed to meet the needs of a fast moving XP project. The aim of the tool was to minimize rework and to automate time consuming paper-pen practices, such as recording story and task items on the board. The tool was used in a project where mobile application software for real markets was produced. The team abandoned the tool only after two releases. Essentially, the tool failed to provide as powerful a visual view as the paper-pen board method. The developed tool also turned out to be too difficult to use in fast-paced development environment. The interview data further revealed that a computerized solution is not by any means self-evident, but rather it has to be able to compete with the best alternate solutions, i.e. manual paper-pen approach in this case, and even provide additional value for the project team.

The experiences, findings and implications of this study should be of interest to any organization considering requirements management tool support for XP projects.

The results emphasize the role of adaptation in tool development. The tool, constructed to support any development method, should take into account the underlying values of the method itself. If this is not the case, the tool is likely to work against the nature of the method.

Future research will be concerned with constructing a new solution addressing the lessons learned that were found relevant during this study. This is to be followed by validating and enhancing the solution in future SW development projects.

References

[1] J. P. Tolvanen, "Incremental method engineering with modeling tools," in *PhD dissertation*. Jyväskylä University Printing House and ER-Paino Ky: University of Jyväskylä, Finland, 1998.

[2] K. Beck, *Extreme programming explained: Embrace change*. Reading, MA.: Addison Wesley Longman, Inc., 2000.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods: Review and Analysis*. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478.

[4] K. Beck and M. Fowler, *Planning extreme programming*. New York: Addison-Wesley, 2001.

[5] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practise Guide*: John Wiley & Sons, 1997.

[6] G. Kotonya, Sommerville, I., *Requirements Engineering: Process and Techniques*: John Wiley & Sons, 1998.

[7] D. Leffingwell and D. Widrig, *Managing Software Requirements - A Unified Approach*: Addison-Wesley, 2000.

[8] O. Gotel, "Contribution Structures for Requirements Traceability," in *Imperial College of Science, Technology and Medicine*: University of London, 1995.

[9] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," presented at First International Conference on Requirements Engineering, 1994.

[10] L. Williams, "The XP Programmer: The Few-Minutes Programmer," *IEEE Software*, vol. 20, pp. 16-20, 2003.

[11] R. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*. Upper Saddle River, NJ: Addison-Wesley, 2001.

[12] F. Macias, M. Holcombe, and M. Gheorghe, "A formal experiment comparing extreme programming with traditional software construction," presented at Proceedings of the Fourth Mexican International Conference on Computer Science, 2003.

[13] S. Ambler, "Lessons in Agility from Internet-Based Development," *IEEE Software*, vol. 19, pp. 66 - 73, 2002.

[14] M. Lippert, S. Roock, R. Tunkel, and H. Wolf, "Stabilizing the XP Process Using Specialized Tools," presented at XP 2001, 2001.

[15] P. O'Brian Holt, "HCI tools, methods and information sources," presented at IEE Colloquium on Usability Now, 1991.

[16] J. Nielsen, *Usability engineering*. San Francisco, CA: Morgan Kaufmann Publishers, 1993.

[17] K. Breitman and J. Leite, "Managing User Stories," presented at International Workshop on Time-Constrained Requirements Engineering (TCRE 02), 2002.

[18] J. Nawrocki, M. Jasinski, B. Walter, and A. Wojciechowski, "Extreme Programming Modified: Embrace Requirements Engineering Practices," presented at 10th IEEE Joint International Requirements Engineering Conference, RE'02, Essen, Germany, 2002.

[19] L. Wagner, "Extreme Requirements Engineering," *Cutter IT Journal*, vol. 14, pp. 34-38, 2001.

[20] I. Hooks and K. Farry, *Customer-centered products : creating successful products through smart requirements management*. New York, NY: American Management Association, 2001.

[21] J. Koskela, J. Kääriäinen, and J. Takalo, "Improving Software Configuration Management for Extreme Programming: A Controlled Case Study," presented at European Software Process Improvement, EuroSPI'2003, Graz, Austria, 2003.

[22] F. Maurer and S. Martel, "Process Support for Distributed Extreme Programming Teams," presented at ICSE 2002 Workshop on Global Software Development, 2002.

[23] M. J. Rees, "A feasible user story tool for agile software development?," presented at Ninth Asia-Pacific Software Engineering Conference, 2002.

[24] M. Lippert, S. Roock, and H. Wolf, *Extreme Programming in Action: Practical Experiences from Real World Projects*: John Wiley & Sons Ltd., 2002.

[25] A. Leon, *A Guide to software configuration management*. Boston: Artech House, 2000.

[26] I. Crnkovic, Asklund, U., Dahlqvist, A., *Implementing and Integrating Product Data Management and Software Configuration Management*. London: Artech House, 2003.

[27] A. Sääksvuori and A. Immonen, *Product lifecycle management*. Berlin: Springer-Verlag, 2004.

[28] J. Kääriäinen, J. Koskela, J. Takalo, P. Abrahamsson, and K. Kolehmainen, "Supporting Requirements Engineering in Extreme Programming: Managing User Stories," presented at ICSSEA 2003, Paris, France, 2003.

[29] M. Hult and S.-A. Lennung, "Towards a definition of action research: A note and bibliography," *Journal of Management Studies*, pp. 241-250, May 1980.

[30] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," *Administrative Science Quarterly*, vol. 23, pp. 582-603, 1978.

[31] D. Avison, "Action research: a research approach for cooperative work," presented at The 7th International Conference on Computer Supported Cooperative Work in Design, 2002.

[32] D. C. Fowler and P. A. Swatman, "Building information systems development methods: synthesising from a basis in both theory and practice," presented at Australian Software Engineering Conference, 1998.

[33] P. Abrahamsson and J. Koskela, "Extreme programming: A survey of empirical results from a controlled case study," To be presented at ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2004),, Redondo Beach, CA, USA, 2004.

[34] T. Dingsøyr and G. K. Hanssen, "Extending Agile Methods: Postmortem Reviews as Extended Feedback,"

presented at 4th International Workshop on Learning Software Organizations, Chicago, Illinois, USA, 2002.

[35] F. Buckley, *Implementing configuration management : hardware, software, and firmware*. Los Alamitos: IEEE Computer Society Press, 1996.

[36] D. Lyon, *Practical CM - Best Configuration Management Practices for the 21st Century*, 2nd ed: RAVEN Publishing Company, 1999.

[37] A. L. Ames, "Users first! An introduction to usability and user-centered design and development for technical information and products," presented at Professional Communication Conference, IPCC 2001, 2001.