

# Supporting requirements engineering in extreme programming: managing user stories

**Jukka Kääriäinen, Juha Koskela, Juha Takalo, Pekka Abrahamsson, Kari Kolehmainen**

VTT Technical Research Centre of Finland

P.O.Box 1100, FIN-90571 Oulu, Finland

Phone: +358 8 551 2191, Fax: +358 8 551 2320

{jukka.kaariainen, juha.koskela, juha.takalo, pekka.abrahamsson}@vtt.fi, kari.kolehmainen@annuminas.com

## Abstract

One objective for the agile methods is to lower cost of changing requirements. Currently the most popular agile software development method is Extreme Programming (XP). XP addresses this issue by simplifying management tasks and documentation while the traditional software engineering places more emphasis on strict control and extensive documentation. Requirements management (RM) is the activity that ensures that requirements are traceable and all changes to requirements are properly handled. In a dynamic and fast moving project with an iterative process, RM may tie up too much resources. Requirements management and configuration management (CM) are only implicitly addressed by XP. It may be that the method developers viewed RM and CM too bureaucratic, heavy weight or ceremonial to be included in XP. Currently, in the XP process, user requirements called as “user stories” are written and managed on paper cards. The objective of this paper is to examine the challenges involved in the requirements management in an XP project. The aim is also to study possibilities to integrate the support for user story management into single tool framework. Based on the study, an approach for managing user stories with the respect of XP’s basic principles is depicted. Our study is based on an empirical XP case study. According to our study, the following issues were recognized especially important when considering the management of requirements in XP. First, XP provides the basic set of practices that should be shaped into the development situation. This means that practices are tailored for the purposes of a project and the organization, and they can be further changed on-the-fly on periodic process assessments, i.e. if they do not work. Thus, the tool support should not force detailed procedures but provide just enough basic abilities for storing, relating and retrieving user stories and tasks. Second, the tool should allow the XP process to remain agile. This means that the tool should not jeopardize XP’s intentions for open communication and lightweight management and documentation. Third, requirements management tool support should be integrated into the project’s overall development environment. This allows the project team to operate via one channel from a user story definition, through implementation, up to testing. Our solution for the management of user stories and tasks is called StoryManager. The solution has been integrated as plug-in into Eclipse –tool integration framework to enable integrated environment for an XP project.

## Keywords

Extreme programming, requirements management, requirements engineering

## 1 Introduction

New software development methodologies called as agile methods have been developed to address the needs for lightweight and faster software development processes [3]. Currently the most popular one is Extreme Programming (XP), an agile development method developed by Kent Beck [4]. One objective for the agile methods is to lower cost of changing requirements. XP addresses this issue by simplifying management tasks and documentation while the traditional software engineering places more emphasis on strict control and extensive documentation. In order to achieve simplicity, XP uses an iterative and incremental software process and very short development cycles. Further, it minimizes documentation and ties up customer involvement into the product development.

Requirements management (RM) ensures that requirements are traceable and all changes to requirements are properly handled [20]. In the development of complex software products, the requirements management can be difficult and effort consuming when detailed traceability is targeted. In a dynamic and fast moving project with an iterative process, this may tie up too much resources. XP strives for efficient use of resources, an early introduction of functional product releases and continuous feedback from the on-site customer. Therefore, it is inefficient if major proportion of resources has to be used for management tasks that do not deliver concrete results.

The objective of this paper is to examine the challenges involved in the requirements management in an XP project. The aim is also to study possibilities to manage user stories and tasks electronically as part of integrated tool framework. Requirements engineering, especially capturing and analyzing user's conception about the system, happens in planning game phase in XP. Thus, our approach focuses on planning game phase of the XP process. First, the paper introduces XP and RM concepts. After that, the related work concerning user story management is considered. It describes also how user requirements are handled in the current XP process and identifies the potential problem areas that arise from the use of manual solution. Then, findings based on an empirical XP case study are introduced. Based on these findings, an approach for managing user stories with the respect of XP's basic principles is depicted. Finally, conclusions and future research activities are identified.

## 2 Background

The following sub-sections introduce XP and RM concepts as well as related work.

### 2.1 Extreme Programming (XP)

Extreme programming (XP) as a concept has emerged in the late 90's along with Kent Beck's book "Extreme Programming explained: Embrace Change" [4]. Along with XP, many more of these "agile" methods have emerged (for an overview see e.g., [2]). XP addresses issues of changing requirements and their cost by simplifying management tasks and documentation. XP uses an iterative and incremental software process in relatively short cycles. Traditionally this type of approach would yield increased management overhead because management activities related to ending and starting iteration have to be executed for every iteration, but these are minimized in the XP process. XP introduces many practices but two techniques which are characteristic to XP are pair programming and test driven development [4]. XP may first seem quite chaotic, but it includes several good engineering practices [18]. However, for example, inadequacy of requirements management practices have raised some concerns [16].

Product development in the XP process starts with "planning game." Planning game can be divided into "release planning" and "iteration planning" [5]. During planning game, customer writes user stories, which the developers estimate and customer then subsequently prioritizes. Planning game is a phase in XP development when requirements, that is stories, are elicited, estimated and selected for release. Planning game is performed for each release. A release is divided into iterations. The subset of stories based on priority and size is then selected for each iteration. This is called iteration planning. Developers then divide stories into tasks and give an estimate for each task. Estimating user stories is difficult in other than very coarse level. On the other hand, estimating tasks is much more easy and accurate because tasks are defined in more detailed and concrete level. The next step in the XP process is development when the iterations are produced and released. Then acceptance tests are used to validate the completion of stories. Our consideration in this paper focuses on planning game phase of the XP process.

## 2.2 Requirements Management (RM)

Requirements management (RM) can be seen as a parallel support process for other requirements engineering processes [20][12]. It ensures that requirements are documented and traceable during product development and changes to them are properly handled.

Requirements identification is an essential pre-requisite for RM. It focuses on the assignment of unique identifier for each requirement [20]. These identifications can be used to unambiguously refer to requirements during product development and management. Further, requirement attributes can be used to record additional information about requirements [13]. Leffingwell and Widrig [13] emphasize that by using attributes, you get better management of complexity of information.

Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in both a forwards and backwards direction [9][10]. Gotel [9] emphasizes the life cycle aspect of the traceability. Requirements form the basis for design and implementation activities, and they should be traceable through product's life. Requirements' traceability is needed, e.g. for verification and change impact analysis activities. Traditional solutions for RT are based on manual traceability tables (e.g. solutions based on spreadsheets) and automatic management of traceability information stored in database management system (e.g. RM tools).

Requirements change management refers to the ability to manage changes to the requirements [12]. It also ensures that similar information is collected for each proposed change and that overall judgments are made about the costs and benefits of proposed change. Requirements are "frozen" when moving to the design phase (requirements baselining). Even if requirement specification is comprehensive, something can change during development, for example, customer's needs or regulations. This causes the need for clear practices that guide how possible changes to requirements are handled.

## 2.3 Related work

Currently, XP has generated a lot of interest from practitioners and academia. However, the management of user stories has received only little attention in the respective XP literature. A user story can be thought as a high level requirement or a user requirement. It is the user's conception about a functionality that the system should provide. RM and configuration management (CM) are only implicitly addressed by XP. It may be that the method developers viewed RM and CM too bureaucratic, heavy weight or ceremonial [7] to be included in XP. Currently, in the XP process, user stories and tasks are written on paper cards [4]. Story and task cards are identified using story and task numbers, and story cards are placed onto the wall with their respective task cards. This procedure was used in the eXpert -project, an empirical XP case study carried out in VTT Electronics [1]. This is a very simple and powerful way of visualizing the traceability between stories and tasks when a project team works in co-located workspace.

Few authors have considered requirements engineering in XP development. Paetsch et al. [17] analyze the commonalities and differences of traditional RE and agile SW development. They further analyze possibilities how agile methods can benefit from traditional RE methods. Breitman and Leite [6] support XP by using a scenario structure to organize information elicited through the user stories. They do not agree with Beck that implemented stories should be discarded but highlight the traceability of stories. Alike, Wagner [21] concludes that the lack of written, traceable requirements can make it difficult to maintain the software over time. He also emphasizes that user stories are not complete requirements, but requirements are actually spread into stories, test cases and code. On the other hand, Wagner [21] states that requirements baselining exists, in some form, in the XP process, because each iteration contains agreed set of stories. Internet-based tool support for distributed XP, called MILOS, has been introduced by Maurer and Martel [15]. Solution supports virtual software teams with communication, collaboration and coordination. The solution allows you to write and manage stories and tasks in electronic format. Lippert et al. [14] claim that a computer cannot be used for the planning game. On the other hand, they further specify 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. However, it is obvious that they still should be printed and put onto the wall if necessary.

Pinheiro [19] discusses requirements research from the point of view of agile methods, especially from XP viewpoint. Paulk [18] considers XP from software CMM (Capability Maturity Model) perspective and concludes that XP has many good practices, but they are not suitable for every occasion. Requirements process modifications in the XP process are introduced by Nawrocki et al. [16]. They assess XP against CMM-model and model introduced by Sommerville and Sawyer [20]. They 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. According to the model introduced by Sommerville and Sawyer, XP supports only few practices. Thus, Nawrocki et al. [16] introduced an extended XP process that address the most of the deficiencies of the traditional XP process, but still remains lightweight. Grunbacher and Hofer [11] complement XP with EasyWinWin requirements negotiation technique. They state that this approach has the following benefits: emphasis of shared vision, more complete stakeholder identification, full perspective for on-line customer and extensive stakeholder involvement in decision-making.

The literature identifies the storage and documentation of requirements as one problem area in XP. This activity can be included and partly automated in XP, in some form, but the challenge is not to jeopardize XP's intentions to remain agile.

### **3 Principal findings from an empirical case study**

This section introduces our findings based on the analysis of empirical case study. For details of this case study, see [1]. According to our study and experiences, the following issues were recognized especially important when considering the documentation and storage of user stories and tasks in XP.

First, XP provides the basic set of practices that should be adapted into the development situation. This means that practices are tailored for the purposes of a project and the organization, and they can be further changed on-the-fly on periodic process assessments, i.e. if they do not work. For example, some attributes in story and task cards were found unnecessary during the experiment. Thus, the tool support should not force detailed procedures but provide just enough basic abilities for defining project-specific user story and task attributes as well as abilities for storing, relating and retrieving user stories and tasks.

Second, the tool should allow the XP process to remain agile. This means that the tool should not jeopardize XP's intentions for open communication and lightweight management and documentation. Paper cards were used in experiment for story and task documentation. Story cards were placed onto the wall with their respective task cards. This was an efficient way of visualizing the dependencies between stories and tasks when operating co-located workspace. In XP methodology, a customer is always present in the project room and communicates with the project team. However, this is not always possible in real life. Thus electronic storage, management and distribution of story and task cards were considered important if the customer cannot always be present.

Third, requirements management tool support should be integrated into the project's overall development environment. This allows the project team to operate via one channel from a user story definition, through implementation, up to testing. The integration should also store and manage all project-related information under the appropriate project. This means that the project personnel can easily find all relevant project-related data from the system.

### **4 Integrated tool support for the management of user stories**

In this section, we introduce our solution for the management of user story and task cards called StoryManager. In specific, the proposed solution has been integrated in Eclipse –environment (Eclipse project [8]). Eclipse is a development environment and a tool integration framework found suitable for XP development. Eclipse –environment was used successfully in VTT's XP case study [1]. Tools are integrated through plug-ins into Eclipse. Currently there are hundreds of plug-ins available. In VTT's experiment, the Eclipse environment contained integrations with CVS (Concurrent Versions System), JDT (Java Development Tooling) and Junit (Testing framework). This framework complemented with StoryManager extension enables integrated tool environment for XP-based development from stories through implementation up to testing. Eclipse allows the user to easily navigate between tool perspectives during product development (Figure 1).

Our intention has been to remain agile since any new solution or practice should not jeopardize the fundamental idea of adaptable and lightweight processes. We maintain that the solution proposed is flexible enough to be tailored for an individual XP project with the respects of a project's needs. On the other hand, it integrates the basic functionality needed for adequate user story and task definition and management in the integrated development tool framework (Figure 2). Integrated tool framework enables the project team to work with one channel throughout the whole development life cycle. During planning game, the team works through StoryManager plug-in. Stories and tasks are stored into MySQL relational database. During implementation and

testing, the team works, e.g. through JDT plug-in and JUnit view. From information management point of view, our approach includes support for requirements management and configuration management. Requirements management (StoryManager) is used in this environment to share and manage user stories and tasks. On the other hand, configuration management (CVS) is used to manage and share code and other documentation.

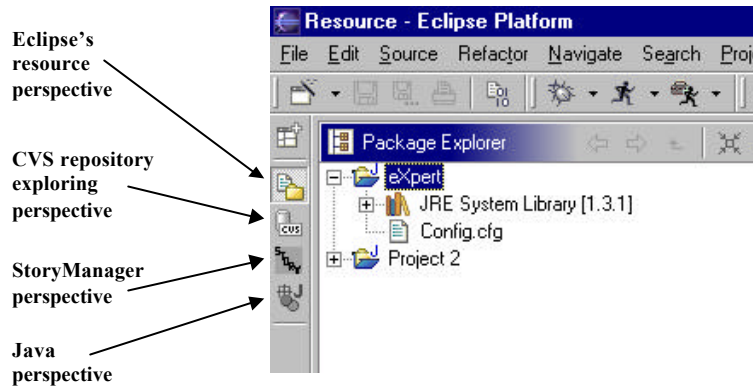


Figure 1: Navigation between perspectives in Eclipse

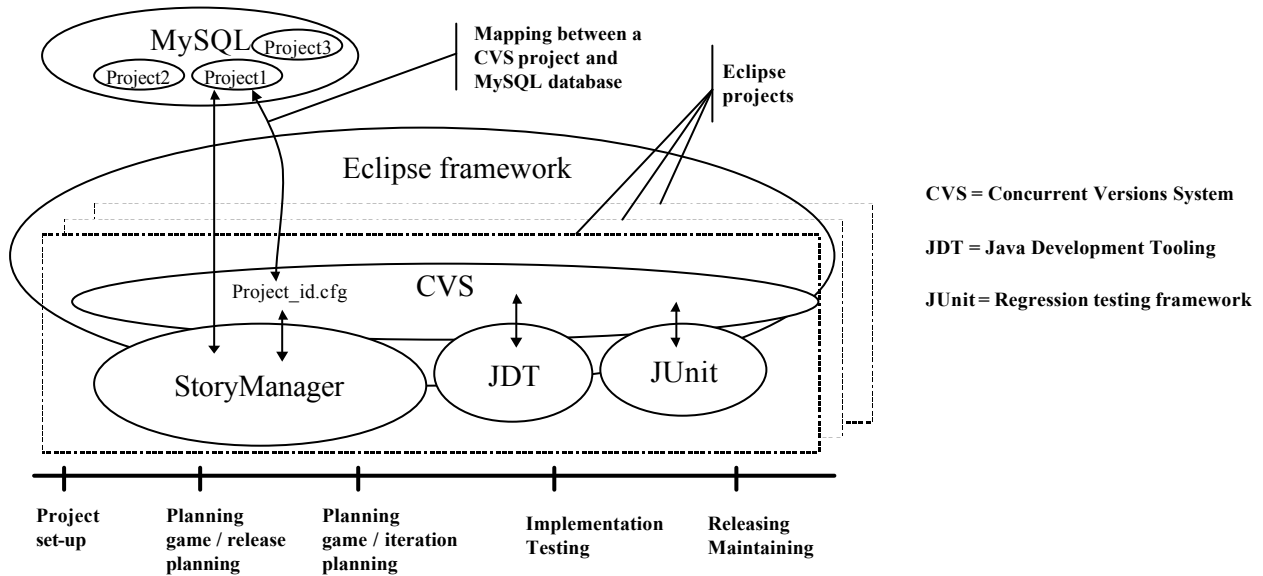


Figure 2: StoryManager as part of integrated tool environment for XP development

The introduction of StoryManager facilities is divided into process steps. Each process step introduces relevant application facilities and explains their usage. Phase "Project set-up" is used to describe activities that need to be performed before planning game:

Project set-up

First, the relevant Eclipse project is selected or created. Attribute definition facility is used to define user-defined attributes for stories and tasks. This facility allows flexible story/task attribute definitions according to the needs of the project (Figure 3). Then StoryManager creates project-specific database under the Eclipse -project. The program creates project-specific configuration file (CFG-file) which will be stored in CVS into project-root (Figure 2). The CFG-file indicates for StoryManager the correspondent MySQL database with the respect of the selected CVS project.

Planning game / release planning

First during the Planning Game -phase, a customer defines stories in cooperation with a project team. The program enables to fill-in story/task cards according to project-specific attributes. The program's AutoID facility is used to create automatically unique identifiers (ID) for stories (and tasks). This facility corresponds with requirements identification activity. Certain attributes are mandatory including:

- ID: automatically created by the system (story\_XXX, task\_XXX)
- Status: Defined, Implementing, Done, Postponed (color codes are used in tree-view to indicate status)
- Release: Not specified, 1, 2, 3, ...
- Iteration: Not specified, 1, 2, 3, ...
- Description: actual text for story / task

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 appropriate. However, the application stores a version history from the item (story/task) which can be used to examine the history of the item (Figure 3). Release and Iteration -attributes illustrate the selection of items for certain release and for certain iteration. Actually this corresponds with requirements baselining facility indicating agreed set of items for certain release and iteration.

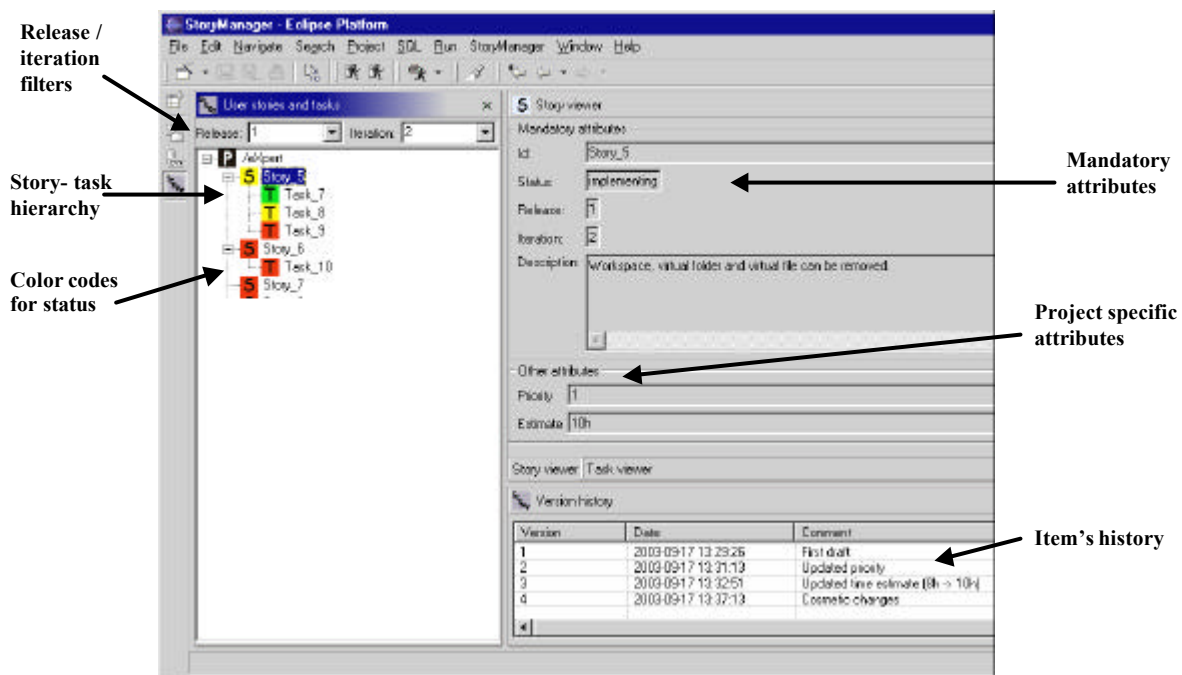


Figure 3: StoryManager main perspective.

#### Planning game / iteration planning

During iteration planning, the certain set of stories is selected for next iteration. This will be illustrated using Iteration -attribute. Then tasks are defined for next iteration. Mandatory attributes and main facilities are same for story and task cards, but optional attributes can vary. In addition to this, the task can be related under a story (traceability between stories and tasks) (Figure 3). In this case, the program stores the linking information in MySQL Link-table. However, a task can be created also under a project root and related afterwards.

#### Process phase independent facilities

Certain facilities are needed despite of a process phase. These facilities include the check out/in, views and reporting. Parallel story/task modification has been disabled to avoid uncontrolled changes when someone is modifying a story or task. When a user check out a story or task from database for modification, the system locks the story or task and it cannot be modified or deleted by others. When the user checks in the item, the system unlocks it and after that other users can modify it. Views are used to describe StoryManager database's content for a user (retrieving information) (Figure 3). The basic StoryManager perspective contains tree-view containing hierarchical story/task structure, story/task -content view and history view. Hierarchical story/task structure can be filtered by using release and iteration attributes. Finally, reporting facilities are used to create reports from the contents of database. This facility can be used if stories and tasks need to be printed and put onto the wall. Currently the following reports are possible: all stories, all tasks, stories and corresponding tasks, stories and tasks based on selected release and iteration (baseline).

## 5 Conclusions and future research

Requirements management and configuration management are only implicitly addressed by XP. Requirements management is needed to ensure that requirements are identified, traceable and all changes to requirements are properly handled. However, in a dynamic and fast moving XP project, traditional RM may tie up too much resources. In this paper, we introduce an approach for supporting requirements engineering in XP by managing user stories and tasks, called StoryManager. Traditionally this has been done manually using story and task cards. Our aim was to consider requirements management in XP and provide integrated, yet lightweight, approach for user story and task management. Integration was carried out in Eclipse –environment. The advantages of solution are the following.

First, the approach integrates all information from requirements (stories) through implementation up to testing under the appropriate Eclipse –project. Users can operate through one environment that store and control project-related data in electronic format including stories, tasks, code and other documentation.

Second, StoryManager allows you to define project-specific attributes for stories and tasks. This is a very simple way to define templates to collect minimum information about stories and tasks with the respect of each project.

Further development of solution includes the validation of the first version of the solution. Further research should also consider integration between stories and implementation/testing. The validation will analyze the applicability of solution for XP and verify if the solution remains lightweight. Bureaucratic and complicated solution just jeopardize simplicity. Empirical validation of the proposed system is currently ongoing. The verification is carried out in the XP experiment project developing mobile application software in Eclipse environment. The experiment project works according to XP practices, but the customer is off-site. Thus, the customer can follow the implementation of stories from the remote office using StoryManager.

The limitation of our solution is that the StoryManager is not a stand-alone program, but it has to be used in the context of Eclipse environment. Therefore, it dictates the development environment. Also the reporting facilities and visualization of user stories and tasks is challenging.

## Acknowledgments

The work was funded by the Technical Research Center of Finland (VTT) and by the National Technology Agency (TEKES). This work has been part of the ITEA project called MOOSE (<http://www.mooseproject.org>) and VTT's strategic research project called AGILE (<http://agile.vtt.fi>).

## References:

- [1] Abrahamsson, P.: Extreme programming: First results from a controlled case study. Euromicro 2003, Antalya, Turkey, 2003.
- [2] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods: Review and analysis, VTT Publication 478, Espoo, 2002.
- [3] Abrahamsson, P., Warsta, J., Siponen, M., Ronkainen, J.: New directions on agile methods: A comparative analysis. International Conference on Software Engineering (ICSE25), Portland, Oregon, USA, 2003.
- [4] Beck, K.: Extreme Programming Explained: Embrace Change. Reading, Massachusetts: Addison-Wesley, 1999.
- [5] Beck, K., Fowler, M.: Planning extreme programming. Addison-Wesley, 2000.
- [6] Breitman, K., Leite, J.: Managing User Stories. International Workshop on Time-Constrained Requirements Engineering, TCRE 02, 2002.
- [7] Cockburn, A.: Agile Software Development, Boston, Addison-Wesley, 2002.

- [8] Eclipse project.: <http://www.eclipse.org/>. Available in 29th September 2003.
- [9] Gotel, O.: Contribution Structures for Requirements Traceability, Ph.D. Thesis, Imperial College of Science, Technology and Medicine, University of London, August, 1995.
- [10] Gotel, O., Finkelstein, A.: An Analysis of the Requirements Traceability Problem, Proceedings of the First International Conference on Requirements Engineering, pp.94-101, 1994.
- [11] Grunbacher, P., Hofer, C.: Complementing XP with Requirements Negotiation. XP2002.
- [12] Kotonya, G., Sommerville, I.: Requirements Engineering: Process and Techniques, John Wiley & Sons, 1998.
- [13] Leffingwell, D., Widrig, D.: Managing Software Requirements - A Unified Approach, Addison-Wesley, 2000.
- [14] Lippert, M., Roock, S., Wolf, H.: eXtreme Programming in Action. John Wiley & Sons, 2002.
- [15] Maurer, F., Martel, S.: Process Support for Distributed Extreme Programming Teams, ICSE 2002 Workshop on Global Software Development, 2002.
- [16] Nawrocki, J., Jasinski, M., Walter, B., Wojciechowski, A.: Extreme Programming Modified: Embrace Requirements Engineering Practices, 10th IEEE Joint International Requirements Engineering Conference, RE'02 Essen, Germany, September 2002.
- [17] Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development. IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2003 / KMDAP2003), June 9-11, Austria, 2003.
- [18] Paulk, N.: Extreme programming from a CMM perspective, IEEE Software , Volume: 18 Issue: 6, Page(s): 19 -26, Nov/Dec 2001.
- [19] Pinheiro, F.: Requirements Honesty. International Workshop on Time-Constrained Requirements Engineering, TCRE 02, 2002.
- [20] Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practise Guide. John Wiley & Sons, 1997.
- [21] Wagner, L.: Extreme Requirements Engineering. Cutter IT Journal, Vol. 14, No. 12, December 2001.