Journal of Theoretical and Applied Information Technology

www.jatit.org

A METHOD FOR REQUIREMENTS MANAGEMENT IN DISTRIBUTED EXTREME PROGRAMMING ENVIRONMENT

[1]ALI AKBAR ANSARI, [2]SAYED MEHRAN SHARAFI, [3]NASER NEMATBAKHSH

[1]Computer Department, Islamic Azad University, Najafabad branch, Esfahan, IRAN

[2]Computer Department, Islamic Azad University, Najafabad branch, Esfahan, IRAN

[3]Computer Department, University of Isfahan, Faculty of Engineering, Esfahan, IRAN

ABSTRACT

Both Distributed Software Development (DSD) and eXtreme Programming (XP) approaches have gained significant Popularity. Both DSD and XP method are growing trends as software business requires quicker quality production at a cheaper price. Requirement Management (RM) is not easy to perform even under the best of circumstances and it becomes more difficult when performed globally because of the nature of distributed development projects and the diversity of stakeholders. This article presents an investigation of the possibility to integrate Requirement Management in Distributed eXtreme Programming. One reason for integrating RM with DXP was that XP emphasizes lightweight documentation in XP based development. We propose the development of model and tool for executing RM in DXP that are required for proper software engineering and that activity should not restrict the efficient execution of XP method in DSD.

Keywords: *Distributed Software Development (DSD), eXtreme Programming (XP), Requirements Management (RM).*

1. INTRODUCTION

Many reports highlight the importance of good requirements engineering (RE). The CHAOS report published in 1995 [22] shows that almost half of the cancelled projects failed due to a lack of requirements engineering effort and that a similar percentage ascribes good requirements engineering as the main reason for project success. Successful projects do manage requirements, failed ones lack requirements processes. On the European side, a survey with over 3800 organizations in 17 countries similarly concluded that most of the perceived software problems are in the area of requirements specification and requirements management [5].

Globally distributed work is taken up as an alternative to single-site mainly because of the economic and strategic benefits it offers. Distributed software development is becoming the norm by promising potential advantages like global resources, attractive cost structures, round-the-clock development and closeness to local markets [7]. Developing software in distributed teams has brought about its own unique set of problems. requirements management in global projects is one of the essential challenges that shall be paid adequate attention. Organizations need to effectively define and manage requirements to help ensure they are meeting customer needs, while addressing compliance and staying on schedule and within budget.

Agile software development refers to a group of software development methodologies aiming to more nimble and lighter development processes, making them more responsive to change. There are studies indicating that it is possible to successfully combine agile methods with distributed projects [18, 17, 20, 16, 21, 15, 19].

Extreme Programming (XP) [1] is undoubtedly the hottest Agile approach to emerge in recent years. the main weaknesses of the XP approach to requirements management is the lack of requirements documentation.

The objective of this paper is finding a solution for managing user stories and tasks in electronic format for change management and maintaining traceability.

This paper is organized as follows: The next section briefly gives an overview on Distributed Software Development, Extreme Programming, and Requirement Management (RM). Section 3 describes about elicitation and managing requirement in XP. In section 4 RM related problems with XP are defined. Section 5 describes our main approach and the proposed solution and finally section 6 concludes the paper.

2. BACKGROUND

This section introduces the concepts of distributed software development, extreme programming, and requirements management.

2.1. DISTRIBUTED SOFTWARE DEVELOPMENT

Many organizations turn toward distributed software development, software development distributed beyond national borders, in an attempt to produce cheap higher-quality software with the shortest development cycle possible [3]. Distributed development means co-operation between several teams located at different sites. This includes large software companies developing a single product out of many parts where each part could be built at a separate location.

Distributed software development is becoming the norm by promising potential advantages like global resources, attractive cost structures, round-the-clock development and closeness to local markets [7]. The promises are intuitive. To unleash the potential, methods and tools for distributed software development are designed to enable geographically dispersed team members to share programming tasks and development practices [6].

The many challenges of DSD are, perhaps obviously related to the effects of increased distance between people. *Distance* has been identified as a key problem and by its very nature introduces barriers and complexity into the management of globally distributed projects. The distance factor involved [23] in three dimensions - geographical, temporal, and socio-cultural poses challenges to communication, coordination as well as control.

2.2. EXTREME PROGRAMMING

Extreme Programming (XP) [1] is undoubtedly the hottest Agile approach to emerge in recent years. XP addresses issues of changing requirements and their cost by simplifying management tasks and documentation. The goal of XP is to produce the software faster, incrementally and to produce satisfied customer [1]. XP is a collection of values, principles and practices to maximize the software quality. It defines some practices, in particular related to requirements elicitation and coding phases in order to make the process lighter and changes adaptable, as well as informal and customer oriented. These practices are organized in "feedback cycles", the most relevant are the phase of requirements gathering ("User Stories"), coding (Pair Programming and Refactoring) and testing, during the development (TDD - Test Driven Development) and validation by the customer (Acceptance Test).

In XP, Development starts with planning game. Planning game can be divided into “release planning” and “iteration planning” [2]. During the planning game, the customer writes user stories. Those cards are estimated by the developer, based on those estimation customer priories them depends on their needs to establish a timebox of an iteration. Developers develop those story cards through pair programming and test driven development. At last customer provides acceptance test to accept the developed functionality. In between they consider all of the XP practices in mind to improve the quality of the software.

2.3. REQUIREMENT MANAGEMENT

Requirements Management (RM) activities are understood to begin before actual requirements engineering process phases and continuing during design, implementation, testing and maintenance phases [11]. On the other hand, Requirements Management means “the systematic process of organizing and storing relevant information about requirements, while ensuring requirements traceability, and managing changes to these requirements during the whole lifecycle of the information system” [13]. Requirement Management includes activities related to maintenance, namely identification, traceability and change management of requirements.

Requirements identification is an essential pre-requisite for requirements management. It

focuses on the assignment of unique identifier for each requirement [11]. These unique identifiers are used to refer to requirements during product development and management.

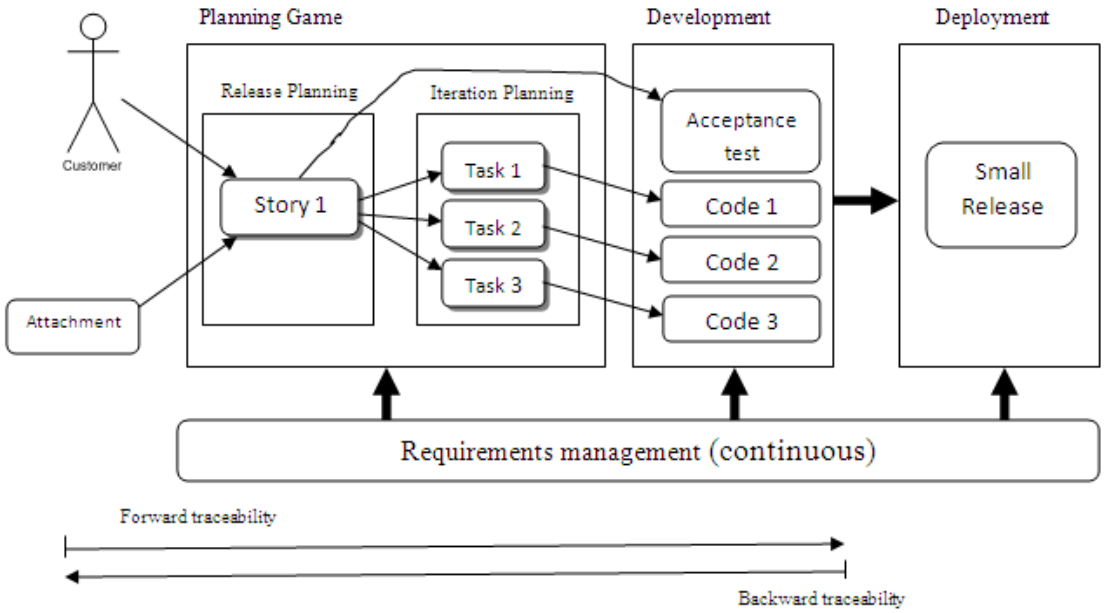

Fig. 1.RE information in XP process

Requirements traceability [9, 10] refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases). Forward traceability is the ability to trace a requirement to components of a design or implementation. Backward traceability is the ability to trace a requirement to its source, i.e. to a person, institution, law, argument, etc [12].

Requirements change management refers to the ability to manage changes to the systems requirements [11]. Requirement change management process defines the set of activities that need to be performed when there are some new requirements or changes to existing requirements.

3. ELICITATION AND REQUIREMENT MANAGEMENT IN XP

Requirements elicitation activity is done during planning game and responsibility of this activity lies largely on customer shoulders. XP employs the use of unstructured requirements gathering techniques referred to as User Stories. User Stories are one of the important aspects of the XP. They are playing vital role in XP. User stories are informal, natural language descriptions of system features that are written on an index card. User stories are composed of three aspects [14]:

- A written description of the story used for planning and as a reminder.
- Conversation about the story that serves to flush out the details of the story.
- Tests that convey and document details and that can be used to determine when a story is complete.

Stories can be decomposed into tasks, quantifiable units of development effort. The decomposition is made by the programmers that also have to estimate how long it would take to implement each task. System development is a succession of such iterations where the requirements are continuously being defined by means of stories.

Today As projects become larger, Distributed and more complex, traceability and change management becomes increasingly difficult to maintain. In small project using XP practices, the lack of documented can be overcome by asking other team members about a particular task, in essence, echoing the original conversation. This solution becomes less practical in distributed development environment. The lack of requirement document causes problems especially when managing changes to requirements and maintaining traceability.

Different objects that relate to requirements and their relations are illustrated in Figure 1. Actors are

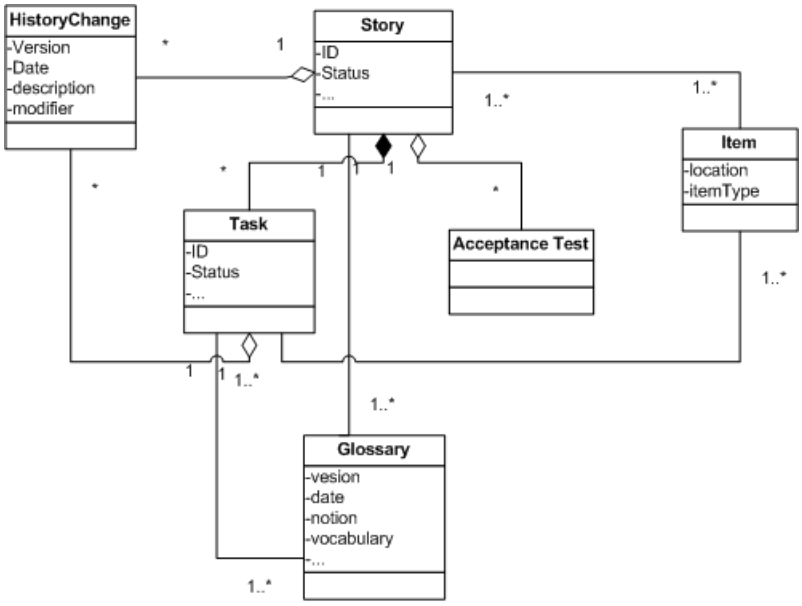

Fig. 2. UML Class diagram of proposed solution

stakeholders. Arrows represent traces or links between items and planning game phase represent information that is stored in tool. Rounded cornered and shaded boxes are objects that contain RE information. Rounded corner boxes in Development phase are implementation or design objects. There also can be hierarchical relations between stories which are also forward traceability relations. Only modification is that stories are written to automated tool. If customer provides source documentation for stories, they are also stored into tool environment. All stored items (story/task) should be under version control. However, in XP only the last item version is relevant. Each user story can be linked to source of user story which can be some document or most often customer. Customer should prioritise all stories and with the help of development team they should identify and select core set for the first iteration. When the team starts coding and selects tasks they also maintain links between tasks and code by selecting related implementation components in tool environments and by assigning them for task.

When stories are ready (all tasks related to that story are ready) and acceptance tests for that story is written link is made between story and acceptance test script.

4. CHALLENGES FOR REQUIREMENT MANAGEMENT IN DXP

According to Beck, once the stories (and their task decomposition) are used, they are to be discarded. This corresponds to the *Embracing change and travel light* concept. In the author's idea there is no need to save the stories once they had their impact on the code because it (the code) is bound to change anyways. In DSD, This is not necessarily good practice [4]**.** In XP there are three sources of knowledge about the software to be maintained [8]: code, test cases, and programmers' memory. If the software remains unchanged for a long enough period of time, the programmers will forget many important things and - what is even worse – some of programmers might get unavailable (for instance they can move to another company). Then, the only basis for maintenance is code and test cases. That can make maintenance hard. It would be easier if one had the requirement documented.

On the other hand, even though stories are the first artifact created by XP projects, the stories are

often the result of a problem statement, request for proposal, or a contract with the customer. Stories often times, after the review process, restructured to another form of requirements artifact. At the end, when a team member is looking at a specific development task, they may lack the understanding and background information to complete the task.

In extremely distributed projects, a team member may not have been present for the conversation, or simply not recall all the information about a particular task. So, Access to documented is indeed very valuable. Hence, an approach is needed that integrates knowledge management and DXP.

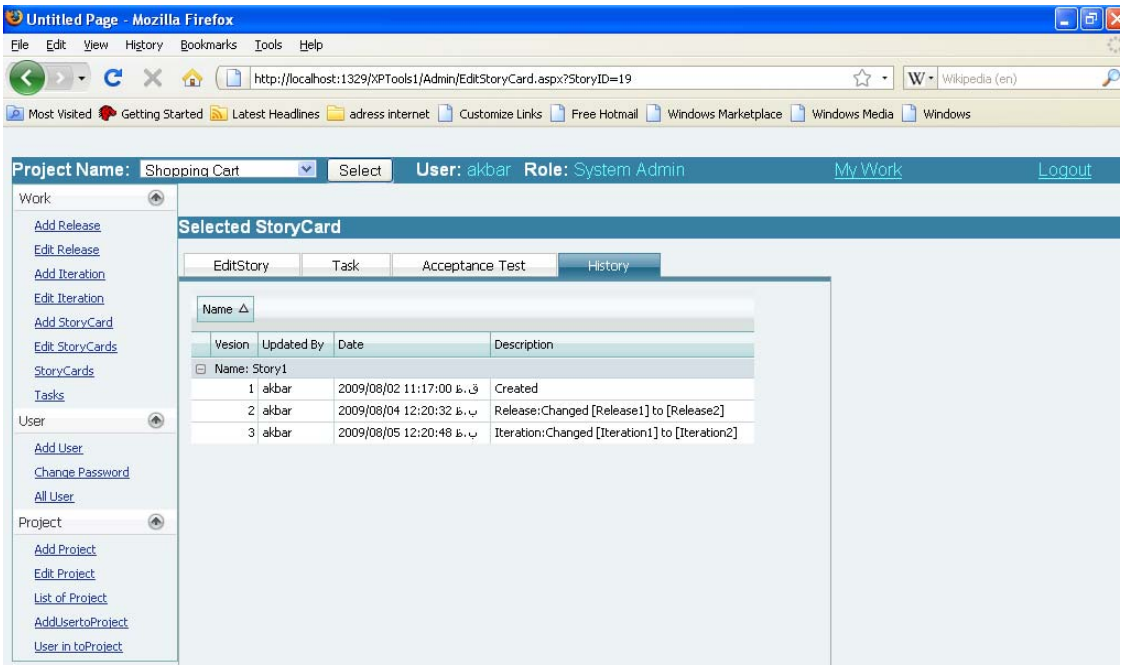

Fig. 3.Item history

5. PROPOSED MODEL AND TOOL

In this section, we introduce our solution for the management of user story and task cards called M.R.D.X.P. (Managing Requirement in a Distributed eXtreme Programming). M.R.D.X.P. was developed to offer the customer and developers a RM tool that is lightweight and visible yet some of the

XP practices. M.R.D.X.P. needed to be easy to use, fully distributed, accessible and have equal facilitation for both sets of users.

It was developed using ASP and Ajax to ensure it could be fully distributed across the Web.

In tool framework which is proposed here, user stories are managed in a database. Tool provides simple version management for user stories with the possibility to make baselines and browse version history. Tool also stores tasks in the same way that it stores user stories. Attributes of stories or tasks are almost entirely up to user to define. State(Defined, In-Progress, Completed, Accepted, Blocked) of tasks are explicitly defined by the tool.

In Figure 2 there is presented data model of proposed system as UML class diagram. Traceability is implemented by storing information about traced items into database. Item entity has multi value attribute item type, which can be one of these: Task, Story or File. Actual traces are stored in link tables which is constructed relationship between item and story or task. Location attribute stores Items identification which can be for example path in tool workspace.

HistoryChange table stores information relevant to version management. Version management in system is linear and on only the last version is preserved. So the purpose of HistoryChange table is gather and provide history information. Thus accessing past versions of user stories is not possible and therefore branching is not possible. Entry is added to History- Change whenever user story or task is modified. HistoryChange stores date and version number of related requirement (Figure 3). Item and HistoryChange entity and their

attributes are managed by tool and used do not need to worry about them.

User Story and Task tables contain information of stories and tasks. Only ID field and multi-value state field are mandatory. ID field is incremental ID of task or story. This way tasks and stories always have unique identifier. Other fields are user definable and used when needed.

Glossary table stored information relevant to result of discussion about user stories between customer and developer as well as vocabulary used by the customer (business). Creation of glossary, guaranteeing that both developers and customer share a common understanding.

6. CONCLUSION

Requirements management is needed to ensure that requirements are identified, traceable and all changes to requirements are properly handled. However, in a Distributed and fast moving Enviroment XP project, traditional RM may tie up too much resources.

Process proposed here resembles original XP approach in great extend with the addition of tool support for managing requirement engineering related information. Tool support makes persistent requirements documentation possible and accessible. Tool also gives rigor for capturing traceability information that would be otherwise lost.

REFERENCES:

[1] Beck, K., "Extreme programming Explained Embraced Change", Reading, MA: Addison-Wesley, 2000.

[2] K. Beck and M. Fowler, "Planning extreme Programming", New York: Addison-Wesley, 2001.

[3] Moe, N. B. & Smite, D., “Understanding lacking trust in global software teams: A multi-case study”. Lecture Notes in Computer Science, 2007.

[4] Breitman, K., Leite, J. C. S., "Managing User Stories", 10th IEEE Joint International Requirements engineering Conference, RE'02 Essen, Germany, September 2002.

[5] European Software Institute, European User Survey Analysis, Report USV_EUR 2.1, ESPITI Project, January 1996.

[6] Canfora, G., Cimitile, A., Di Lucca, G. A. & Visaggio, C. A., "How distribution affects the success of pair programming", International Journal of Software Engineering and Knowledge Engineering, 2006.

[7] Damian, D. & Moitra, D., "Global software development: How far have we come?", IEEE Software, 2006.

[8] Nawrocki, J., et al., "Extreme Programming Modified: Embrace Requirements Engineering Practices", 10th IEEE Joint International Requirements Engineering Conference, RE'02 Essen, Germany, September 2002.

[9] Gotel,O. , Finkelstein, A. ,"Contribution structures", Proceedings of RE’ 95, 2nd International Symposium on Requirements Engineering. York, England. Los Alamitos, California: IEEE Computer Society Press, 1995.

[10] Gotel O. and Finkelstein A., "Revisiting requirements production", Software Engineering Journal, 1996.

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

[12] Wieringa, R.J., "An introduction to requirements traceability", Technical Report IR-389, Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam, September 1995.

[13] Grehag, Å., "Requirements Management in a Life-Cycle Perspective - A Position Paper", In Ben Achour-Salinesi, C., Opdahl, A.L., Pohl, K. and Rossi, M. (Eds) Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ’01, Interlaken, Switzerland. Essenere Informatik Beiträge, 2001, pp. 183-188.

[14] Cohn, M., "User stories applied for Agile Software Development", Reading, MA: Addison-Wesley, 2003.

[15] J.Sutherland, A.Viktorov, J.Blount, N.Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams", IEEE International Conference on System Science, 2007.

[16] K.Sureshchandra, J.Shrinivasavadhani, "Adopting Agile in Distributed Development", IEEE International Conference on Global Software Engineering 2008, p.217-221.

[17] M.Simmons, Internationally Agile, InformIT March 15th , 2002.

[18] M.F.Nisar, T.Hameed, "Agile Methods handling Offshore Software Development Issues", Proceedings of INMIC, 8th IEEE

International Multitopic Conference 2004, p.417-422.

[19] M.Kircher, P.Jain, A.Corsaro, D.Levin, "Distributed eXtreme Programming", Proceedings of the International Conference on eXtreme Programming and Agile Methods 2004, p.147-154.

[20] M.Fowler, "Using an agile software process with offshore development", 2006. (Available: http://martinfowler.com/articles/agileOffshore.html)

[21] M.Farmer, "DecisionSpace Infrastructure: Agile Development in a Large Distributed Team", Proceedings of the Agile Development IEEE Conference 2004, p. 95-99.

[22] CHAOS, Software Development Report by the Standish Group 1995.

[23] Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., Ó Conchúir, E. "A Framework for Considering Opportunities and Threats in Distributed Software Development", In: Proceedings of the International Workshop on Distributed Software Development, Austrian Computer Society 2005, p. 47–61