World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:1, No:4, 2007

Requirements Management in a Distributed Agile Environment

Paul Prior, Frank Keenan

***Abstract*—**The importance of good requirements engineering is well documented. Agile practices, promoting collaboration and communications, facilitate the elicitation and management of volatile requirements. However, current Agile practices work in a well-defined environment. It is necessary to have a co-located customer. With distributed development it is not always possible to realize this co-location. In this environment a suitable process, possibly supported by tools, is required to support changing requirements. This paper introduces the issues of concern when managing requirements in a distributed environment and describes work done at the Software Technology Research Centre as part of the NOMAD project.

***Keywords*—**Agile, Distributed, Requirements Management, XP.

I. Introduction

Many reports highlight the importance of good requirements engineering (RE). McPhee [1] suggests that requirements activities should account for 25% of the total development effort. The Standish Group [2] survey found that incomplete requirements (12.3%) and changing requirements and specification (11.8%) were significant contributors to project failure. Also, Taylor [3] reports that 70% of projects failed at the requirements definition stage and 80% also claimed that clear and detailed requirements were a critical success criterion.

Recently Agile Methodologies (AMs) [4] have become established as an approach for software development. A survey conducted by Shine Technologies [5] highlighted that 92.8% of respondents felt AMs had made their team more productive and 84% saw an increase in the quality of products delivered. Williams et al [6] also highlights the potential improvements that can be made by using Extreme Programming (XP). Holz et al [7] describe Agile approaches as methodologies that seek to reduce documentation overhead while increasing informal communication between team members. Although different approaches exist, they all subscribe to the Agile Manifesto [8]. This promotes communication and collaboration and the ability to respond to change. However, greater customer participation is expected.

As AMs encourage a high degree of customer collaboration, the impact of inconsistencies and misinterpretations that can contribute to project failure is decreased. However, teams often find themselves distributed across many buildings, towns, continents or even time zones [9][23]. Organizations using AMs need to adapt to this situation [24]. Currently Agile practices work best in environments that are co-located and although some work has been done to support a distributed deployment [10][11] challenges still exist.

The NOMAD research project involves three partners, the Software Technology Research Centre (SToRC), the Telecommunications Systems & Software Group (TSSG) and the Institute of Art, Design and Technology in Dun Laoghaire (IADT-DL). The purpose of the NOMAD project is to develop wireless tools for use in a student environment. Two sites worked as developers (SToRC and TSSG) and the third acted as customer, thus providing an opportunity to develop a knowledge sharing process solution for distributed development. A trial development and deployment has taken place in late 2003/early 2004 with a second trial scheduled for early 2005.

This paper gives an overview of the RE and AMs adopted. It further describes work conducted between February 2004 and January 2005 which describes one such distributed process at work, resulting in the production of a Requirements Management (RM) tool for distributed environments.

II. Requirements Engineering

RE can be described as the task of capturing, structuring, and accurately representing user requirements so that they can be correctly embodied in systems which meet those requirements [12]. Traditionally the RE process consists of five key stages [13]: Feasibility Study (determining the validity of entering into a development cycle), Elicitation and Analysis (dealing with the gathering and refining of requirements), and Specification and Validation (specifying the requirements in a readable format and ensuring their credibility). It is essential that any approach to RE should enhance a development teams ability to elicit the correct requirements from the customer, manage these and in collaboration develop the required product.

RM is essential to successful RE. It is the process by which a system to notify the people involved that the requirements are being addressed is established [28]. It is concerned with

Manuscript received January 11, 2005. Paul Prior is with the Software Technology Research Centre, Dundalk, Co.Louth, IRE (phone: 353-879665296, e-mail: paul.prior@dkit.ie).

Frank Keenan, is a lecturer in Software Development with Dundalk Institute of Technology (DKIT) at Dundalk, Co.Louth IRE. He is also part of the Software Technology Research Centre and the Irish Software Engineering Research Consortium (ISERC). (e-mail: frank.keenan@dkit.ie).

ensuring the state of requirements is visible at all times and that progress in dealing with these is apparent to both customers and developers.

Tool support is often provided to assist RE activities. Wieringa and Ebert [14] describe the most popular used today. Some such as Analyst Pro [15], C.A.R.E [16] and DOORS [17] are RE tools that provide features such as requirements classification, traceability, configuration management and analysis. Others, such as Reqtify [18] and RTM Workshop [19] are RM tools and include features such as change notification, source locking and impact analysis.

Tools used with AMs are lightweight and easily attainable. Cockburn [20] describes some lightweight tools that can be used; these include whiteboards, digital cameras, poster sheets, index cards and post-it notes.

III. The XP Planning Game

XP [21] is one of the more widely adopted AMs and was chosen as the basis for the NOMAD process**.** It is built on well defined practices designed to reduce documentation to an adequate amount while increasing customer feedback and communication. Although it is recommended that all practices be implemented, some are more relevant depending on the working environment. The principle that most encapsulates the elicitation and sharing of requirements is the Planning Game. The metaphor of a Game is used for ease of explanation, which can be described as follows.

A. Game

Each planning game has a goal, pieces, players and rules much like a game of chess. It is important that each part interacts successfully to get the best result.

1) Goal: The goal of the game is to put the maximum amount of value into a requirement.
2) Pieces: The major playing piece in the game is a user story described on an index card. Each requirement is written down and has an associated business value, cost and acceptance test.
3) Players: Each player has a role, however the most common roles are that of developers, business or customer, each has expert knowledge of their own domain.

B. Moves

The moves within the game are designed to obtain the best result. The steps involved are as follows.

1) Write Story: A story although containing enough value to assign a cost, is a commitment for further conversation and is traditionally written on an index card [22].
2) Estimation: Developers estimate the difficulty of stories. At this stage stories can be merged or split into different stories, as business and the developers feel fit.
3) Commitment: The estimated stories are prioritized and committed to an iteration. This can be driven by release date or by the story itself.
4) Change Management: This is initiated if business changes the value of a story, if a story is split, if development has overcommitted on a story or a new or temporary story is introduced. This would result in a re-estimation of the story and therefore a re-commitment.

The Planning Game allows for the capture, storage and exchange of knowledge. However, change management does not solely take place within the planning game. As XP is built on principles such as constant feedback it facilitates change through flexible iterations and allows for the re-prioritization of requirements based on changing customer preference. XP also uses acceptance testing to validate requirements. If a customer does not accept a requirement this can also result in a re-estimation and evaluation of the requirement in question.

IV. The NOMAD Process

The process in use within NOMAD merges the Planning Game with concepts of knowledge responsibility, RM and constant communication. The eight main activities of this process are as follows.

A. Feasibility Study

During this activity a brainstorming session takes place in which knowledge, experience, ideas and preconceptions are shared. An overview of the system in question in the form of a system metaphor is generated.

B. Requirements Elicitation

Both developers and customers participate in this stage of the process, with the customer defining acceptance tests for each requirement specified in the iteration. This is a simple paragraph explaining the minimum requirements that need to be met before the requirement can be released. A combination of prioritization and conversation is used to elicit, refine and prioritize the requirements. A maximum of 10 requirements are baselined at a high-level and these should be sufficient to cover the project. The outputs from this stage are the baselined high-level requirements.

C. Specification

The customer details the requirements via user stories and conversation in the specification stage. This allows for the knowledge to be organized in a manner suitable for estimation. The outputs to this stage are the refined, specified requirements.

D. Estimation

Each iteration is estimated by developers using a range of techniques tailored for the particular project that is currently being worked on. This estimation is forwarded to the customer and it is they who ultimately give the iteration the go ahead. If a change is made the iteration is re-estimated by development and evaluated by the customer for re-scheduling. The output to this stage is the estimated iteration.

E. Iteration

An iteration consists of the cyclic use of simple design, using pen and paper, followed by test first development. The customer may also wish to refine acceptance tests at this point.

The output from this stage is an operational prototype and acceptance tests ready for inspection.

F. Release Planning

Upon completion of the iteration, the customer evaluates the work to ensure it is consistent with that specified in the user story and resulting acceptance test. If the developers have been successful in their interpretation and management of the knowledge then the code can go to integration, otherwise, it is re-estimated and scheduled for another iteration. Therefore, the output from this stage is a rejected or accepted iteration.

G. Integration

Integration of the accepted iteration takes place continuously involving each iteration. This allows the customer to see the project as it progresses and gives a visual indication to progress. CVS [25] is used as a shared knowledge repository, allowing both distributed teams to integrate their code simultaneously while source control makes sure mutual exclusion is ensured. The output from this stage is an integrated project.

H. Release

When accepted by the customer the complete project is tested and released. The output from this stage is a completed product.

As well as these distinct stages, a constant stream of communication is upheld from customers and developers using common tools such as messenger [26] and forum based discussion boards [27]. This line of communication is also used for discussion between our partners and SToRC. This insures a steady flow of knowledge between both distributed teams throughout the project lifecycle.

V. Process Evaluation

Following the initial NOMAD trial a questionnaire and interview session took place between the process team and the various developers involved in the project. This was designed as a basis to improve the process for the second trial.

The feedback received was honest and useful with most developers agreeing the process adopted had contributed significantly to the overall success of the project. It was determined that constant communication and share of knowledge via messaging and forums dramatically increased the prospect of delivering a product that the customer wants. However, some felt that face-to-face contact would also be needed as it generates a useful bond between team members. In addition, a greater breakdown of roles would be interesting as each customer or developer has specific knowledge in an area they are involved in. It was also suggested that the customer should be given some direction to spark interest or to give an indication as to what can be achieved. This would then allow the relevant knowledge to be extracted, eliminating time and effort costs further on in the lifecycle.

Ultimately it was decided an integrated tooling platform that tackled some of these issues would be most beneficial

VI. M.A.R.D.I

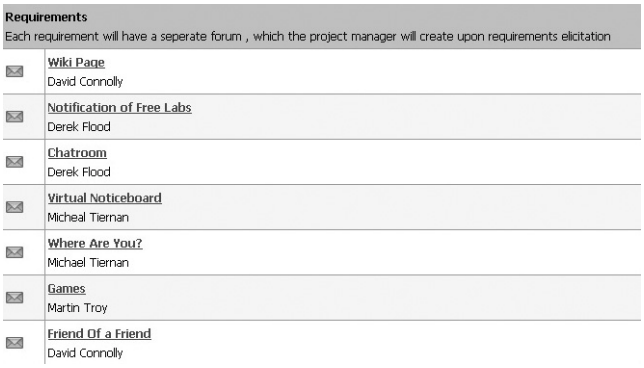
M.A.R.D.I. (Managing Agile Requirements in a Distributed Development Environment) is an evolutionary prototype that was developed to offer the customer and developers a RM tool that is lightweight and visible yet encapsulates some of the points highlighted above. The focus group in question was a combination of distributed developers and the NOMAD customer population (i.e. students at DKIT). With this in mind M.A.R.D.I. needed to be easy to use, fully distributed, accessible and have equal facilitation for both sets of users.

It was developed using JSP (Java Server Pages) and JavaScript to ensure it could be fully distributed across the Web. It uses an XML-based repository and includes facilitation for the following.

1) Prioritized Requirements Overview: The initial screen a user sees is the prioritized requirements, each with a name and unique ID. From here both the customer and developer has a complete view of the project.

2) *Real Time Discussion:* An overview of the requirements being discussed can be found in the overview page; however, a more detailed forum was developed for U.I. discussion and bug reporting.

FIGURE I
M.A.R.D.I Discussion Forum

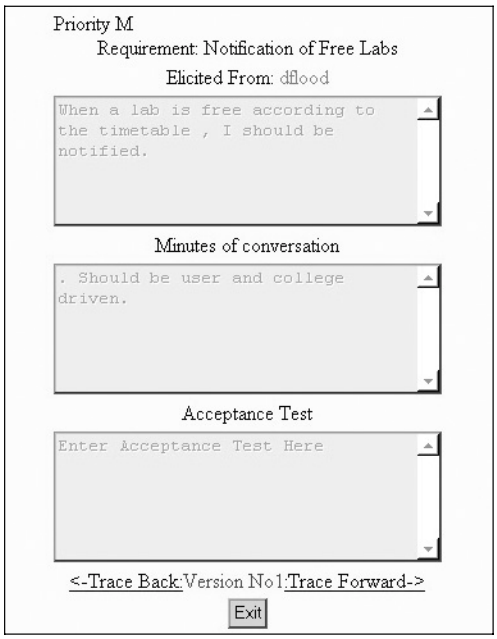


3) Requirements Locking: This allows a developer to freeze a requirement if they are working on it. No other developer can sign the requirement out or have access to change it. It is designed to enforce a sense of responsibility for the requirement and the knowledge it contains.

4) Change Management: Change in any project is inevitable, but it is the management of this change that is important. In M.A.R.D.I. change is user driven. It is the responsibility of the developer who has the requirement locked or the project manager to determine the feasibility and repercussions of any change that is suggested.

5) Version Control: This involves a process by which a requirement can be traced as it evolves throughout the project. It is available to all users to the system and saves a snapshot of a requirement each time a change is made.

FIGURE II
M.A.R.D.I Version Control



VII. Conclusions And Future Work

This paper has introduced issues of concern when sharing knowledge in a distributed development environment. It presents an overview of the process adopted for the NOMAD project. In particular it describes the M.A.R.D.I. prototype tool that has been developed for managing requirements in this environment. However, further research is needed into overcoming the difficulties with sharing knowledge with distributed teams. A second trial has been scheduled for early 2005 with results available in mid-2005.

References

[1] C. McPhee, A. Eberlein (2002) Requirements Engineering for Time-to-Market Projects, *Proceedings of the 9th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS2002),* Lund, Sweden

[2] Standish Group International, Chaos Report, [Online], Available: www.projectsmart.co.uk/docs/chaos_report.pdf, 1995

[3] A. Taylor (2000, Jan), "I.T Projects Sink or Swim", *The Computer Bulletin*, pp24-26

[4] M. Fowler (2003, Apr), The New Methodology, [Online] Available: www.thoughtworks.com/us/library/newMethodology.pdf

[5] Shine Technologies, Agile Methodologies Survey, [Online], Available: www.agilealliance.org/articles/reviews/ShineTechnologies1/articles/AgileSurvey2003.pdf , 2003

[6] L. Williams, W. Krebs, L. Layman, A. Anton (2004), Towards a Framework for Evaluating Extreme Programming, *Proceedings of the Empirical Assessment in Software Engineering (EASE),* Edinburgh, Scotland, Available: http://collaboration.csc.ncsu.edu/laurie/publications.html

[7] H. Holz, F. Maurer: Knowledge Management Support for Distributed Agile Software Processes. *Advances in Learning Software Organizations, 4th International Workshop, LSO 2002*, Chicago, IL, USA, August 6, 2002, Revised Papers. Lecture Notes in Computer Science 2640, Springer, 2003

[8] K. Beck, M. Beedle, A. VanBennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Greening, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schawber, J. Sutherland, D. Thomas, Agile Manifesto, [Online] Available: www.agilemanifesto.org, 2001

[9] P. E. McMahon (2001, Nov), "Distributed Development: Insights, Challenges and Solutions", *CrossTalk*, pp4-9 , Available: http://www.stsc.hill.af.mil/CrossTalk/2001.nov/mcmahon.asp

[10] F. Maurer, M. Sebastien, [Online] "Process Support for Distributed Extreme Programming Teams", Available: sern.ucalgary.ca/~milos/papers/2002/MaurerMartel2002a.pdf , 2002

[11] M. Kircher, J. Preshant, L. D. Carsaro, D. Levine "Distributed Extreme Programming", *eXtreme Programming and Flexible Processes in Software Engineering*, Xp2001, Villasimius, Sardinia, Italy, 2001

[12] www.Dictionary.com, [Online] , http://dictionary.reference.com/search?q=requirements+engineering&r=67

[13] I. Sommerville ,"Software Engineering" (6th ed), Addison-Wesley Professional, 2000, Available: http://www.awprofessional.com/bookstore/product.asp?isbn=020139815X&redir=1

[14] R. Wieringa,C. Ebert (2004, Mar/Apr), Practical Requirements Engineering Solutions [Online] , *IEEE Soft*, pp16-18. Available: http://csdl.computer.org/comp/mags/so/2004/02/s2toc.htm

[15] Analyst Pro, Available: http://www.analysttool.com

[16] C.A.R.E (Computer Aided Requirements Engineering), Available: www.sophist.de

[17] DOORS (Dynamic Object Oriented Requirements System), Available: www.telelogic.com

[18] Reqtify, Available: www.tni-valiosys.com

[19] RTM (Requirements and Traceability Management) Workshop, Available: www.chipware.com

[20] A.Cockburn (2004, Nov), What the Agile Toolbox Contains, Humans and Technology for Crosstalk, *Crosstalk*, pg4-7, Available: http://www.stsc.hill.af.mil/crosstalk/2004/11/0411Cockburn.html

[21] K. Beck, "Extreme Programming Explained: Embrace Change" (1st ed) , Addison-Wesley Professional, 1999, Available: http://www.awprofessional.com/titles/0-201-61641-6

[22] R. Jeffries (2001, Aug 30). "Essential XP: Card, Conversation, Confirmation", [Online], Available: http://www.xprogramming.com/xpmag/expCardConversationConfirmation.htm

[23] J. D. Herbsleb, D. Moitra (2001, Mar/Apr), Global Software Development [Online], *IEEE Soft,* pp16-20. Available: http://csdl.computer.org/comp/mags/so/2001/02/s2toc.htm

[24] M. Summons (2004, May), Distributed Agile Development and the Death of Distance, Sourcing and Vendor Relationships*, Executive Report Vol. 5, No 4*. [Online]. Available: http://www.thoughtworks.com/au/library/

[25] CVS (Concurrent Versioning System), Available: http://www.gnu.org/software/cvs/

[26] Msn (Microsoft Network) Messenger, Available: www.weaddress.com

[27] MvnForum, Available: www.mvnforum.com

[28] T. Murphy (2003, Apr)., Mastering the Requirements of Requirements Management: Application Delivery Strategies Integration & Development Strategies, *Meta Practice 2020*