Full citation: Licorish, S., Philpott, A., & MacDonell, S.G. (2009) A prototype tool to support extended team collaboration in agile project feature management, in Proceedings of the International Conference on Software Engineering Theory and Practice (SETP-09). Orlando FL, USA, ISRST, pp.105-112.

A Prototype Tool to Support Extended Team Collaboration in Agile Project Feature Management

Sherlock Licorish, Anne Philpott and Stephen G. MacDonell
SERL, Auckland University of Technology
Private Bag 92006, Auckland 1142, New Zealand
{sherlock.licorish, anne.philpott, stephen.macdonell}@aut.ac.nz

Abstract

In light of unacceptable rates of software project failure agile development methodologies have achieved widespread industry prominence, aimed at reducing software project risks and improving the likelihood of project success. However, the highly collaborative processes embedded in agile methodologies may themselves introduce other risks. In particular, the fluid and diverse nature of agile team structures may mean that collaboration regarding what is to be delivered becomes more challenging. We have therefore developed a prototype tool intended to enable all stakeholders to have greater access to the features of the emerging system irrespective of their location, via remote feature management functionality. Software engineering experts have evaluated the initial prototype, verifying that it would enhance collaboration and is likely to assist teams in their handling of feature management.

1. INTRODUCTION

Among the software systems development methodologies to gain recent dominance have been those characterized as agile. In agile methodologies there is a gradual surfacing of the software through extensive co-operation, with team members interacting constantly in a common space employing a 'speculate-collaborate-learn' approach [1]. Beznosov and Kruchten [2] contend that the aims of such an approach are to reduce software project failure and development costs. More generally, Abrahamsson *et al*. [3] and Kuppuswami *et al*. [4] state that agile methodologies can reduce failure in software projects by mitigating software project risks. While the likelihood and impact of some risks may well be reduced through the use of agile methods, Kirk and Tempero [5] and Sharp *et al*. [6] argue that agile methodologies may themselves present complexities that might result in *additional* risks in the software process. Added risks may be associated with: minimal upfront planning which can result in rework due to oversight; frequent and ongoing customer involvement which may lead to disagreement and increase project cost; the need to manage a diversity of skills and personalities within highly interactive project teams; a lack of shared vision and domain knowledge among project stakeholders; and a lack of documentation which can result in poor communication during the project [5-7].

In recent years there has been steady adoption of agile methodologies in practice [8], and the body of evidence supporting their use continues to grow. The bulk of that research evidence, however, has understandably considered the impact that the use of agile methods has had on *conventional* risks. The contention that there are possibly *new* risks to consider therefore presents us with a research opportunity. Providing a degree of automated support for the potentially delicate handling of interactions among team members during feature management could be beneficial in terms of ensuring that the team can operate as intended – with customer representatives as genuinely active and embedded members irrespective of the their location – but with reduced likelihood of disagreements occurring among the wider team and exposing the entire project to risk. This research

therefore addresses the support for agile project contexts with particular emphasis on mitigating the risks that may arise from intensive and ongoing collaboration in feature management, leading to the development and preliminary evaluation of a suitable prototype tool.

The remainder of this paper is structured as follows. In the next section we review prior related work with particular attention directed to the involvement of customers in software teams, both under agile and more conventional methodologies. We then describe the development and preliminary evaluation of our prototype tool as a vehicle to assist diverse and/or dispersed teams in the feature management activity. We acknowledge the limitations and threats to the work in Section 5. We then draw conclusions from our work and describe further research opportunities.

2. RELATED WORK

Extensive and active customer involvement has long been cited as a key to software project success [9,10]. In the context of this study, customer involvement is taken to mean the active engagement of one or more of the intended users of the software system in requirements specification, software testing, and other development practices. The endorsement of customer involvement is especially dominant in agile processes; an extreme example of user involvement is evident in XP's onsite customer practice [11]. While customer involvement may be beneficial in principle, and has indeed been shown to be a factor in some project successes, it has long been acknowledged that larger teams, and particularly those with diverse membership, present additional communication and management challenges compared to their smaller and more homogeneous counterparts.

Some research goes so far as to suggest that there are in fact several pitfalls to extensive customer involvement. For instance, a high degree of early involvement can have an inflationary impact on the expectations of customers regarding the likely performance of the system being developed. Evidence for such an assertion can be found in studies reported by Grisham and Perry [12] and Tesch *et al*. [13], both of which found that customer satisfaction was influenced by their prior expectations. These studies established that when team performance is higher than customer expectations there is greater conformity and higher customer satisfaction. However, when the reverse occurs, customers are (naturally) dissatisfied.

Sharp *et al*. [6] studied nineteen software developers in an activity session and found that heavy customer involvement threatened software project success in a number of ways. While their findings may not generalize to all software projects, the results indicate that the incidence of conflicting views, skill differences, customer exposure to sensitive information such as schedule slippages and technical issues, and the opportunity for customers to assert influence in developers' decisions can lead to stakeholder clashes and may therefore negatively influence project success.

Since customer satisfaction is a central goal (perhaps *the* central goal) of the software developer in agile contexts [14], it is necessary that developers seek to stabilize scope, price, and duration to keep customers satisfied in what is often a changing environment. Ceschi *et al*. [15] assert, however, that it may be difficult to stabilize these variables in highly collaborative and iterative contexts. This could clearly pose challenges for agile project teams. In addition, agile methodologies generally emphasize the necessity of customer involvement to ensure software quality. However, Siakis and Siakis [16] posit that quality for the customer is mainly associated with ease of use (after the software is released) and suitable functionality to match appropriate pricing. These are not *necessarily* linked to ongoing active customer involvement in the development process.

In summary, the findings of previous studies suggest that extensive customer involvement is most beneficial when teams are successful, when customers and developers share similar views, and when there is minimal skill difference between the customer(s) and the developer(s). In addition, customer interest in the software development process is linked primarily to software ease of use and functional adequacy in relation to pricing [16]. Accordingly, the extension of the team to include onsite customers may be problematic under circumstances in which the above conditions are not met – for instance, when the developers perform poorly, or where the team inclusive of the customer representatives lacks balance in skills or personalities. Therefore, finding a means of facilitating optimal collaboration among team members may reduce potential conflict and the negative impacts associated with such interpersonal conflict. In addition, such means may also ensure involvement in instances when onsite participation is not feasible. One way to optimize interaction with customer representatives during feature management is to extend their ability to contribute via a remote tool interface. Such an interface should simulate aspects of the face-to-face environment, providing the customer with the opportunity to initiate features, and to participate in the management of other aspects of the process, but without the intensive and constant onsite engagement commonly promoted.

Previous studies have demonstrated that software tools can enable and/or improve communication in projects [17,18]. For instance, it has been proposed that the use of groupware systems enhances project communication and produces improved project outcomes [19,20]. Therefore, a tool to improve the extent and quality of customer contributions, particularly in the determination and management of software features, could add value to software project management. An assessment of existing project management tools based on the work of Kelter *et al*. [21] and VersionOne [22] indicates that these tools have not been built with the intent of enabling remote extension of the onsite customer. As a result, the objective of the current study is to bridge this gap with a prototype feature management tool that extends distributed customer interaction, and enhances team collaboration, through a remote interface.

3. THE ASRM TOOL

This section describes the development of a prototype tool called the Agile Social-Risk Mitigation Tool (ASRMT), designed in part to support collaborative feature management by providing remote access via a web interface to all stakeholders during agile software development [23]. The intent is to facilitate active involvement and collaboration irrespective of time or stakeholder location, thus supporting the work of dispersed teams. It is also hoped that increased collaboration would reduce the likelihood of conflict or miscommunication regarding software features.

We intended in the first instance to develop the tool rapidly and iteratively, as a lightweight and therefore readily usable prototype, rather than a fully viable product. Our goal in this regard was to have the tool used and evaluated by a small group of practitioners so that it could be refined (based on their feedback) before being deployed and assessed more formally in live project settings. Several choices exist for web application development that would fit well with such an approach. Among these, Java technologies and Visual Studio.NET seem popular [24,25]. Our prototype tool was developed using the ASP.NET framework using Visual Basic and some JavaScript for client side validation [26]. The development architecture for the tool is shown in Figure 1.

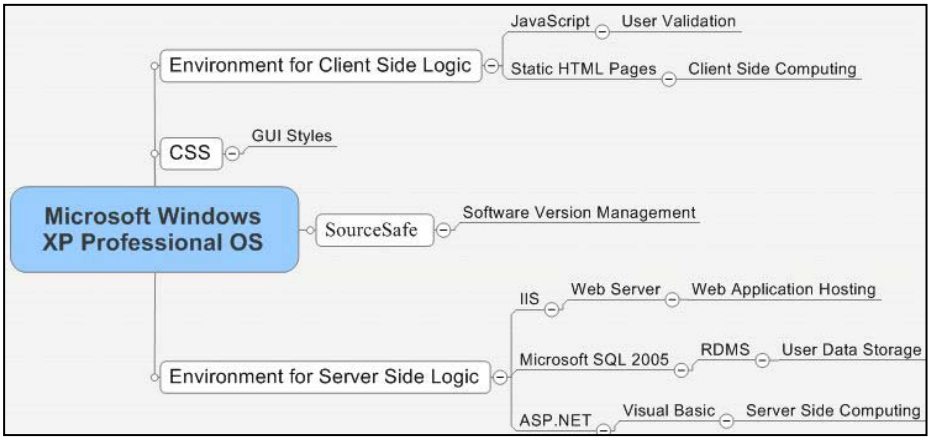

Figure 1. ASRMT development architecture

The initiation, specification and prioritization of software features, a sequence of standard and ongoing activities in agile software development, can be tracked either on- or off-site using ASRMT. Customer stakeholders can actively contribute to the management of features for all projects in which they are involved.

Feature information initiated by customer team members is from that point jointly managed by both developer and customer members using ASRMT. The status of each feature is visible to all concerned at any time and follows the partially iterative sequence of states depicted in Figure 2, illustrated in the tool screen shot shown in Figure 3.

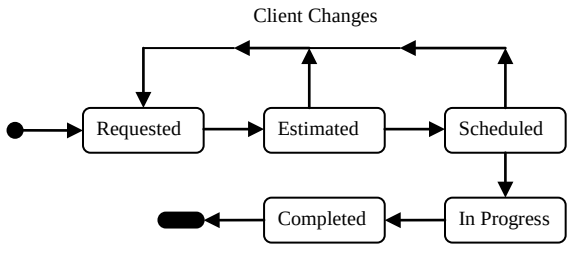

Figure 2. Feature traversing states

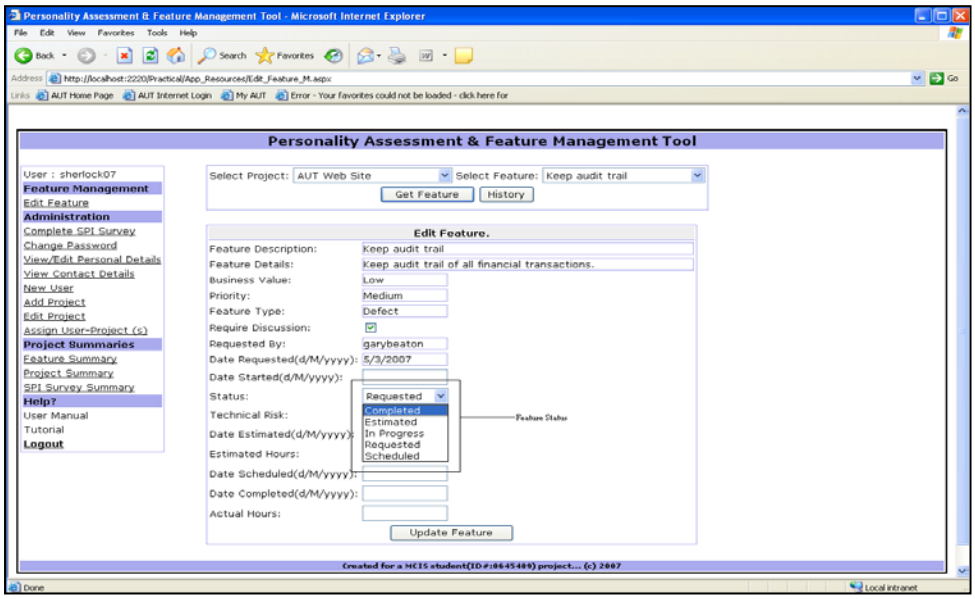

Figure 3. Feature editing screen

Add Feature: ASRMT enables customer team members to add features. In order to do so they must provide information relating to the Project, a Feature Description (short statement of not more than 200 words), Feature Details (any additional details regarding the feature), an indication of perceived Business Value (Low, Moderate, or Significant, features that are Significant being most important for the business at the time of entry), the feature's Priority (High, Medium, or Low – features that are High in priority should attract most of the developers' attention), Feature Type (New Feature, Defect repair, or Enhancement), and whether the feature needs to be discussed in a face-to-face meeting. New features are tagged with the default status 'Requested'. (Reporting based on all of these fields is also supported.)

Edit Feature: Customer team members are able to edit information for features previously added when the feature carries the status 'Requested', 'Estimated', or 'Scheduled'. If a feature's status is 'Estimated' or 'Scheduled' and it is edited, its status reverts to 'Requested'. A log is maintained for all changes made to features. When the assigned Developer team member is ready to estimate a requested feature and so enters estimation information (Date Estimated and Estimated Hours), the feature status is automatically changed to 'Estimated'. If the feature is then scheduled (once Date Scheduled is entered), its status is automatically changed to 'Scheduled'. Once the feature's Date Started field is entered, the feature status is then automatically updated to 'In Progress' (and the customer team members can no longer directly edit the feature). Finally, the feature status is automatically changed to 'Completed' once the Actual Hours and Date Completed are entered. In addition to considering a feature's Business Value and Priority in terms of effort allocation, project managers and developers are able to categorize and process features based on an estimation of their technical risk (Low, Medium, or High).

ASRMT offers a range of project summaries for feature information (for example: New Feature, Defect Repair, and Enhancement tracking, at various states of completion). ASRMT also provides real-time feature tracking information support for customer and developer team members, in both detailed and summary forms. This allows all stakeholders to observe project progress while still being physically remote from the rest of the team. In addition, the decision making of project managers is supported through the information provided by ASRMT.

4. ASRMT EVALUATION

In keeping with the research aims and the present prototype form of the tool, ASRMT has been informally tested by a small number of software engineering experts. Seven participants were involved in the evaluation process. These participants were agile software developers with varying levels of development and project management experience who were therefore well placed to comment on whether they felt the tool would help agile teams to optimize collaboration during feature management.

The tool was installed on a local server, on which the participants completed a scenario-based evaluation

comprised of two parts. The first part of the evaluation asked each participant to test ASRMT's functionality using 23 tasks, in the roles of project manager, developer, and customer, while the second part of the evaluation was designed to solicit feedback regarding participants' impressions of the tool and their use of it while working through the scenarios. While questions may arise in relation to validity for randomly constructed evaluation instruments [27], it is important to note that the questions for the ASRMT user evaluation were not randomly selected. Rather, this evaluation was adopted from an earlier exercise reported by Lewis [28]. Lewis' instrument has been previously assessed for reliability and validity, and recommended for usability evaluations (see, for example, [29,30]).

The scenarios used in the first part of the evaluation required that each participant carry out 'typical' team member activities; for instance, step 5 of the project manager's set of tasks was conveyed as follows: "Next you need to change an ongoing project – the Credit Card System. Edit a feature previously added using the Edit Feature menu. The status of the feature determines what data is added to the feature; for example, if the values for 'Estimated Hours' and 'Date Estimated' are entered, the feature status automatically changes to **Estimated**. Let the feature traverse states by updating its associated information (enter the Date Scheduled, notice the status changes to **Scheduled**). (Features entered by clients have 'Requested' status by default, features then traverse status in the following order: Estimated, Scheduled, In Progress, and Completed)."

Task 6 of the customer scenario required the following: "You want to know how the team is getting on overall with the project. Go to the Project Summaries Section of the main screen and get a summary. (Note: you are only allowed to see project summaries for projects that you have been added to by the Project Managers.) Get a feature summary for the Credit Card System project – information should exist pertaining to features previously entered (Insert Start Date (22/02/2007), End Date (15/04/2007), Filter by Priority, then by Status, then by Risk Rating)."

Having completed the set of tasks, participants then moved to part two of the evaluation. This comprised 11 questions. Seven closed questions, each conforming to a four-point Likert scale, were used to evaluate ASRMT's stability and the users' learning experience. Two closed questions conforming to a Likert scale and two further open-ended questions were used to assess ASRMT's usefulness, to consider whether ASRMT addressed the research objectives (explained to the participants at the beginning of the evaluation exercise), and to seek participants' overall impressions and recommendations for improving ASRMT:

1. ASRMT would be useful if used in live projects.
2. ASRMT offers functionality to address the features discussed in the 'ASRMT purpose' section at the beginning of this document. [This section spelled out the challenges associated with team collaboration in such projects and the desire to support as much as possible the interaction among all team members.]
3. In terms of your overall impression of ASRMT…
    a. List any negative aspects
    b. List any positive aspects
4. Please outline any suggestions you have for improving ASRMT.

The possible responses to the questions conforming to a Likert scale included 'strongly agree', 'agree', 'disagree', and 'strongly disagree' options. The answers were linearly scaled from one to four where a 'strongly agree' choice was represented by one and four represented a 'strongly disagree' choice, offering participants no neutral choice such as 'neither agree nor disagree'. This approach was deliberately selected to force participants to express an opinion. While there may be threats to reliability for usability evaluations employing such an approach, given that the participants were all trained software engineers with a range of levels of experience, and so represented the target population, it is believed that this option presents a low threat to the reliability of the findings [27].

All participants completed the evaluation in full, with the average time taken being just under one hour. In general, participants felt that the concepts and ideas underpinning the development of ASRMT were excellent. Six of the participants thought ASRMT was easy to use. Of the seven participants, three reported encountering a small number of faults while using ASRMT. However, all participants reported that they were able to successfully complete the scenarios, that ASRMT was easy to learn to use, and that they recovered easily and quickly from any errors.

Of the seven participants, five reported that ASRMT was simple and satisfying to use, and four believed that ASRMT would be useful and effective if used in live projects. The three participants who did not agree that ASRMT would be useful if used in live projects felt that the tool needed usability improvement before it would be suitable. All of the participants, however, believed that ASRMT offered functionality to address the challenges of collaborative feature management in keeping with the purpose of optimizing interactions and minimizing project risk. In terms of the participants' overall impressions of ASRMT, all participants

believed that the notion of extending customer (and other stakeholder) interaction via a remote interface such as that in ASRMT would be useful. Regarding the tool's ease of use, five of the participants believed that ASRMT's simplicity and ease of use would enhance agile project management.

Among the recommendations for improvement, five participants believed that a few of ASRMT's user interfaces could be improved, and one participant suggested that ASRMT might need enhancement if it was to be implemented in large projects. Participants suggested that additional guidance for user tasks be provided. In addition, one participant also suggested that ASRMT could be extended to include additional functionality such as a discussion feature and automatic e-mail reminders, which would be likely to further assist project participants.

5. LIMITATIONS AND THREATS

There are at least two limitations to our study, both of which relate to the evaluation of the ASRM tool. The first arises from the limited scale and artificial nature of the sample tasks participants were asked to undertake. Even though the ASRMT user evaluation scenarios were meaningful, in that they were tasks typical of those undertaken in feature management, due to resource constraints the ASRMT was not used in the management of live software projects. In such projects there could be many project members occupying varying roles, and there may be a need to coordinate and manage many concurrent development tasks, perhaps across a portfolio of projects. Further ASRMT user evaluations should therefore be carried out in live project environments.

The second limitation relates to the scale and form of the evaluation. The tool was evaluated by just seven participants. While these seven individuals were representative of *some* of the intended users of the tool, being experienced software engineers, they were not themselves project customers nor were they communicating with other developers. The degree to which their responses could be considered representative of stakeholder perceptions of the tool and its usefulness is not known. Further assessment of the tool by project managers, customers and other developers, preferably in concert, would therefore be beneficial.

6. SUMMARY, CONCLUSIONS AND FUTURE WORK

Regardless of the methodology employed, evidence shows that software development continues to be a very challenging endeavour. This study found that, while agile methodologies are increasingly likely to improve some aspects of software project development and management, the human collaboration practices common in agile methods may also introduce social risks, and such risks could be critical – especially in circumstances where the project team begins to under-perform.

With such issues in mind, a prototype tool was designed and developed. The tool was intended as a low-overhead complement to face-to-face team interaction, providing stakeholders with a lightweight means of maintaining involvement in feature management without the risks arising from intensive and constant interactions. ASRMT was verified by a small number of software engineering experts. In scenario-based testing the tool was found to be easy to learn and use and sufficiently lightweight as not to present unjustified additional overhead. The expert assessors were also strongly supportive of the concept of remote stakeholder involvement. The ASRMT user evaluation findings suggest that the tool is likely to be useful to agile developers, and should improve their handling of collaboration-related risks.

While the purpose of ASRMT in the context of agile projects was to reduce the incidence and severity of risks associated with potentially disruptive interactions, ASRMT may provide a vehicle through which customer involvement could be further strengthened in non-agile projects. Since the literature also shows that too little customer involvement may be problematic, *balancing* customer involvement is key. Therefore, the ASRMT solution may support risk mitigation in two ways: it should enable project managers to deal effectively with complex interactions in diverse teams (as per agile methods), but it may also help managers to engage more frequently and intensively with customers if their lack of input under traditional methodologies begins to threaten project progress.

Irrespective of the enhancement just described, in order to draw any stronger conclusions regarding risk mitigation and project success it would be necessary to utilise ASRMT in live project settings – the logical next step for this work.

7. REFERENCES

[1] Highsmith, J. (2000) Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York: Dorset House.

[2] Beznosov, K., & Kruchten, P. (2004) Towards agile security assurance. Proc 2004 Workshop on New Security Paradigms, Nova Scotia, Canada.

[3] Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003) New directions on agile methods: a comparative analysis. Proc 25th ICSE, Portland OR, USA.

[4] Kuppuswami, S., Vivekanandan, K., Ramaswamy, P., & Rodrigues, P. (2003) The effects of individual XP practices on software development effort. SIGSoftSEN 28(6) 1-6.

[5] Kirk, D., & Tempero, E. (2006) Identifying Risks in XP Projects through Process Modeling. Proc Australian Software Engineering Conference.

[6] Sharp, H., Robinson, H., & Segal, J. (2004) Customer collaboration: successes and challenges in practice systems. Report TR2004/10, Computing Dept, Open University.

[7] Hulkko, H., & Abrahamsson, P. (2005) A multiple case study on the impact of pair programming on product quality. Proc 27th ICSE, St. Louis, MO, USA.

[8] Anon (2008) Martinig & Associates: Adoption of agile methods. Retrieved March 2008, from http://www.methodsandtools.com/

[9] Clavadetscher, C. (1998) User involvement: key to success. IEEE Software, 15(2), 30-32.

[10] Jiang, J. J., Klein, G., & Balloun, J. (1996) Ranking of system implementation success factors. Project Management Journal, 27(4), 50-55.

[11] Beck, K. (2000) Extreme Programming Explained: Embrace Change. Reading, MA: Addison-Wesley Longman.

[12] Grisham, S. P., & Perry, E. D. (2005) Customer relationships and Extreme Programming Proc 2005 Workshop on Human and Social Factors of SE.

[13] Tesch, D., Jiang, J. J., & Klein, G. (2003) The impact of information system personnel skill discrepancies on stakeholder satisfaction. Decision Sciences, 34(1), 107-127.

[14] Kettunen, P. (2007) Extending Software Project Agility with New Product Development Enterprise Agility. Software Process: Improvement and Practice, 12(6), 541-548.

[15] Ceschi, M., Sillitti, A., Succi, G., & De Panfilis, S. (2005) Project management in plan-based and agile companies. IEEE Software, 22(3), 21-27.

[16] Siakas, K. V., & Siakas, E. (2007) The agile professional culture: a source of agile quality. Software Process: Improvement and Practice, 12(6), 597-610.

[17] Boehm, B., Grunbacher, P., & Briggs, R. O. (2001) Developing groupware for requirements negotiation: lessons learned. IEEE Software, 18(3), 46-55.

[18] Damian, D. E. H., Eberlein, A., Shaw, M. L. G., & Gaines, B. R. (2000) Using different communication media in requirements negotiation. IEEE Software, 17(3), 28-36.

[19] Ali Babar, M., Kitchenham, B., Zhu, L., Gorton, I., & Jeffery, R. (2006) An empirical study of groupware support for distributed software architecture evaluation process. Journal of Systems and Software, 79(7), 912-925.

[20] Ali Babar, M., Kitchenham, B., & Jeffery, R. (2007) Comparing distributed and face-to-face meetings for software architecture evaluation: a controlled experiment. Empirical Software Engineering, 13(1), 39-62.

[21] Kelter, U., Monecke, M., & Schild, M. (2003) Do we need 'Agile' software development tools? LNCS v.2591, Springer Berlin/ Heidelberg, 412-430.

[22] VersionOne. (2006) Agile tool evaluator guide. Retrieved Oct 10 2006, from http://www.versionone.com

[23] Angioni, M., Carboni, D., Pinna, S., Sanna, R., Serra, N., & Soro, A. (2006) Integrating XP project management in development environments. Journal of Systems Architecture, 52(11), 619-626.

[24] Atsuta, S., & Matsuura, S. (2004) eXtreme Programming support tool in distributed environment. Proc Computer Software and Applications Conference.

[25] Kaariainen, J., Koskela, J., Abrahamsson, P., & Takalo, J. (2004) Improving requirements management in extreme programming with tool support - an improvement attempt that failed. Proc Euromicro SEAA.

[26] Flanagan, D. (2001) JavaScript: The Definitive Guide. (Fourth ed.). CA, USA: O'Reilly.

[27] Kirakowski, J. (2000) Questionnaires in usability engineering: a list of frequently asked questions. Cork, Ireland: Human Factors Research Group.

[28] Lewis, J. R. (1995) IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation

and Instructions for Use. Intl Jnl of Human-Computer Interaction, 7(1), 57-78.

[29] Calisir, F., and Calisir, F. (2004) The relation of interface usability characteristics, perceived usefulness, and perceived ease of use to end-user satisfaction with enterprise resource planning (ERP) systems. Computers in Human Behavior 20, 505–515.

[30] Vuolle, M., Kallio, T., Kulju, M., Tiainen, M., Vainio, T., and Wigelius, H. (2008) Developing a Questionnaire for Measuring Mobile Business Service Experience. Proc MobileHCI, Amsterdam, 53-62.