Requirements management in students' software development projects

Pekka Mäkiaho, Timo Poranen, Zheying Zhang

Abstract: *In this paper, we study requirements management practices in students' software development projects. The 12 projects studied applied iterative development models and agile practices. We analyze tools usage, methods, processes, common problems, risks, and challenges in requirements management. We also research changes in requirements statuses and conduct a more detailed analysis for status changes in three projects. As a result, we propose guidelines and suggestions to teachers and project managers based on our findings.*

Key words: *Requirements management, requirements statuses, tools, student projects*

INTRODUCTION

Teaching the basics of project-based work is part of studies in all the universities that offer degrees in Computer Sciences. In addition to project management, main activities in software development projects include requirements management, software design, implementation and testing. Understanding requirements correctly is essential for a successful project, because in a worst-case scenario misunderstandings might lead to the implementation of a completely wrong product. Thus, teaching requirement engineering is an essential part of the computer science studies [7,12].

In this paper, we conduct an extensive study on requirements management practices in students' software development projects. We analyze tool usage, methods and processes, risks and challenges in 12 software development projects conducted during the fall semester in 2015. All the projects implemented a different software product based on the needs of their respective clients. The teams had freedom to use different tools and practices in their requirements management activities. We also analyze the status changes of the requirements throughout the project life cycle. Our data was gathered using Moodle questionnaires and weekly reports written by the project managers.

The structure of the paper is as follows. In the second section, we present the background knowledge of software development projects and requirements management. In the third section, we explain the used questionnaire and weekly report format in details. Next, we analyze and discuss the collected data from the perspective of requirement management methods, tools, and processes. The last section concludes the study and gives some suggestions on teaching project work.

REQUIREMENTS MANAGEMENTS IN SOFTWARE DEVELOPMENT

The traditional sequential and stage-gated software development models, such as the waterfall model [16], relied heavily on prediction and documentation. The project was planned and scheduled, and the required features and functionalities were documented up-front, the software architecture was defined in a design document, and a test plan was used to define test cases and testing process. It is usually challenging to know and define all the required information of a software system at the beginning of the project. Therefore, iterative and incremental [4], and later also agile software development models began to gain popularity.

Agile projects feature an intensive communication process with users and do better at harnessing changes for the customer's competitive advantage [5,15]. Changes are accommodated by implementing a product through a series of iterations and maintaining a

dynamic backlog of requirements to be done. Each iteration implements the set of requirements in the backlog that have the highest priority at that time. The agreed changes, such as new requirements, as well as changes to existing ones are prioritized against the remaining backlog contents and allocated to future iterations. Therefore, the requirement and its priority are continuously tuned to make sure that the highest value is provided to the customers as quickly as possible. This type of change control forms the basis of agile requirements management and it consists of activities such as requirements tracing, impact analysis, requirements updating and version control. Project teams manage requirements in an interactive and just-in-time manner.

There is a wide range of tools supporting requirements management [2,17]. Besides the traditional word processors and the commercial, sophisticated requirements management tools such as Rational DOORs family [9], many general-purpose file storage and sharing services like Google Drive offer a platform that allows several different users involved in common tasks to achieve their goals. These tools facilitate the process of creating, editing, sharing, and discussing requirements. In addition, there are also tools dedicated to source code sharing and management, such as GitHub [11], or project management tools, such as JIRA [3] and Redmine [10], which provide requirements management capabilities for prioritizing backlogs, allocating requirements to iterations, monitoring requirement status and linking requirements to other related artifacts.

Moreover, tracking requirements status has an important role in monitoring project progress, as well as retrospectively analyzing the potential problems and improvements in the requirements management process. The status of a requirement describes the state of this requirement at a particular time, and it forms one of the attributes for the requirement. For example, the requirements which have been requested by project stakeholders and added into the backlog can have a status 'New', the ones under implementation have a status 'In progress', the requirements which have been implemented and accepted by acceptance testing have a status 'Done', 'Rejected' requirements are those which are not planned to be implemented in any upcoming releases. Different projects may define the requirement status categories differently. No matter what statuses are given in a project, a requirement always has one status at a given time in the project life cycle, and its status is updated when specified transition conditions are satisfied. The requirement status changes when the development work on a project is progressing. Monitoring the status of each requirement throughout development work provides a precise gauge of project progress, and can help to illustrate how the project is approaching its goal of delivering the product that meets the expectations of the client.

Agile and traditional projects handle requirements differently in various respects, particularly with regard to the timing and depth of requirements activities and the extent of written requirements specification [11,15]. In addition, the effectiveness of communication between clients and the development team, the cost and schedule estimation, the business value based requirements reprioritization, and minimal documentation form the main challenges in agile requirements management practice [8,15]. Obviously, the practice of requirements management is at a cross-section of multiple disciplines. On one hand, knowledge of techniques and tools is essential in a project, but on the other hand, soft skills such as communication, negotiation, problem perception and interpretation are inseparable from a successful project.

Both the hard and soft knowledge and skills are of importance in university education. Some studies [7,12] have discussed the challenges of teaching the requirements engineering course by providing students with competencies in multiple disciplinary and in real project settings, and proposed and evaluated pedagogical strategy and approaches to motivating students for acquiring knowledge and skills in real-life software development environments. In this paper, instead of the curriculum design of a course, we identify and analyze the problems the students face in projects offered by local

industry and research groups, and propose guidelines on requirements management in student projects. *Our research questions are: what are the common challenges of requirements management in student projects (Q1) and, what are the tools used in student projects and how they applied for requirements management (Q2).*

PROJECT DATA

During the fall semester in 2015, 22 students participated in a Master's level course called Software Project Management (SPM) and 40 students in a Bachelor's level course called Project Work (PW). These students formed 12 project groups in which the SPM course students worked as project managers and the PW course students as developers. All the project managers had passed the PW course previously or had the corresponding knowledge. The managers had also passed a course called Software Project Management theory, or they were participating in that course at the same time. The course called Requirement engineering was not mandatory as preceding studies, but it was recommended. All teams had different clients and project topics.

Within the projects, students were free to select the tools they wanted use. The university provided basic software development and design tools such as Subversion, Balsamiq [1] and Redmine, but it was not compulsory to use these. The course supervisors helped the project teams to select a suitable set of tools.

The development model could also be chosen freely: all the teams chose to use Scrum [13] with some variation, for example skipping daily meetings or having them online. Students were asked to specify the 'definition of done' in their project plans and to keep on tracking the state changes of every requirement using the statuses 'New', 'In progress', 'Done' and 'Rejected'.

The projects began in week 37/2015 and the final reviews with supervisors and clients were held in January 2016. In addition to the final review, the teams held three mandatory reviews with the supervisors and the other stakeholders. The teams reported their progress via weekly reports which were sent to the course supervisors and clients. The predetermined report forms included the working hours used, commits to the version control, passed test cases and overall number of test cases and the number of requirements for each state.

Data was gathered also by using two Moodle questionnaires: the first one was prepared in the mid phase of the projects, at the beginning of November, and the second one after the projects had been finalized. In order to get answers to our research questions, we asked the teams both direct questions concerning the challenges and the tool usage, and more indirect questions that could have indicated if there had been some problems during the project work. In the first questionnaire, the students were asked questions about how they had elicited the requirements, what tools they had used and whether they had had any challenges in managing the requirements. The project managers were also asked how the requirements' statuses had been updated and whether they had an agreed process for handling the requirements' status change.

The second questionnaire included questions such as how the prioritization was conducted and if there had been any changes in the priorities, how the requirements had changed and how the changes had affected to the project, how the requirement tools helped the team, and if whether there had been any changes in the tools usage during the project. The project managers were also asked whether there had been any changes in the agreed process, and whether the weekly reporting of the requirements' statuses had helped them to observe the project progress. In addition to these questions, all the students were asked to give free comments on challenges on requirements' management.

ANALYSIS OF DATA

All the SPM and PW students answered to both of the Moodle questionnaires. We also received more than 200 weekly reports from the teams.

In this section, we first give an analysis of requirement changes in three sample projects. Then, we analyze requirements management processes and requirements elicitation methods of the studied projects. After that, we go through the findings from change management and prioritization used by the different teams. This is done to obtain an overall picture of how the requirements were handled during the projects. Then, we introduce the tools used in the requirements management. Finally, we list reported challenges and problems.

EXAMPLE PROJECTS

We selected three example projects (A, B and C) which refined the requirements to reasonable level and reported them satisfyingly. Figures 1, 2 and 3 illustrate how the requirement statuses changed during the project.

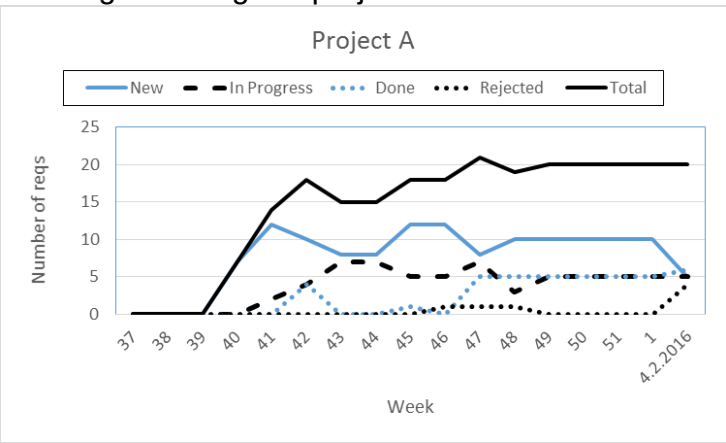

Figure 1: Requirements status changes in Project A.

In Figure 1, it is easy to identify some inconsistency in reporting within Project A: in weeks 42 and 47, the total number of the requirements has been decreased; the requirement should never disappear, only the status can be changed Moreover, the number of requirements with status 'Done' increased at first and then decreased back to zero and never increased above 5. This is because on week 43 the supervisor reminded the team that according to the project's own 'definition of done' all the requirements transferred to 'Done' status should have been acceptance tested. This team also mixed agile and waterfall and left the testing to the end of the project and when they run out of the calendar time, testing was partially skipped and the untested requirement stayed 'In progress'. It was also due to the lack of time that 5 requirements were transferred from 'New' to 'Rejected' after week 1/2016.

The project managers of Project A reported that the client kept on giving new requirements and were not able to prioritize them. The developers reported that as there were so many requirements not yet done and new ones came in all the time, the project felt like a chaos and they had motivation problems.

Project A tried to follow Scrum with 2-4 weeks sprints. However, they had problems in keeping the sprint and product backlogs separated and in concentrating only on finishing the tasks in the sprint backlog.

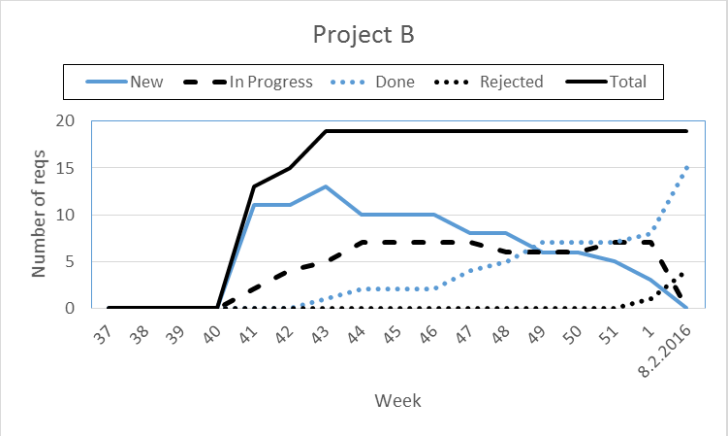

Figure 2: Requirements status changes in Project B.

In Figure 2, Project B stabilized the requirements already on the 3rd week. The amount of the requirements having status 'In progress' increased from zero to seven during the 1st month and then remained the same almost to the end of the project. The number of requirements with status 'Done' increased steadily until week 1/2016 when a bunch of requirements were tested and accepted. At the same time some of the requirements were rejected due to lack of time.

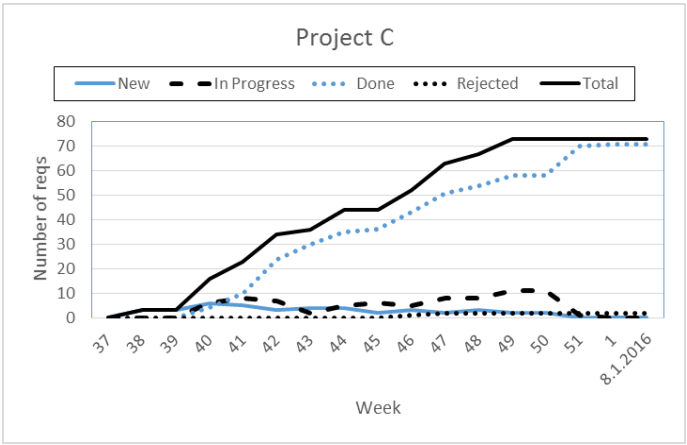

Figure 3: Requirements status changes in Project C.

Project C was a research project in which the client had originally only three high level requirements. The team elicited 73 requirements in total during the period starting two weeks from the project initialization and ending three weeks before the final review. The team was very motivated and their sprints were only one week in length. Sometimes they took new requirements under the implementation and finalized those in one or two days in the middle of the sprint. The Figure 3 shows the statuses only on the bases of the weekly reports, so the real number of requirements 'In progress' could be a bit higher. However, the workload was divided equally on each week and the size of product backlog (requirements having the status of 'New') never grew too much, which was the case in Project A.

REQUIREMENTS MANAGEMENT PROCESSES

Seven teams out of the total 12 told that they had no process for the requirement management. Five teams had an agreed process for managing product and sprint backlogs and for updating requirements statuses.

No teams reported any changes on the process during the project.

REQUIREMENTS SOURCE AND ELICITATION

The main source of the requirements was the client. The requirements were elicited and analyzed together with the project managers, the team members and the clients before they were accepted to the project backlog. In one project, the team members further specified the technical requirements: non-functional requirements that need to be fulfilled because of the technology used. In Project C (Figure 3), the client only proposed 3 requirements and the team elicited 73 requirements in total based on these high-level requirements. Due to the client's business trip during the project, there were difficulties in communicating with the client. The project was research-oriented and the team elicited requirements independently, without guidance from the client. Many requirements were added to the product backlog and were even implemented and tested during the client's temporary period of absence. The implementation was tested by the client when he returned. However, none of the requirements had to be rejected or re-implemented after the late testing. There was a communication challenge in project C and thus a risk that the implementation would not meet the client's expectations. However, this risk did not realize.

Techniques for requirements elicitation included interviews (4 groups), use cases (4), brainstorming (3), prototyping including mockups and flow diagrams (5) and evaluation of the application's preview version (1).

CHANGE MANAGEMENT AND PRIORITIZATION

Most of the projects (9/12) reported that there were no significant changes in the requirements, or there was only some refining before the implementation work started. At the beginning, the requirements were either so well specified that no changes were needed, or they were on such a high level that new requirements needed to be elicited based on them. Two teams mentioned that due to the lack of time, a feature was implemented as a simpler version than specified in the original requirement. This can also be seen as a symptom of poorly specified requirements, and that these requirements had not been properly defined as sub-requirements. Three projects reported that the late changes in requirements affected the re-implementation work to be done.

Half (6/12) of the teams set the priorities together with the client based on their importance to the client and the difficulty level in their implementation. The rest had more informal prioritization criteria, for example "Project managers set the priorities", "We implemented what could be done in the current phase" or "The requirements elicited first were implemented first".

Two teams rejected requirements in the end of the project because the lack of time. Another team had the priority changes approved by the client - another just reported that they changed priorities but did not document the changes.

REQUIREMENT MANAGEMENT TOOLS

Teams were free to select the tools they wanted to use for managing requirements. The list of requirements was maintained with the help of traditional documenting tools such as Excel or Word (2 groups). Shared documents on GoogleDrive were used by 6 groups. Tools for supporting requirement management were used in more than half of the groups: Redmine (6), Trello [14] (3), Github (2), VisualStudio and BitBucket.

Changed in the tool usage were reported by only two teams. One team used Excel from the beginning and tried to take Redmine to use in the middle of the project, but reported that it did not work for them. Another team used GitHub as a version control tool from the very beginning, and during the project they realized that it can also be used for requirement management and took that into use.

Comparing the study that concentrated on tools used in similar circumstances on academic year 2011-2012 [6], the management tools used then were Redmine, Jira, Kanbanery and different wikis. Of those, only Redmine was used now.

CHALLENGES

Seven teams out of twelve reported challenges with incomplete specification. Both the project managers and the team members would have wanted more specifically documented requirements from the client. Another big challenge was communication: it was reported by five teams. These two problems are related: especially in agile development, the specifications are evolving throughout the project lifespan and communication is very important in this process.

Three teams mentioned that the client did not fully understand the importance of prioritization or that there were other prioritization issues involved. In the example project A, having many undone requirements without prioritization made the developers believe that they will not be able to finish the project and this decreased motivation. Three teams mentioned the difficulty of estimating the work amount. Three teams faced problems with the tool used for requirement management: they either did not have a tool or the tool was not suitable for managing requirements. Two teams reported that they had motivation problems because the number of the requirements was huge and increased further during the project. Two teams also reported the changing of the requirements as a problem.

SUMMARY AND CONCLUSIONS

Even though the project teams reported that they applied agile methods, a kind of 'waterfall mind-set' still tended to prevail; changes on the requirements were perceived as problems and client was expected to provide exact specifications at the beginning of the project. Thus, it should be stressed to the students that agile welcomes the change of requirements and the team should be prepared to the changes.

There was one team which had motivation problems when there seemed to be too many requirements in their project. Despite the fact that the project team had selected Scrum as a framework, their requirements were not prioritized and the project backlogs and the sprint backlogs were not clearly distinguished. Maybe the principles of agile methods should be revised at the beginning of the courses and those should be documented in the project plan.

Only two teams out of twelve did not use pure requirement management tools. The team with the lack of prioritization was the other team which did not.

When studying the graphs afterwards, one can easily notice that in Project A, there were reporting problems and that the number of 'New' requirements kept increasing but the number of 'Done' did not. The project manager and supervisor of the project could have tried to find out what caused this. A tool for showing these graphs online could prove to be beneficial to both the supervisor and to the project managers.

These projects were conducted during project work courses. In addition to fulfilling the projects' goals, the aim was also to teach project work and software project management to the students. Even if a project fails either in scope or in time, it can provide a good learning experience. This shall be kept in mind when trying to find solutions to the challenges the projects faced.

REFERENCES

[1] Balsamiq, a web-based mockup-tool. http://www.balsamiq.com, 2016.

[2] Decker, B., E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-Based Stakeholder Participation in Requirements Engineering. IEEE Software, 24(2), pp. 28-35. 2007.

[3] JIRA, a bug tracking, issue tracker and project management software. https://www.atlassian.com/software/jira, 2016.

[4] Larman, C. Iterative and Incremental Development: A Brief History. Computer, 36 (6): 47–56, 2003.

[5] Leffingwell D., Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, 1st Edition, Addison-Wesley, 2011.

[6] Mäkiaho, P., T. Poranen. Tool usage in students' software projects. In Proceedings of INSPIRE, pages 63-76, 2012.

[7] Mohan, S., S. Chenoweth. Teaching Requirements Engineering to Undergraduate Students, Proceedings of the 42nd ACM technical symposium on Computer science education, pages 141-146, 2011.

[8] Ramesh B., L. Cao. and R. Baskerville. Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal, 20(5), pages 449–480, September 2010.

[9] Rational DOORS family, scalable solutions for requirements definition and management, http://www-03.ibm.com/software/products/en/ratidoorfamihttp://www-03.ibm.com/software/products/en/ratidoorfami, 2017.

[10] Redmine, a web-based project management and issue tracking tool. http://www.redmine.org, 2016.

[11] Salo, R., T. Poranen., and Z. Zhang. Requirements Management in GitHub With Lean Approach. In Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST'15), pages 164-178, 2015.

[12] Scepanovic, S., L. Beus-Dukic Teaching Requirements Engineering: EUROWEB experience, ECSAW'15 Proceedings of the 2015 European Conference on Software Architecture Workshops. Article No. 38.

[13] Scrum - iterative and incremental agile software development framework. https://en.wikipedia.org/wiki/Scrum_%28software_development%29, 2017.

[14] Trello, a web-based project management application. http://www.trello.com, 2017.

[15] Wiegers, K., E. and J. Beatty, Software Requirements, 3rd ed. Microsoft Press, 2013

[16] Winston, R. Managing the Development of Large Software Systems. In Proceedings of IEEE WESCON 26, 1970.

[17] Zhang, Z., M. Arvela, E. Berki, M. Muhonen, J. Nummenmaa and T. Poranen. Towards Lightweight Requirements Documentation, Journal of Software Engineering and Applications 3(9), pages 882-889, 2010.

ABOUT THE AUTHORS

PhD student Pekka Mäkiaho MSc, Faculty of Natural Sciences, University of Tampere, Finland, Phone +358 50 556 6266, E-mail: pekka.makiaho@uta.fi.

University lecturer Timo Poranen, PhD, Faculty of Natural Sciences, University of Tampere, Finland, E-mail: timo.t.poranen@uta.fi.

University lecturer Zheying Zhang, PhD, Faculty of Natural Sciences, University of Tampere, Finland, E-mail: zheying.zhang@uta.fi.