Timo Poranen (ed.)

Software project management summaries 2013

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 22

TAMPERE 2013

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 22
AUGUST 2013

Timo Poranen (ed.)

Software project management summaries 2013

SCHOOL OF INFORMATION SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 978-951-44-9210-5

ISSN-L 1799-8158
ISSN 1799-8158

Preface

This report contains summaries of project management articles published in international scientific journals and conferences. The summaries were written as a compulsory task for the "TIETS19 Software Project Management, Theory and Practice – Theory" –course held spring 2013.

The summaries were written in English or in Finnish. The summaries are not in any specific order; only English language summaries are first. All summaries have three sections: Introduction, Results and Conclusions.

Timo Poranen

Tampere, August 2013

An Approach for Modeling Architectural Design Rules in UML and its Application to Embedded Software

A. Mattsson, B. Fitzgerald, B. Lundell, and B. Lings, ACM Transactions on Software Engineering and Methodology, volume 21, number 2, article 10, 2012

Background

An important phase in software development is the design of the software architecture. For this purpose, the high-level structure is presented as a set of components and interfaces; also a set of architectural design rules for infrastructure usage is defined. Currently, these rules are expressed informally and reviewed manually, which is an error-prone and time-consuming work.

A current suggestion for modelling design rules is to express them in a textual form and to address them to the resulting design, but this approach is not sufficient in cases when actual elements are not known yet. In addition, this approach becomes a bottleneck in a project where other processes have been automated.

Other formal approaches such as predicate logic, set theory and UML at the meta-model level have been proposed. In cases when architectural design rules are used to constraint the detailed design which is made using UML, the usage of UML must be restricted by those rules. However, they are not easy to understand and to use by both architect and developers because it requires more extensive knowledge about UML and OCL expressions become quite complex. Considering this, a more intuitive approach was needed, that will be easier to understand and to model, it will not require any knowledge about UML meta-model and it will be enough automated.

Results

The approach developed in this work is based on the purpose of achieving the objectives mentioned in the background part, i.e. developing design rules using UML, making it enough automated and usable in a real-life development project. A literature review, focused on the role of architectural design rules, was performed to achieve the first objective. For the second objective a tool that automatically checks the correctness of architectural rules was developed. To achieve the third objective, architectural rules of an existing system were modelled according to the newly developed approach.

The developed approach is using a set of transformations from model constructs to a UML profile. The set contains seven transformations, which are as follows:

1. A class with a stereotype is transformed into a new stereotype with the metaclass of the old stereotype extended, unless the second transformation applies.
2. If there is a metaclass, then there is already a class representing the language metamodel and is not transformed into anything.
3. If there is a role in the metamodel on the far end of an association of two metaclasses, then the multiplicity of the role for an element of the second metaclass stereotype shall be constrained to the multiplicity in the first stereotype.
4. If an attribute matches an attribute of class stereotype in the metamodel, then it is transformed into a constrained on that attribute

5. If for the attribute no matches were found in the class stereotype, then it is transformed into a stereotype attribute.
6. Any constraint of a class is copied to architectural rules profile, considering the class stereotype.
7. A generalization relationship from two classes is transformed into a generalization from the second class stereotype to the first class stereotype in the architectural rules profile.

This set of transformation is general and complete, but because it is still too complex to use, an additional UML-specific transformation set was defined. The purpose of the additional set is to override the main, in case when both of them apply.

To achieve the goal about making the architectural rules enforcement automating, a tool and a new method were developed. The tool is built in C++ and the model reader component is independent from other tool's parts, but is limited in reading models (currently, only Rhapsody model).

In order to prove that the developed approach is applicable to a real-life project, the architectural design rules of an existing system were successfully modelled according the transformation set proposed.

Conclusions

As an important part of software architecture, designing architectural rules needs an approach to allow design automation, UML usage without extended knowledge about UML metamodel and real-life project applicability is needed.

All these objectives were achieved in this work by developing a new approach to modelling architectural design rules. To allow automation, a tool was developed, which is used for Rhapsody modelling. The tool consumes approximately 200 man-hours, which is considered to be a relatively small task. In order to test the applicability, the approach was applied to an existing system, for which from the total 66 rules only eight could not be modelled because of their judgemental character. In addition, the developed approach is easy to use and understand by architects and developers, because it allows the rule to be modelled at an abstraction level close to the rules modelled.

Iulia Adomnita

Challenges with Software Verification and Validation Activities in the Space Industry

R. Feldt, R. Torkar, E. Ahmad, and B. Raza, in Proceedings of Third International Conference on Software Testing, Verification and Validation (ICST), pages 225-234, 2010

Background

Space related systems and applications need to be highly-reliable. The industry has to follow a set of particular standards and prescribes engineering processes and methods in order to achieve such a desirable quality. One of the standards associated with space systems is European Cooperation for Space Standardization (ECSS) which is the focus of this paper. ECSS is analyzed and evaluated, based on two sample companies which are applying those standards in their validation and verification activities (VVA), if its standards are cost-effective and suggest probable ways to make it more efficient without losing quality.

This paper contributes to the current challenge in three different ways. Firstly, it analyzes the identified and prioritized challenges in VVAs of different organizations in the space industry. Secondly, it evaluates the effects of ECSS on those VVAs and last but not the least, comes up with a proposal solution for existing challenge concerning V&V processes and their usage in the space industry.

Results

The study started with initial investigation that conducted using a web-based survey, an in-depth review of documents at the two sample companies- SSC and RUAG- in order to gather information regarding: a description of processes, software tools used, common defect types, defect detection techniques and also techniques to combine different VVAs.

In the next step, a connection between current defects and VVAs, current challenges and issues in V&V (verification and validation) processes and also effect of ECSS in addition to identification of solutions are evaluated. Finally, evaluation of solutions was conducted using questionnaires and informal discussions with V&V experts in the companies.

The questionnaires divided to four main themes that surveys respondents in terms of their understanding of ECSS standards, effectiveness of V&Vs, efforts required for different VVAs, and change in a particular VVA if ECSS is not relevant. For example, for the first theme, knowledge distribution of ECSS among both companies is identical or for the second theme, for RUAG the most effective VVAs are considered to be ‘code review’, whereas for SSC ‘unit testing’ is considered to be more effective. Also for the third theme, For RUAG, validation testing and system testing requires more effort compared to other VVAs, whereas for SSC it is system testing.

Regarding ECSS issues, the challenges and issues in Table 1 were identified during the two case studies. The empty cells indicate that it was not an issue or challenge for that particular company. For instance, taking ‘knowledge’ from the table, “at SSC the ECSS knowledge is unevenly distributed between individuals” (p. 5) or taking ‘inflexibility’, “at RUAG they find it hard to make changes to requirements during an ECSS project” (p. 5).

Factor	Challenge/Issue	SSC	RUAG
Reusability	Documenting quality when reusing development artifacts	Critical	Critical
Resource-intensive	Showing compliance takes resources from increasing quality	Critical	Important
Interpretation	Difference in interpretation of ECSS	Important	Critical
Increments	Limited support for (integration-driven) development in increments		Critical
Galileo standard	Differences between ECSS and Galileo		Critical
Knowledge	Distribution of ECSS knowledge in organization	Critical	
Innovation	ECSS limits innovation in processes, methods and tools	General	Important
Inflexibility	Hard to make changes/introduce new requirements during project		Important
Requirements	Documenting requirements for compliance proof	Important	
Tailoring	Unclear how to tailor ECSS	General	

Table1: Challenges and issues related to ECSS standards.

Regarding VAVs issues, the challenges and issues in Table 2 were identified.

Factor	Challenge/Issue	SSC	RUAG
Requirements	Unstable and non-testable requirements	Critical	Critical
Testing environment	Defects in testing environment and tools	Important	Critical
Integration testing	Limited focus on integration testing of software components		Critical
Reviews and inspection	Inadequate internal formal reviews and inspection	Critical	General
Unit testing	More focus on structural coverage than black box testing		Important
Independent V&V	Test cases are not reviewed by independent developer/tester	Critical	

Table2: Challenges and issues related to VAVs standards.

Conclusions

Based on identified issues and challenges recommendations were proposed in order to address the mentioned issues. Suggested recommendations include:

- Clarify reuse of artifacts and VVAs. Evaluate cost effectiveness.
- Allow alternative processes and consider simpler/quicker ways for process evolution to occur.
- Clarify relationship and motivate differences between ECSS and other common standards.
- Develop lightweight ECSS training material.
- Ensure the standard is primarily goal-driven. Clarify explicitly for big trends, e.g. model driven development, how they can be incorporated.

To sum up, in this study a variety of research methods namely, combination of questionnaires, document analysis and interviews were conducted in order to distinguish root causes of existing challenges in ECSS standards. Possible ways to address current issues in ECSS standards were introduced.

Reza Ahmadi

Interpretative Case Studies on Agile Team Productivity and Management

C. de O. Melo, D. S. Cruzes, F. Kon, R. Conradi, Information and Software Technology, volume 55, issue 3, pages 412–427, 2013

Background

A large amount of research has been conducted in Agile methodologies leading to potential improved productivity, shortening development time, and reaction times to market changes in the field of software development. It is accepted that industry has adopted Agile methods. Research has been extensively carried out to identify factors that contribute and influence productivity in software development. There are four main factors generally discussed. The product being developed (Characterization of the specific software), People (team members capabilities, experience, motivation), Project (management and resourcing), and processes (tools and software methods). This paper reports that little empirical evidence is known to be available to support this research. Using empirical evidence future research can be focused to those critical areas that will cause most productivity gains, and financial impacts. This paper outlines a detailed industrial case study on three companies over a period of 6 months to answer the question "Which factors do have impact, and in what way, on team productivity when using agile methods?".

Research has shown that team composition & allocation, external dependencies, and staff turnover are the main influences on Team Productivity. To answer the question the authors conducted this extensive study formulating first a thematic map. They compared this with their conceptual framework for agile development productivity. From this the conceptual framework is improved and more detailed reasoning provided based on the empirical results along with agile team productivity and management factors.

Productivity is controversial and varies due to context. There is no general consensus on measurement of software productivity for this reason. For this study the authors define productivity using team perception to quantify team productivity as a common measurement across the three companies.

Results

The research model chosen is started with Input-Process-Output (IPO) teamwork effective frameworks. IPO is a well known generic model in software development which has more recently been adopted for teamwork effectiveness and conceptual framework modeling for quantitative and qualitative studies. Initially a conceptual framework is proposed based on this IPO model where inputs, process and outputs are based on the agile principles. Figure 1, page 414. Inputs used are Individual group characteristics, Stage of team development, Nature of task, Organizational context, Supervisory behaviors. Outcomes are Agile Productivity, Attitude and Behavioral. Group processes are Internal and External. Note that Inputs directly, or indirectly via group processes, drive the outcomes as shown in this conceptual framework.

A multiple case study was conducted for six months based on three companies in Brazil. Selection criteria for Companies chosen was that they have used agile methods for at least 2 years, Projects in place for at least 6 months with at least 4 workers co-located, along with each business in different segments and geographical location. The length of the study was chosen to be six months to remove or identify productivity factors over time.

Company 1 provides financial systems, uses XP/Scrum and has one week cycles, employs 400 people and has 33% staff turnover. Company 2 provides E-Commerce, uses XP/Scrum/Lean principles and has 3 or 4 week cycles, employs 120 people and has 40% staff turnover. Company 3 is an Internet content provider, uses Scrum/XP/Lean principles and 3 week cycles, employs 200 people, and has 35.3% staff turnover. Table 3, page 419, further details the degree of adoption of agile methods in detail within each company. The data collected was retrospective document analysis, semi-structured Interviews and non-participant direct observation field studies.

Interviews were conducted by an author experienced in requirements elicitation and interviewing techniques. Interviews lasted one hour and were recorded. Interviewees were informed of the importance of the study, and included developers, project managers, product owners with differing experiences. No details were given to bias their information. Appendix A, page 425 gives the interview guide. Questions ranged from motivational, to influencing factors on productivity, and how the team operates.

Non-participative observation protocol is given in Appendix B, page 425. It outlines the questions the observers should consider and frequency they should be examined. The observational protocol questions were designed to detect factors in every day practices that effect productivity of the team. Analysis first consists of constructing a thematic map, often used in qualitative data to identify, analyze and report themed information. The thematic map shown in Fig4, page 417, identifies three themes. Inter-team coordination, Team member turnover, and Team design choices along with reasoning. At a later stage further information is added regarding impacts affecting these themes and whether they have positive or negative impact. The thematic map findings were discussed and reported using the conceptual framework. With this combination the thematic maps limited interpretive power and the frameworks ability to provide clarity and focus the research claims are better extended and communicated for discussion.

Team member turnover is a productivity factor producing cost in separation, advertising and recruiting, The relevance to this research is the team productivity is affected until the team member is integrated and up to speed, and loss of effective contributors when someone leaves. There was a positive side found when new members contribute new ideas and experience to the team. These positive and negative effects were clearly identified in interviews and retrospectives from the companies.

Team Design Choices affects productivity. This research identified desirable team attributes as full time allocation, diversity, skills, team size, co-location. Interviews identified that experienced and less experienced members provide more cohesion and flexibility, less conflicts. Small team size was found to improve communication and alignment, responsibility and commitment. Bigger teams have more communication and conflict resolution, more people to understand and align to the big picture. Social interaction of co-location improved communication and removed boundaries within hierarchical organized companies. Workspace layout considering desk positions and proximity also affected how well co-location worked. Although considerable influence is exerted by team design choices, most of the decisions are made by people outside the team, or their control.

Inter-team coordination affected productivity due to shared resources, prerequisite constraints, simultaneity constraints, relationships of tasks and sub tasks dependencies between teams. Inter-team coordination processes are then needed to manage these dependencies. Examples of inter-team dependencies found were external customers, QA, releasing, testing, integration, production, reuse of components under development or maintained in other teams. All three companies in interviews and retrospectives identified impediments waiting for resolution by external dependencies like this. Some teams owning components to be reused were not committed to the same goal, only the to the task to be completed. Other dependencies working on different schedules, like QA and integration.

Roles were identified as being used to influence and enforce practices and choices on teams. An example used was an administrator using their role to tell the team how to proceed in administrator activities, yet not belonging to the team or committing to the same goals.

In further discussion the influencing factors for team productivity are further explored. Productivity was particularly sensitive to team management. Agile teams take responsibility for managing their own work and behavior, yet others usually make decision about goals, team structure and organizational support.

Revision and extension of the conceptual framework during discussions was made. Figure 6, page 421, identified specifically staff turnover and effects on the agile team productivity using IPO, now including positive or negative contribution to the productivity. Figure 7 , page 421, identified specifically the team design choice factors and effects on agile team productivity. Whilst Fig 8, page 422, identified specifically Inter-team coordination factors and effects on agile team productivity. An extensive and more in depth elaboration is given for the working and reasoning of these drawing from the data that has been studied. The limitations of this study are well thought out and explained. Every effort was made to give correct finding and remove bais and false information .Credibility was promoted using well known and established techniques. Confirmability was ensured using multiple researchers working through the data. Dependability was ensured audit trail and traceability. Transferability was ensured by full disclosure of context settings, data extraction, synthesis processes used.

Conclusions

A detailed multi-case study has been given based on empirical research to find “Which factors do have impact, and in what way, on team productivity when using agile methods?”.

This is a very rich paper about effects on team productivity, which should be recommended to read for anyone running agile teams, and form a basis for further empirical research. Major factors identified are Team member turnover, Team Design Choices affects productivity, Inter-team coordination productivity. The conceptual model initially given has been further detailed in three individual conceptual models for each of those contributing factors and how they effect productivity which are soundly backed up by the research data.

Future research could be directed to give guides for managers and organizations setting up agile teams as they play such an important factor outside the control of the teams themselves. Multiple team dependency management has been identified as a performance hit for many years as well as highlighted in this research. Dean Leffingwell was quoted and has a book defining how larger organizations can use agile release trains to synchronize multiple team releases. This paper showed the problems due to asynchronous release cycles. It would be good to compare teams following Dean Leffingwell's suggestion of the agile release train for bigger organizations as a comparison solution for the issues highlighted here. An example is Nokia, which has implemented this synchronous Agile release trains and similar problems existed, but a proper study would be invaluable to extend this area and compare with the results found here.

This study was Brazilian companies. Although I believe this would be representative, a further study in different cultures would further solidify findings to remove any cultural influences that may affect productivity, for example was it the agile practices and people fitting into the team that was accounting for high turnover. Or is this a cultural factor which is influencing productivity but normal in these situations and not because of the agile team setup?

Andrew Cox

Interpreting an ERP-implementation Project From a Stakeholder Perspective

A. Boonstra, International Journal of Project Management, volume 24, issue 1, pages 38-52, 2006

Background

ERP systems are sets of integrated applications that can provide a total solution to an organisation's information system needs by addressing a large proportion of business functions including financial, accounting, human resources, supply chain and customer information. They support a process-oriented view of the business as well as business processes standardised across the enterprise (*Shehab, Sharp, Supramaniam, & Spedding, Business Process Management Journal, volume 10, issue 4, pages 359-386, 2004*).

This article aims to uncover some important dimensions of the organisational change issues around ERP-implementation projects by focusing on how ERP-implementation can impact the interests of stakeholders around the ERP system and how these groups may react by trying to influence the course of events and to alter the design in ways that are more consistent (p. 38).

Results

The study shows how the outcome of ERP-implementation can be affected by the meaning, which various stakeholders attach to such systems, and by the actions they take throughout the project.

Stakeholders possess one or more of three relationship attributes: Power, Urgency and Legitimacy. Combining these attributes generates eight types of stakeholders (p. 40-41):

1. *Dormant stakeholders* possess the power to impose their will on a firm but, by not having a legitimate relationship or an urgent claim, their power remains unused.

2. *Discretionary stakeholders* possess legitimacy, but have no power for influencing the firm and no urgent claims. There is no pressure to engage in a relation-ship with a stakeholder.

3. *Demanding stakeholders* exist where the sole stake-holder relationship attribute is urgency: those with urgent claims, but having neither legitimacy nor power.

4. *Dominant stakeholders* are both powerful and legitimate. Their influence in the relationship is assured, since by possessing power and legitimacy they form the dominant coalition.

5. *Dependent stakeholders* are characterised by a lack of power, but have urgent and legitimate claims. These stakeholders depend on others to carry out their will. Power in this relationship is not reciprocal and is advocated through the values of others.

6. *Dangerous stakeholders* possess urgency and power but not legitimacy and may be coercive or dangerous. The use of coercive power often accompanies illegitimate status.

7. *Definitive stakeholders* possess power legitimacy and urgency. Any stakeholder can become 'definitive' by acquiring the missing attributes.

8. Non-stakeholders possess none o f the attributes and, thus, do not have any type of relationship with the group, organisation or project.

The literature referred to a single case study that was focused to trace the dynamic nature of stakeholder behaviour (p. 41) and unfolding of ERP Implementation process. The study tried to identify different sets of stakeholders, interpretation of technology by the stakeholders, affect of ERP implementation on the interests of the stakeholders and influence on the implementation process.

The research shows that implementation of ERP is a socially, politically and technically complex under-taking that influences nearly every aspect of organisational functioning and thus affects the interests of many different stakeholders during the various stages of the project. ERP-systems are building blocks of organisational strategy. This means that ERP-implementations resemble the nature of the strategy processes, which often have emergent and adaptive characteristics rather than planned. Changes in the external environment, the emergence of new opportunities, and other unanticipated events make departure from initial ideas often inevitable (p. 50).

Different stakeholders can interpret ERP-systems in different ways, given their own histories, interests, self-images, prospects and views. Some groups perceive the system as a means to realise certain new company wide objectives, while others see the system as a way to regain lost power or as a threat to legitimate local interests (p. 49).

ERP-implementation is a dynamic process, which means that views, which are held by the stakeholders at one point in time, may change during the project. This may depend on various reasons, including cognitive, political and opportunistic ones (p. 50).

Conclusions

ERP-projects is a complex cock-tail of rational assessment mixed with various perceptions, quests for power, leadership and subtle processes to gain support for further progress of the project (p. 50).

ERP-implementations challenge vested interests, and lead to opposing views from various players. Implementation includes multi level activities, which are not the exclusive area of a single project manager (p. 50).

Moreover, good project management of ERP-projects goes far beyond the technical implementation of the system. ERP-implementation can be perceived as an organisational change project and should be managed accordingly.

Different stakeholders may change their roles during the project. Establishing more stable structures where members of senior management play a role, also during the implementation phase, may support the realisation of the initial project goals. Project managers should be aware of this and stimulate top management presence (p. 51).

Eashan Salhotra

How Do Top Managers Support Strategic Information System Projects and Why Do They Sometimes Withhold This Support?

Albert Boonstra, International Journal of Project Management, volume 31, issue 4, pages 492-512, 2013

Background

It is generally agreed that the support of top management in strategic information system projects is an important success factor. The relationship between IS projects success and top management support is quite a studied subject, but there is only little reliable knowledge about the types of behavior top management support contains.

In prior studies top management support has been treated as a single construct, something that can be given or withheld. The paper of *Albert Boonstra*, however focuses on the concept of top management support.

There are three main questions the research aims to answer: "*1) What behavioral types are associated with top management support for strategic IS projects? 2) How can these behaviors be placed in a coherent framework? and 3) Why do managers sometimes withhold these types of support?*" (p. 501).

Results

In the paper the author introduces a framework of top management support behavior types and aims. The framework is developed by using a multiple case study design. The research involved five case studies of IS implementation projects. All of the projects were company-wide, involving multiple business units or integrating IT systems, strategic in nature and complicating in terms of duration, budget and user groups. All companies had several levels of management and their IS project already in the roll-out phase.

Organizations and projects involved in the case studies were as follows: an ERP project of a manufacturer, aiming at aligning the companies systems' with the business strategy of the company; a CRM implementation project of a large financial service provider, aiming at improving customer service and including a vast number of employees in more than 100 local branches; an ERP integration project of a dairy food company, aiming at integrating the company's numerous IT systems; an e-government project of a city council, aiming at providing e-services for the residents of the city; and a electronic he alt record integration project of a hospital, aiming at integrating 24 different electronic patient record systems inside the organization.

The research team prepared the interview questions by first studying other research models and theories on the topic and familiarizing with the case study organizations and projects through other data sources, such as observations and internal project documentation. In addition to the representative of the organization's top management, other project members, such as business managers, project managers and workers, were also interviewed. An average of 7 interviews was made in each organizations with an average of 5 organization members.

Based on previous research literature and the research material, five top management behavioral categories were identified. These categories were used to classify the supportive top management behaviors identified during the research. The five categories were: 1. resource management of financial, material and human resources, 2. establishing and

enforcing of the IS project's and organizational structures, 3. enthusiastic communication about the project, 4. sufficient knowledge regarding the management and content of the project, and 5. using of the power to resolve possible conflicts and to protect the project team. The framework for top management support also includes four agendas that different types of behavior aim at. These agendas were "*Accommodating the implementation project*", "*Reshaping organizational context*", "*Adapting the IS to the organization*" and "*Dealing with stakeholders*" (p. 503). When combining the aim with the category as a result the research introduced a framework with total of 20 definitions of top management support behavior types, and then analyzed how many of each support behavior type was detected in each IS project.

Cross-case analysis revealed variance in overall top management support in the case study projects. But also the perceived adequacy of top management support variated. The support of top management was perceived adequate when "*the overall amount given by the top management matches the perceived amount needed*" (p. 508). The amount and type of support needed fluctuated in different stages of the project.

To answer the last of the three questions about the reasons why top management withholds supports, the researchers found out that top management struggled with the capabilities to give support: there was not enough expertise, resources nor time. Another reason was the ever changing goals and context of the IS project. It is hard to give support to a project if it starts to look like a failure. And the disagreement among other top managers of the right support type was also seen as a possible reason for a low level of support.

Conclusions

The main conclusion of the research seamed to be that "*the top management support is a context-dependent balancing act involving several types of support, with various intensities, aimed at a range of goals which may change over time*" (p. 509). The flexibility of top managers in how, when and at what intensity they give their support was seen as one of the key issues. As well as understanding that different kinds of IS projects in different kinds of environments have different needs when it comes to top management support. There is no one-and-only right way to act.

Siiri Tammisto

Project Scheduling With Limited Resources Using a Genetic Algorithm

J. R. Montoya-Torres, E. Gutierrez-Franco, C. Pirachicán-Mayorga, International Journal of Project Management volume 28, issue 6, pages 619–628, 2010

Background

The article is focused on solving a Resource-Constrained Project Scheduling Problem (RCPSP) with a certain kind of genetic algorithm. In earlier project scheduling models taking the resource constrictions in all levels in mind were not taken to count in the algorithm, but were the responsibility of the project manager to take in mind differently in all parts of the project. The current approach handled in this research is based on the idea of taking resource restrictions in mind in project management with efficient scheduling based on object oriented programming model. The object orientation allows the project to be cut into instances based on the concept of activity list. These instances can also be seen as chromosomes of the project which can create evolution in project due to certain kinds of mutations. The mutation is based on crossover of two instance nodes. The start and ending nodes are time valued for 0 and therefore can be called as dummy nodes. Genetic algorithm is based on simulating the real evolution methods inside a software project. Evolution strategy is based on selecting the best individual node or individuals and then uses it or them to generate a set of new individuals in the next generation. In a project based on such algorithm there would be constant evolution and progress to some direction in the project considering the scheduling the object orientation in programming issues and resource management (as the algorithm takes the available resources in count).

Results

The algorithm discussed to be genetic that was used in this article was programmed with Java. Computational experiments were ran on the algorithm. The experiments were experimental on their nature. The goal of the experiments was to count the crossover probability and mutation probability with certain amount of individual values, the amount was either 100 or 150. Over 2400 instances were used. Also the algorithm was compared with other state of the art algorithms previously used in scientific literature. From the researches referred to only other algorithms based on the genetic approach were taken for evaluation. The tables in the article show that there was sufficient enough results by the algorithm. Though the solution value never was greater than 10% and when the amount of individuals was raised from 100 to 150 the differences with previous experiments with other similar algorithms decreased. Taken in count in the computational progress there was also a fitness function involved to define the optimality of chromosomes of the certain individual taken account. The fitness function was based on the idea of evolution strategy and supports the idea of mutation. Though low values seemed to give more valuable results towards empirical studies around the topic. The result bound conclusion proposed that is not possible to find actually any groundbreaking optimal solutions for instances of higher value in node quantity in reasonable computational time.

Conclusions

The research is again only one computational and object oriented programming based

approach in creating a genetic algorithm for organizing efficient project scheduling. The results are not groundbreaking but they were encouraging for the research team according to the article. Their further research will include non-renewable resources and other objective functions to include activity costs.

In connection to this I might appoint some criticism towards the article that these issues were not handled clear enough in the begging of the article in my point that which kind of resources and costs the researched algorithm in fact was able to handle, but of course it is understandable that when talking chromosome level in any topic the perfection must be built in levels and new features will always be added due to evolution, not just in the algorithm but also in the research.

Maybe a right algorithm eventually would solve core problems in resource-constrained project scheduling as evolution in earth has created a species such as humankind to solve some problems that the previous primates were not able to solve, but maybe it could be arguable if this is all just based on computation, but this is just my opinion. I would say that good scheduling is more up to good humanistic management not just calculated results.

Mikko Myllylä

Building Theories of Project Management: Past Research, Questions for the Future

J. Söderlund, International Journal of Project Management, volume 22, pages 183-191, 2004

Background

The field of project management has been an issue of controversial debate over past decades. The aim of this discussion is to promote the standardization and certification programs for project managers. To the day several institutions and associations have been established with prime goal of bringing professionals together and creating an effective network. The members of these networks come from various disciplines such as psychology, pedagogy, business administration, organization theory, industrial engineering and sociology.

Despite the rapid development seen in recent years the field of project management has been blamed for too narrow focus and almost an entire lack of empirical studies. Project management researches have been accused for being too focused on studying the reasons for success and failure of projects instead of examining the big picture. As a result many important questions that are at the core have not been addressed properly.

The diversity of the field of project management introduces many challenges that need to be tackled. While increasing the complexity of the field the multidisciplinary nature of project management also complicates an agreement on fundamental ideas of project management research. The aim of the original paper of this summary is to raise a discussion and debate about some fundamental theoretical issues related to project management research.

There are two major lines in project management field to follow. One line traces the intellectual roots of project management research to various types of planning techniques. This approach presents project management as a specific problem-solving method with strong connotations to optimization theory and applied mathematics.

The other dominant lineage traces the project management research to completely different intellectual roots. The process of managing projects is seen as a broader discipline where projects represent empirical entities. The biggest difference between these two main theoretical traditions is in the way they look at definition. One tradition traces its intellectual roots to the engineering science and applied mathematics where another tradition leans towards social sciences with special interest in the organizational and behavioral aspects of project organizations.

One of the fundamental controversial topics in project management research is associated with the perspective to look at projects. For some researches projects are nothing else than a way of looking at industrial and organizational activity. Thus researching into projects is more a matter of looking to capture the unique, complex and time limited processes of interaction, organization and management.

The alternative point of view is to stress the importance of providing knowledge and theories about the organization and management of projects. This approach emphasizes the importance of project dimension as a universal entity. Typically in a project context the universal elements are normally uniqueness, task complexity and time-limitedness.

Results

It doesn't come as a surprise that the majority of modern organizations are relying on projects. The term project itself is often used to describe the observed pattern of organization or interaction. The important question to be answered is why firms and projects exist in the first place. Answering this question is not only the matter of philosophical discussion.

The major role of theories and models is to enhance our understanding of the field or phenomena we're interested in. Despite the broad attention in the scientific literature there are only a few studies discussing the behavior of project organizations in theoretical terms. Partially this is due to the lack of available options. Many of models and techniques found in the project management field fail to explain or increase our understanding about the behavior of project organizations.

The existing project-oriented literature does not explore thoroughly all the aspects of projects. Especially the difference between "temporary" and "permanent-like" organizations has not been discussed extensively. As a result many important questions have not been answered. How do people react on time pressure and control by deadlines and milestones? These sorts of questions have to be tackled in order to create new and evolve existing theories.

Conclusions

In the profit seeking business world organizations usually relies mostly on measurable metrics. This mindset inevitably translates also to the project management research. Areas like critical success factor are of particular interest. The downside of concentrating only on critical success factors is that it easily ignores the behavioral dimension of project organization. When thinking about the success criteria it is necessary to broader the traditional triple constraint model (cost, time and conformance to specifications).

Elvis Okemou

The Strength and Weakness of Requirement Engineering (RE) Process

A. Haron and S. Sahibuddin, in Proceedings of the 2nd International Conference on Computer Technology and Development (ICCTD 2010), pages 56 - 59, 2010

Background

Requirements engineering is an important part of the software project development. The goal of RE is to understand the needs of the stakeholders and by being able to provide the software engineer with tools and techniques which help understanding the process better, in order to have a software that meets the client's needs. There is also a question as to why there is a need of RE in the developments of software projects, The need is that RE helps the developer to better understand before the actual development process begins, this helps in reducing the cost of rework in a software lifecycle. The RE process is a part of the software project management cycle, RE ensures the overall quality of each stage of the product development. In simple terms requirements are the specifications of the product, meaning they are the descriptions of how the product or the system should behave. This study tries to reduce the gap with comparison to the current requirements practices to the appropriate of requirement best practices.

Results

There is a Gap analysis made in order to identify the critical issues of the organization, when the analysis was made there were six components which were listed out. They were identified after the implementation of some software projects. They were divided into six components, therefore, *managers, stakeholder, developer, business rules, business process and technology,* thus making them all either inter related or directly related to each other. (p. 56)

The first component is Managers: The main process of business is to deliver services to the public sector, in due course they have improved their services by creativity and innovation. The manager of the software project plans everything from the implementation to the layouts to the time frame of the project. The manager holds himself responsible for the requirements, release and the product life cycle. This also means that the success of the product depends on the skills and competence of the manager. This paves to the risks and success factors they need to take into consideration when facing challenges, some of them include defects, delays etc. (p. 57)

The second component is Business Rules: when developing a project there are many things which need to be taken into account one such important factor when it comes to Business rules is to provide support for an organizations business strategy keeping in mind the business success. When there is an initiative taken to improve the service delivery there are some constraints in achieving the goals. The main constraint being the people and the management for approval. (p. 57)

The third component is business process: The success of the software project depends on the business process we follow; they keep changing from time to time depending on the system application. This also determines the complexity of the project development. The projects which are most successful are those which have a excellent process and a clear team structure. When a stakeholder requests for a requirements change this aligns with the process change also this helps the requirements provider either a stakeholder or a customer to contract

and realize a software solution. (p. 57)

The fourth component is stakeholder: The most important component in the business process is the stakeholder. They are the ones who know what requirements are needed. Some of the issues that occur are in understanding the stakeholders' needs. In some situations there is a miscommunication between the developers. This is due to the gap between the analysts and the stakeholders; this is also due to the variation of the future needs of the system users with the actual needs. (p. 57)

The fifth component is the technology: The key factor when it comes to technology is that it should meet international standards. There are also constraints in meeting the expectation of the stakeholders, this is minimized with two things namely mapping and capability. This is then compared to the current infrastructure, software and hardware. Then there is the system of online where the stakeholders receive the services anywhere, anytime. This results in globalization of the standard. (p. 57)

The sixth component is developer: The developers act as middle managers between the stakeholders and the managers. Any technical decision regarding the project needs to be consulted with the developers. Communication between the developers and domain experts is needed. The knowledge of the developer often aligns with the technology and helps in fulfilling the stakeholders' expectation with regard to proper market research and globalizing the project. (p. 57)

This paper suggested the RE process that were experienced by IT personnel. There are Elicitation process; Analysis and Negotiation Process; Documentation Process; Management Process; and Validation and Verification Process. This leads to the study implemented to identify the strength and weakness of RE process.

Strength:

The needed of the systems: This is the need to improvise the service delivery with comparison to other country or among application for the organization.

To improve the current process: The goal of the system is to develop a system faster, cheaper, and safer. The goal of the improvement process is to minimize defects and maximize productivity. The task of documentation helps in identifying the weakness and potential risks which are very important for RE process.

This paper also specifies the reasons for choosing components Of The Shelf (COTS),one being which the system serves as an existing surrogates, two being the adaption of either the whole or a part of the solution, three being the level of confidence it increases in the users. Furthermore the paper discusses the models and techniques used for the system development, and the need to choose the right kind of techniques for the project. (p. 58)

Weakness:

Lack of RE knowledge: Most IT personnel lack the knowledge of RE as they come from the background of programmers, testers, managers. The developers perceive the RE process to be less important when compared to other phases of development.

Lack of REP elements: Requirements Engineering Process involves managers, domain experts, developers who share different skills, competences, backgrounds. These include disciplines such as elicitation, analysis, review, communication identification of problems. RE also emerges as a process of learning in which they are prioritized, negotiated, evaluated and documented.

Lack of international standard implementation: It's not an easy task to adopt the available standard, the management maps well the guidelines and hints in applying the concepts,

approaches to the software projects. This implementation of international standard does not ensure a successful project. They need to be properly tailored to the level of the project in hand.

Difficulty to adopt RE model: The difficulty is due to the methodologies taken into account based on the experience of previous failures. It's hard to convince the methods or model applied will work for another environment.

Less exposure on RE technique: There are minor problems such as the success to the size of the projects, type and people involved in the project. There is no single solution which works for all the RE problems, as this is due to the diversity in the stakeholders. (p. 59)

Conclusion

This paper analyzed the RE process to go through a plan, first, proper training for the IT personnel in order to get familiarized with the RE process, second, exposure to real time environment to gain knowledge, third, the level of expertise in the field for the IT personnel is very important. As a conclusion, this research found strength and weaknesses in RE in order to reduce the gap of current requirement practice in Public Sector based on issues identified. The finding of strength and weakness of RE will help us to reduce the gap between current practice in the Organization with appropriate practice.

Thrushna Nalam

Factors Associated with the Software Development Agility of Successful Projects

J. Sheffield and J. Lemétayer, International Journal of Project Management, volume 31, issue 3, pages 459-472, 2013

Background

This paper from Jim Sheffield and Julien Lemétayer aims to find an answer to the question of which factors in a software project imply an agile way of software development. Agile software development methods are nowadays widely adapted and often preferred instead of the plan-driven traditional software development methods. These different schools of thought can also be (and very often are) mixed together. The authors point out that in order to get the most out of the project, the level of agility in development should be as good a match as possible to the properties of organization and to the development team.

The data gathering of the research was done in two steps. In the first step, a group of project managers were interviewed. Based on the interviews and literature research, a set of key indicators of software development agility were selected for a short web questionnaire.

The questionnaire was sent to a large number of project groups. The answers were then analyzed with statistical methods to find the key factors that would correlate best with software agility in the projects. In the literary, there is discussed a large number of concepts and measures that indicate software agility. In this research Sheffield and Lemétayer aimed to find the most important measures that would correlate most with the development agility of the project.

The hypotheses for the research were that there are factors in the project environment and in the project itself that are associated with the approach to the projects' development model. These values would reflect the development agility of the project. The factors being researched were split into two groups: To those belonging to the project environment, and to those belonging to the project itself.

Results

The answers of the questionnaire seemed to point out that the organizational culture influences the level of development agility in the project. In many cases, the development routine, no matter whether it was plan-driven or agile, was found out to be an integral part of the company culture. This factor (organizational culture) was the most important factor *in the project environment* and was found to support the hypothesis H1. Of the project factors, the "empowerment of the project team" was found to have the highest correlation with the software agility of a successful project and was found to support the hypothesis H2. In comparison of H2 to H1, the project team empowerment was found to have the highest correlation of all the found factors, it being even higher than the organizational culture factor.

In the project managers' interview stage, the level of experience of the project team was deemed to have a high importance in the successfulness and selecting the *appropriate* level of agility to the project. This "level of experience" measure includes the project team's education level. However, the level of experience of the team did not correlate with the actual level of software agility, measured in the second phase of the research. The interviews with the project managers also showed that the certifications and individual training level of the developers was not an important factor when determining the project agility level. This was

most likely due to the fact that the project members received a comprehensive training and induction to these rather large scale projects involved in the research.

The two main factors that support the original hypotheses enclose also a number of variables that explain each factor more in detail. The "empowerment of the project" -factor has relations to a number of variables which were analyzed with factor analysis. The analysis revealed, that some components (variables) of the factor have a more important role that others. The most important ones were found to be agility supported by the customer, procedural empowerment and close customer collaboration.

Conclusions

The implications of the research are manifold, according to the authors. The research revealed that both the project environment and the project clearly affect the software development agility of the project. Secondly, it is shown that the appropriate level of agility of the project does have a significant correlation, and thus should be judged on a project-by-project basis by the practitioners, rather than always utilizing the most hyped or person's favorite development method. From the results and interviews it is also easy to see why the project team, project management, customer and the top management need to agree on the correct level of development agility.

The study lays a basis on a more intelligent approach to selecting an appropriate development model by first setting a number of key indicators that should be evaluated when a new project is about to be started. The result of the evaluation would determine which level of agile approach would be the most optimal. It is possible that the indicators would indicate a not-so-optimal environment and starting point for the project. In this case, a mixed approach utilizing both the traditional and agile methods might then be used.

The study goes on to show that in order to the project to be successful, its development model needs to have a correct level of agility in relation to the factors in the project and the project environment. This implicates that there is not one correct development method for software projects, but in order to successfully carry out the projects, the top management and project managers would need to be aware of the importance of selecting the correct development method. This research paper would act as a fine starting point for further research to the field. It is also acknowledged by the authors that more research is needed in this area, as this research does have its limitations and more accurate results might well be accomplished with larger data sample and more detailed interviews and data analysis.

Lauri Maasola

Requirements Engineering and Agile Software Development

F. Paetsch, A. Eberlein, and F. Maurer, in Proceedings of Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 308-313, 2003

Background

Requirements engineering is the first phase in software development process. Requirements are specifications of the services that the system should provide, the constraints on the system and the background information that is necessary to develop the system, requirements are, conditions or capabilities needed by a user to solve a problem, condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document, and documented representation of that capabilities. Agile Development is software development process model .Agile methods are code oriented approach, it is less documentation centric, it depends on clients involvement in the process.

Extreme Programming XP

It is one of the agile approaches, it depends on co-presence of the whole team and collecting feedbacking to locate the current situation, XP introduce simplicity and team communication. It focus on general software process rather than concentrate on requirements.

Scrum

It concentrate of how the team work together to produce quality product , it adopt values like flexibility, adaptability and productivity. Scrum rely on three techniques backlog, sprints and daily scrum. Requirements Engineering process represented in Scrum as backlog it list down the requirements. Backlog prioritize features, functions, and bugs .Backlog considered and incomplete requirements document .During sprint there is no flexibility for requirements change . Sprint meeting are similar process to requirements review process.

The Crystal methodologies

Are collection of methodologies from which members are tailored for specific project depending on project characteristics .Different members are indexed by colors (Clear, Yellow, Orange, Red, Magenta, Blue, Violet) representing heaviness. Crystal clear and orange are similar techniques to requirements engineering.

Feature Driven Development (FDD)

FDD focus on two phases of the software development process (design and implementation) with short iteration .At first phase class diagram is it represent entities , attributes ,methods and interrelationships .methods introduce system functionalities that are the basis for features, those feature are prioritized in features list and reviewed by domain members , a weekly 30 meeting is established for features status review, the meeting is reported and that report can be compared to requirements tracking document.

Dynamic Systems Development Method (DSDM)

It is rapid application development framework. Requirements are elicited at early phases (feasibility and business study), as development process advance new requirements are elicited . Requirements Engineering techniques can be used during development process. The main focus of DSDM is testing it spread over the whole development process.

Adaptive Software Development (ASD)

It is Iterative , incremental prototyping development process for large and complex systems. After a review of working application a change request is proposed .

Results

Requirements Techniques in Agile development

The main intersection between requirements engineering and Agile development is customer involvement , there should be a propriate customer representative during the process. Different RE techniques can be applied to AM methods as following

Interviews

All agile methods use interviews as the primary techniques to obtain required information and clarify misunderstanding .This kind of interaction is a trust builder between developers and customers.

Prioritization

It is iterative technique over development process to prioritize required features, as new requirements arise the prioritization repeated to get business value.

JAD Sessions

It encourage customer involvement ,heavily used in Adaptive Software Development and Dynamic System Development Methods, used at early phase to understand the new system. Sessions are documented for further enquiries.

Modelling

As discussed previously it is the main technique used in Agile modeling (AM) . it is used to provide common understanding of the system components during development process . the main difference between models in AM and RE , models are volatile and not documented in AM while it is persistent and documented in RE .

Documentation

Requirements documentation It is not primary goal in most Agile methods, but it is used in some methods like Scrum (Backlog).

Validation

Review meeting is the basic validation technique, it is frequently used by both parties(customer and developers) to ensure schedule validity and features implemented .In review meeting customer can suggest requirements changes .In addition acceptance tests can be run by customers to validate software.

Ethnography, Brainstorming and observations

Those techniques heavily used in RE, but it is not explicitly declared in agile methods although it is recommended in requirements elicitation process for completeness and consistency.

Conclusion

Software development process consists of four activities, requirements specification (that is concerned with defining final product and related constraints on it operation), development, validation and evolution. There are different Software process models the most common waterfall model, V-model, RAD, and agile methods.

Requirements Engineering following activities elicitation, analysis, specification and validation. Requirements and agile methods are common in interacting directly with stakeholders thus requirements techniques for example brainstorming, JAD sessions and interviewing are applicable and effective in AM.

Ahmed Dawoud

Improving Test Efficiency Through System Test Prioritization

H. Srikanth and S. Banerjee, Journal of Systems and Software, volume 85 pages 1176-1187, 2012

Background

Software testing is an important part in software development, which also is a time consuming, complicated and expensive process. Software testing is the last opportunity for companies to fix the problems before they deliver the product to the users and clients. However, not all the companies are resourceful for software testing. Therefore, how to ensure the quality of software testing with the limitation of budget and resource is the research question in the article. There are many types of testing, the authors limit the topic in the system testing.

In the background part, the authors did the research of Test Case Prioritization (TCP). By using TCP, the test cases were ordered and effectiveness of testing activities was increased. However, the TCP were generally applied in white box which was not system level. The authors consider the goal of topic is to apply TCP to black box test and decrease the failure rates in the system level. The authors did some further research about related approaches. There are mainly three parts of related work: coverage-based test case prioritization, software reliability engineered test (SRET) and model based test prioritization. The mainly strategy in coverage-based test case prioritization was Average Percentage of Faults Detected. It helped detect the failure rate quickly, but not effective when the faults are not equally severe. Software reliability engineered test (SERT) used an operation profile to estimate the functions relative customers. Even though SERT is a positive approach in test prioritization, the authors decided to modify and maintenance the SERT to be resource intensive and purpose non-operational. The authors consider customers assign is the priority scheme in the further research. Model based approach was low cost but very depend on the accurate information which the developers and testers provided.

Results

The authors proposed an approach named Prioritization of Requirements for Test (PORT) in this article. Then authors evaluate this approach in two case studies comparing random prioritization.

There are four factors in the PORT: customers-assign priority (CP); developer perceived implementation complexity (IC); fault proneness of requirements (FP); requirement volatility (RV). The authors give the reasons of these factors selecting. For customers-assign priority (CP), customers can grade in the requirements from 1 to 10 point. 10 point means the highest priority from the customer side. By doing research of Munson and Boehm's study, authors consider the CP is necessary and important. The testers should test the requirements with higher priority. Developer perceived implementation complexity (IC) is to measure the difficult level of implement a requirement. Testers should give the priority of requirements with high complexity grade. According to Amland studies, a high complexity requirement tends to result to more faults. Development team identifies the fault proneness of requirements (FP) by using their previous experience that which requirements are likely to be error prone. The requirements volatility can be range from 1 to 10, which identify how likely the requirements will be changed. The changing of requirement usually results into the failure of the project.

The stakeholders should include customer, requirements analyst, maintenance engineer, developer/architect and tester. The stakeholders grade the requirements in four factors and the four factors values are used to compute the Prioritization Factor Values (PFV). The PFV are used to compute the weight priority (WP) of test case. The authors validate the PORT by analyzing the severity of failures. There are four levels of severe: highly severe, medium severe, less severe and least severe. The authors evaluate the PORT by applying the approach into two case studies comparing with random prioritization. The case studies are industrial cases and the results show that the PORT is useful and easier to detect the failure in the test.

Conclusions

In the PORT approach, some positive approaches in software testing have been used, such as SRET and TCP. But the authors modified some approaches in the consideration of testing efficiency in system level. Personally, PORT is a good approach in software testing and almost all the stakeholders are involved. Customer-assigned priority (CP) is especially an important factor in improving system test efficiency since no customers are involved in the deep level like this in the previous approaches.

He Ye

Agile Project Management in Product Design

M. Molhanec, in Proceedings of the 31st International Spring Seminar on Electronics Technology, pages: 399-403, 2008

Background

This article considers the role of project management in product design.

Nowadays, competitive marketplace demands high quality and a fast track from the beginning to finished product. Project management has an important effect on developing product quickly with high quality. Therefore, for being successful, powerful project management should be applied to the product development process. In order to improve project management within product design, the author uses some knowledge of software project management in addition to ICT and KM tools.

Project management is not an easy process since achieving all of the project goals is difficult. There are many challenges such as time, scope, budget and etc. In this article, the primary software project management, the Object Oriented Software Process (OOSP) by Scott W. Ambler, is used.

Traditional project management method in product design doesn't consider the whole lifecycle of development product. These approaches focus on developing new product rather than preceding and improving the old version. As a result, the simple routine design management encounters failures. However, in the product lifecycle management (PLM), the entire lifecycle of product from the beginning to manufacture and service is considered.

PLM, in simple term, is a business strategy for creating a central product management and environment and the technology used to access this information. PLM is based on CAD (computer-aided design), CAM (computer-aided manufacturing), and PDM (product data management) and it can be defined to connect various stakeholders during the whole lifecycle of product from concept to retirement. In other words, PML is the collaboration of methods, stakeholders, and processes with set of tools mentioned along all stages of a product's life.

It should be noticed that PLM is not a single software product. It is actually the combination of software tools and methods together to either manage the whole process of product or single stages of lifecycle. Many software applications have developed to organize and integrate different stages of a product lifecycle. One of the most important goals of PLM which makes it different from traditional project management is to collect information that can be used for other projects and to organize the development of many products at the same time.

There are several PLM lifecycle models in industry, but the one emphasizing hardware-oriented product is shown in this article. (p. 400)

Phase 1: Conceive

Imagine, specify, plan, and innovate

In this fundamental stage, the requirements from customers, company, marketplace, and in general stakeholders' point of view are defined. Then, more technical features can be outlined according to these requirements. At the same, the main functional aspects of product and basic concept of product design are specified at this stage.

Phase 2: Design

Describe, define, develop, test, analyze, and validate

Detailed development and design of the product from beginning, and testing to launching is identified at design stage. It is also included the redesigning to improve existing products. This phase covers many engineering disciplines such as mechanical, electrical, software. Finally, simulation, validation, and augmentation tasks are carried out using CAE (computer-aided engineering).

Phase 3: Realize

Manufacture, make, build, procure, produce, sell and deliver

After finishing the design phase, the method of manufacturing is defined. This also involves analysis tools for simulation of operations. Moreover, documentation work for sales product in marketplace is done alongside engineering tasks. For instance, engineering information transferred to a web based sales.

Phase 4: Service

Use, operate, maintain, support, sustain, phase-out, retire, recycle and disposal

Managing of service information takes place in final phase of the product lifecycle. This includes providing service engineers and information for back up, support, repair and maintenance of products. The “end of life” to every product should be considered since it may have side effects.

However, this traditional approach has other limitation. They can describe project management freely and several approaches exist but there is no definite standard method. Therefore, the author presents agile project management by Scott. W. Ambler to apply in his approach.

Results

The approach used in this article is based on agile project management by Scott Ambler. The lifecycle of the object-Oriented Software Process (OOSP) consists of four process patterns. A process pattern is the collection of activities such as techniques and tasks to solve a specific software process problem. In other word, it describes the verified solutions to general software process problems like design pattern. Ambler noted that process patterns are an excellent mechanism for communicating approaches to software development that have proven to be effective in practice. A significant feature of a process pattern is that it defines the process that should perform and it does not specify the exact detail of how to do the tasks. (p. 400)

There are four project phases with OOSP which is similar to the traditional approach: Initiate, Construct, Deliver, and Maintain and Support. These phases represent the interactions between the various stages for a single project phase. There are also fourteen stages in the OOSP: Justify, Define and Validate Initial Requirements, Define Initial Management Documents, Define Infrastructure, Model, Program, Test in the small, Generalize, Test in the large, Rework, Release, Assess, Support, and Identify Defects and Enhancements (change management). A stage describes the tasks, which are usually done in iterative manner in a single project phase. However, project phases are performed in a sequential manner in OOSP. (p. 401)

In addition to the OOSP’s phases and stages, there are more important tasks which are crucial to the success of project and they can be applied to all stages during the whole product life cycle. These tasks include quality assurance, risk management, training and education, people

management, reuse management, metrics management, and deliverables management. (p. 401)

Several strong and weak points exist for the OOSP. As strong points, the OOSP is a comprehensive methodology that considers the whole lifecycle of software development. It shows that large-scale software project is sequential in the large and iterative in the small opposed to traditional approaches that are not iterative. The OOSP lifecycle contains a maintenance and support phase which plays an important role in the success of projects and reminds developers to consider support and maintenance during the development of software. Nevertheless, the OOSP can be amazing for software developers that prefer to focus on small part of process. Moreover, like Open Process the OOSP suffers from market challenges.

As it mentioned above, Ambler methodology is used for software development. And the author transfers this methodology (the agile project management) into general project management in the product design. The main difference between software and product project management is also specified. Both software and product project managements are the same in high level of abstraction but there are different in details (process pattern stage). However, product design project management is more confusing since it is more general and heterogeneous. Therefore, the author suggests following augmentations in order to transfer Ambler approach to product project management: (p. 402)

Changes in the second phase (Construct): in construct phase, Producing Stage instead of Programme Stage in Ambler approach is proposed since Producing Stage is more general. And there is not any change in other stages as it does not matter that we produce software or other technological product. Moreover, some tasks like measurements are easier to do in technological products than software products. (p. 402)

Usage of ICT and KM Tools: the author strongly recommends that Information and Communication Technology (ICT) and Knowledge management (KM) tools are necessary for supporting the project process which these tools are not used in Ambler approach. Moreover, in the proposed approach, UML standard diagrams for states and activities are used. Mind Mapping and Topics Map will also be used for thinking process. (p. 402).

Conclusion

Product design project management can be improved by transferring some knowledge from software project management. Both kinds of project management are almost similar in low level and there are differences in high level of abstraction which include outputs (hardware or software) and detail of one phase (Construct phase).

Furthermore, proposed approach is based on using the UML standard for graphical diagrams of process and activities. The ICT and KM tools are also used since they can support the whole process of project management.

Golnaz Sabet Nejad

Successful Extreme Programming: Fidelity to the Methodology or Good Teamworking?

S. Wood, G. Michaelides, C. Thomson, Information and Software Tecnology, volume 55, issue 4, pages 660-672, 2013

Background

Software development process today requires a group of individuals working effectively as a team and it is crystal clear that the performance of a great teamwork far exceeds the work of individuals acting independently. Extreme programming (XP) is an agile software development methodology which particularly emphasizes good teamwork. The practices which sets XP method apart from other agile development methods are all interdependency oriented such as pair programming, collective code ownership and frequent, short team meetings (p. 660). XP practices are generally categorized into 4 categories which are "Foundations (Testing, Pair programming, Refactoring)", "Customer planning (Planning game, Customer access, Short releases, Stand up meetings)", "Craftsmanship (Sustainable pace, Simple design, Use of metaphor)" and "XP-specific team factor (Continuous integration, Coding standards, Collective code ownership)".

While it is proven that agile methods, particularly extreme programming have positive effect on the output of project teams, the studies and the empirical investigations to back this and to provide concrete evidence have been lacking. The question the researchers ask is "Is the adherence to the XP methodology per se that is the main ingredient in the success of XP software development or the established dimensions are more important to success?"

The intention of this article was to address that issue as well as to find out the impact each part of the XP practices have on performance by extensive research. In order to do that the researchers have amassed a sample from various software development teams and assessing their attributes such as team cohesiveness, general characteristics, and the eventual project outcomes.

Results

The main study was compiled by using the "method of multiple hypotheses" which aims to encourage impartial research. So the research team first outlined 9 different competing propositions associated with the performance of projects. Those hypotheses are as follows:

1. XP practices, XP-specific team factor, and general team factors are each uniquely associated with team performance.
2. General team factors are uniquely associated with team performance.
3. XP practices are uniquely associated with team performance.
4. The XP-specific team factors are uniquely associated with team performance.
5. XP practices and the interactions between them and (a) the XP-specific team factor and (b) general team factors, are associated with team performance.
6. Team cohesion and the interaction between it and XP practices are uniquely associated with team performance.
7. The interaction between team practice and XP practices is associated with team performance.
8. XP practices are uniquely associated with team performance, and (a) the XP-specific team factor and (b) general team factors, including cohesion mediate

relationship.

9. The XP-specific team factor is uniquely associated with team performance, and team processes and cohesion mediate relationship. (pp. 664-665)

The researchers have tested these competing hypotheses by assessing the association between the performance of certain number of real-life project teams and their devotion to agile specific methods and general team factors. There have been total of 40 teams mainly consisted of university students who have gone through formal courses in agile methods and subsequent sessions. The study is made out of the results of questionnaires the researchers have taken from each team members after the project has finished. The main variables of the study are XP practices measured by the adherence to XP, team inputs measured by degree of participation etc and finally team cohesion. The questionnaire which was used for the study was enclosed in the appendix and the example question is as follows:

-'Foundations': What percent of your work was done in pairs?

The results have from the data collection have supported the hypotheses number 1 as the samples showed all three variables are uniquely associated with performance and showed that all XP practices and the XP specific team factors have great impact on team performance. But the impact is not always positive such as pair programming, testing and refactoring have attributed to some of the teams' poor performance while the customer related practices had more positive impact. Overall the result indicates that the concentration on some particular areas of XP methodology or the negligence of the method altogether is not likely to work while the customer planning and team code management are the key to successful project (p. 670).

Conclusions

The article shows that not all the parts of the XP practices have positive impact on the performance of the team. While the customer planning and XP-specific team factor are positively related to performance the foundations' relationship is negative. All in all the results have strong credibility and accuracy as the method for compiling the sample is not based on the participants' assessment of their own performance.

Byambadorj Dulamsuren

A Project Manager's Optimism and Stress Management and IT Project Success

D.C. Smith, M. Bruyns, and S. Evans, International Journal of Managing Projects in Business, volume 4, number 1, pages 10-27, 2011

Johdanto

Informaatioteknologian projektien epäonnistumiset ovat edelleen maailmaan laajuisesti hyvin yleisiä. Epäonnistumiset ovat jatkuneet siitäkin huolimatta, että projektien vetäjien toiminnan tueksi on kehitetty erilaisia toimintamalleja ja metodeja.

IT-projektipäällikköiden pätevyys voidaan jakaa kahteen eri osa-alueeseen, koviin ja pehmeisiin taitoihin. Tämä artikkeli perehtyy näistä taidoista jälkimmäisiin eli pehmeisiin taitoihin. Pehmeitä taitoja ovat henkilön persoonalliset ja emotionaaliset taidot, joista tässä artikkelissa keskitytään optimismiin ja stressiin. Tämän tutkimuksen päämääränä on tutkia minkälainen vaikutus projektipäällikön optimismilla on IT projektin onnistumiseen, sekä minkälainen vaikutus projektipäällikön stressin hallinnalla on projektin onnistumiseen.

Tässä tapauksessa optimismilla tarkoitetaan lähinnä toivoa. Toivolla taas tarkoitetaan henkilön odotuksia siitä, että tavoitteet pystytään saavuttamaan. Liian suurella optimismilla voi olla vakavia seurauksia projektin onnistumiselle ja se onkin usein syy miksi projektit epäonnistuvat. Stressistä puhuttaessa ei aina välttämättä tarkoiteta huonoa. Stressi voi tasosta riippuen vaikuttaa joko positiivisesti tai negatiivisesti.

Results

Tämän artikkelin tutkimus toteutettiin tarinankerronnan avulla. Tarinankerronta on tutkimusmenetelmä, joka mahdollistaa tutkijoille niin hiljaisen kuin tarkan tietämyksen keräämisen.

Tutkimukseen otti osaa 12 erittäin kokenutta projektipäällikköä, jotka työskentelivät suurissa organisaatioissa. Kaikki nämä henkilöt olivat työskennelleet IT-alalla vähintään 10 vuotta ja johtaneet useita suuria projekteja. Tutkimuksen henkilöiden tarinat nauhoitettiin ja kerättiin haastatteluiden avulla.

Tutkimuksen lopputuloksen mukaan optimismilla on positiivista vaikutusta projektin lopputulokseen. Optimismin vaikutus näkyi hyvin esimerkiksi siten, että se loi paljon paremman suhteen niin tiimin sisäisesti, kuin myös ulkopuolistenkin osapuolten kanssa. Tiimin sisäisten suhteiden paraneminen vaikutti siihen, että tiimin jäsenet olivat paljon halukkaampia ja ahkerampia työskentelemään projektipäällikölle. Ulkopuolisten suhteiden parantumisen johdosta projektit saivat paljon enemmän ohjausta ja tietoa näiltä osapuolilta. Tutkimus ei suoranaisesti kertonut optimismin ja projektien onnistumisen suhteesta, mutta esille tuli se, että optimismi vaikuttaa projektipäällikön kykyyn johtaa projekteja. Eräs tutkimukseen osaa ottanut henkilö sanoikin, että optimismi nitoo projektityöskentelyn kannalta tärkeät asiat yhteen. Joten hänen mielestään optimismi vaikuttaakin yli 50% asioista.

Tutkimuksen mukaan stressi voi vaikuttaa projekteihin joko positiivisella tai negatiivisella tavalla. Stressi voi olla projektiympäristössä hyvin terveellistä ja voi johtaa korkeaan motivaatioon. Positiivista stressiä voidaan luoda esimerkiksi haastamalla tiimiläisiä, sekä palkitsemalla tiettyjen tavoitteiden saavuttamisesta. Stressistä voi tulla kuitenkin erittäin huono asia, mikäli rasituksesta tulee liian suuri. Negatiivinen stressi vaikuttaa esimerkiksi tuottavuuteen, sekä fyysiseen ja henkiseen terveyteen. Stressin hallinnan kannalta olisi

erittäin tärkeää, että pystyttäisiin asettamaan realistisia tavoitteita sekä jokaisen yksittäisen henkilön tulisi tietää omat rajansa.

Tutkimuksen lopputulos olikin se, että olemalla realistisen optimistinen ja hallitsemalla stressiä projektipäällikkö pystyy parantamaan projektin onnistumisen mahdollisuutta. Tutkimuksen mukaan seuraavat asiat ovat tärkeitä näiden hallinnan kannalta:

1. Kunnon projektisuunnitelma ja siinä pysyminen vähentää epätietoisuutta ja näin ollen lisää optimismia ja vähentää stressiä.
2. Onnistumisia pitää juhlia tietyn väliajoin, koska se lisää tiimin optimismia.
3. Ammattimainen käyttäytyminen lisää optimismia.
4. Motivointi ja yksilöiden huomioon ottaminen vähentää stressiä
5. Vahvat yleiset johtamistekniikat auttavat hallitsemaan stressiä.
6. Hermojen säilyttäminen tiukassa paikassa auttaa vähentämään stressiä.
7. Ulkopuolisten väliintulojen minimointi vähentää stressiä.
8. Kevyt stressi projektin aikana parantaa kuitenkin tehokkuutta.

Yhteenveto

Tutkimuksen perusteella optimismilla ja sressin hallinnalla on projektin onnistumisen kannalta positiivista vaikutusta. Näissä molemmissa tapauksissa pitäisi pystyä löytämään oikea tasapaino, koska molemmissa ääripäät eivät ole hyväksi projektille. Oikealla tasapainolla on mahdollista saada tiimi työskentelemään mahdollisimman tehokkaasti.

Artikkeli antoi varsin hyvän kuvan siitä, miten nämä asiat voivat vaikuttaa projektien onnistumiseen. On aivan ymmärrettävää, että niin liika kuin liian vähäinenkin optimismi voi olla projektille tuhoisaa. Lievä positiivinen optimismi on hyvästä, kunhan vain ollaan tietoisia siitä mihin oikeasti pystytään. Liiallinen stressia voi olla projektin onnistumisen kannalta erittäin vakava asia. Liiallisen stressin takia yksilö voi menettää vaikka koko työkykynsä. Stressin suhteen tulisi löytää jokaisen oma taso ja hieman haastaa sitä. Haastavuuden avulla työ pysyy mielekkäänä ja näin ollen työn teho korkealla. Näiden asioiden hallinta voi parantaa projektin onnistumisen mahdollisuutta hyvin merkittävästi.

Lasse Mäkinen

Professionalization, Risk Transfer, and the Effect on Gender Gap in Project Management

M.-J. Legault ja S. Chasserio, International Journal of Project Management, volume 30, issue 3, pages 697–707, 2012

Johdanto

Pohjois-Amerikassa ja Euroopassa IT-alalla työskentelevistä ainoastaan viidennes on naisia, vaikka yleisesti korkeakoulutetuilla aloilla naisten osuus on tätä suurempi. On myös selvitetty, että IT-alalla työskentelevistä naisista 74 % mainitsee pitävän työstänsä, mutta silti jopa puolet näistä jättää työnsä työuran keskivaiheella. Tällainen työvoimakato on yritysten kannalta hyvin kriittistä, sillä työntekijät ja heidän tietotaito ovat alalla merkittäviä resursseja.

Perinteisesti projektivetoisiin teknologiatyöpaikkoihin on liitetty joustavuus, hyvä palkkaus, työn itsenäisyys ja henkilökunnan mahdollisuus vaikuttaa työhönsä. Artikkelin tekijöiden tarkoituksena on ollut tuoda esille mahdollisia ongelmakohtia, mitkä voivat liittyä projektin hallintaan. Tämän lisäksi on haluttu selvittää, miksi naisten osuus IT-alalla työskentelevistä on niin pieni, ovatko projektit todellisuudessa joustavia ja kuinka paljon työntekijällä on mahdollista vaikuttaa omiin työoloihinsa.

Artikkeliin on haastateltu työntekijöitä seitsemästä Kanadalaisesta yrityksestä. Näistä yrityksistä viisi on pieniä B2B-yrityksiä, joiden liiketoimintaosaaminen koostuu muun muassa multimediasta ja IT-palveluista. Lisäksi tutkimukseen on otettu mukaan kaksi selvästi suurempaa yritystä, näiden yritysten työntekijöistä haastatteluun on otettu ainoastaan IT-osastolla työskentelevät. Haastatteluita on tehty yhteensä 88 kappaletta, vuosien 2001 ja 2002 välillä. Haastateltavana on ollut henkilöstö- ja projektipäälliköitä, sekä perinteisiä IT-työntekijöitä. Naisten osuus haastateltavista on ollut noin puolet.

Tulokset

Artikkelissa mainitaan, että projektien hallinnan ongelmat ovat syntyneet siitä, että lopputuotteena ei ole ollut varsinainen massatuote, vaan ne ovat olleet räätälöityjä palveluita asiakkaan tarpeisiin. Projektit noudattivat kuitenkin samaa kaavaa siten, että projektien hallintaan vaikutti huomattavasti se, millainen sitova sopimus on tehty toimittajan ja asiakkaan välille.

Projektisopimusta tehdessä täytyy olla jonkinlaiset arviot muun muassa tulevista työmääräarvioista, aikataulusta ja hinnasta. Räätälöidyille tuotteille on tyypillistä, että työntekijät tarvitsevat suhteellisen vapaat kädet, jotta innovaatioita voi syntyä. Tämä taas aiheuttaa ongelmia projektin hallinnan suhteen, koska projektipäällikön on vaikeampi hallita kehitystä ja tehdä arvioita tulevasta. Arvioiden tekemistä vaikeuttaa myös projektin aikainen vaihe, ja se että asiakkaan toiveet voivat vaihtua useaan otteeseen. Näin ollen projektisopimukseen liittyy aina merkittäviä taloudellisia riskejä. Tämän lisäksi, jos sopimukseen kirjatut arviot ovat menneet toteuttavan yrityksen kannalta pahasti pieleen, voi asiakkaalle muodostua suhteeseen tavallista määräävämpi asema.

Artikkelin mukaan rivityöntekijöiden kannalta tilanne ei ole kovin hyvä, sillä heidän mielipiteitä kuullaan harvoin sopimuksia tehtäessä. Näin ollen heidän mahdollisuutensa vaikuttaa tuleviin työtehtäviin ja työtahtiin on varsin pieni. Perinteisesti projektitiimiä edustaa projektipäällikkö, joka tekee tarvittavat arviot muiden työntekijöiden osalta. Koska nämä arviot ovat yleensä hyvin summittaisia, joutuvat projektityöntekijät olemaan valmiita ylitöihin lyhyelläkin varoitusajalla, jotta projekti saataisiin ajoissa valmiiksi. Tarvittavien

ylityötuntimäärien suuruus riippuu mahdollisesti pieleen menneestä työmääräarviosta.

Kyselytutkimuksessa nousi esille, että monet työntekijät työskentelevät yli 40 tuntia viikossa, jotkut jopa yli 50. Osa haastateltavista joutui tekemään töitään myös kotona työpäivän päätteeksi. Tämä taas ajaa työntekijän hankalaan asemaan, sillä jos työntekijä ei ole valmis joustamaan projektien aikataulujen mukaan, voidaan kokea, että hän ei ole sitoutunut ja motivoitunut hoitamaan työtehtäviään. Yritykset kokevat sitoutumisen puutteeksi myös sen, jos työntekijä pyytää helpotusta työ- ja kotielämän yhteensovittamiseen. Tästä johtuen ylityöt ja kasvavat työmäärät ovat yksi merkittävimmistä syistä siihen miksi naiset jättävät tai vaihtavat teknologia-alan työpaikkansa. Suurin osa ylitöitä tekevistä naisista on lapsettomia.

Yrityksille ylitöiden teettäminen on kallista, sillä laissa määrätyt kohdat asettavat vaatimukset ylityökorvauksille. Tämän vuoksi ylitöitä ei virallisesti suositella, työntekijät voivat halutessaan tehdä ylitöitä palkatta, jotta projekti valmistuu ajoissa. Monet suostuvat tähän varjellakseen ammattiylpeyttään, sillä myöhästynyt tai epäonnistunut projekti voi vaikuttaa heidän maineeseensa, ja tätä kautta myötävaikuttaa potkujen saamisessa. Osittain tästä syystä naiset eivät ole niin halukkaita ottamaan projektin johtotehtäviä vastuulleen, koska se tarkoittaisi enemmän venyviä työpäiviä ja -tunteja. Venyvät työpäivät aiheuttavat taas ylimääräistä stressiä ja jännitteitä, kun aikatauluihin pitäisi sovittaa myös perhe-elämän vaatimukset.

Yhteenveto

Artikkelissa mainitaan useita piirteitä, miksi perinteinen projektimuotoinen työskentelytapa ei ole paras mahdollinen menettelytapa rivityöntekijän näkökulmasta. On olemassa tilanteita jolloin työntekijän henkilökohtainen vapaus pienenee selvästi, ja toisaalta projektipäällikölle tai asiakkaalle muodostuu vääristynyt asema. Tilanne ei ole ihanteellinen myöskään projektipäällikölle, jonka täytyy aikatauluja tehdessä joustaa sekä asiakkaan että työntekijöiden suuntaan. Yleensä asiakas tai palvelun tilaaja on kuitenkin se taho kenen toiveiden mukaan joustetaan enemmän.

IT-ammattilaiselle pahin riski on, että projekti ei valmistu aikataulussa, jolloin mahdollisesti saa maineen epäpätevänä työntekijänä. Tämä taas vaikuttaa tulevaisuuden työnsaantiin. Välttääkseen tämän he saattavat tehdä paljon ylitöitä, jotta projektit valmistuvat aikataulussa ja henkilökohtainen työportfolio puhtaana. Paljon ylitöitä tekevät nähdään myös sitoutuneina työntekijöinä, jolloin heitä palkitaan myös muita enemmän muun muassa ylennyksillä.

Nykyäänkin on vielä käytäntönä, että naiset ovat enemmän vastuussa pienten lasten hoidosta. Tästä johtuen naiset jättäytyvät uransa keskivaiheella IT-alan työpaikoista, tai eivät ota vastaan vastuullisempia tehtäviä, koska työaikataulujen yhteensovittaminen perhe-elämän kanssa on vaikeaa.

Markus Salomaa

To Bridge or to Bond? Diverse Social Connections in an IS Project Team

J.Y Han and A. Hovav, International Journal of Project Management, volume 31, issue 3, pages 378-390, 2013

Tausta

Tietojärjestelmät ovat yhä tärkeämpiä organisaatioiden toiminnan kannalta. Siitä huolimatta tietojärjestelmäprojektien epäonnistumisprosentti on hyvin korkea. Yksi merkittävä tekijä projektien onnistumisen kannalta on tutkimusten mukaan tiedon jakaminen. Tietojärjestelmäprojektit ovat kuitenkin usein lyhytkestoisia, mikä osaltaan vaikuttaa ryhmän jäsenten alttiuteen jakaa tietoja ja taitoja keskenään. Tietojärjestelmäprojektien toteuttaminen vaatii myös useiden eri alojen asiantuntijoiden panosta, mikä lisää projektiin osallistuvien henkilöiden yhteistyötaitojen merkitystä. Sosiaalinen pääoma nouseekin merkittäväksi tekijäksi projektin onnistumisen kannalta.

Sosiaalinen pääoma, sekä sisäinen että ulkoinen, kattaa kolme ulottuvuutta: rakenteellinen, relaationaalinen ja kognitiivinen. Rakenteellinen sosiaalinen pääoma tarkoittaa projektiryhmän sisäisiä ja ulkoisia viestintäkanavia. Relaationaalinen ulottuvuus kattaa sosiaalisten suhteiden sisältämät voimavarat kuten luottamuksen ja normit. Toimijoiden välille muodostuvat yhteiset merkitykset, käsitykset ja tulkinnat määrittävät kognitiivisen ulottuvuuden sosiaalisessa pääomassa.

Aiemmat tutkimukset tarkastelevat sosiaalisen pääoman vaikutusta tietojärjestelmäprojektien onnistumiseen joko sisäisten tai ulkoisten ulottuvuuksien kautta. Sisäisen pääoman on todettu lisäävän ryhmän yhteenkuuluvuutta ja siten tukevan projektin toimintaa. Ulkoisella pääomalla on sen sijaan löydetty ristiriitaisia vaikutuksia projektin toteutumiseen. Ulkoisista lähteistä saatu tietotaito voi vaatia resurssien irrottamista ryhmän sisäisestä työskentelystä ja siten vaikuttaa negatiivisesti projektin tulokseen. Ulkoisella tietotaidolla onkin todettu positiivinen vaikutus projektitulokseen, kun se toimii lisänä ryhmän omiin kykyihin ja tietoihin.

Sosiaalinen pääoma kattaa siis sekä ryhmän jäsenten keskinäisen tiedonjakamisen että verkostoitumisen kautta saatavan ulkoisten tahojen tarjoaman tiedon. Tämän tutkimuksen tavoitteena on aiemmista tutkimuksista poiketen selvittää ja arvioida sekä sisäisen että ulkoisen sosiaalisen pääoman vaikutuksia tiedonjakamiseen ja sen myötä tietojärjestelmäprojektin onnistumiseen. Sisäisestä sosiaalisesta pääomasta käytetään termiä sitoutuminen ja ulkoisesta termiä verkostoituminen. Sitoutumisella arvioidaan olevan positiivinen korrelaatio sekä tiedonjakamisen halukkuuteen että projektintulokseen. Verkostoitumisen oletetaan vaikuttavan suoraan projektitulokseen sekä toissijaisesti tiedonjakamisen halukkuuteen sitoutumisen kautta. Lisäksi projektin laajuudella sekä ryhmäkoolla oletetaan olevan vaikutusta projektin toteutumiseen.

Tulokset

Tutkimuksessa tarkasteltiin henkilöitä, jotka ovat osallistuneet tai paraikaa osallistuvat tietojärjestelmäprojektin toteuttamiseen. Hypoteeseja testattiin kyselytutkimuksen avulla, johon osallistui 177 vastaajaa. Kyselytutkimuksen kysymykset käyttivät seitsemänkohtaista asteikkoa ja niiden arviointiin käytettiin pilottitutkimusta, jonka myötä kysymykset asetettiin lopulliseen muotoonsa. Esitetyn mallin arviointiin käytettiin PLS-analyysia. Tutkimuksessa käytettyjen muuttujien pätevyyttä arvioitiin kirjallisuusanalyysilla, asiantuntijaraadin avulla

sekä suorittamalla ennakkokokeita, jotka osoittivat muuttujien kuvaavan tarkoituksenmukaisesti kuvattuja ilmiöitä.

Tutkimuksen perusteella luotiin malli, jossa asetettiin sitoutuminen sekä verkostoituminen koetun projektituloksen ennakkotapauksiksi. Lisäksi sitoutumisella oletettiin olevan vaikutusta ryhmän jäsenten halukkuuteen jakaa tietoa keskenään. Kyselytutkimuksen tilastollisen analyysin perusteella voitiin todeta sitoutumisen korreloivan positiivisesti halukkuuteen jakaa tietoa sekä koettuun projektitulokseen. Sitoutumisen lisäksi myös verkostoituminen vaikuttaa positiivisesti kokemuksiin projektin tuloksesta. Verkostoituminen korreloi positiivisesti myös tiedonjakamisen kanssa silloin, kun projektiryhmä on vahvasti sitoutunut. Näiden lisäksi ryhmäkoolla oli negatiivisen vaikutus projektin toteutumiseen, kun taas projektin laajuudella ei ollut siihen vaikutusta.

Pohdinta

Tärkein havainto tutkimuksessa oli, että sekä sisäisellä että ulkoisella sosiaalisella pääomalla voidaan parantaa projektin toteutumismahdollisuuksia ja sen tuloksia. Sitoutuminen kasvattaa ryhmän jäsenten halukkuutta jakaa tietoa keskenään ja siten edistää toiminnan tehokkuutta. Verkostoitumalla jäsenet voivat tuoda projektiin ulkoista tietotaitoa, joka myöskin edesauttaa ryhmän toimintaa.

Tutkimustuloksista voidaan tehdä projektijohtamiseen liittyviä johtopäätöksiä. Esimiesten tehtäväksi asettuu sellaisen työympäristön luominen, joka edesauttaa sitoutumista. Projektiryhmän rakenne tulisi olla tasapainossa siten, että ryhmässä tapahtuu sitoutumista välittömästi muodostamisen jälkeen, mikä myöhemmin mahdollistaa verkostoitumisen kautta saatavien tietotaitojen hyödyntämisen.

Jatkotutkimushaasteina voidaan nähdä tutkimustulosten vertaaminen eri kulttuureja edustavien työntekijöiden kesken. Alati kansainvälistyvät tietojärjestelmäprojektit vaativat työntekijöiltä yhteistyötä yli kulttuurillisten rajojen. Eri kulttuureissa sosiaalisen pääoman painotukset voivat vaihdella, mikä tulisi ottaa huomioon valittaessa kunkin projektinryhmän jäseniä.

Ville Siltala

Acquiring and Sharing Tacit Knowledge in Software Development Teams: An Empirical Study

S. Ryan and R. V. O`Connor, Information and Software Technology, volume 55, issue 9, pages 1614-1624, 2013

Johdanto

Ryan ja O`Connor käsittelevät artikkelissa ohjelmistokehitysprosessin inhimillisiä tekijöitä ketterien ohjelmistokehitysmenetelmien näkökulmasta nostaen esiin sosiaalisen vuorovaikutuksen tärkeyden hiljaisen tiedon (*Tacit knowledge*) välittämisessä kehitysryhmän sisällä. Aiemmassa tutkimuksessaan Ryan ja O`Connor kehittivät tavan mitata hiljaista tietoa kehitysryhmissä. Siihen liittyy TMS-malli (*Transactive memory system*), joka selittää tiedon jakautumista ryhmissä sekä sosiaaliset elementit, kuten keskustelut, ryhmän jäsenten kesken.

Ryan ja O`Connor pitävät asiantuntijoiden välistä tiedon jakamista tärkeänä tekijänä ryhmän työn onnistumisessa, siksi he erityisesti selvittävät artikkelissa miten ohjelmistokehityksessä ryhmät sekä hankkivat ja jakavat tietoa että miten hiljainen tieto ja TMS-malli vaikuttavat ryhmän onnistumiseen työssään.

Tulokset

Ryan ja O`Connor tutkivat sosiaalisen vuorovaikutuksen ja ryhmän hiljaisen tiedon sekä sen kehittymisen yhteyttä Internet-kyselyssä, joka suuntautui Irlantilaisten ja Iso-Britannialaisten yritysten työntekijöille. Kyselyn tuloksia vertailtiin kehitettyihin teoreettisiin malleihin ja niiden perusteella arvioitiin mallien todenmukaisuutta.

Kyselyssä selvitettiin seuraavilla tavoilla ryhmän hiljaisen tiedon kehittymistä, vuorovaikutuksen määrää ja laatua, TMS-mallin mukaista tiedon jakautumista ryhmässä sekä ryhmän suorituskykyä:

1. Hiljaista tietoa arvioitiin Ryanin ja O`Connorin kehittämällä ryhmän hiljaisen tiedon mittaristolla.
2. Sosiaalisen vuorovaikutuksen laatua ja määrää mitattiin tutkimalla ryhmän jäsenten keskenäistä vuorovaikutusta ja sitä oliko vuorovaikutuksella haluttuja lopputuloksia ryhmän jäsenten mielestä.
3. TMS-mallin mukaista tiedon jakautumista mitattiin Lewisin kehittämällä erikoistumista, uskottavuutta sekä koordinointia mittaavalla mallilla.
4. Ryhmän suorituskykyä arvioitiin sekä tehokkuuden että tarkoituksenmukaisen toiminnan suhteen, mikä Ryanin ja O`Connorin mukaan on vaikeaa IT-alalla.

Tuloksien mukaan sosiaalisen vuorovaikutuksen laatu korreloi hiljaisen tiedon muodostumisen kanssa ryhmässä, mutta määrällä ei ollut suuria vaikutuksia. Lisäksi sekä TMS-mallin mukainen tiedon jakautuminen että hiljaisen tiedon määrä ennustavat ryhmän tehokkuutta, mutta eivät tarkoituksenmukaista toimintaa.

Yhteenveto

Artikkelissa Ryan ja O`Connor esittivät tutkimuksensa siitä miten sosiaalinen vuorovaikutus on yhteydessä ohjelmistokehitysryhmien hiljaisen tiedon kehittymisessä positiviisella tavalla. Tulokset ovat myös yhdensuuntaisia aikaisempien tutkimusten tulosten kanssa, mutta viittaavat myös siihen, että hiljaisen tiedon määrä vaihtelee paljonkin eri kehitysryhmien välillä.

Yhteenvetona Ryan ja O`Connor esittävät mallin hiljaisen tiedon hankkimisesta ja jakamisesta ryhmissä kuvassa 2 (s. 9).

Ryan ja O`Connor huomauttavat, että kyselyn toteuttamistavan johdosta jotkin kyselyn vastauksista saattavat olla vääristyneitä, mutta tilastotieteellisten menetelmien avulla ne on yritetty ottaa huomioon.

Artikkeli keskittyi lähinnä sosiaaliseen vuorovaikutukseen hiljaisen tiedon hankkimisen välineenä, mikä ryhmäkontekstissa on ymmärrettävää, mutta samalla Ryan ja O`Connor sivuuttavat ryhmän jäsenet yksilöinä, jotka myös ryhmän ulkopuolella hankkivat tietoa. Tutkimuksessa ei myöskään kiinnitetty huomiota hiljaisen sekä eksplisiittisen tiedon suhteeseen, mikä jättää jälkeensä monia kysymyksiä ja vähentää artikkelin hyötyä käytännössä laajemmassa kontekstissa.

Antti Varjoinen

Project managers in Global Software Development Teams: a Study of the Effects on Productivity and Performance

R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F. José García-Peñalvo, and E. Tovar, Software Quality Journal, January 2013

Tausta

Ohjelmistokehitys muuttuu jatkuvasti muun maailman kanssa globaalimpaan suuntaan. Tämä kysyy ohjelmistokehitys projektien johtajilta entistä enemmän tietoa ja taitoja, kun jo valmiiksi kompleksiseen prosessiin lisätään ulkomailla toimivan projektihenkilöstön tuomat haasteet.

Tutkimuksen tarkoituksena oli kartoittaa ja vertailla projektin johtajien kykyä toimia niin perinteisessä paikallisessa yksikössä toteutetussa projektissa, kuin ulkomaille ulkoistetussa projektissa.

Tulokset

Projektin onnistumista mitattiin mm. käytetyn ajan, tehdyn työn ja tuottavuuden avulla. Tämän lisäksi hyödynnettiin ns. 360 asteen palautetta, jossa projektin johtajat saivat palautetta kaikilta projektin osapuolilta.

Tutkimukseen osallistui kahdeksan projektinjohtajaa kahdesta maasta. Kullakin johtajalla oli kaksi projektia, joissa oli kussakin kaksi aliprojektia. Toinen aliprojekteista toimi paikallisessa yksikössä ja toinen oli ulkoistettu ulkomaille. Palautteen antoon osallistui kahdeksan viiden hengen tiimiä joihin kuuluivat projektin johtajan esimies, toinen projektin johtaja, paikallinen projektiryhmäläinen ja paikallinen asiakas.

Tuloksista kävi ilmi, että ulkoistetut projektit valmistuvat hitaammin ja projekti johtajien piti tehdä enemmän työtä ulkoistettujen projektien kanssa. Myöskin tuottavuus todettiin kautta linjan heikommaksi ulkoistetuissa projekteissa kuin paikallisissa ja virheiden määrä suuremmaksi. Tuottavuuden heikommuuteen epäillään tarvittavan työpanoksen kasvua ja työnjohdon organisointia. Kaikkiaan projektin monimutkaisuus ja ulkoistamisesta johtuvat kompleksit huonontavat kaikkiaan ulkoistettujen projektien tuottavuutta. Kahden eri projektin välille ei suurta tilastollista eroa saatu. Kuitenkin toisessa projektissa oli havaittavissa hieman kohentunut aikataulujen pitäminen, joka voi johtua lisääntyneestä luotosta organisaatioiden välillä tai ensimmäisen projektin aikaan tapahtunut hienoinen yhtiökulttuurin osmoosi.

360 asteen palautteen tuloksien perusteella projektien työntekijä asemassa olleet arvostelivat johtajiaan kaikkein kriittisimmin. Näiden esimiesten arviot olivat kuitenkin positiivisempia. Huomattavaa kuitenkin on, että samat esimiehet usein päättävät ulkoistamiskäytännöistä, joten heidän on myös oltava positiivisia lopputuloksesta. Samaa voitaneen sanoa muista projektin johtajista. Toisaalta projektin ryhmän jäsenet kritisoivat johtajiaan tiukemmin, jopa niin että esimiesten ja jäsenien välille löydettiin statistista eroavaisuutta. Ryhmän jäsenet antoivat lähes kautta linjan huonompia arvosanoja johtajilleen, kuin muut palautteen antoon osallistuneet. Epäilyksen alla onkin, että kahden hyvin erilaisen aliprojektin vetäminen samaan aikaan voi vaikuttaa lopputuloksiin.

Yhteenveto

Tulokset esittävät heikompia tuottavuuslukuja ulkoistetuilla kuin paikallisille projektiryhmille. Täten voidaan osoittaa, että ulkoistetulla projektilla on negatiivista vaikutusta projektin johtajan tuottavuuteen. Suurimmaksi vaikeudeksi epäillään koordinaation, kommunikaation ja kontrollin puutetta ulkoistetussa projektissa. Projektin johtajien olisikin hyvä kehittää kommunikaatio ja ajankäyttö taitojaan näiden ongelmien lieventämiseksi. Ulkoistetun projektin hallinnointi on kuitenkin haastavaa ja joissain tapauksissa kuitenkin saadaan valmista aikaiseksi lähes samaan tapaan kuin paikallis projekteissa.

Petri Arpiainen

Managing Projects in a Games Factory: Temporality and Practices

P. Stacey and J. Nandhakumar, in Proceedings of the 38th Annual Hawaii International Conference on System Sciences, page 234a, 2005

Johdanto

Pelien markkinaosuus viihdeteollisuudesta kasvaa ja uusia pelejä julkaistaan päivä päivältä lisää. Peliprojektien myöhästely ja epäonnistuminen on arkipäivää edelleen. Pelejä on tutkittu paljon mutta vähemmän niiden luomisprosessin kannalta vaan enemmän psykologian ja sosiologian kannalta.

Tässä artikkelissa Stacey ja Nandhakumar tutkivat miten peli studiot tuottavat ja hallinnoivat peliprojektia. Tarkoituksena oli lisätä tutkittua tietoa kyseisestä aiheesta sekä selvittää mitkä asiat vaikuttavat erityisesti peliprojekteihin ja niiden hallinnointiin. Stacey ja Nandhakumar esittävät että “pelien kehittäminen on erilaista kuin yleisesti ohjelmiston kehittäminen sillä peli ei ole varsinaisesti ratkaisu mihinkään ongelmaan vaan sen on tarkoitus tuottaa tunteellista tyydytystä”.

Artikkelissa käsitellään termiä ”ajallinen rytmi”, jolla tarkoitetaan aikaa, joka muodostuu jokapäiväisten asioiden tekemisestä. Määritelmä löytyy teoksesta Orlikowski, W.J. And J. Yates, It's About Time: Temporal Structuring in Organisations. Organization Science, 2002. 13(6): p. 684 – 700.

Tulokset

Stacey ja Nandhakumar tutkivat useiden pelistudioiden projektien hallintaa haastatteluilla tammikuun ja huhtikuun välisenä aikana vuonna 2004. Studiot olivat Singaporesta, jossa valtio on kiinnostunut luomaan menestystarinoita paikallisista firmoista viennin kohottamiseksi. Studioista haastateltiin keskeisiä vaikuttajia projektissa ja haastatteluiden tukena Stacey ja Nandhakumar hyödynsivät projekti dokumentteja ja vuokaavioita. Artikkelissa Stacey ja Nandhakumar keskittyvät yhteen esimerkki tapaukseen, CGS:ään.

Pelin kehittämisen erona yleisesti ohjelmiston kehittämiseen on asiakkaan varsinainen puute. Peliohjelman vaatimukset ovat tiimin asettamat ja käytännössä pääsuunnittelijan päässä. Stacey ja Nandhakumarin mukaan koska pelit ovat “oikukkaita” eivät kehittäjät välttämättä seuraa tarkkoja kehitysmalleja vaan saattavat kehittää peliä “purkauksin” populaari kulttuurin vaikutuksesta.

Stacey ja Nandhakumar selvittivät, että pelin kehittämistä ei kannata katsoa lineaarisella kuvakulmalla, eli GANTT, PERT jne. Vaan ajallisilla rytmeillä. Pelin kehityksessä pelaustestaaminen (Play-testing), henkikunnan vaihtuminen ja kehittäjien tunnetilat luovat erilaisia ajallisia rytmejä jotka yksilötasolta heijastuvat koko henkilökuntaan ja sieltä ylöspäin. Pelin kehittämiseen vaikuttaa myös ulkopuoliset ajalliset rytmit kuten messut ja markkinoinnin kannalta tärkeät ajat kuten joulumyynti. Valtio voi luoda ajallisia rytmejä kehitykseen esimerkiksi rahoituksella. Pelin kehittämisessä sisäisten ja ulkoisten ajallisten rytmien tunnistaminen ja hallinta on keskeistä, mutta ajalliset rytmit ovat hyvin muuttuvia eikä niiden ennustaminen ole helppoa.

Pelaustestaamista tapahtui pelikehityksessä päivittäin, se on eräänlainen kehityksen alisykli jonka Stacey ja Nandhakumar määrittelivät olevan “...a rapid feedback between (re)conceive,

(re)design,(re)code, (re)play-test), and (re)evaluate.”. Pelaustestausta kehitystiimi muokkasin sopivaksi jokaista ongelmaa kohden ja määritteli ajan ja tarpeen sitä vastaavaksi. Pelaustestaaminen oli tärkeä osa pelin innovointia ja toi esiin uusia ideoita, esimerkiksi bugeista tuli uudentyyppisiä toimintoja. Pelaustestaamisen kehitysideat hidastivat kuitenkin kokonaiskehitystä, ja siten on tärkeä olla niin kutsuttuja “sankareita” jotka ovat sellaisia osaajia, jotka pystyvät hallitsemaan tätä kokonaiskehityksen hidastusta.

Yhteenveto

Pelikehityksessä tarkasti jäsennetyt kehitysmallit ja jopa ketterät mallit ovat haitaksi luovuudelle silloin, kun pelin kokonaisuus on epäselvä. Ongelmana on vaatimusten määrittelemättömyys, sillä pelillä ei ole niinkään varsinaisia asiakkaita jotka voisivat antaa ohjelman vaatimukset. Kun tilanne on selkeä, kuten esimerkiksi jatko-osan kehittäminen, kannattaa kehitysmallit ottaa mukaan, mutta niiden tulisi huomioida sisäiset ja ulkoiset ajalliset rytmit.

Stacey ja Nandhakumar olivat löytäneet hyviä huomioita pelikehityksen ja yleisesti ohjelmistokehityksen osilta. Aihealue on kuitenkin hyvin laaja ja he ottivat kaksi asiaa käsittelyyn tutkimuksessaan. Artikkelin teksti oli hieman hajanainen siksi lopputulos tuntui epäselvältä. Uskon että artikkeli antaa kuitenkin hyvän kuvan pelikehityksestä ja yleisesti ohjelmistokehityksen eroista. Aihetta tulisi kuitenkin tutkia pitemmälle ja keskittyä analysoimaan suppeampaa aluetta esimerkiksi pelaustestaamista.

Miikka Kähkönen

Distributed Agile: Project Management in a Global Environment

S. Lee and H-S Yong, Empirical Software Engineering, volume 15, number 2, pages 204-217, 2010

Tausta

Nykyään suuret ohjelmistoprojektit toteutetaan usein ulkoistamalla ne yhteen tai useampaan maahan. Kohdemaissa paikalliset toimijat ovat nykyisin tottuneet käyttämään niin sanottuja ketteriä ohjelmistonkehitysmenetelmiä kuten Scrumia, mutta koko projektin mittakaavassa näitä strategioita on ollut vähemmän käytössä. Seiyoung Lee ja Hwan-Seung Yong kertovat artikkelissaan Yahoo!:n projektista, jossa Scrumia käytettiin projektin hallintaan niin paikallisella kuin globaalilla tasolla.

Tulokset

Lee ja Yong vertailivat Yahoo!:n kahta valmista My Yahoo! -projektia, joissa palvelun kansainvälisten versioiden toteutus ulkoistettiin. Zorro, joka oli projekteista aikaisempi, toteutettiin perinteisemmällä ulkoistamisstrategialla, jossa californialainen kehitystiimi antoi kansainvälisille tiimeille ohje- ja työlistan, joita heidän tuli noudattaa ja toteuttaa. My Yahoo! -versioista uudempi, Chameleon, toteutettiin käyttämällä Scrumia niin paikallisten tiimien sisällä kuin niiden ja Californiassa sijainneen ydinkehitystiimin välillä. Artikkelin kirjoittajista toinen, artikkelissa nimeämätön henkilö, osallistui Chameleon- ja Zorro-projekteihin ja pystyi näin ollen kertomaan omakohtaisista kokemuksistaan.

Kirjoittajien suosikiksi näistä kahdesta ohjelmistonkehitysmenetelmestä nousi Scrum, jonka edut sekä globaalissa että paikallisissa projekteissa ovat suurin piirtein samat: Siinä missä esimerkiksi perinteisempi vesiputousmalli on suhteellisen kankea ja paikallisille kehitystiimeille vaikeammin toteutettavissa oleva on Scrum ketterämpi ja helpommin hallittava. Vesiputousmallista kankeamman tekee se, että siinä tiimeille annetaan projektin alussa työlista, joka niiden pitää toteuttaa. Se on myös usein työntekijän kannalta raskaampi, koska siinä on vaikeampi hahmottaa projektin kehittymistä ja omaa panosta suhteessa kokonaisuuteen.

Kansainvälinen Scrum toteutettiin monessa ”kerroksessa”: Paikalliset ryhmät muodostivat oman Scruminsa, heidän ryhmänvetäjänsä kuului alueellisten johtajien ryhmään, josta muodostui kansainvälinen Scrum. Kansainvälisen Scrumin vetäjänä toimi californialaisen ydinryhmän Scrum Master. Käyttämällä tällaista rakennetta muodostui kansainvälisten tiimien ja ydinryhmän välille parempi yhteys ja projekti oli kokonaisuutena paremmin hallittavissa. Scrumin hyötyihin kuuluu perinteisesti myös se, että projekti on jaettu pienempiin osiin, jolloin se tuntuu mielekkäämmältä siihen osallistuvien ihmisten kannalta. Pienempien osatavoitteiden myötä tiimien jäsenet näkevät työnsä tulokset nopeammin kuin esimerkiksi vesiputousmallissa, joka lisää heidän motivaatiotansa ja intoa osallistua projektiin. Yhteydenpito kansainvälisessä Scrumissa tuli luonnollisesti suorittaa käyttämällä erilaisia etätyöskentelyn menetelmiä, kuten puhelimia, pikaviestiohjelmia, sähköpostia ja videoneuvottelulaitteita, eikä Scrumkaan poistanut kieli- ja kulttuurieroista syntyviä ongelmia.

Vaikka Lee ja Yong pitävät Scrumia tavallisia ohjelmistonkehitysmenetelmiä parempana, on heillä parannusehdotus siihen miten Yahoo!:n Chameleon toteutettiin. Heidän ehdotuksessaan

paikallisten tiimien vetäjät kuuluisivat alueellisiin Scrumeihin, joiden Scrum Masterit taas olisivat osallisena myös ydintiimissä Californiassa. Täten paikallisilla tiimeillä olisi mahdollisuus vaikuttaa myös kehitysalustaan, jolloin niden työ helpottuisi kun ne tekevät palvelusta omia versioitaan.

Yhteenveto

Lee ja Yong kehuvat varsin vuolaasti ketterän ohjelmistonkehitysmenetelmän hyötyjä verrattaessa sitä perinteisiin ja antavat erilaisten raporttien tuloksia tukemaan näitä väitteitä, mutta kuten he itsekin myöntävät, on kahden eri projektin vertailu vaikeaa. Erot saattoivat johtua myös erilaisesta kehitysalustoista, joita projekteissa käytettiin.

Ville Saarinen

The Scrum Software Development Process for Small Teams

L. Rising and N.S. Janoff, IEEE Software, volume 17, number 4, pages 26-32, 2000

Johdanto

Scrum on suosittu ketterä ohjelmistokehitysmalli. Scrumissa ohjelmistokehitys tapahtuu inkrementaalisesti ja iteratiivisesti, joten se sopii erityisen hyvin projekteihin, joissa ei ole mahdollista määritellä kaikkia ohjelmiston vaatimuksia etukäteen tai määritellyt vaatimukset muuttuvat usein.

Artikkelin kirjoittajat työskentelivät AG Communication Systems nimisessä telekommunikaatioyrityksessä, jossa he olivat törmänneet toistuviin ongelmiin New Business Opportunity (NBO) projekteissa. Projekteille oli tyypillistä vaatimusten muuttuminen kesken projektin ja monesti toteutus piti aloittaa ennen kuin edes kaikki alkuperäiset vaatimukset olivat tiedossa. Koska Scrum sopii erityisesti juuri tämän tyyppisiin projekteihin alkoivat artikkelin kirjoittajat testaamaan kehitysmallin käyttöönottoa yrityksen projekteissa.

Tulokset

Artikkelissa kuvaillaan Scrum ohjelmistokehitysmallia ja sen soveltumista erilaisiin ohjelmistoprojekteihin ja raportoidaan kolmen pilottiprojektin kokemuksista Scrumin hyödyntämisestä käytännössä. Koska suurin osa artikkelista koostuu Scrumin periaatteiden kuvailemisesta aloitan tiivistelmän Scrumin esittelyllä ja kerron lopuksi pilottiprojekteissa kertyneistä kokemuksista.

Useassa kohdassa artikkelia painotetaan, että Scrum ohjelmistokehitysmalli on tarkoitettu pienille enintään kymmenen hengen ryhmille. Projektin on siis oltava riittävän pieni tai ainakin jaettavissa pieniin riittävän itsenäisiin osiin, jotta Scrumin käyttämisestä saadaan hyötyä. Ryhmän koon rajoittamista perustellaan artikkelissa muun muassa yhden yhteisen fokuksen tärkeydellä ja sosiaalisen laiskottelun (social loafing) ilmiöllä.

Scrum, kuten muutkin ketterät ohjelmistokehitysmallit, on iteratiivinen ja inkrementaalinen kehitysmalli. Scrumissa projekti jaetaan useisiin lyhyisiin osiin, sprintteihin. Sprint on ennalta sovittu aikaväli jonka aikana tuotetaan yksi inkrementti tai iteraatio tuotteesta. Jos sprintin aika ei riitä kaikkien ennalta sprintin aikana tehtäväksi sovittujen ominaisuuksien toteuttamiseen, toteutettavia ominaisuuksia karsitaan prioriteettien mukaisesti, mutta sprintin aikaväliä ei venytetä. Scrum on siis kiinteää aikataulua noudattava kehitysmalli.

Scrum-projektin projektinjohtajaa kutsutaan Scrum masteriksi. Scrum master johtaa projektia, seuraa sen etenemistä ja varmistaa, että kaikki edistyvät työssään. Scrum masterin tehtäviin kuuluu myös muun muassa päivittäisten Scrum palaverien vetäminen ja projektin riskien hallinta.

Olennainen osa Scrum kehitysmallia on myös päivittäiset Scrum palaverit. Scrum palaverien on tarkoitus olla lyhyitä fokusoituja 15-30 minuutin tapaamisia, joissa jokainen ryhmän jäsen vastaa kolmeen kysymykseen. Mitä hän on saanut aikaiseksi edellisen palaverin jälkeen? Mitkä ongelmat estävät hänen työnsä edistymistä? Mitä hän aikoo saada tehdyksi seuraavaan palaveriin mennessä? Palaverien tarkoitus ei ole jäädä ratkomaan eteen tulleita esteitä vaan informoida koko ryhmää esteistä, jotta ryhmäläiset voivat auttaa toisiaan palaverien välillä osaamisiensa mukaan. Muita tärkeitä palaverin funktioita on informoida Scrum masteria ja

ryhmäläisiä projektin etenemisestä, informoida ryhmäläisiä tuotteen prioriteeteista ja auttaa riskien hallinnassa.

Jokaisen sprintin lopuksi ryhmä tuottaa uuden inkrementin tai iteraation tuotteesta. Sitten projektiryhmä tapaa sidosryhmien kuten johdon ja asiakkaiden edustajien kanssa. Sprintin tapahtumista raportoidaan ja sprintin tuloksena syntynyttä tuotetta arvioidaan. Projekti voidaan päättää minkä tahansa sprintin päätteeksi tai sitä voidaan jatkaa. Jos projektia jatketaan, projektisuunnitelmaa ja vaatimuksia muokataan valmistuneen tuotteen arvioinnin ja sprintin aikana kerääntyneiden tietojen pohjalta.

Artikkelin kirjoittajat painottavat, että, vaikka Scrum on hyvä ohjelmistokehitysmalli tietyntyyppisiin ohjelmistoprojekteihin, sitä ei tule seurata orjallisesti. He suosittelevat mallin hyödyntämistä aina kunkin projektin ja tilanteen ominaispiirteet huomioiden. Esimerkiksi yhdessä kolmesta pilottiprojektista koettiin käytännöllisemmäksi pitää Scrum palaveri vain joka toinen päivä ohjeistetun päivittäisen palaverin sijaan. Toisessa projektissa taas vaatimuksia muutettiin kesken sprintin, jotta voitiin paremmin reagoida yrityksessä sprintin aikana tapahtuneeseen stategiamuutokseen, vaikka Scrumin ohjeiden mukaan ulkopuolisten tekijöiden ei pitäisi antaa vaikuttaa suunnitelmaan kesken sprintin.

Artikkelin kokeilussa kirjoittajat esittelivät Scrum kehitysmallin periaatteet kolmelle AG Communication Systemsin projektiryhmälle ja suostuttelivat heidät käyttämään niitä yhden pilottiprojektin ajan. A-ryhmä kehitti uutta simulaattoria GTD-5 EAX vaihdejärjestelmän ohjelmistokehittäjille. B-ryhmä kehitti uutta tuotetta pienille puhelinpalvelukeskuksille. C-ryhmä kehitti uutta toiminnallisuutta GTD-5 EAX vaihdejärjestelmälle. Kaikissa kolmessa pilottiryhmässä Scrum koettiin hyödyllisena mallina ja kaikki kolme projektia saatettiin päätökseen onnistuneesti.

A-ryhmässä Scrum paransi varsinkin ryhmän yhteishenkeä ja yhteistyökykyä. Projektin ohjelmistokehityksessä kohdatut ongelmat hahmotettiin koko ryhmän yhteisinä ongelmina sen sijaan että ne olisivat jääneet ohjelmoijien yksilöllisesti ratkottaviksi ongelmiksi. Myös työn jatkuva edistyminen muuttui konkreettisemmaksi ja palkitsevammaksi.

B-ryhmässä ongelmia aiheutti kehitysryhmän muut projektin ulkopuoliset työt. Päivittäiset palaverit ja jatkuva edistymisen seuranta tekivät hyvin näkyväksi, että osalla ryhmästä oli toistuvasti liikaa tehtäviä. Ryhmässä päädyttiinkin käsitykseen, että Scrum sopii parhaiten sellaisiin projekteihin, joissa kaikki ryhmäläiset voivat keskittyä yksinomaan kyseiseen projektiin. Päivittäisten palaverien koettiin helpottavan erityisesti uusien ryhmäläisten ohjastamisessa.

C-ryhmässä Scrumin hyödyllisimmäksi osaksi koettiin päivittäiset Scrum palaverit, jotka koettiin tehokkaaksi välineeksi tiedon jakamiseen ja edistymisen seurantaan.

Pilottiprojekteissa kerättyjen kokemusten sekä omien mielipiteidensä ja päättelyn pohjalta artikkelin kirjoittajat listasivat Scrum kehitysmallin yleisiksi hyödyksi seuraavat yhdeksän kohtaa. Kehitettävä tuote jakaantuu hallittavankokoisten osien sarjaksi. Projekti edistyy vaikka vaatimukset muuttuisivat useampaan kertaan. Kaikesta tulee läpinäkyvämpää. Kehitysryhmän kommunikaatio paranee. Onnistumiset projektin aikana ja lopuksi tulevat yhteisiksi. Asiakkaat saavat aikataulun mukaisesti toimitetut inkrementit. Asiakkaat saavat tihein väliajoin nähdä miten kehitettävä tuote todella toimii. Asiakassuhteet kehittyvät ja luottamus asiakassuhteissa paranee. Malli luo onnistumisen kulttuuria, jossa kaikki odottavat projektin onnistuvan.

Yhteenveto

Pilottiryhmien kokemusten perusteella Scrum on hyvä ohjelmistokehitysmalli erityisesti muuttuvissa liiketoimintaympäristöissä tapahtuvaan ohjelmistokehitykseen. Sillä on kuitenkin rajoituksensa. Se ei sovi suurien projektiryhmien hallitsemiseen. Scrumin käyttö

myös edellyttää, että projektiryhmän jäsenet pystyvät sitoutumaan projektiin tarpeeksi sekä osallistumaan tiheästi pidettyihin säännöllisiin palavereihin.

Artikkeli antoi hyvän kuvan Scrumin käyttöönoton hyödyistä ja rajoituksista ohjelmistoalaa varmasti melkohyvin edustavan yksittäisen yrityksen tapauksessa. Kovin tieteellistä lähestymistapaa tai laajempaa kuvaa ketterien ohjelmistokehitysmenetelmien omaksumisen vaikutuksista artikkelista ei kuitenkaan löydy. Lisäksi pilottiprojektit olivat varsin lyhyitä. C-ryhmän projekti oli vain yhden kahdeksan päivän sprintin mittainen, ja A-ryhmän projekti oli jo edennyt pitkälle ennen Scrumin käyttöönottoa. Olisi ollut mielenkiintoisempaa lukea kokemuksia Scrumin käytöstä pidemmissä projekteissa.

Jarkko Huotari

An Investigation into Agile Methods in Embedded Systems Development

C. Oliveira Albuquerque, P. Oliveira Antonino, and E. Yumi Nakagawa, in Proceedings of ICCSA 2012, Part III, LNCS 7335, pages 576-591, 2012

Johdanto

Sulautetuilla järjestelmillä viitataan järjestelmiin, jotka on suunniteltu suorittamaan tiettyjä toimintoja laitteessa, joka koostuu usein myös muisa laitteista (=hardware) ja mekaanisista osista. Niiden tarve on viime aikoina kasvanut hurjasti ja samaan aikaan niistä on tullut monimutkaisempia. Tällaisia järjestelmiä käytetään usein myös sellaisissa yhteyksissä, missä ihmishenkiä on kyseessä, esimerkiksi lentokoneet tai autot, jolloin jo kehitysprosessissa on huomattava mahdolliset virheet. Tämän vuoksi kehitettäessä sulautettuja järjestelmiä on siis keskityttävä olennaisesti tehokkuuden ja luotettavuuteen ohella myös turvallisuuteen.

Agiilit metodit pyrkivät tekemään tuotteesta valmiin nopeammin karsimalla turhasta hallinnosta ja monimutkaisista vaiheista. Niissä tärkeimpiä ominaisuuksia ovat kommunikointi ja mukatuvuus. Asiakas pyritään pitämään tietoisena kehityksen vaiheista ja tuloksista, sekä näin ollen tyytyväisenä koko ajan. Esimerkkeinä kiinnostusta herättäneinä kehitysmetodeina mainittakoon eXtreme Programmine (XP) ja Scrum.

Kilpailun kasvaessa yritykset joutuvat kehittämään järjestelmiä yhä nopeammin ja tehokkaammin, jolloin agiilit metodit tulevat väistämättä korvaamaan vanhat menetelmät. Näitä kahta on kuitenkin syytä tutkia vielä enemmän yhdessä, jotta saadaan kaikki potentiaalinen hyöty julki.

Tulokset

Tutkimuksessa käytettiin metodina systemaattista kirjallisuuskatsausta. Tavoitteena oli tunnistaa kaikki merkittävät tutkimukset, jotka käsittelivät agiileja metodeja sulautettujen järjestelmien yhteydessä. Tutkimukseen löydettiin yhteensä 494 tutkimusta 7:stä eri tietokannasta, joista 51 päätyi lopulliseen kirjallisuuskatsaukseen. Systemaattinen kirjallisuuskatsaus sisälsi kolme vaihetta:

1. Suunnittelu
2. Etsinnän suorittaminen ja informaation keräys
3. Kvalitatiivisen ja kvantitatiivisen analyysin tulosten raportointi

Suunnitteluvaiheessa valitut tutkimuskysymykset olivat:

1. Mikä on nykyinen tila agiilien kehitysmetodien käytössä sulautettujen järjestelmien kehityksessä?
2. Mitkä agiilit metodit ovat suosituimpia sulautettujen järjestelmien kehityksessä?
3. Mitä hyötyjä, haasteita ja rajoitteita liittyy agiilien kehitysmetodien käyttöön sulautettujen järjestelmien kehityksessä?

Ensimmäisen tutkimuskysymyksen tuloksista kävi ilmi, että agiileja metodeja on käytetty ja kokeiltu teollisuudessa sulautettujen järjestelmien yhteydessä. Kehitysmenetelmiä ei välttämättä käytetä sellaisenaan, kuten saatettaisiin tehdä "perinteisessä" ohjelmistokehityksessä, vaan hieman muokattuina sopivammiksi sulautetuille järjestelmille. On kehitetty jopa uusia agiileja metodeja, jostka keskittyvät järjestelmien rajoituksiin ja turvallisuuteen vielä enemmän.

Suosituimpia agiileja metodeja olivat eXtreme Programming, jota oli käytetty 54%:ssa 13:sta tutkitusta organisaatiosta, kun taas Scrumia oli käytetty 27%:ssa. Nämä tulokset ovat yhdenmukaisia muiden tutkimusten kanssa, jotka ovat myös päätyneet osoittamaan eXtreme Programming:n ja Scrumin suosion. Myös näiden kahden yhdistelmää on käyetty, joissa on koitettu ottaa käyttöön molempien parhaita puolia, kuten esimerkiksi muuten käyttämällä eXtreme Programming:a, mutta adoptoimalla Scrumista sprintti ominaisuuden. Suosituin käytetty ominaisuus on kuitenkin testausta painottava kehitys Test Driven Development (TDD), mikä on varsin ymmärrettävää ottaen huomioon tuvallisuuden ja laadun tärkeyden sulautetuissa järjestelmissä.

Kolmas tutkimuskysymys paljasti, että agiilit metodit tuovat sulautettujen järjestelmien kehitykseen nopeutta, tuottavuutta, sekä virheiden vähentymistä. Ylipäätään siis järjestelmän laatu parantuu. Haasteina mainittiin vanhasta luopuminen. Yrityksissä on totuttu vanhoihin tapoihin, joten uuden käyttöönotto, tässä tapauksessa agiilit metodit, on hankalampaa aluksi kuin vanhan käyttäminen. Harvalla työntekijällä on kokemusta agiileista kehitysmetodeista. Rajoitukset koskivat lähinnä ymmärryksen puutetta otettaessa käyttöön uusi kehitys menetelmä. Yrityksen ilmapiiri täytyy olla sellainen, että se kannustaa uuden kokeilemista ja uusia ideoita. Rajoitukseksi olivat muodostuneet myös sertifikaatit, joita yritykset käyttävät. Ne voivat edellyttää esimerkiksi itsenäistä testausvaihetta järjestelmälle, joka ei kuitenkaan sisälly agiiliin kehitykseen.

Johtopäätökset

Vaikka agiilit kehitysmetodit eivät välttämättä ensiksi tunnu sopivan sulautettujen järjestelmien kehitykseen, on niillä kuitenkin tutkimuksen tulosten mukaan ylipäätään positiivinen vaikutus. Yksi tärkeimmistä tuloksista on myös se, että on jo olemassa tutkimusta näiden kahden yhteistoiminnasta, mutta kuitenkin tulisi enemmän panostaa aiheen tutkimukseen. Erityisesti keskittyen siihen, mitkä metodit ja tavat sopivat parhaiten sulautettuja järjestelmiä kehitettäessä.

Teemu Ruotsalainen

Task Coordination in an Agile Distributed Software Development Environment

D. K. M Mak and P. B. Kruchten, in Proceedings Canadian Conference on Electrical and Computer Engineering, pages 606-611, May 2006

Tausta

Artikkeli käsittelee tehtävien (task) jakamiseen liittyviä ongelmia sovellettaessa ketteriä ohjelmistokehitysmenetelmiä kehitystiimin ollessa maantieteellisesti hajautettu. Ketterät menetelmät eivät ole aina yhteensopivia maantieteellisesti hajautetun ohjelmistokehityksen kanssa. Monet näistä menetelmistä vaativat suoraa ja myös epäformaalia kommunikaatiota, joka ei ole mahdollista tiimin ollessa maantieteellisesti hajautettu. Tehtäväjaon suorittamisen kannalta tämä asetelma on ongelmallinen useasta syystä. Ensinnäkin, tieto, jota projektipäälliköt tarvitsevat tehtäväjaon toteuttamisessa on kerättävä erikseen sillä se ei muodostu projektipäälliköiden ja muiden tiimin jäsenten välisessä kanssakäymisessä. Toisekseen, projektipäällikön rooli tiedonvälittäjänä korostuu ja voi muodostaa informaation liikkumista hidastavan esteen.

Kirjoittajat pyrkivät tarjoamaan ratkaisun tehtäväjaon ongelmiin ketterän ja hajautetun ohjelmistokehityksen alueella. Kirjoittajat ehdottavat ratkaisuksi menetelmää, joka toimii projektipäälliköiden apuvälineenä ratkaistaessa tehtäväjakoon liittyviä kysymyksiä kuten missä järjestyksessä tehtävät olisi parasta suorittaa ja kenelle tiimin jäsenelle tietty työtehtävä tulisi osoittaa.

Kirjoittajat tunnistavat neljä työnjakomenetelmän ominaisuutta, jotka ovat merkittäviä työnjaon onnistumisen kannalta. Menetelmän tuottama työnjako tulisi olla läpinäkyvä tiimin jäsenten suuntaan, menetelmä ei saisi rajoittaa liikaa tiimin jäsenen omaa harkintaa työnsä organisoimisessa, menetelmän tulisi mahdollistaa prosessimalliin sisältyvät tehtävät ja myös ad-hoc-tehtävät ja menetelmän tulisi mahdollistaa tiimin jäsenten osallistuminen työnjaon toteuttamiseen.

Tulokset

Ongelma-alueen kannalta keskeisiä käsitteitä ovat prosessimalli ja projekti. Prosessimalli on jonkin ohjelmistokehitysprosessin yleinen kuvaus. Prosessi sisältää toisistaan riippuvaisia aktiviteetteja, joiden suorittaminen vie prosessia alusta kohti loppua. Aktiviteettien suorittamisesta on vastuussa jokin prosessikuvauksessa ilmaistu tekijärooli. Aktiviteetti voi vaatia toisen aktiviteetin suorittamisen ennen kuin se aloittaa. Aktiviteetti voi liittyä johonkin prosessimallissa kuvattuun välituotteeseen.

Projekti on jonkin prosessimallin ilmentymä eli konkreettinen ohjelmistoprojekti. Projektiin liittyvät keskeiset käsitteet ovat tiimijäsen, tehtävä ja artefakti. Käsitteet vastaavat prosessimallin tekijärooleja, aktiviteetteja ja välituotteita samassa järjestyksessä.

Artikkeli esittää artikkelin ongelma-avaruuden ja aiemman tutkimuksen pohjalta kolme ongelma-aluetta kuvaavaa mallia: tehtävämalli, tiimin jäsen -malli ja artefaktimalli. Mallit kuvaavat kunkin käsitteen niihin liittyvän informaatiosisällön kannalta. Tehtävämallin

menetelmän kannalta keskeiset tiedot ovat tehtävää välttämättä edeltävät muut tehtävät, tehtävän aikavaatimukset sekä sen mahdolliset suhteet projektin artefakteihin. Tiimin jäsenen tärkeimmät tiedot ovat tiimin jäsenen taidot, kokemus prosessiin liittyvistä artefakteista sekä tiimin jäsenen nykyhetkinen työtaakka. Artefaktiin liittyvät tärkeimmät tiedot ovat sen valmistumiseen liittyvä tila ja mahdolliset riippuvuudet toisiin artefakteihin.

Artikkelissa ehdotettu työnjakomenetelmä on kaksivaiheinen. Ensimmäinen vaihe on tehtävien suoritusjärjestyksen eli kunkin tehtävän painoarvon laskeminen ja toinen tehtävän osoittaminen parhaiten soveltuvalle tiimin jäsenelle. Menetelmä tuottaa jokaiselle tiimin jäsenelle henkilökohtaisen tehtävälistan, jossa tehtäville on annettu suositus niiden suoritusjärjestykseen.

Tehtävien painoarvon ratkaisevat artefaktien ja tehtävien suhteet, tehtävien keskinäiset riippuvuudet sekä tehtävien aikataulu. Tehtävän painoarvo on sitä korkeampi mitä useampi tehtävä riippuu siitä artefaktista, jota tehtävä muokkaa. Tehtävien keskinäiset suhteet nousevat merkittäviksi siinä tapauksessa, että tehtävä ei vaikuta minkään artefaktin tilaan. Tämä on mahdollista esimerkiksi tilanteessa, jossa tehtävä/aktiviteetti ei kuulu käytössä olevan prosessimallin kuvaukseen vaan on ns. ad-hoc-tehtävä. Aikataulu otetaan huomioon soveltamalla kriittisen polun menetelmää. Korkeimman painoarvon aikataulun kannalta saa tehtävä, joka on projektin kriittisellä polulla ja jonka välttämätön aloitusajankohta on lähimpänä nykyhetkeä.

Menetelmän toinen vaihe jakaa painotetut tehtävät tiimin jäsenien kesken. Tehtävien jaossa painotetaan tiimin jäsenten taitoja, työtaakkaa ja aiempaa kokemusta käsittelyssä olevasta artefaktista. Kunkin tiimin jäsenen nykyhetken työtilanne otetaan huomioon eräänä tiimin jäsenen ominaisuutena. Työtaakan merkitys painottuu tehtävän vaatiman erityisosaamisen suhteen; mitä enemmän tietyn työtehtävän suorittaminen vaatii erityisosaamista, sitä vähemmän annetaan painoarvoa tiimin kunkin jäsenen työtaakalle.

Päätelmät

Kirjoittajat päättelevät menetelmän sopivan hajautettuun ketterään ohjelmistokehitykseen useiden seikkojen perusteella. Menetelmä lisää järjestelmän läpinäkyvyyttä ja tuo esiin työnjakoratkaisujen taustalla olevat tekijät. Järjestelmä lisää joustavuutta sillä tiimin jäsenillä on mahdollisuus valita suorittamansa työtehtävät oman tehtävälistansa puitteissa. Menetelmän laskukaavat mahdollistavat jouston organisaatioiden ja tilanteen mukaan.

Teemu Jyrkämä

Opening Up to Agile Games Development

P. Stacey and J. Nandhakumar, Communications of the ACM volume 51, issue 12, pages 143-146, 2008

Tausta

Pelien kehittäminen ja erityisesti suunnittelu on huomattavan epävakaa prosessi. Peliin sisältyvien tärkeiden ominaisuuksien lopullista lukumäärää ei mitenkään voida tietää kehitysprosessin alkumetreillä. Kuluttajat odottavat pelien olevan hauskoja, viihdyttäviä ja toisaalta myös haastavia. Mutta vasta kun peliä on kehitetty tarpeeksi pitkään, eli pelistä on olemassa prototyyppi, on näiden asioiden havaitseminen mahdollista.

Tutkijat Stacey ja Nandhakumar lähtivät selvittämään, miksi pelejä kehittävät yritykset käyttävät Agile-kehitysmallia vain osittain. Kyseinen malli vaikuttaa tutkijoiden mielestä sopivan erityisen hyvin pelien kehitykseen, mutta nykyisellään yritykset sisällyttävät toimintaansa vain joitain Agile-kehityksen periaatteita ja metodeita. Tutkimuksessa haluttiin erityisesti selvittää, mitä Agile-kehityksen hyviä puolia pelinkehityksessä käytetään. Lisäksi tärkeää oli ymmärtää mikä sai aikaan Agile-ajattelun yrityksissä ja otti sen yleiseen käytäntöön.

Tarkempaan tutkimukseen valittiin kolme eri pelistudiota ympäri maailmaa. Studioiden yhdistävä tekijä oli erikoistuminen mobiilipeleihin, vaikka yksi studioista valmisti myös perinteisiä tietokonepelejä. Tutkijat eivät halunneet tehdä tavanomaisia haastatteluita valmiine kysymyksineen, vaan lähtivät prosessiin mukaan avoimemmin mielin. Jo valmiiksi mietittyjen kysymysten sijaan he pitivät vapaamuotoisia keskusteluja kehittäjien kanssa ja järjestivät ryhmätapaamisia. Tavoitteena oli selvittää perinpohjaisesti jokaisen yrityksen tavat tehdä asioita ja pelien yleinen kehitysprosessi. Tutkimusta tehtiin vuosien 2004 ja 2005 välisenä aikana noin puolitoista vuotta, jotta tutkijat näkisivät tarkemmin kehitysprosessien sisään. Suojatakseen firmoja, tutkijat kutsuivat yrityksiä peitenimillä CGS, Miko ja Goo.

Tulokset

Tutkimuksessa keskityttiin erityisesti arvioimaan CGS yrityksen viittä henkilöä, jotka vastasivat eri alueista pelin kehitysprojektissa. Mukana olivat pelisuunnittelija, projektin johtaja, pääohjelmoija, graafinen ohjelmoija ja harjoittelijaohjelmoija.

Tutkijat näkivät kehityksen eräänlaisena bumerangina. Aluksi kehityksessä lähdetään konseptitasolta, sitten siirrytään suunnitteluvaiheeseen ja tämän jälkeen koodausvaiheeseen, joka johtaa pelin testaamiseen. Tutkijat havaitsivat, että *play-testing* eli pelin testaaminen pelattavuuden tarkistamiseksi ja ohjelmointivirheiden tuhoamiseksi esiintyi useasti haastatteluissa teemana. Mikäli *play-testing*-vaiheessa havaittiin ongelmia esimerkiksi pelattavuudessa, piti peliin tehdä muutoksia useilla eri osa-alueilla. Kun konseptitasolla tarvittiin uudelleentyöstöä, heitettiin peli takaisin konseptitasolle. Jos taas konsepti oli toimiva, mutta muutokset tekniseen tai taiteelliseen toimivuuteen olivat tarpeellisia, siirrettiin pelin kehitys takaisin suunnittelutasolle. Mikäli pelissä havaittiin ohjelmointivirheitä, siirtyi peli tietysti takaisin koodaajien käsiin.

Pelien testaus oli tutkijoiden mukaan tärkeämpää kuin perinteinen ohjelmointivirheiden etsiminen. Pelin toimivuutta uudelleenarvioitiin joka päivä, mikä vastaa Agile-kehitystä erittäin paljon. Pelien testaus olikin aina viimeinen pysäkki ja jos ongelmia havaittiin, lähti peli uudelleen kiertoon. Joskus myös *play-testing*-sessioiden yhteydessä havaitut, erityisesti graafiset bugit koettiin hyväksi, sillä ne saattoivat inspiroida uutta suunnittelua. Välillä

projektin johto ei kuitenkaan ollut paikalla hyväksymässä uusia ideoita projektiin, ja työntekijät joutuivat itse päättämään onko idea tarpeeksi hyvä, jotta se kannattaa lisätä peliin. Tämä kuvasti tutkijoiden mukaan selvästi Agile-menetelmän rohkeutta.

Pelinkehityksessä ei ole aina varsinaista asiakasta, vaan yhtiö tekee peliä itselleen ja toimii näin itse asiakkaana. Tällöin yhtiön täytyy itse päättää asioista ja päätyä yhteisymmärrykseen monista seikoista. Tutkijoiden tekemien haastatteluiden mukaan parhaat pelit syntyvätkin hyvästä kommunikaatiosta. Yhtiöt harrastivat viikoittaisia tapaamisia, joissa äänestettiin asioista, joista kaikki eivät olleet yhtä mieltä. Kun kehityksessä saavutettiin joku isompi tavoite, kerättiin koko yhtiö sihteereistä lähtien pelaamaan peliä ja jakamaan mielipiteitään ja ideoitaan. Haasteeksi muodostuikin ideoiden ja mielipiteiden paljous, joiden sisällyttämisestä peliprojektiin päätti lähinnä yhtiön johto. Työntekijät saattoivat kuitenkin toisinaan äänestää ylivoimalla jonkin ominaisuuden kehitykseen.

Kehitysprosessin aikana yhtiöistä erosi useita työntekijöitä muun muassa siitä syystä, että he eivät saaneet ideoitaan läpi toteutukseen. Tämä lisäsi työntekijöiden työpaineita, sillä kehityksen täytyi jatkua vaikka tekijätiimistä puuttui jokin tärkeä osa-alue. Tällöin joku toinen työntekijä joutui korvaamaan poistuneen henkilön paikkaa kunnes yhtiö sai palkattua uuden työntekijän.

Päätelmät

Agile-menetelmä on tutkijoiden mukaan tullut pelinkehitykseen, koska kehitysprosessin aikana on tarvittu rohkeutta, itseluottamusta ja aloitekykyä selvitä *play-testing* keskusteluiden aiheuttamasta päätöstulvasta nopeasti. Lisäksi johtajien halu palautteeseen, avoimuus ja halu jakaa tuote muun yhtiön kanssa on auttanut asiassa. Tutkimuksessa havaittiinkin, että mikäli työyhteisössä ilmenee pessimististä asennetta pelien ominaisuuksien kehittämismahdollisuuksiin, se vaikuttaa suoraan myös työmotivaatioon. Lisäksi irtisanomistilanteissa muiden tekijöiden on täytynyt tehdä aloitteita ja improvisoida.

Tutkijoiden mukaan Agile-kehityksen edistämiseksi on yrityksissä käytettävä erityisesti viittä ohjeistusta. Kehittäjätiimin tulisi päästää koko yhteisö pelin testaukseen ja arvioimiseen mukaan, jotta palaute olisi mahdollisimman laajaa. Yrityksen tulisi myös arvostaa kehittäjätiimin yksilöitä ja kuunnella näiltä saatua palautetta. Ilmapiirin täytyy olla positiivinen, jotta ihmiset haluavat tuoda mielipiteitään esille ja mahdollisesti esittää uusia ideoita. Yrityksessä tulisi myös olla terveellinen väittelyn ilmapiiri, jossa jokainen voi myös olla eri mieltä kehitysideoista. Sen sijaan että työntekijät työskentelevät aina samoissa työpisteissä tuttujen työkavereiden kanssa, tulisi heidät välillä asettaa mukavuusalueidensa ulkopuolelle erilaisiin työtehtäviin. Tämä rutiineista irrottautuminen kasvattaa myös työntekijöiden osaamisaluetta.

Panu Hokkanen