Timo Poranen (ed.)

Software project management summaries 2012
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 2012.

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.

We hope that these summaries help students to familiarize themselves into various aspects of (software) project management.

Timo Poranen
Tampere, December 2012
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
</tr>
<tr>
<td>What practitioners consider to be the skills and behaviours of an effective people project manager</td>
<td></td>
</tr>
<tr>
<td>Software engineering challenges in game development</td>
<td></td>
</tr>
<tr>
<td>Project segmentation - A tool for project management</td>
<td></td>
</tr>
<tr>
<td>Improving management of outsourced software projects in Pakistan</td>
<td></td>
</tr>
<tr>
<td>IT project management control and the control objectives for IT related technology (CobiT) framework</td>
<td></td>
</tr>
<tr>
<td>Users as knowledge co-producers in the information system development project</td>
<td></td>
</tr>
<tr>
<td>Coopetitive relationships in cross-functional software development teams: How to model and measure?</td>
<td></td>
</tr>
<tr>
<td>Requirements engineering and the creative process in the video game industry</td>
<td></td>
</tr>
<tr>
<td>An investigation into the governance of information technology projects in South Africa</td>
<td></td>
</tr>
<tr>
<td>Agents assisted software project management</td>
<td></td>
</tr>
<tr>
<td>Quality management metrics for software development</td>
<td></td>
</tr>
<tr>
<td>An empirical analysis of the relationship between project planning and project success</td>
<td></td>
</tr>
<tr>
<td>Information systems project management: an agency theory interpretation</td>
<td></td>
</tr>
<tr>
<td>Integrating system analysis and project management tools</td>
<td></td>
</tr>
<tr>
<td>Coordination in co-located agile software development projects</td>
<td></td>
</tr>
<tr>
<td>Can distributed software development be agile?</td>
<td></td>
</tr>
<tr>
<td>Obstacles to decision making in agile software development</td>
<td></td>
</tr>
<tr>
<td>Lean software management: BBC worldwide case study</td>
<td></td>
</tr>
<tr>
<td>Benefits realization management: Panacea or false dawn?</td>
<td></td>
</tr>
<tr>
<td>Empirical findings on team size and productivity in software development</td>
<td></td>
</tr>
<tr>
<td>Software process improvement success factors for small and medium web companies: A quantitative study</td>
<td></td>
</tr>
<tr>
<td>Testing in the software development life-cycle: now or later</td>
<td></td>
</tr>
<tr>
<td>The effects of schedule-driven project management in multi-project environments</td>
<td></td>
</tr>
<tr>
<td>A lightweight framework for describing software practices</td>
<td></td>
</tr>
<tr>
<td>A documentation framework for architecture decisions</td>
<td></td>
</tr>
<tr>
<td>Research on agile project management with Scrum method</td>
<td></td>
</tr>
<tr>
<td>Rules and tools for software evolution planning and management</td>
<td></td>
</tr>
</tbody>
</table>
What practitioners consider to be the skills and behaviours of an effective people project manager


Background
There has been a growing number of companies using project management methodologies and processes since 1990s. Project management is also management of people and an effective people project management can increase performance and productivity of a project. This can create a competitive advantage. This research uses literature review, face to face interviews and focus groups meetings to collect information about managing people skills. This research proposes what are considered to be the most important skills and behaviours of an effective people project management.

Results
This research deals with meanings people put on their observations. That is the reason why a constructivist interpretivist approach with a phenomenological research paradigm has been chosen. Qualitative research data is analysed using a quasi-judicial method. The research starts with literature review. Important skills and behaviours are collected and analysed to be used in the later phases in the research. The literature review reveals a number of people management skills but it doesn't provide evidence which skills make an effective people project manager. The literature review doesn't answer to the research questions.

Main research questions for the research are (p. 996):

1. What are the most important skills practitioners consider make an effective people project manager?
2. What specific behaviours do practitioners associate with each of these skills?

Next phase of the research is face to face interviews of ten practitioners who have 3 to 15 years of experience as a professional project manager. These five male and five female work in the telecommunications, banking, consultancy and engineering industries. They were asked questions what they considered to be skills and associated behaviours of an effective people project manager. A focus group meeting is the third phase of the research. It was held with the same practitioners from the interviews. The focus group is used for validating and checking the reliability of the findings from the literature review and interviews (p. 997). Practitioners used their own experiences and suggested skills and behaviours what they considered to
be an effective people project manager (p. 997).

Important part of the research is the triangulation of the literature and empirical data. The theory (literature) is tested against empirical data and it is possible to counterbalance strengths from one to another.

Six skills and associated behaviours are identified. The research suggests that project managers would benefit from adopting these skills and behaviours to improve efficacy of their projects.

Efficient people project management skills and behaviours (p. 1000) are:
- **Understanding behavioural characteristics**: Understanding of the relationship between behaviours and feelings, being genuine, open and honest, showing authentic concern for others and adopting repertoire of behaviours for different situations.
- **Leading others**: Inspiring others to be more creative, applying appropriate leadership style for situations, informing others about manager's wishes.
- **Influencing others**: Influencing or impressing others so that they can see the benefits, selling others the benefits.
- **Authentizotic behaviour**: Showing an open concern for others, accepting people who they are and empowering people. Understanding the needs other people have and make people feel good about the work, the project and themselves.
- **Conflict management**: Having open and honest discussions to find root causes of the conflicts. Concentrating on work issues, being loyal, trustful and helpful, being prepared to compromises. Observing behaviours to sense conflicts early.
- **Cultural awareness**: Developing, displaying and applying an awareness of other cultures in a team. Understanding values and beliefs of other cultures.

One of the main findings in the research is that knowing the skills and acquiring competences is not enough. Practitioners need to apply the skills, observe the outcomes and modify them if needed to make them work even better. It is a continuous process where behaviour and its impacts matter.

Literature suggests some people project managers are play-acting rather than showing authentic behaviours. The problem here is that people behave consistently with their beliefs: it does not matter what they say when their behaviour shows their real beliefs. According to research play-acting is acceptable behaviour only when applied time to time in situations, not continuously.

These skills and behaviour can be adopted by project managers anywhere in the world. Expectations of the behaviour of managers may vary in different areas of the world and there may be some corporate cultural differences also. The behaviours are not limited for application in any specific area of the world or industries.

**Conclusions**
Results of this research are surprising. There are literature and research of people project management skills but the associated behaviours are not discussed. The practitioners in
the research consider the skills don't make anyone an effective people project manager but it is the application of the skills, behaviour, that drive the effectiveness of the skills.

The research shows how people project managers need to have understanding of behaviours and needs of other people. Play-acting doesn't work. It's challenging to apply changes to behaviour when the changes have to be genuine and the behaviour has to be authentic.

Efficient people project manager is not only making people to work but it can also improve the successful delivery of projects. This article and research is very interesting for every people project manager or for someone who wants to be an effective people project manager one day.

Anu Kauppi
Software engineering challenges in game development


Background

Video game development has several unique characteristics which create additional challenges for the management of the projects. The article describes these challenges and highlights some engineering practices that can be used to help dealing with them. Game developers need to combine code, graphics, sound, music, etc. and create an entertaining and fun experience for the player. The diversity and the sheer amount of these assets is a major problem. Game projects today are huge. Estimating the scope of the project is difficult and the problem is often made worse by feature creep (Adding new features during development). A single game project may involve hundreds of people and take years to complete. People involved come from several different disciplines, which makes effective communication challenging. The employees are often divided into teams by their specialty (programming team, animation team). This helps the information sharing between people of same discipline but makes communication between groups more difficult.

Game development can be roughly divided into three phases: preproduction, production, and testing. During preproduction the game designers create the game design document (GDD) and prototypes are created and tested to find the fun elements for the game. During production most of the assets are created. Developers still usually create prototypes, iterations and increments of the game. This is the phase, where feature creep happens. The changes made in this phase may result in changes to the GDD. Unmanaged changes create problems with functionality, schedules, etc. Testing is usually the last phase before the game is sent to the publisher. The testers look for defects and push the game to the limits. The process seems to fit the waterfall model, but the production phase breaks the model with iterations and increments.

Game developers often use third party software engines for physics engines, game engines, AI, etc. This reduces the costs but there are some issues to be considered. The decision on which engines to use should be carefully considered. Some engines are best used on certain types of games. Developers also need to be careful not to make their games too similar to other games made using the same engine.

Results

Game developers are starting to acknowledge the need for improving their development
processes. Because the industry is so different from other software industry, the managers need to understand the current process and carefully consider which development processes work which ones do not. They need to find out, which methodologies work best with their teams and organization.

The spiral model may work well for large projects. Project manager can develop a series of tasks derived from the GDD. Requirements engineering skills are very important at this phase. GDD usually contains both stated and just implied requirements for the game. The manager must clarify all of them for the production teams. Using requirements engineering the manager can reduce errors that are caused by miscommunication between designers and the production team. Risk management is also important to handle feature creep. New features should be carefully considered as to how important they are and are they worth the extra effort. Using requirements engineering and risk management also helps with project scope problems, which are often result from translation of the GDD to a project plan. The scope estimations can never be exact, but they can help to predict success of budgets and schedules.

The cycles of the spiral model can be planned so that the game is playable at the end of each cycle. This would enable better testing of the game during development. It would also help the team to see the potential of the game and to adapt to possible changes in the design and schedule. Scrum is another model that produces incremental iterations of the product. In scrum the goal of every cycle is to have usable software. Different methods could also be combined. For example agile methods could be used during preproduction phase, when prototyping and iterations are needed. In production phase the more formal spiral process could be used. Companies need to invest in training their managers, so that they have the necessary skills to evaluate different methods and identify the ones that are right for their teams and for the project.

**Conclusions**

Management need to be trained to understand different methodologies, so that they can select the ones that are right for their team and project. Agile methods could work well for the preproduction phase for fast iterations and prototyping. The manager needs to use requirements engineering when transferring the GDD to project plan and estimate the scope of the project with the help of the teams involved. Risk management should be used to deal with changing requirements. At the production phase a more formal spiral model is recommended. The tools and pipeline should be optimized for the handling of large amounts of diverse assets. The teams need to understand that they need to work together. Game development is always a group effort. Third party components must be carefully evaluated to determine which one is the best choice for the job.

Game development is in many respects different from the rest of the software industry. The challenges that rise from these differences are unique, but they can still be overcome by using some of the methods used in traditional software engineering. The companies need to invest in researching these methods and adapting them to game development.

Esko Vankka
Project segmentation - A tool for project management


Background

In the 1950s, most of projects were managed using network-based project management techniques, like CPM, PERT or PERT-COST. Although result using these techniques were good, there are some situation where they are not applicable. This is due to the fact that some assumptions have to be accepted in the project for the correct working of networks-based techniques, and in some projects is not possible to accept these assumptions.

One of these assumptions is that for using these techniques project activities must have a clear beginning and ending point. In some projects, specially in complex ones, activities may change over time, and some decision taken during planning phase may become inaccurate. Other of the assumptions that is mentioned in the article is that the activity-sequence relationship of the project can be specified and arranged in a directed network, but in some projects, the sequence relationship cannot be specified beforehand.

Network models' assumptions are valid for many projects however there are other projects where these assumptions cannot be accepted. The use of networks-based techniques for managing a project that do not satisfy these assumptions may produce mistakes. This paper proposes an alternative method based in segmentation, this new method can be used in projects that do not satisfying the requirements for applying networks techniques.

Results

For the segmentation method can be used, the project must meet certain requirements:

- The project can be divided into similar segments.
- The activities of the project should be known beforehand, and all the segments should have a similar set of activities.
- The performance order of the activities should be known and be the same for all the segments.
- The duration and cost of perform each activity on each segment can be estimated.
- The actual duration and cost of performing each activity on each segment can be measured.

The main idea of the segmentation method is to divide the project into several subprojects, the number and size of the subprojects may vary depending of the original one and all the subprojects require a similar set of activities. The project planner can
determine the number and size of the subprojects considering that if the number of segments is high, the project planning will be harder, by the way, the segments size will be smaller so the project duration will be shorter.

In the paper, a project planning is proposed in order to calculate the planned time and the planned cost and to compare them to the results obtained without using segmentation. The evaluation model consist in calculate the planned finish time \( P_{F_{ij}} \) of each activity on each segment by adding the planned starting time \( P_{S_{ij}} \) and the planned duration \( P_{D_{ij}} \) of these activities. Also is possible to estimate the planned project duration \( P_{D} \) as the difference of the finish time of the last activity and the starting time of the first one (p. 17 equations 1 and 2).

To demonstrated the proposed evaluation model, a project example is designed (p. 17, tables 1,2, and 3). As shown in table 1, the project has 5 activities, and it can be divided into 3 segments. The duration of each activity in each segment is shown, as well as, the duration of each activity without considering the segmentation process. By the results of tables 2 and 3 we can check that using segmentation the project estimate duration will be shorter than without using segmentation.

The results of the evaluation model results are only a schedule that is an approach of the estimated time for each activity, when the project is being developed, a project control system is needed for controlling if the times are being met. For this task, a technique called "Line Of Balance" (LOB) is used. This technique consist in calculate two variances (start variance and finish variance) for each activity on each segment. These variances are calculated as the difference of the real time and the planned time (p. 18 equations 4 and 5). A positive value means that the activity property, starting time or finish time, is developed ahead the plan. If the value is 0 means that the activity result is exactly as planned and a negative value means that the activity result failed to met the plan. For the activities that finish according to the plan, a third variance value is calculated, cost variance (p. 18 equation 6).

For detecting activities that are consistently late, activity variances are used. Activity variances are calculated by adding the values of the same activity in each different segment (p. 18 equations 7,8 and 9).

The previous example is continued. This time we have the status of the project in the fourth week of development. Some activities were developed as planned but another ones were late (p. 18 table 4). Is possible to detect which activities are the problematic ones by the calculation of variances and activity variances (p. 18 tables 5 and 6).

**Conclusions**

In some projects, specially those whose activities are quite long comparing to the project length, the use of network-based project management techniques do not provide good results. To solve this problem, the article introduces the project segmentation technique, that it is based on dividing the project in independent segments with a variable size. The segmentation gives to the project manager more flexibility for planning the project, and the number of segments can be increased for getting a faster development and more control in the execution phase.
An evaluation model is proposed for supporting the segmentation choice, and a project control system is also proposed for checking that the project times and costs are being fulfilled.

Eugenio Gómez Martínez
Improving management of outsourced software projects in Pakistan

Z. Jalil and A. Hanif, in proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, pages 524-528, 2009

Background

According to statistics presented in the paper, software industry today has the highest rate of project failure in the world. Despite of considerable investments in project management training, the project managers are not performing as well as expected.

Offshore project management is a field of growing importance as offshore IT outsourcing to Pakistan and other countries is a key corporate strategy for many organizations today. The avoidance of certain issues and associated risks in outsourced software projects is essential to promote success. The authors make a division of 5 typical issues: misconceptions in requirements, communication barriers, too high expectations from the client, cultural differences and possibly distributed testing of the software. In contrast to in-house developed software projects, software management tasks in outsourced projects get distributed among the outsourcer and the outsourcing company. Smooth and continuous communication between these two parties is essential if the project is to succeed.

Results

Particularly in Pakistan, the project manager in the outsourcing company is supposed to handle all communication tasks in addition to all technical tasks. Despite of the Pakistani government's efforts to promote its software industry and outsourcing, it is clear that there are still some prominent issues that need to be addressed. As is the case with India and China, most software projects that are outsourced to Pakistan are over budget, over schedule and sometimes not completed at all. To remedy this situation, the authors have completed a survey of 22 outsourced software projects in Pakistan. The results of this survey identify several common problems regarding project management in these projects.

The survey form was distributed to software project managers in the surveyed projects. The gathered data included information on the project, project managers and utilized project management policies. The authors divide their observations and analysis of the data into 11 different categories: 1) inexperienced and uncommitted project managers, 2) training level of project managers is very low, 3) small and medium-sized projects, 4) satisfaction level of stake holders, 5) deviation from budget and schedule, 6) focus of project managers, 7) communication management, 8) team management issues, 9) top three risks involved in outsourced projects, 10) top lessons learned, 11) monitoring and
control.

According to the results in categories 1 and 2, the low experience and commitment levels of the project managers can be considered to have an impact on the project's outcome. As a side note, category 3 specifies that most of the outsourced projects in Pakistan are projects with budgets less than $100,000.

In category 4 the authors observe that some 65% of the project stakeholders are not satisfied with the management of the surveyed projects. However, in category 5 the authors conclude that there has been significant progress in project management as almost 70% of the outsourced projects meet their budget and 65% meet their deadline. The main reasons for deviation from budget and schedule are perceived to be issues regarding requirement management and communication management.

The authors' analysis of the data in category 6 shows that the project managers' primary focus is on requirement analysis, architectural design and implementation. Continuing on, the authors state in category 7 that 85% of the project managers thought that communication skills have a considerable impact on the project's success. In category 8 it is revealed that the top two issues in team management have to do with communication, respectively. This trend also continues in category 9, where it is shown that most project managers considered lacking communication management as the most prominent risk in an outsourced project.

Category 10 specifies some of the lessons learned by the project managers in the outsourced projects. Predictably, the most prominent lesson is found out to be communicating well and according to plan. Other common lessons include identifying risks early and managing resources efficiently. The authors conclude that most of the lessons learned “pertain to poor communication and inefficient requirement management” (p. 527).

The final category identifies the two most effective methods for monitoring and control as weekly status meetings with clients and weekly progress review meetings. The authors outline that the project managers emphasized the importance of regular communication between the client and the development team as well as communication amongst the team members.

**Conclusions**

As a conclusion, it is determined that communication management is the most crucial area that is currently not implemented in a satisfactory way in outsourced software projects in Pakistan. It is suggested that the responsibility could be addressed directly to an “outsourcing manager” who would specialize in communication management.

Aku Hänsänen
IT project management control and the control objectives for IT related technology (CobiT) framework


Background

It is clear that project management control of project progress has become increasingly important. Edward Bernroider and Milen Ivanov explore the usage, value and structure of frameworks in their study and focus on the popular Control Objectives for IT and related Technology (CobiT) framework. CobiT is a framework created by ISACA released in 1996. Its main purpose is in defining and aligning business goals with IT goals and IT processes with 34 generic processes to manage IT.

In this article they focus on two main goals: To explore use and success of control frameworks from the point of CobiT framework; and to investigate use and implications of individual metrics following suggestions from CobiT.

The article defines the three major success factors of IT projects: costs, time and quality. These factors are more commonly known as the Iron Triangle. It is suggested that a wider set of metrics needs to be applied in order to measure the project success. (p. 326)

Results

In this paper Bernroider and Ivanov research IT project management control systems use in organizations. The set-up was guided by critically questioning suggestions of CobiT – framework. Researchers set out to see whether control variables are valid, important and used in project management controlling practice.

The research was conducted with comparisons with literature to questioning whether the metrics suggested in CobiT sufficiently capture the scope of project management measures needed.

The main part of the study was based on a cross-sectional field and web based survey that was targeted to project management professionals, information system auditors and IT consultants. A pre-test was conducted in order to validate the questionnaire. With the results of pre-test the actual questionnaire was modified in order to focus on the main topic of the study. The actual survey recorded 266 partial and 58 fully completed questionnaires. Researchers used SPSS v16 to support their data analysis. (p. 330)

The survey questioned the feasibility of CobiT metrics for assessing efficiency and effectiveness of project management processes, but also revealed that statistical tests do not indicate that the use of individual metrics promotes or hinders IT project success.
Results also show that more quantifiable facts relating to time and costs rank first followed by stakeholder perceptions. (p. 330-331)

The results indicate that the metrics suggested by CobiT are regarded capable of being used and important enough in order to improve the success of IT projects. According to survey researchers found out that overall adoption rate of CobiT for Project Management control frameworks was a lot lower than other reports suggest. Individual metrics are seen in very important part and seem to be confirming the multiple attribute character of project management performance management. (p. 332)

**Conclusions**

The research verdict was two-sided in terms of CobiT’s project management coverage. The project management control assessment factors of CobiT were too abstract for specific applications but on the other hand controlling for post-implementation reviews seem to be regularly missing in other studies.

It was obvious that CobiT alone does not significantly promote the success of projects. Researchers found out on the other hand that individual experience, the frequency of large projects and the size of organizations all positively correlate with success rates.

The researchers stated that the CobiT framework seemed to be viable yet very generic construct to measure project management performance. On the other hand industry practice may find difficulties in implementing an effective project performance management system based alone on CobiT because of its generic nature and focus on program management relevant for IT governance.

Antti Kiiskinen
Users as knowledge co-producers in the information system development project


Background

Information system development (ISD) is a process where developers design systems based on requirements gathered from users. Through research it has been found that projects are canceled or can't be completed within budget because the designed system does not meet the users' requirements. One major cause of this kind of ineffective system development is that the users are not involved in the development process. This can also cause extra costs and time required to fix the system to meet the users' requirements. Users should be included in the development process because it increases productivity and improves their attitudes towards the system.

Service-dominant logic is an emerging perspective that suggests that customers may be seen as value co-producers. Co-production is a process that is based on collaboration between producers and users. ISD may be viewed as a value co-production process where users and developers work close together to design a system that supports the organizations' daily operation and is based on defined user requirements.

The ISD process can be separated into two stages: system design and development. This study emphasizes the customers' valuable role in these two stages. Customers can provide requirements and facilitate the development process.

This study's research question is "How can users act as knowledge co-producers in the design and development stages?". The study hypothesizes that users can ensure the success of the project by contributing their expertise to facilitate requirements determination and improving the system development process.

Results

The data for the study was collected in two steps. First, letters were send to all 359 institute members of the Information Management Association (IMA) in Taiwan. IMA's goal is to improve IT usage and communication among IS professionals. Members willing to participate in the study were then contacted via telephone. They were introduced the main purpose of the study and explained the data collection procedures. They were also asked how many projects were completed recently in their organization.

Then, a questionnaire was send to 750 project managers, team leaders and senior members, whose contact details had been collected in the first stage of the study. 279 people returned the questionnaire. 19 of the responses were not included in the analysis because they contained missing values. This means that the valid response rate of the
The study was 34.6 percent.

The research variables were measured by using multi-item scales adopted from previous research studies. Likert scale from 1 (strongly disagree) to 7 (strongly agree) was used in all of the questions.

The variables used in the study were requirement determination, user-IS relationship, common knowledge, user review and project performance. There were also two control variables used, called task uncertainty and system complexity.

Requirement determination refers to synthesizing the ISD members' and users' knowledge to define user requirements and finally to design the system. User-IS relationship refers to "the level of mutual trust, respect, reciprocity and closeness of relationship between users and developers during the ISD project" (p. 30). Common knowledge refers to the developer's knowledge of the new application areas and user's overall knowledge / expertise in IS development methods and processes. User review is assessed by asking if users reviewed and approved developer's work in a certain time period. Project performance refers to the development process success.

The two control variables were task uncertainty, which refers to "the difference between the amount of information required to perform the task and the amount of information already possessed by the organization" (p. 31), and system complexity, which refers to "the perceived level of complexity associated with the analysis and design of an information system" (p. 31).

The study had four hypotheses:

H1. The level of common knowledge is positively associated with the effectiveness of requirement determination.

H2. The magnitude of the impact of common knowledge on requirement determination is influenced by the quality of the relationship between users and developers.

H3. The effectiveness of requirement determination is positively associated with project performance.

H4. The magnitude of the impact of requirement determination effectiveness on project performance is influenced by the degree to which the user participates in the reviewing process.

The results of hypotheses testing showed that common knowledge had a positive effect on requirement determination. This supported H1. The path from requirement determination to project performance was also significant, which supported H3.

The results also showed that the magnitude of the requirement determination impact on requirement determination was connected with the level of user-IS relationship. This means that H2 was supported. Opposed to, the magnitude of the requirement determination impact on project performance was only mildly connected to the level of user review. This means that H4 was partially supported.
Conclusions

The purpose of this study was to understand how users could serve as knowledge co-producers in the design and development process. It was proposed that "better design quality can be obtained when users and developers process knowledge about each other's domain" (p. 34). Users should participate in the review process to make sure that "the integrated knowledge is carried out effectively by the developers" (p. 34). Users may also detect or expose non-functional designs in the early stage of the process and this way help to reduce unnecessary costs.

Saila Oldén
Coopetitive relationships in cross-functional software development teams: How to model and measure?


Background

Software development project success is influenced by several factors. Perhaps one of the major factors is the collaboration of individuals. A team consisting of software industry experts has an alternative - a group that is a mixture of various fields. These cross-functional teams aim to bring multi-disciplinary skills and knowledge together to enable an effective implementation of a project. Previous studies have shown that cooperation, working towards a mutual goal, within a team helps promote innovation and diversity of solutions. A team consisting of one field of industry may not identify as wide range of problems and solutions as a cross-functional team would (p. 1096).

Cross-functional teams have also disadvantages. Team members with different backgrounds and point-of-view may have different personal goals and therefore promote competition. The competition might reduce team performance. The competition might arise, for example, when programmers want to develop their own skills by learning a new programming language or while the business people who would rather emphasize the rapid completion of the project. Team managers might highlight some project properties above others to achieve better result with a particular indicator. The studies have shown that competition is healthy in certain level but too fierce competition has clear disadvantages (pp. 1096-1097).

Results

The goal of this study was to seek answers to how competition and cooperation (coopetition) between cross-functional software development team members could be modeled and measured. The study shows a multi-dimensional conceptualization and the means for project managers to understand and measure cooperation and competition within one or between several software development teams (p. 1097).

The method of the study was questionnaire survey. The 7-point Likert scale was used through all thirteen survey questions of which concerned cross-functional team’s cooperative task orientation, cooperative communication as well as relations between members of the team (cooperative interpersonal relationships). Concerning the competitive components, tangible and intangible assets allocation of resources were researched. Tangible resources, for example, could be the organization's financial and
human resources. The intangibles could be strategic intellectual power or the leaders' attention toward an aspect of a project. Conceptualizations of these subjects were made on the basis of a literature survey (pp. 1097-1099).

The survey was carried out online to Australian software development project managers (115 usable responses). The group of respondents consisted of internal and outsourced project managers. Challenge of the projects and the size of the teams varied greatly. The respondents could be regarded as project management professionals (p. 1099).

The results were analyzed with partial least squares regression (PLS). The analysis resulted in values for each component as presented in Figure 1. A value shows how significant one component is and how it is related to other components. For example, task-orientation in a cross-functional team is in relation with project commitment with coefficient of 0.81 and assistance between team members with coefficient of 0.83 (p. 1100).

Figure 1 (p. 1102).
The results showed that the individual components were not significant in either of the sides, competition and cooperation, but the combination of several components made the difference. The study results support the viability of a multidimensional model in evaluating team’s cooperation and competition. The conceptual model does not take a position on what level of competition or communication, for example, is appropriate for the project's success nor gives advice for project managers to react on problems but it gives a value which can be monitored and compared (pp. 1101-1103).

Conclusions

This article shows that cross-functional team’s have a number of challenges. A negative action might override the benefits of a positive action without a contradiction is noticed. The proposed evaluation model can help detect such a change in behavior. The model gives an idea of what kinds of things should be considered when assessing the team's performance. This is perhaps useful in large-scale projects that contain a number of groups that should work as efficiently as possible. This evaluation model gives the project manager an opportunity to correct the situation when necessary.

My experience on the subject of team dynamics agree with the results of this article. I have seen when experts in different fields may want to inadvertently emphasizes the importance of their personal interest or pledge valuable information. These factors do nothing to improve the project outcome. As the study findings say, these factors should be identified at an early stage to maintain the project’s viability.

Johan Laitinen
Requirements engineering and the creative process in the video game industry


Background

Requirements engineering is an important part of software engineering, but in video game development its practices are not as clear. A video game should be "fun" and "entertaining" and those are requirements which are not always too well understood and which may lead to problems. There are professionals from many different fields (art, music, graphics, computer science, etc.) in video game development who need to interact with each other without fully understanding each other. Development usually starts with game design document (GDD), which is a thorough non-formal document written by the game designers and is like a requirements document of the preproduction team.

This study tries to develop a formal process for defining the requirements by first locating the causes for the most costly errors by reviewing some literature from the field of requirements engineering and video game development, then analyzing reports from a game developer magazine, and finally by having a look at real video games.

Results

Findings from the literature review are that emotional factors are an important part of video games, and that "there are no established techniques for eliciting emotional requirements." (p. 2) Also the person responsible for generating requirements is usually the game designer who has little or no interest in mathematical representation of the requirements. So the designer has to work with the production team to translate the requirements in some form of natural language. (p. 2)

After preproduction phase, the GDD needs to be transformed into specification and requirements for the production and starting of the traditional software development process. They state that requirements errors are some of the most costly to fix and should be fixed as early as possible, but still it is a common reason for failure of a project. They also found out that game design documents seem to never be "complete" at the end of the project.

Another main part of the study was a review of "Postmortems" columns in "Game Developer Magazine" which consisted of 50 reports about developed games. Each report contained 5 entries of "what went right" and "what went wrong" in the project. These reports and entries were categorized into five categories: (1) pre-production, (2) internal, (3) external, (4) technology and (5) schedule. They found out that a category that was found contributing to the success of the project also contributed to the failure, so a single category is just as likely to contribute to the success as to the failure. (p. 5)
Last main part of the study related to real examples from real games and their development and focuses on three issues: documentation transformation, implication creating emergent requirements and the effects of a priori knowledge.

Transforming GDD to a requirements or specification document builds up into a massive document and requires lots of different skills which implies a team effort resulting in significant costs which may lead to minimizing of the documentation effort by management. (p. 6)

Game design and the GDD is full of implied information, and identifying these implications requires careful analysis. They found three types of implications: implications that can be directly derived from the materials (GDD, etc.), implications from the general knowledge of the game-domain and implications from the knowledge of implementation details such as the target architecture. Some of these implications may easily take many person-months of development time and lead to rejection of ideas, which in turn can have negative impact on the creative process of the design team. (p. 6)

The paper has an example of a puzzle description in a GDD, which requires a lot of priori knowledge about both preproduction and production realms for anyone who is trying to formalize it. They also note that this job is usually left for a junior staff member who lacks that knowledge. GDD's also often contain lots of visual content that is very difficult to present in a formal manner. This also reveals problems in the design that might have features that are impossible to complete because of technological constraints, leading to need for dialog between different teams. (pp. 7-8)

**Conclusions**

From these results the authors concluded that project management issues are the greatest contributors to the failure or success in video game development. Many of the issues in seem to be because of poor management of the transition from preproduction to production. Also implied information should be detected as early as possible and priori knowledge should be applied in transitions from preproduction to production.

They estimated that game studios will shift to more formal processes to increase success when projects get more complex.

Timo Korhonen
An investigation into the governance of information technology projects in South Africa


Background

Many organizations have been using Information Technology (IT) to embrace their business, improve performance and increase their productivity. IT is seen as an integral part of organization business strategic objectives and processes in many organizations. In order to realize the value of IT in various organizations, they need to apply established both IT and IT projects governance principles as part of overall organization’s strategic objectives.

Unfortunately, organizations that uses general project governance principles in managing IT projects have encountered many problems including delaying the completion of IT projects, over or under budgeting and poor quality of project outputs. This implies that specific attention needs to be given to the management of IT projects (p. 662).

This paper looks at what degree IT governance is adopted by South African organizations (p. 662), and determining the relationship between corporate governance, IT governance and IT project governance. To achieve these objective findings from the literature survey (theory) were compared with the actual practice found in South Africa organizations to determine whether gaps do exist and, if so, to recommend improvements (p. 662). Through qualitative interviews the study has found that many South Africa organizations adhere to formal corporate governance principles, but many organizations do not comply with IT governance principles. Moreover the literature revealed that there is a lack of formal guidance regarding IT project governance, and therefore investigated the practices of governance of IT projects in the South African context.

Results

In order to achieve the objective of the study, the authors compared the findings from the literature survey (theory) with the actual practice found in organizations to determine whether gaps do exist and, if so, to recommend improvements (p. 662).

In the literature review the authors discussed the relationship between corporate governance, IT Governance and IT project governance. The main objective of corporate governance is to ensure the sustainability of organizations through what would be considered best practice (p. 662). There are many good governance principles and practices in different countries where organizations need to adhere to. In South Africa the government established Code of Governance Principles known as King 2 and the King 3. In recognizing the importance of IT in business in South Africa, the latest version of this
code has included full Chapter 5 to its principles guiding the governance of IT. This chapter emphasis the integration of IT in every business process including organization objectives and risk management; gives the directions for board to monitor and evaluate IT investments and expenditure. IT investments are implemented within the organization through IT projects (p. 663). This is what provides a link between IT governance and IT projects, and therefore to the management of IT-related projects and programmes (p. 663).

There number of international guidelines for IT governance as it is for Corporate Governance, but Control Objectives for Information and related Technology (COBIT) is popular and widely used. Based on literature, there is no formal guideline for IT projects governance. The relationship between Corporate Governance, IT governance and IT project Governance can be shown in Fig 1 (p. 663).

To determine the actual situation in South African organizations context, qualitative methods based semi-structured interview were used for this study. The semi-structured interview was developed based on the findings of the literature survey that focused on three aspects namely the interviewee's role and responsibilities within the organization, the notion of corporate governance and project governance within the organization, and the summation of the business strategies of the organization as well as the perception of the interviewee regarding organizational success. The interviewees participated in the study came from a wide variety of organizations ranging from the banking environment, mobile telecommunications environment, agricultural as well as petrochemical industries with wealth of experience in their respective organizations (p. 665). They have worked in their organizations for average duration of 7 year and have occupied high ranked positions in their organization such as CIOs or portfolio managers.

In short the results of the interview revealed that

- Many organizations in South Africa comply with corporate governance such King 2/3, SOX or Basel II. In terms of IT Governance requirements such as COBIT, the study has shown that very few organizations do comply with IT governance requirement. This was because these principles were not mandatory by legislation as Basel for financial organizations.

- In relation to how project governance was practiced within the context of corporate and IT governance. The study showed that many organizations use the structures of
steering committees in the absence of formal guidelines. At project level, many organizations have project-related steering committees (composing of project owner and the major stakeholders) responsible for execution of the project. This committee reports to executive committee (composed of CIO of various business units within the organization) which guides on the strategic alignment and prioritization of projects within the organization. The executive committee reports to the organization board. The board is a top-level in an organization which oversees the overall direction the organization. While it was revealed that steering committee meets once per month, project steering committees met twice or once per month. The relations between various steering committees based on the study is shown in the figure 2 below (p. 668)

![Fig. 2. Relationship between organisational committees.](image)

**Conclusions**

IT has been the key factor in enhancing organization performance, increasing competitive advantage and improves processes within the organizations. As a result IT is a major investment cost comprising nearly half of many organization budgets. Unfortunately IT projects in many organizations encounter a lot of problems including unintended project deliverables, over or under budgets and takes long time to complete.

The study revealed variations in terms how projects are managed in different organization, but the general trend indicates that many organizations in South Africa have adopted the steering committee approach. The executions of the project is done at the project steering committee level, the quality assurance and monitoring is conducted by project management committees but in other organizations, is conducted by executive committees.

Although there are formal principles for both cooperate governance and IT governance, it was revealed that most of South Africa organizations do not comply with IT governance requirements such as COBIT. Many organizations tend to comply with principles if they are mandatory by legislation. It is very important of these organizations to adhere to these requirements in order to improve the quality of the IT-related investments and therefore enhance profits.

Joel S. Mtebe
Agents assisted software project management


Background

Automatic tools for managing and assisting management of software projects are starting to become a common item in software businesses but still require improvement. Management of projects still relies much on human interaction which can suffer from unpredictable and egoistic behavior, stressful situations and lack of skilled people. Recently research in intelligent systems has been increasing and shows promising results in automating tasks that usually are done by people.

Sethuraman et al. proposes an intelligent agents based system for assisting management of software projects. This system composes of agent programs that assist people who are working in a project thus reducing their burden and human errors. Intelligent agents are software components that act according to some specified behavior models. Each agent has these basic characteristics: a role, one or more goals and sensing of environment. Agents can be designed to perform consistently in every situation and are less prone to error.

Benefits from automated assistance systems are shown to bring increase in quality and productivity. Since the business opportunities for this field are large and growing, these benefits can bring considerable profit.

Sethuraman et al. uses the Gaia methodology to model and Jade (a Java framework for agent programming) to give a detailed example of an intelligent agents based system.

Results

Managing software projects composes of many distinct activities or modules. Intelligent agents approach is found to be applicable in especially quality review management, project monitoring, control and status reporting. Also it is shown to improve productivity and quality of these activities (page 1).

Sethuraman et al. focuses on studying quality review management of a software project management process, the purpose of which is to organize review sessions with right people in each necessary state of the project.

Gaia is an intelligent agent oriented software methodology that specifies intelligent agent system in terms of roles that have the following attributes: responsibilities, permissions, activities and protocols. Protocols consist of purpose, initiator, responder, inputs, outputs and processing. Gaia model also defines interaction protocols between roles, services.
provided by roles and communication links between agent types.

Sethuraman et al. gives a detailed description of agent system for quality review management. In their analysis they discovered the following roles: quality manager assistant, review assistant and software engineer assistant. Quality manager assistant is responsible for initiating the whole review process. This includes reading through the quality review schedule document, searching for suitable reviewers, scheduling dates, sending requests and so on. Review assistant receives requests from quality manager assistant and responds. Software engineer assistant makes sure that the material for review (source code) is submitted in time. So all agents in this system act as assistants to actual persons.

Sethuraman et al. presents a case study for a quality review process. The setup for this case forms of one quality manager assistant agent deployed on quality managers laptop, three review assistant agents on reviewers laptops and two software engineer assistant agents on engineers laptops. All laptops are connected via LAN or extranet.

The quality review assistant reads the project plan document that includes all the necessary data for the review from Excel sheet and initiates the review process by sending requests to review assistant agents. These agents inform their users (actual reviewer) by prompting a dialog to accept or reject the request. Quality review assistant sends messages to software engineer assistants to upload code for the review and forwards this information to review assistant agents. Finally when reviewers are done the quality review assistant closes the review.

**Conclusions**

Intelligent agents systems have been shown to bring benefits in some subtasks of managing software projects. Sethuraman et al. states that the use of intelligent agents system increases quality and productivity approximately 5% per project. In mid to large scale IT companies this improvement is significant.

A lot of the activities in managing software projects are mostly repetitive tasks which are prone to human errors. As computing power increases and methods and algorithms become more sophisticated it is expected that intelligent systems become more common in management of projects in not just the field of software development.

Tomi Fagerlund
Quality management metrics for software development


Background
It is very common that software project will not stay in budget and will have schedule overruns. Even if the project doesn’t have any previous matters, it might not be usable and has to be reworked. Because of these problems more attention has been paid to the management of software quality. Despite of the tools and models that have been created to help with management issues, software quality problems still exists. One example of a model is COCOMO. It is one of the earliest and most widely used software project cost-estimation models. There is one big down side in COCOMO. It doesn’t take into account the quality of project management. Management is very important factor and if it is poorly done, the cost of the project can increase rapidly.

Results
In this paper a quality management metric (QMM) is introduced. This metric is repeatable and it is computed from quantitative answers to a structured set of inquiries. The questionnaire consists of two parts: questions with answers yes, no or not and paired choices. There are four areas of software management that the questionnaire addresses: risk management, planning management, people management and requirements management. The goal of the questionnaire is to develop a metric that could be compared and ranked.

The first part of the questionnaire contains pair choice questions. The answerer chooses one of the two statements which better described his program. The statement doesn’t have to match exactly, but it should be the closest one. The right tendency of the program was proven with many similar questions with different wording. Each of the four sections has a maximum score of 70.

The second part of the questionnaire contains yes-no-not applicable questions. This kind of questions can be easily compared and it is user-friendly. Each choice has an associated point value depending on the importance of the question. Each section has a maximum score of 62.

Questions of the management requirements section doesn’t seek to determine the quality of the software project by a simple answer. All the answers created the structure laid out by the program manager. In estimation and planning section, questions don’t seek an answer to a specific estimation technique. Goal is to find out the whole estimation process by asking if used technique was appropriate and how well was it implemented.
People management section is divided into four subsections, because of the highly weighted importance factors. Sections were: human resources, leadership, communication and technical competency. Each of the subsections has their own questions depending on the subject. The risk management sections goal is to find out if the program manager has an active risk management program and ways to monitor the risks. For each risk there should be integrated mitigation strategy.

The QMM Summary Score Sheet consists of points from each of the two questionnaires. Points are multiplied by their relative Importance Coefficient. The final score can be calculated after the weighted score of all the four sections are known.

**Conclusions**

The QMM indicated that questionnaire correlated between QMM scores and the success of the program. This means that QMM is a viable method for measuring the quality of software project management. With QMM program manager might be able to detect discrepancies between him and a team member’s perspective. If the manager’s score is much higher than teams overall score, he or she might think that management practices are successfully used, but they are actually not.

Arto Laurila
An empirical analysis of the relationship between project planning and project success


Background

In general, the project manager’s goal is to lead the project to completion, on time, within the budget, and especially fulfill the planned performance or end-product goals. Project manager has to prepare the project plan and watch that the plan is followed in order to end up in the success of the project. Dvir et al. attempt to provide in their study some information about the relationship between project planning and project success. The aim of the study is to show, in what level the project planning helps the project succeed, by taking several viewpoints into account. Three aspects relating to project planning were considered: a) requirements definition, b) development of technical specifications, and c) project management processes and procedures. These three aspects were reflected, in the means of project success, to the perspectives of a) end-user, b) project manager, and c) the constructing office.

Many researchers are of the opinion that a project is a special task that has not been done before. In fact, at the initial planning stage, it is impossible to exactly know the precise tasks to be performed to complete the project, not to mention the exact project costs. Some may even end up with the conclusion that no initial project planning should be done. For example, Andersen (p. 90) proposed that the standard planning approach should be replaced with so-called milestone planning, where a milestone describes what is to be done in the specified stage, but not the exact way the stage should be carried out.

However, the necessity of the initial planning phase will necessarily affect the other phases in the project life cycle. The authors of the paper also found out in another their study that the origination and initial phase have the most influence on project success, and additionally the project planning documents and preparation of formal design has a positive effect on the time constraints and cost evaluation of the project. These literature-based researches lead the authors to define three levels of project planning. End-user level focuses mainly on the functional characteristics of the project end product. Technical level focuses on the project team’s technical specifications and requirements of the project deliverables. The project management level focuses on the demanded planning of the activities and processes so that the technical work can proceed effectively.

The previous approaches for measuring project success may not reflect the end-user needs and they also have a considerable possibility for difficulties in commercializing the final product.

Moreover, success means different things to different people; so comprehensive success
criteria have to be considered. A few researchers have identified three aspects of project performance for measuring success: the implementation process, the perceived value of the project, and client satisfaction with the delivered project.

The author and a few other researchers used 13 success measures based on a previous research, summarizing the measures into four dimensions: 1) Meeting design goals, 2) Benefit to the customer, 3) Commercial success, and 4) Future potential. However, the authors finally selected three main dimensions of them in this study: the end-user, the project manager, and the contracting office.

Results

The authors based their study on the results of more than a hundred defense research and development projects performed in Israel. Although all the projects did not directly relate to software projects, they have common characteristics shared by computer projects management as well. The data, planning and success variables were measured by using structured questionnaire items and interviews.

A few concepts must be clarified here. The end-user represents the parties who will be using the systems the project is intended to. The end-user also specifies the initial functional requirements. The contracting office is a unit of the Ministry of Defense. The contractor is a commercial firm that has been granted the contract for performing the project. The project manager is here an employee of the contractor who is responsible for the success of the project. The questionnaires were filled out during three years after the completion of the projects. The questionnaire focused on several aspects of the research and development process. In the research of the authors, the analysis remains limited to variables related to project success and project planning.

According to previous researchers, project success may be measured in three criteria: 1) meeting planning goals, 2) end-user benefits, and 3) contractor benefits. The authors of the paper used these same success criteria in their questionnaire items and presented the statistical results in three corresponding tables as the result of the survey. Each of these tables contains several properties indicating the success level of the attribute they belong to, in a 1–7 scale, where 1 represented total failure and 7 total success. Each of these properties is a result from several observations, or factors, which together affect the final value of the property. In addition to these, they included into the questionnaire an item associated with overall success of the project, by using the same 1-7 scale.

How did the authors measure the amount of effort invested in project planning then? They used the following three dimensions, described in three tables: 1) development of functional requirements, 2) development of technical specifications, and 3) implementation of project management processes and procedures. Each of these dimensions consists of several questionnaire variables measuring its various aspects. The authors took the variables from previous work of Shenhar et al. (p. 93).

Most of the data analysis and results are based on the correlation between averaged results of the three project planning measures and the four project success criteria (three measures plus one overall success measure). The authors found several interesting correlations between these measures. It should be noted, that correlation means a
meaningful statistical relationship between two sets of statistical data.

First, they found that “there is a high correlation between the capturing and developing of the functional specifications of the end product and the definition of the technical specifications”. However, there seems not to be correlation between the implementation of planning procedures and the quality of the functional and technical specifications of the end product. Second, the authors found no correlation between the implementation of planning procedures and the various success dimensions. Third, there is positive correlation between the development of functional and technical specifications, and meeting planning goals and achieving the final benefits of the contractor. However, this correlation seems not to be statistically significant in this case.

The most significant finding is that all four success measures are highly *inter-correlated*, which means that projects thought to be successful are also successful for all stakeholders of the project.

**Conclusions**

Too much planning may kill the creativity of developers, but at least a minimum amount of planning is always needed in projects. Planning does not automatically lead to project success, but lack of planning is likely to lead to project failure. On the contrary, some projects may strictly follow planning, but will lead to failure because the actual customer benefits are ignored.

Generally speaking, planning is a central point particularly in connection with project management. A few professional project management standards, such as *PMI Guide to Project Management Body of Knowledge* (p. 94), emphasize the need to invest in project management processes and project planning.

The main conclusion of the research was that on the project and its deliverable requirements should be put a considerably amount of effort. The study also confirms previous researches related to project requirements and specifications; a significant positive relationship between the amount of project planning and the project success can be found. This study has also revealed the relationship between project success and end-user benefit. The overall success measure seems to reflect the end-user’s satisfaction the results of the project.

However, the study by the authors is solely based on the measurements of project success and planning, so a more comprehensive research could be performed, with more variables or factors involved. The main conclusion is stated to be that no effort should be spared, but the project goals and its deliverables requirements should be clearly defined in the initial stage of a project. On the other hand, the authors claim that a minimum level of project planning tools is required, but no matter what kind of tools are utilized. This aspect, the chosen tools for project planning, would require more precise researches, in order to find in what level the chosen planning tools reflect the general success of the end product.

Jouni Kähkönen
Information systems project management: an agency theory interpretation


Background

The agency theory has been built from the point of view of the development group. There are two major roles in agency theory: principal and agent. The manager is the principal and the employees are agents. In some organizations the principal may be as well the functional manager as also the project manager.

The success of an IS project can be understood in multiple ways. One is the point of view of the manager. The project is successful if it has been completed on time and it has produced the desired product. From the point of view of the employee, on the contrary, the project may be held successful if the employees have overcome some major difficulties or they have been able to increase their own skills. What affects to the outcome of an IS project is environmental variables which may cause difficulties in terms of succeeding. These variables will be discussed in detail later.

Results

The survey was conducted by interviewing managers from enterprises which represent different branches of industry and which are of different sizes. Among other things, the interview contained questions on contracts and compensations, monitoring employees’ work and creating routine for the developers.

When it comes to agency theory in IS projects the main thing is the type of contract. With the contract it is possible to determine whether an agent gets rewarded based on the outcome of his work or based on the time he has spent on the project (behavior-based contract). Considering the agency theory, the outcome-based contract type should be the better contract type because it makes the agents work in the best interest of the principal. This approach also makes monitoring futile in most cases because agent’s reward is based on the outcome. Monitoring can, however, result in better outcome than leaving it undone. The rewards are often something else than financial rewards, for example technical training or flexible work schedule.

Monitoring is performed in order to keep the principal up to date of the agents’ work. As a tool, Microsoft Project was the most commonly used project management software. The most popular techniques were periodic progress reports and periodic team meetings.

There might occur goal conflicts when agent’s interests are different from those of the principal. With outcome-based contracting the possibility of such conflict may be minimized whereas the behavior-based contracting does not have such effect. The goal of
an agent may be to get the produced code into use or simply to produce as good code as possible in order to create high quality systems. So when project managers aim to deliver a project in time and in agreed condition, the developers might concentrate on keeping themselves valuable in the labor market, i.e. they might aim to develop themselves instead of completing the project.

Shirking is a common problem. In IS development that means that the agents use their time on other things than those they should be doing. Such activities may be for example helping others instead of concentrating on own tasks, surfing the Internet and playing games, educating themselves on a skill that is of minor or none importance considering the project. Sometimes developers might demand more and more precise instructions from the client instead of beginning the actual work.

Lying, exaggerating and not reporting everything are forms of privately-held information. That means that an agent is working for his own interest, is willing to keep the manager away from him etc. and thus using means mentioned above to succeed. The most common type of privately-held information was over-reporting the percentage of completed work. That means it is not uncommon for the developers to report the project being 80 % completed while the reality is something much less.

Another feature in agency theory is task programmability. Task programmability means the rate in which the behavior of an agent can be predicted. The more routine the work has the better it is programmable. That also correlates with the use of outcome-based contracting. The lower the task programmability rate is the frequenter is the use of outcome-based contracting.

There are four reasons according to the agency theory why the IS projects fail. First reason is the little use of outcome-based contracting, secondly the projects lack of sufficient monitoring. Third reason are goal conflict, shirking and privately-held information which together cause delays etc. The fourth reason is that organizations are somehow not capable to exploit task programmability.

Agency theory is not only a simple theory but tries to be also a tool for managers to improve IS development projects. Therefore it builds a framework which consists of the six aspects: contract type, monitoring, goal conflict, shirking, privately-held information and task programmability. IS development project managers should be able to reflect their project against these aspects and thus avoid the failure of the project.

Conclusions

The agency theory is a means for project managers to better improve the IS development project they are working on. The theory is, however, not yet completely tested on IS projects. In future research, agency theory should be implemented on various IS projects and then create a statistical analysis which could show the wider applicability of the theory on IS projects. Another implication for research is that the agency theory does not cover all possible risks there are in an IS project. Therefore also risks outside of agency theory should be studied and their impact on project success investigated.

The article was written already in 2003, only two years after the agile manifesto. Therefore it would be also worthwhile to study whether agile methods make the agency
theory obsolete or can they work together. If it is possible to combine agency theory framework as management tool with agile methods, it might, in my opinion, lead into interesting results when it comes avoiding failure of an IS project.

Teemu Keskinen
Integrating system analysis and project management tools


Background

There are many CASE tools like Rational-Suites are trying to automate the entire ISLC and commercial tools. One conclusion that emerges from a thorough review of both areas is that tools for systems modeling and software engineering are much more heterogeneous than project management tools. Gantt and Pert diagrams have become dominant project management modeling tools and are currently included in standard PM software such as MS Project. A survey of 1000 project managers has found that 48.4% use MS Project, 8.5% use MS Excel, and the rest use Gantt/Pert-based tools from other vendors. More than 50% use Gantt/Pert-based project management software to manage every project, independent of its application domain and characteristics.

By contrast, there are three commercial CASE software packages demonstrate the heterogeneity of tools in the area of software engineering. The first one is PowerSoft offers Power-Designer that supports DFDs, for procedural system modeling, and Entity Relationship Diagrams (ERD), for database modeling. Secondly, Oracle’s Designer 2000, supports Functional Hierarchy Analysis and ERD. Lastly, the most famous tool is that Rational offers is Rose, a UML modeling tool.

There are some questions to consider:

1. Is the gap between CASE and PM tools bridgeable?
2. What can be done to smooth the transition between CASE and PM tools?
3. How can analysis and design data, collected by CASE tools, become directly available to planning and control processes, supported by PM tools, without being subjectively interpreted or biased?
4. Is there a way to improve software modeling and engineering by introducing a managerial perspective in addition to the technical perspectives?

The answer in the paper is:

1. Yes, it is possible to bridge the gap between CASE and PM tools using mapping models as the required bridges. DFD mapping into Gantt–Pert Diagram, for example, could be regarded as validity and feasibility proof of this concept.
2. Transition between CASE and PM can be achieved using existing commercial tools. There is no need to develop new tools, which will enable analysis tasks as well as management tasks. It is sufficient to look for solutions to bridge tools from both areas.
3. Analysis and design data, collected by CASE tools can become directly available to
PM tools. This can be achieved if and only if repositories of both kinds of tools are shared.

4. A managerial perspective is essential, as evident from many case studies reflecting the need to introduce a managerial perspective in addition to the technical perspectives.

Results

Mapping model. The normal problem is that the lengthy and costly ISLC is fostered by disparate tools without effective consolidation of data generated throughout the overall ISLC. Thus, the present study focuses on mapping DFD and FH systems modeling objects into Gantt and Pert project management and control diagrams, thereby integrating these two ISLC aspects.

Basic mapping of DFD objects into Gantt diagram objects:
1. Each of the external entities are represented only once for input (if they produce input) and only once for output (if they produce output).
2. Each Read Only (RO) data store and each Read/Write (R/W) data store are represented only once for input and for output.
3. Each basic flow appears only once in every Gantt diagram.
4. Each basic process appears only once in every Gantt diagram.
5. OR connections between flows are not represented in the absence of parallels in Gantt diagrams. Logical connection traits between flows can, however, be included within basic process characteristics, thus maintaining mapping completeness.
6. A general process is represented by means of a summary task, i.e. a grouping of activities and flows under a general name.
7. General flows are those that connect between summary tasks.

The basic mapping model can improve the output Gantt diagram:
1. Enabling Gantt’s activity dependencies to be different from DFD dependencies. In the basic mapping model, Gantt activities are in the same sequence order as in the DFD.
2. Decomposing feed flows from and to external entities into the following types: User interface, request for a report, and fixed-time or real-time activity alerts.
3. Decomposing data-writing flows into the following types: Insert, Update, and Delete.
4. Applying the additional possible attributes of DFD components listed in Table 1.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Process</th>
<th>Flow</th>
<th>Entity</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>And/or alert</td>
<td>Arrival manner</td>
<td>Internal/external</td>
<td>Internal/external</td>
</tr>
<tr>
<td>2</td>
<td>And/or output</td>
<td>Frequency</td>
<td>On/off line</td>
<td>Access protocol</td>
</tr>
</tbody>
</table>
A repository of a commercial tool containing such a schema will enable integrated managerial and engineering data interchange.

The repository contains three main components:

1. <DFD Components Dictionary>—Tables: [DFD COMPONENTS], [COMPONENT TYPES], and [PRECEDENCES].
2. <Data Items Dictionary>—Tables: [DATA ITEMS], [ITEM SYNONYMS], [DOMAINS], and [RELATION GROUPS].
3. <Ascription of Data Items to Basic Flows>—Table: [DATA ITEMS in FLOWS].

Conclusions

Estimating development cost and time is difficult, no matter which methodology is applied, for the following three reasons:

1. First, estimation must be based on quantitative data, which are available only after providing the results of analysis to members of the development team who, for the most part, are not involved in the analysis stage.
2. Second, methods for evaluating development time and cost are hardly an integral part of system analysis and design tools. The same holds for methods for estimating the response time of the system under development.
3. Third, accepted models such as COCOMO, are based on a presumptive assessment of archaic factors such as program size and LOC. Today’s fourth generation software provides little in terms of certainty or accuracy.

In the absence of an integrating layer, the mapping model presented in this paper has the following advantages:

1. The mapping model enables effort and cost estimation for information system development to be an integral part of conventional analysis and design methodologies, i.e. DFD and FH. This is due to the possibility of deriving assessments directly from system analysis and design raw data, as opposed to relying on aggregates or other secondary sources.
2. The mapping model enables extension of the development effort and cost assessment process, beyond the mere aspects of <Time> and <Resources and Budget>, to include also <Complexity and Quality>, yielding expected system response times, and data volume traffic.
3. The mapping model bases the estimation process on the general project management and control tools Gantt and Pert, which are widely used and supported (e.g. MS-Project).
4. Use of Gantt and Pert diagrams enables dynamic control of the estimation, based on
reports regarding actual progress. This provides for projected-to-actual comparisons of cost, and moment-to-moment updates of CPM calculations, as opposed to the static control methods customary in the field of system building.

5. Use of Gantt and Pert charts allows “drilling down”, into the system code design, including master routines and system major service routines. This differs from current methods used for information systems development, which are limited primarily to the area of functionality.

6. The potential of a detailed drill-down concerning system code design provides for engaging and integrating the technical team (development managers) as early as the analysis stage. This results in a more reliable and accurate estimation base for the entire system development project.

By reading the paper, I can see it shows the feasibility and validity of translating system analysis objects into project management objects. The mapping method aims to integrate common information system analysis and design methodologies with standard project management and control tools, can potentially improve estimation, planning, and control of software development projects, in terms of cost, time, and management.

Xie Liwei
Coordination in co-located agile software development projects


Background

Agile software development has proven itself to be a successful software development model. There has been a lot of research on effective coordination and effective coordination has been acknowledged to be a critical element especially in software development. However, the studies on effective coordination have been conducted prior to agile software development taking place in common use. The article identifies agile practices with coordinative functions and develops a model for coordination in agile software development projects.

Results

To explore on-going software development projects and to be able to build up a theory, the research was conducted as a positivist multi-case study research. There were three projects (cases) selected. All of them had co-located teams. All selected projects were from different type of organization using different type of development contracts. This ensured variability between selected projects. All three projects had different kind of coordination difficulties and different level of experience on agile software development projects. All selected projects located in the capital of New Zealand.

Data was collected mainly in semi-structured interviews. Data also included field notes and photographs from visits at work sites and project documents. Data was analysed in two different ways: Dependency-focused analysis and Activity-focused analysis. This approach was used to find proper associations between dependencies and coordination mechanisms.

A theoretical model of coordination in co-located agile software development projects was created. This was achieved by combining evidences found in qualitative analysis of the data. Theory is supposed to act as a starting point for understanding coordination in agile software development projects. Theory is not proposed to be a finished product.

The model has three distinct components: Synchronisation, Structure and Boundary Spanning. These components are mechanisms (activities and artefacts) that form a coordination strategy. Synchronisation stands for activities and artefacts used to synchronise the project team knowledge of project (e.g. weekly meeting). Structure has three categories of coordination mechanisms: Proximity (Are all members in same open-plan room?), Availability (Are members always available to work for the project?) and Substitutability (Can team members do each other’s work?). The third component Boundary Spanning occurs when team or one of its members is interacting with someone
outside project team.

There is also Coordination Effectiveness in the model. Coordination Effectiveness is an outcome of a coordination strategy. Coordination Effectiveness is divided to implicit and explicit coordination. Explicit coordination contains physical objects and can be described with “Right place”, “Right thing” and “Right time”. Implicit coordination occurs without explicit message passing and can be described with “Know why”, “Know what is going on and when”, “Know what to do and when”, “Know who is doing what” and “Know who knows what”.

The model also presents nine propositions which describe what type of coordination strategy should be used under different conditions and how different kind of coordination strategy leads to a different kind of coordination effectiveness. First four propositions focus on how coordination strategies increase coordination effectiveness. The latter propositions suggest what kind of coordination strategy should be used in certain circumstances.

The first proposition emphasises the importance of a) synchronisation and structure when the customer is included in the project team and b) synchronisation, structure and boundary spanning coordination mechanisms when customer is external. Second proposition indicates that synchronisation activities increase implicit coordination effectiveness. Third proposition addresses structural coordination to increase implicit coordination effectiveness. According to fourth proposition boundary spanning mechanisms increase explicit coordination effectiveness.

Proposition 5 expresses that increasing frequency of iterations will help to maintain coordination effectiveness under conditions of high project complexity. Changing the priorities of stories will help to maintain coordination effectiveness when project has uncertainty as proposition 6 points out. Propositions 7 and 8 illustrate the superiority of project team focusing on a single project relative to team working on multiple projects when it comes to the terms of proximity and availability. Proposition 9 extends coordination maintenance in situation of uncertainty by suggesting to increase boundary spanning if the customer is not part of the project team.

**Conclusions**

The article provides a theoretical model of coordination in co-located agile software development projects. The model is not completed and further work is needed.

Even though the propositions in the new theoretical model are common knowledge to any experienced and self-respecting scrum master, it is important to have this kind of theoretical base for further studies on coordination in agile software development projects.

The model provides guidance to those who are undertaking or adapting their agile approach. The model guides to pay attention to synchronisation, structure and boundary spanning in coordination in agile software development projects.

Markus Leinonen
Can distributed software development be agile?


Background

Methods used for global, distributed development and agile development methods are fundamentally quite different. Whilst agile methods are based on highly informal and low-threshold communication, team cohesion and fast adaptation, it’s commonly thought that in order to work efficiently, distributed software development should rely on formal, verified process.

For this paper, a study was conducted in order to research whether it’s possible to combine these methods and gain the benefits of both methods. On the study, case study using semi-structured interviews was conducted on three companies using agile distributed software development methods. As a result, the researchers found five groups of practices that tackle the challenges of the distributed agile development.

Results

Both agile and distributed software development methods have their own challenges related for example to communication, lack of control, lack of trust or team cohesion. During this study the researchers were able to identify five “new challenges” for agile distributed development (p. 42).

1. Communication need vs. communication impedance (Challenge 1)
2. Fixed vs. evolving quality requirements (Challenge 2)
3. People vs. process oriented control (Challenge 3)
4. Formal vs. informal agreement (Challenge 4)
5. Lack of team cohesion (Challenge 5)

From the case study conducted on the three software companies using agile distributed development, the research group found these successful practices for overcoming these challenges and combining the best aspects of these traditionally contradictory development methods.

Continuously adjust the process. Instead of fixating for a certain set of practices offered by a specific development framework, the companies successfully evolved their development practices suiting for their needs at the time and altered for example their way of specifying requirements.

Facilitate knowledge sharing. For balancing between the agile informality and the documentation needs of distributed development, the companies facilitated several
practices. Common project repositories were used for maintaining the necessary documentation. Iteration time-boxes were modified in order to allow teams to gain better understanding of the domain and requirement prioritization was adjusted to favor well understood functionality in the beginning of the projects in order to ease the learning curve.

**Improve communication.** Communication was noticed to be a major challenge especially when related to asynchronous working hours and working on vast distribution on different time zones. Project coordinator roles were emphasized and even increased in order to balance the coordination responsibilities and formal channels were facilitated to support also informal communication. All these actions however were based on the fact that communication in distributed agile development team should be constant: whether through meetings, online chats, video conferences or emails.

**Build trust.** Building trust is essential when using development methods of minimal formal control. Team cohesion was empowered by frequent visitations – enabling that precious face-to-face –time between representatives. It was also detected, that high-performance teams with prior experience on working together achieved more cohesive groups.

**Trust but verify.** Contradictory to agile methods traditionally, it was observed that the companies used a lot of practices to formally verify their processes. Distributed quality assurance – teams or representative persons – were used to review the development process and the most critical artifacts from informal communications were documented in order to supplement the communication.

These sets of practices all address the five challenges the research group identified in the beginning of the study. Challenge 1 was addressed by improving the communication and facilitating knowledge sharing – constant communication through different channels. Challenge 2 was addressed by trust but verify practices; using distributed and constant quality assurance enables control over quality without a heavy contract structure. Challenge 3 was addressed by continuously adjusting the process and trust but verify sets of practices: continuously evolving development environment is usually providing just enough tools to the current situation and again, constant quality assurance allowed the companies to gain the level of control needed for the organizational norms without the cost of flexibility. Challenges 4 and 5 were addressed by the build trust practices: cutting back the formality of the communication and trust gained through it enabled the teams to adapt the rapidly changing environments. By enabling site visitations, the team cohesion was improved even more to support the environment of working towards a common goal.

**Conclusions**

Based on findings from the case study compared to the challenges presented on the study, it can be said that incorporation of agility to distributed software development is possible. Acknowledging the challenges is essential in order to find the proper practices for addressing them. Practices presented on the paper are demonstrations on how these challenges can be approached.

Personally, I find the study and results very consistent to my own experiences of
distributed agile development. Communication and team cohesion are hard enough challenges in non-distributed development environment, let alone when working with different cultures, languages and time zones. However, I find that agile software methodologies bring value to distributed work as well when applied with care and consideration of the current project status. Tweaking and evolving the agile practices is something that is rarely talked about, but I think that this paper and the practices demonstrate the value of embracing the change on agile practices as well.

Jenna-Riia Oldenburg
Obstacles to decision making in agile software development


Background
Earlier research has shown that decision making is more effective when done in group than when done by individual, because teams can pool knowledge and information. However, the obstacles to decision making in agile software development environment are poorly understood. This research shows what kind of decisions are made in four subsequent stages of iterative development cycle (planning, execution, review and retrospective), and what are the obstacles when facing these decisions.

Results
The research was done by interviewing 18 agile team members from focus group of 43 agile developers and managers across 5 organizations (p. 1242). The participants were attendees at a large software development conference.

The work was done in two phases. A focus group research was conducted where the participants had open discussion on decisions made during particular periods of an iteration. First, the group was split into smaller groups where different types of decisions were identified and written to posters on a wall. Then the findings were discussed within the whole focus group. One researcher acted as a facilitator and the other observed the session while taking notes. The topics covered in discussion were (p. 1242):

- How decisions are made in the context of ASD teams
- Decisions participants make in the four Agile periods
- Participant perceptions of decision making obstacles to decision making during the four periods of the iteration cycle
- Issues and complexities related to decision making (to flesh out any other obstacles)

When an obstacle was identified by participant he was asked to provide evidence or examples for the researchers. In the end, 32 decisions (p. 1244) and six different obstacles to decision making were identified by the focus group. The identified decisions indicate that agile teams tend to focus more on tactical rather strategic decisions, which is likely because of the short time-boxed development iterations (p. 1244). Each identified obstacle was further researched in separate case study by having face-to-face interviews with selected members of the focus group. A total of 18 interviews were conducted to illustrate and add examples to the obstacles found in the focus group phase (p. 1242).
The identified obstacles to decision making are:

“Agile team members are unwilling to commit to a decision and rely on the Scrum Master for decisions”. Agile team members are often not willing to make the decision, leaving the responsibility to Scrum Master or some other authority. Lack of commitment regarding architectural decisions is typically due to lack of sufficient expertise in cases when problem is complex, and in case of measurement decisions there can be difficulties in choosing correct metric when uncertainty affects the decision maker, e.g. when measuring progress against DOD criteria (p. 1245).

“Agile teams face conflicting priorities for decisions”. In agile context the team hierarchy is flat, including the customer representatives. This creates conflicting priorities in decision making, stressing the prioritization work and relationships (p. 1245).

“Decisions are based on unstable staff availability during an Agile iteration”. If team members are pulled to other projects during iteration, the context of the situation is changed and scope decisions are impacted. In addition to work contribution, the pulled member might have information the other members don't (p. 1246).

“Agile team members are not implementing decisions and are relying on others for decisions”. If the decisions or changes to decisions are not tracked it might result in decision not being implemented. Environment where decisions do not result in implementation can cause people to stop making decisions (p. 1246).

“Agile team members are not taking ownership of decisions despite ASD team autonomy”. Lack of ownership or accountability of decisions may cause a situation where decisions are made easily, but not followed through. In the case study this was especially visible in strategic decisions made on retrospective, where everybody agreed on problems but no one took the ownership of decisions until the team started make actions and owners clear (p. 1247).

“Collaborative group decision making on ASD teams prevents experts from making decisions”. ASD team members may lack empowerment to make decisions, causing uneducated decisions to be made and preventing the team from working in self-organized fashion (p. 1247).

The obstacles were mapped to a DDM principles to show which DDM principles the obstacles are related (p. 1246). The mapping is shown in Figure 1.
Figure 1: The decision making obstacles mapped to DDM principles (p. 1246).

Conclusions
The research examined the types of decisions made in planning, execution, review and retrospective phases of agile iteration cycle, and obstacles when making the decisions. The iterative cycle tends towards tactical short term decision making, as opposed to strategic long term decisions, because the decisions are based on the goal of the period where the decision is made. The six identified decision obstacles are unwillingness to commit to decisions, conflicting priorities, unstable resource availability, and lack of implementation, ownership or empowerment. The focus of the research was identification of the obstacles. Finding solutions and measuring impact is left for future research.

Pekka Kauppila
Lean software management: BBC worldwide case study


Background

The term Lean comes from Toyota’s way to do more with less compared to other companies in automobile industry. The case study examines how the ideas of lean, such as reducing lead time by reducing work-in-progress (WIP), pull and continuous improvement, can be applied to software project management. This is based on the idea that although software development is somehow different from the product development of manufactured products, the principles can be applied at higher level. Lean as a management approach has already been applied to different industries. Previously there have been successful applications of single concepts from Toyota Production System such as kanban in the area of software development.

Results

The research hypothesis for this single case study was that the application of lean ideas would improve the capability of a software development process (p. 22). Data for this study was gathered from several different sources, such as direct observations, semi-structured interviews and statistical analysis of outputs, to assist accuracy. The case study focused only on a single development team employed by BBC Worldwide for over a 12-month period. The lean approach was only implemented on the team, so there was a constraint of working with existing corporate standards when dealing with things that could be not decided by the group itself.

Team used value stream mapping to draw the work content of all the stages of the development life cycle to kanban cards. Then the amount of WIP allowed for each stage was determined using the bottlenecks. The concept of Minimum Marketable Features (MMF) was also used. It is the smallest independent entity which returns value to customer in form of delivering a subset of customer requirements. Team used visual management tools such as information radiators and kanban boards to make the progress transparent. Daily standup-meetings were held to check the status of each task and if needed, to decide a proper countermeasure for emerging problems.

The improvement of capability was measured through several different metrics. The predictability and effectiveness were measured by lead time, development time and release frequency per month. Quality was measured by life defects per week and the continuous improvement. Lead time is the net working days from the customer approval of requirements to the delivery of the required feature. Development time shows the duration of a task from the point it was released to work to the point it was considered
Release frequency per month shows the number of items released to customers. Life defects are bugs reported by customers during a week plus the bugs still open. Continuous improvement measured total number of problems identified and the number of days problems prevented further work.

The results of the case study showed the software was being delivered with 47% less variance and on average 37% quicker. The variation in delivery times was reduced by 78% from 30.5 days to 6.8 days. The mean time to develop fewer and smaller software features declined by 73% from 9.2 to 2.5 working days. (p. 26.) As expected, the frequency of releases increased. The variation of live defects reduced, which indicated that the bugs were being fixed more quickly. The mean number of bugs open after each week decreased (p. 26). It meant that the problems with the process were resolved more quickly. Also the total number of problems identified increased, but the average number of days work was “blocked” by the problem fell sharply. (p. 27)

Analysis shows that Lean with low WIP and pull approach meant that the size, complexity and volume of work input were reduced. Cycle time was improved by reducing variance through minimizing the size of the units of work and continuous improvement. Team was able to deliver new functionality faster and more predictable.

The more frequent releases reduce technical and market risks as it allows customer to evaluate tangible products. The data on bugs and continuous improvement shows that bug rates fell. Problems were solved more quickly which also improved productivity. However, Lead time could be further reduced by reducing the time when proposed ideas transformed to decomposed engineering ready.

There are also drawbacks. Lean does not work well with targets and milestones and standardized corporate reportings as it is an emergent approach. To truly deliver value to customers will require the development team proactively to work with customers to define and analyze their problems and to see if business value is actually created. This can be seen as a too big of an enlargement to team’s current job description. Lean also requires managers to be facilitators which could be difficult for certain managers. Team itself could show reluctance as staff is not used to on being encouraged to identify problems.

With short iterations and daily checks, it might be easy to think that Lean and Agile are alike. However, Lean differs from Agile approach in several ways. Lean is based on pull compared to push and batch model in Agile. In Lean, data is seen as source of empowerment and not as control tool for management. In lean is not possible to “cherry pick” work tasks as in Agile, but the staff needs to be multiskilled and able to work on almost any area. Lean approach also requires collaboration of team members in solving the problems preventing work.

**Conclusions**

The use of lean methods such as visual management, team based problem-solving, smaller batch sizes and statistical process control can indeed improve software development. Methods allowed good use of resources and fast response-time with a focus on delivering highest value to the customer. Lean also encouraged other beneficial
improvements, such as developing team skills and reducing staff turnover (p. 30). However, lean is not a substitute for professional software engineering practice. It also has a few drawbacks as it might not fit well with existing corporate standards.

Katja Sorjamaa
Benefits realization management: Panacea or false dawn?


Background

Benefits realization management (BRM) has always been a debatable topic for project and program management. It is an essential part of management practice where public money or sources are to be used effectively. Benefits realization management is therefore important to be practiced in an effective way ensuring that projects are done in a beneficial way.

According to Bradley (2006), BRM has been originally developed in 1980s and 1990s when people started realizing that they should get returns or benefits of their investments. This management technique then became popular and finally in 2009 a ‘Benefits Management Specific Interest Group (SIG) was founded by the Association for Project Management (APM) in UK. Now with benefits management, several other management techniques such as change management, performance management and portfolio management is also integrated which as a whole is responsible for maintaining benefits from projects.

As mentioned in the title of the paper, author here has mainly focused on answering the doubts of public, where they get confused on whether applying benefits management is an actual benefit of the project or is it just something which has nothing to do with the change in management practice for higher benefits.

This paper focuses on how organizations actually operate and to what degrees and in what ways BRM is being used in an organization. There are different studies that try to explore in detail why benefits oriented practices were not being adopted systematically.

Jenner (2009) studied on the reasons of failure of different ICT projects and tried to come up with the reasons behind unsuccessful management practice of these ICT projects. Ashurst et al. on a different study suggested that the reasons for unsuccessful projects could be due to the lack of awareness, clients and consultants delegating responsibility of benefits management to each other or unexpected and unintended consequences (p. 344). This applies to all the organizations including IT (IS) industry where there is less attention given to manage and realize benefits. Still the question arises, is it a panacea or false dawn?

Results

The study of this paper concerns on why BRM is not followed straightforward, done by evaluating the past experiences. The author of this paper used to work as a government
officer in the 1990’s and early 2000s, hence is using his experience in management practice.

For exploring the past benefits in BRM, author studied on the practices of BRM in a regeneration program that was carried out in north of England. The main goal of elaborating on this study is to visualize what changes are needed in the current day practice of BRM. This program was named as ‘Area Based Initiatives (ABI)’, funded by UK Government.

Below shown benefits hierarchy in regeneration makes a clear overview of activities, outputs, outcomes, objectives and vision of the ABI regeneration program.

![Benefits hierarchy in regeneration](image)

Fig. 1. Benefits hierarchy in regeneration.
This project was under tight benefits management supervision of Government, ensuring that it would benefit public sector. The way in which benefits management practice was done would start from local partnerships submitting funding bids based on disadvantage area, education etc. (p. 345). Benefits hierarchy maintained in that regeneration program is shown in Fig. 1. Along with this there was annual reporting and individual evaluation also during middle and at the end of the program (p. 345). Author mentioned this program in a hope that there will be a lesson from this to current BRM practice in organizations. However, there were also various issues encountered in different phases like defining benefits and collecting data, setting targets, attributing benefits, weighting different benefits and loss, timescales for benefits realization which were analyzed and adopted in the management process.

Conclusions

The main conclusion that the author has tried to mention in this article is that even though there is not a huge benefit by adopting benefits management system for your project, it certainly is not a false dawn. The main thing that needs to be mentioned is that ignoring the complexity of BRM if it is practiced in the organizations also in such where BRM is not imposed to apply, it is definitely beneficial. Author with the case study of the regeneration program (Fig 1) has tried to give facts on such benefits. With such practice, benefits management will fill the gap between change management and program management. (p. 350)

This paper has reflected upon several potential benefits of using benefits realization management practice in a project. It has tried to clear the doubts of management practitioners that BRM will not be a false dawn if applied as a management paradigm over traditional management practice. However, its relation to profile/portfolio management and change of project/program management still needs to be discussed. A further elaboration on practical implications of BRM also seems to be studied. Overall, it was an informative paper with lots of information and a case study of regeneration program to light upon authors’ idea was worth mentioning.

Anju Thapa
Empirical findings on team size and productivity in software development


Background
The article deals with analysis of the relationship between team size, productivity and other variables in Software Development. The analysis is based on The International Software Benchmarking Standards Group’s preprocessed repository data.

The main preconception behind the study is that there's a strong relation between productivity (that is, efficiency) and team size, in addition to programming language and development platform. The aim of the study is to systematically examine this idea by using statistical analysis and thereby provide some useful information to help software project management decision making and estimation.

Results
The International Software Benchmarking Standards Group's (ISBSG) software project management repository is being evaluated in the article. The data set taken from the repository can be comprehensively filtered with different type of parameters. For example, the data set for this analysis was filtered with following parameters: Data Quality Rating (A and B included), Unadjusted Function Point Rating (A and B included), Count Approach (IFPUG, NESMA, Albrecht, Dreger and the variants of the previously mentioned included), Recording Method (Staff hours (Recorder) included) and Resource Level (Level 1 included). Regarding to nominal attributes, some homogenization was done to the fields with equivalent value and further data set filtering was carried out based on nonrepresentative frequency values. The analysis described in the article was done based on the ISBSG database release 10.

It's stated that in 75% of the data the team size has been less than 10 people and the productivity is worst in the projects with average team size (ATS) equal or more than 9 people. One factor that makes difference in the productivity level is whether the project is new development or enhancement project, the latter being more productive. The possible reason for this is that the development team already knows the application and the domain.

As the previously mentioned preconception suggests, there's productivity difference between chosen programming languages. Visual Basic (VB) seems to be the most productive, whereas C++ and Java are among the less productive languages. The difference between C++ and Java (in favour of C++) is considered to arise from lack of experience in Java projects. In general, fourth generation languages seems to have better overall productivity values than the languages that are considered to be numbered among
third generation.

It's pointed out that some statistical productivity difference between different organisational types was expected. Anyhow, with the exention of Ordering and Voice Provisioning, this preconception seems to be false. Since these two exceptional types have only approximately 9% representation in the repository, this dissimilarity does not have a significant importance.

A simple model to relate productivity with the most important relating attributes is presented in the article. This kind of model can be used by project managers to make rough estimations about the relation of productivity and the number of personnel working in the project.

Conclusions

Three main threat types for the validity of the study is presented: Construct validity (how accurately the variables used measure the concept that they should measure), Internal validity (how accurate are the conclusions in the data preprocessing stage) and External validity (how representative is the data used in the study).

The study shows that there exists a relation between productivity, team size and the programming language used in the project. As anticipated, the study confirms that projects with 9 or more team members are less productive than projects with under 9 people team size. Further on, the enhancement projects are more productive than new projects and mainframe projects are less productive than projects done with other platforms.

In future, the ISBSG repository will evolve and further statistical and data mining research will be made to study its quality, as well as other similar repositories. The general assumptions regarding the productivity will be furthermore tested with the data collected from these repositories.

Tuomas Granlund
Software process improvement success factors for small and medium web companies: A quantitative study


Background

This research paper is about the quantitative study of software process improvement (SPI) success factors for small and medium web companies of Pakistan. 21 participants from 11 different software companies were taken into account and data retrieved through interviews were analyzed quantitatively. Authors have differentiated between traditional and web applications from different angle to focus specifically on software process improvement (SPI) for web development companies. Pakistan was chosen for case study because it has rapidly growing of many small and medium web development companies which export about US $ 1.4 billion (p. 482).

Result

The primary concern of this research paper is to evaluate the software process improvement (SPI) success factor for small and medium Web companies. Semi structured, open ended interviews were conducted with 21 participants from 11 different Pakistani software companies and data were quantitatively analyzed using the Glaserian strand of grounded theory research procedures in order to evaluate the software process improvement success factor. SPI success factors play a major role in the implementation of successful process resulting in quality software, lesser development time and enhance productivity (p. 482). A study was made to illustrate how critical SPI success factors for software companies were identified in Habib’s, Pino’s and Khan’s review.

A systematic literature reviews (SR) on the studies that were related to SPI for small and medium Web companies were made and it was noticed that none of the studies have suggested a specific model for Web SPI. The SR identified the need of new model that specifically deal with small and medium Web companies for evaluating SPI. Many SPI factors such as automated tool support for process, use of measurements and metrics, project monitoring through reviews, feedback from discussion, employee training and many more were identified from SR. After SR, a replication study was made to find out whether the theoretical model of SPI success factors proposed by Dyba which was related to small and large software companies, would also be applicable to small and medium
Web companies (pp. 483-484). The result of replication study proved that there are still some other critical SPI success factors for small and medium Web companies.

The quantitative research methodology was employed and data were analyzed using grounded theory procedure as proposed by Glaser and Strauss. Data were collected from the professionals of Pakistani Web companies using open ended interviews. The interviews were taken in native language (Urdu) and translated into English. The translated information was then transcribed into codes using coding techniques. Open and focused coding was used for transcription and theoretical coding was used for establishing relations among different categories as explained by the principle of grounded theory suggested by Glaser and Charmaz.

The product of this research is the initial exploratory framework of SPI success factors for small and medium Web companies. It includes the core categories of SPI success factors for small and medium Web companies and the relationship among them. The categories were identified by performing open and focus coding techniques and relationships among them were identified during theoretical coding. The different categories identified in the initial exploratory framework of SPI success factors for small and medium Web companies are SPI success, SPI goal and benefits, automated tool support, client support, communication, company vision, cost benefit analysis, employment support, gradual approach, higher management support, SPI consultancy, SPI implementer role, SPI measurements, SPI supportive policy, tailoring of processes, applying knowledge about processes and SPI awareness program (pp. 500-501).

Conclusions

This research has identified software process improvement (SPI) success factors for small and medium web companies using quantitative research methodology under the principles of grounded theory. Studies like systematic literature reviews and replication studies were performed to make sure that SPI success factors for small and medium Web companies exists or not. It should be noted that there were no such research that has resulted the SPI success factor specifically for small and medium Web companies. Semi structured, open ended interviews with 21 professionals from 11 different Pakistani small and medium software companies were conducted in native language (Urdu) to collect the data and the collected data were translated into English and then transcribed upon which open, focus and theoretical coding were done in order to achieve important categories and their relationships. The core categories and their relationships were integrated together into a framework of SPI success factors of small and medium Web companies. The findings of the research have contributed a lot in the field of Web development by small and medium companies and have opened the area for further exploration and research.

Sundar Kunwar
Testing in the software development life-cycle: now or later


Johdanto

Virheettömyyden saavuttamiseksi käytetystä keinoista, ohjelmakoodin katselmoinneista ja testaamisesta, on tullut keskeinen osa ohjelmistokehitysprosessin elinkaarta. Kuitenkin, on vähän todisteita siitä, että resurssien lisääminen näihin kehitysvaiheisiin yksistään vähentäisi virheitä tai kiehittäisi ohjelmiston kokonaislaatua.

Tutkimuksen tarkoituksena oli simuloida kahta vaihtoehtoista toimintatapaa testauksen toteuttamiseksi ja selvittää, missä ohjelmistoprojektin elinkaaren vaiheissa testausta kannattaa toteuttaa. Ensimmäinen näistä toimintatavoista oli testauksen suorittaminen vasta sen jälkeen, kun kaikki ohjelmakoodi on kirjoitettu. Toinen oli testauksen suorittaminen rinnakkain, kunkin kehitysvaiheen päättäessä.

Tulokset
Tutkimuksessa käytettiin tutkimusmenetelmänä hypoteettisen ja geneerisen ohjelmistokehityspjektin simulaaatiota, mikä mahdollisti tutkimustulosten yleistämisen eri kehitysalustoille ja -kielille.

Esimerkkiprojektin kehitysprosessi jaettiin kolmeen vaiheeseen: kannattavuustutkimukseen, toimintojen suunnitteluun ja ohjelmointiin. Kehitysvaiheet esintyivät aina edellä mainitussa järjestyksessä, mutta niiden välillä saattoi myös olla päällekkäisyttä, esimerkiksi ohjelmointi voitiin aloittaa, kun 60 % toiminnallisuudesta oli suunniteltu. Ylläpitöa ei sisältyny esimerkkiprojekteihin, vaikka se tunnustettiin kriittiseksi vaiheeksi menestyksekkään suunnittelun ja toteutuksen kannalta.

Tutkimuksessa mitattiin kunkin simuloidun projektin kohdalla projektiin vaadittua aikaa kuukausina, projektiin käytettyä työmäärää miestyökuukausina ja projektin kustannuksia dollareina. Näiden mittayksiköiden valintaan päädyttiin siksi, että voitaisiin tunnistaa ne ohjelmistoprojektin vaiheet, jotka vievät eniten aikaa, työpanosta ja rahaa. Ohjelmistotuote määriteltiin luotettavaksi, kun se on 95-prosenttisesti virheetön.

95-prosenttisesti luotettavan tuotteen valmistamiseksi viemä aika ei vaihdellut merkittävästi eri toimintatapojen välillä. Ensimmäisellä toimintatavalla, eli testauksella

Sen sijaan toimintatapojen väliset erot ohjelmistoprojektin kustannuksissa olivat dramaattiset. Projektissa, jossa testaus toteutettiin kehityksen päätteeksi, saatiin kokonaiskustannuksiksi $15 298 000, kun rinnakkaisella testauksella se oli vain $4 083 000. Tämä ero selittyi sillä, että mitä myöhemmin ohjelmistoprojektin elinkaaren aikana virhe havaitaan, sitä kalliimmaksi sen korjaamisen kustannukset nousivat, koska järjestelmä joudutaan mahdollisesti tekemään uudelleen ensimmäisessä vaihtoehdossa virheitä piti etsiä ja korjata kalliissa ylläpitovaiheessa 3.03 kuukautta ja toisessa vain 0.22 (olettaen, että neljä ihmistä tekevät tätä kokoaikaisesti).

Yhteenveto

Tutkimuksen perusteella testauksella ei ole merkitystä ainoastaan lopputuotteen luotettavuuden kannalta, vaan se vaikuttaa myöskin projektin kokonaiskustannuksiin. Testausta tulisi siksi tehdä niin usein kuin se on ohjelmistoprojektin puutteissa mahdollista. On myös olemassa tekijöitä, jotka vaikuttavat ohjelmistoprojektin vaatimaan aikaan, työpanokseen ja kustannuksiin. Organisaatioiden pitäisi kyetä tunnistamaan nämä tekijät, ja kontrolloimaan niitä, jotta projekti olisi mahdollisimman menestyksekäs.

Pyry Kallio
The effects of schedule-driven project management in multi-project environments


Johdanto

Useat organisaatiot toimivat tyypillisesti moniprojektiympäristössä, jossa on samanaikaisesti työn alla useita projekteja ja kehitystiimit jakavat resurseja, kuten työntekijöitä. Organisaatiot toimivat paineen alla, koska tuotteet on saatava markkinoille ajoissa.

Projektissa projektipäällikkö arvioi tehtäviin töihin kuluvan ajan, kun käytössä on annetut resurssit. Määritelmän mukaan aikataulupaine (schedule pressure) on jännite, joka aiheuttaa kuilun projektipäällikkön arvioimien jäljellä olevien työpäivien määrän ja aikataulun sallimien tosiasiallisten jäljellä olevien työpäivien määrän välillä.

Aikataulupaine on pieni, jos projektin näyttää olevan aikataulussa ja näyttää valmistuvan ilman ylityötunteja. Vastaavasti aikataulupaine on suuri, jos projekti on tavoitettaessa virstanpylväitä ilman yliytöitä tai projektin ja sen resurssien suhteessa on käytössä ylimääräistä työvoimaa.

Aiempien tutkimusten mukaan moniprojektiympäristössä toteutettavat projektit viivästyvät usein aikataulupaineen vuoksi. Aikataulupainetta aiheuttaa osaavien työntekijöiden usein tapahtuva vaihtuminen projektista toiseen. Ylimmän johdon näkökulmasta saattaa olla houkuttelevaa kaapaata projektityöntekijöitä meneillään olevista vähemmän tärkeistä projekteista myöhemmin alkaneisiin liiketoiminnan kannalta kriittisiin projekteihin, kun organisaatiolla ei ole käytössään ylimääräistä työvoimaa.

Tulokset


Artikkelin tehtävänä on kuvata projekteihin määrättyjen resurssien kaappaamisen lyhyen ja pitkän aikavälin haitallisia vaikutuksia. Lyhyellä aikavälillä resurssien kaappaaminen saattaa varmistaa sen, että myöhässä oleva projekti saadaan valmiiksi ajoissa. Pitkällä aikavälillä resurssien kaappaaminen aiheuttaa aikataulupainetta, sillä resurssien varastointi vaatii esimerkiksi aikataulupainetta, joka on tuottelua saatettamista markkinoille ennen kilpailijoita.
Aikatauluvetoineen johtaminen, jossa käytetään resurssien kaappausta, lisää aikatauluupainetta kierteenomaisesti. Näin organisaation kyky saattaa projekteja loppuun aikataulun mukaisesti huononee.


Tuomari-autovalmistajan projekteihin kuului useita kehitysprojekteja, kuten auton komponenttien parantaminen tai tuotteessaa olevan viennin korjaaminen. Useissa projekteissa käytetään samoja komponentteja, jolloin moniprojektiympäristön rinnakkaiset kehitystehtävät vaikuttavat toisiinsa.

Tutkimuksen aikana tehdynä tietokonesimulaatiolla järjestettiin koetilanteita, joissa aikatauluupaineen määraa säädeltiin. Tehdylä kyselytutkimuksen perusteella tiedettiin, että projektien aloitushetkessä on usein viiveitä, jotka johtuvat ylimmän ja keskijohdon välisen budjettineuvotteluiden viivästymisestä.


Simulaation tuloksena todettiin, että moniprojektiympäristössä aloitettavan projektin alkamishetken myöähystyminen aiheuttaa aikatauluupainetta ja viivästymisiä muihin meneillään olevissa projekteissa. Simulaation oletuksena oli, että organisaatio toimii ilman ylimääräistä työvoimaa, jolloin ainoa mahdollisuus nopeuttaa yksittäistä projektia on työvoiman kaapamisen muista projektiteosta.

Simulaation tuloksena todettiin, että moniprojektiympäristössä aloitettavan projektin alkamishetken myöähystyminen aiheuttaa aikatauluupainetta ja viivästymisiä muihin meneillään olevissa projekteissa. Simulaation oletuksena oli, että organisaatio toimii ilman ylimääräistä työvoimaa, jolloin ainoa mahdollisuus nopeuttaa yksittäistä projektia on työvoiman kaapamisen muista projektiteosta.

Simulaation avulla tutkittiin myös äärimmäisiä tilanteita, kuten projektin alkamista useita kuukausia myöhemmin, jolloin todettiin myöhästämistä alkaneen projektin vaatineen lopulta enemmän aikaa kuin sen aluksi arvioitiin vaativan.

Lopputuloksena todettiin, että moniprojektiympäristössä johtamiskäytäntöä on suuri vaikutus projektien etenemiseen aikataulujen mukaisesti. Resurssien siirtämisessä projektien välillä tehdut virheet voivat vaikuttaa haittallisesti useisiin projekteihin moniprojektiympäristössä.

Yhteenveto

Moniprojektiympäristössä tapahtuvan resurssien siirtämisen vaikutukset voivat olla pitkäkestoisia ja mahdollisesti haittallisia organisaation toiminnalle. Kun työvoimaa

59
siirretään meneillään olevasta projektista toiseen, työvoimaa menettäneet projektit kokevat aikataulupainetta.

Projektin edistymiselle on haitallista paitsi työvoiman kaappaaminen pois projektista, myös työvoiman vaihtuminen edestakaisin projektien välillä. Aikataulupaineen lisääntyminen voi aiheuttaa kierteen, joka hidastaa kaikkia meneillään olevia projekteja. Lyhyen tähtäimen korjausliikkeet viivästynneiden projektien edistämisestä saattavat aiheuttaa pitkän tähtäimen viivästymisä koko organisaation toiminnassa, kun muut meneillään olevat projektit viivästyvät.

Juha J. Kari
A lightweight framework for describing software practices


Johdanto

Kaikki ohjelmistoyritykset käyttävät samankaltaisia käytäntöjä ja toimintatapoja kehittäessään ja luodessaan tuotteita ja palveluita, mutta käytännöt voivat olla hyvin eri lailla käytössä eri yrityksissä. Artikkelin tavoitteena on selvittää, miten menestyneet ohjelmistoyritykset ovat sopeuttaneet toimintatapansa sopimaan omaan kontekstiinsa. Tämänkaltaisia tutkimuksia ei ole juurikaan tehty, sillä suurin osa ohjelmistoyritysten käytäntöihin liittyvistä tutkimuksista käsittää tietyn metodin tai käytännön vaikutusta tehokkuuteen ja menestykseen.

Tulokset


Alkuperäisenä ideaan oli viitekehysen käyttö ohjelmistokehityksen toimintatapojen monipuoliseen tutkimukseen, mutta käyttää laajennettiin tutkimuksen edessä. Viitkehystä voidaan käyttää myös seuraaviin tapauksiin:

- selventämään päätavoitteet, mitä toiminnalla pyritään saavuttamaan
- luokittelemaan kaikki käytännöt, mitä tarvitaan perustoiminnon toteuttamiseen
- ymmärtämään mitä käytäntöjä projektin osapuolet käyttävät ja kuinka tehokkaita ne ovat päämäärän saavuttamisessa
- kokonaiskuvan rakentaminen kehitystyöstä oikeiden työntekijöiden näkökulmasta.

Kirk ja Tempero ovat keränneet todisteita kehikon mahdollisuksista toimia monipuolisten tutkimuskohteiden apuna, kuten he ovat artikkelissa ehdottaneet. Artikkelin kuvassa 6 (s. 591) on todistekartta viitkehysen toimivuudesta. Osaan kartan soluista tehdyt tutkimukset todistavat oikeaksi, mutta osaan vaaditaan edelleen lisätutkimuksia.

Kirk ja Tempero ovat keränneet todisteita kehikon mahdollisuksista toimia monipuolisten tutkimuskohteiden apuna, kuten he ovat artikkelissa ehdottaneet. Artikkelin kuvassa 6 (s. 591) on todistekartta viitkehysen toimivuudesta. Osaan kartan soluista tehdyt tutkimukset todistavat oikeaksi, mutta osaan vaaditaan edelleen lisätutkimuksia.

**Yhteenveto**

Tutkimuksen tuloksena kehitettiin viitkehys, joka tueeksi muita ohjelmistokehityksen käytäntöihin liittyviä tutkimuksia. Tutkimus eroaa aiakaisemmista siten, että siinä ei pyritä kertomaan, miten ohjelmistokehitystä pitäisi tehdä, vaan selvitä, että eri metodit ja käytännöt toimivat eri kontekstissa. Viitkehys kehitettiin kokemuksen perusteella ja se kuvaavasti, kuinka toimintatavat ovat implementoitu tietyn tavoitteen saavuttamiseksi.
Artikkelia varten tehtiin kolme tutkimusta, joista ensimmäisessä kysyttiin kolmelta uusiseelantilaiselta ohjelmistoyritykseltä viitekehykseen liittyviä kysymyksiä. Toisessa ja kolmannessa tutkimuksessa viitekehystä kokeiltiin käytännössä. Viitekehyksen menestyksekäs käyttö näissä kolmessa tutkimuksessa todistaa, että sitä voidaan käyttää tutkittaessa ohjelmistoyrityksia eri näkökulmista. Tutkimuksien aiheita voivat olla:
- käytäntöjen puutteen ja erilaisen ymmärtämisen aiheuttamat riskit projekteissa
- yritysten vertailu hyväksikäyttämällä tietoja käytäntöjen kontekstisidonnaisuuksista
- toiminnan tehostaminen tunnistettujen toimintatapojen kautta
- yritysten luokittelu niiden käyttämiens toimintatapojen mukaan.

Elina Koivulampi
A documentation framework for architecture decisions


Johdanto


Yksityiskohtanäkökulma keskittyy asioihin, jotka liittyvät päätöksien perusteluihin. Näkökulman tueksi kirjoittajat ovat laatineet dokumentaatiopohjan, jonka avulla päätökset ja niiden perustelut on helppo dokumentoida. Pohjaan merkitään mm. päätöksen nimi, nykytila, päätösryhmät, ongelma, päätös (ratkaisu), vaihtoehtoiset ratkaisut ja versiohistoria.

Suhdenäkökulma keskittyy arkkitehtuuripäätösten suhteisiin, eli esimerkiksi siihen miten jokin tehty päätös on vaikuttanut muihin päätöksiin. Suhdenäkökulman pohjalta laadittuun näkymään merkitään arkkitehtuuripäätösten tila, eli onko se esimerkiksi hyväksytyt tai hylätty.

Sidosryhmänäkökulma paljastaa sidosryhmien vaikutuksen tehtyihin arkkitehtuuripäätöksiin. Koska aina ei ole mahdollista dokumentoida koko perustelua jollekin päätökselle, voidaan dokumentoida se, mikä sidosryhmä on vaikuttanut johonkin päätökeen, eli kuka osaa perustella, miksi jokin päätös on tehty. Sidosryhmänäkymän merkitään, minkä sidosryhmien edustajat ovat vaikuttaneet mihinkin arkkitehtuuripäätöksiin ja miten. Kronologinen näkökulma keskittyy arkkitehtuuripäätösten järjestykseen, eli siihen mitkä päätökset on tehty missäkin vaiheessa suunnitteluprosessia.

Tulokset

Kirjoittajat testasivat laatimaansa dokumentaatiokehystä tapaustutkimuksessa selvittääkseen, miten se soveltuu tarkoitukseensa. Tapaustutkimuksen kohtena oli projekti, jossa kerätään internetistä haittaohjelmia, joita suoritetaan tarkoitusta varten pystytetysä verkossa ja joita analysoidaan uhkien löytämiseksi. Projektitiimiiin kuuluu neljä kehittäjää, joista kaksi vastaa arkkitehtuurista. Projektiin dokumentointiin käytetään
yrityksen sisäistä wikiä.


Johtopäätökset

Tutkimus osoitti, että arkkitehtuuripäätösten dokumentoinnin vaadi liikaa aikaa tai vaivaa. Tutkimuksessa selvisi myös, että päätösten dokumentointi helpottaa kommunikaatiota eri sidosryhmien välillä. Tutkimus osoitti, että kehys soveltuu hyvin käytettäväksi teknisten arkkitehtuurien arvioinnissa, mutta ei-tekniisi arkkitehtuureihin soveltumisesta ei löytynyt näyttöä, koska tapaustutkimuksessa ei ollut sellaista.

Analyysi osoitti kuitenkin myös, että neljä eri näkökulmasta eivät täysin kata kaikkia sidosryhmien huolenaiheita. Sen takia kirjoittajat jatkokehittivät laatimaansa dokumentaatiokehystä lisäämällä siihen viidennen näkökulman, joka keskitteytyi niihin asioihin, jotka vaikuttavat arkkitehtuuripäätösten tekemiseen (force).

Miia Ketolainen
Research on agile project management with Scrum method


Johdanto


Kritiikin jatkotuotteena syntyivät ketterät menetelmät, joita voitaisiin luonnehtia suuntalinjakokemaksi, jonka lanseerasi vuonna 2001 joukko konsultteja ja ohjelmistonkehittäjiä Agile Manifestona tunnettuna Internet-sivuna. Tämä manifesto tarjoaa 12 periaatetta, joiden tarkoitus on edistää ohjelmistokehitystä. Nämä periaatteet voidaan tiivistää neljään perusarvoon: (1) arvostamme yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja, (2) toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota, (3) asiakasyhteistyöä enemmän kuin sopimusnepuolellista sekä (4) muutoksiin reagoimista enemmän kuin suunnitelmassa pysymistä.

Manifeston jälkeen on kehitetty useita erilaisia menetelmiä, jotka pyrkivät toteuttamaan sen periaatteita, tarjoamalla konkreettisia työkaluja "ketteryden" saavuttamiseksi. Yksi näistä menetelmistään on scrum, ja se onkin saavuttanut tukevan jalansijan viime vuosien aikana.

Tulokset

Ohjelmistonkehitys on itsessään tieteenalana vielä varsin nuori, ja ketterät menetelmät vielä huomattavasti nuorempi. Tästä syystä ketteriin menetelmiin kohdistuvat tutkimukset, etenkin tapaustutkimukset, ovat edelleenkin harvassa, joskin tilanne on koko ajan kääntymässä paremaksi. Esimerkiksi Scrumiin liittyviä tapaustutkimuksia alkaa pikku hiljaa olla tarpeeksi tieteellistä kompetenssia varten.

Yksi tällainen on Kiinassa, Wuhan Yliopistossa tehty tapaustutkimus. Kyseinen tutkimus kohdistui paikallisten maan käytöstä vastaavien viranomaisten sisäiseen käyttöön toteutetuun ohjelmistoprosessiin, jossa käytettiin kehitysmenetelmänä scrumia. Tämän johdosta projektiin keskeisiä vaatimuksia olivat ohjelmiston kyky käsitellä käytettävissä olevaa informaatiota kattavasti ja tarkasti, ollen samalla helposti operoittava ja turvallisuuksääkökökulman huomioonottava.

Ohjelmistonkehityksestä vastasi kahdeksan henkinen työryhmä, jonka eri rooleihin

Kommunikaatio tapahtui päivittäisissä kymmenen minuutin tilannepalavereissa, sekä joka perjantai järjestettyssä tapaamisessa, jossa seurattiin mm. kehittäjien mielipiteitä ja tyytyväisyyttä, sekä käytiin puolituntia teknisten asioiden läpikäyntiin. Lisäksi ryhmän jäsenet kävivät keskenään keskusteluja sekä sähköisten viestien viestintä välityksellä, että kasvotusten.

Tutkijaryhmä keräsi tulokset 3-5 iteraatiokierroksen välein, jolloin edellisten kierrosten statistiikka taltioitiin. Lisäksi ongelmat (tekniset ja muut) analysoitiin, jotta voitiin päätellä onnistumistekijät.

**Johtopäätökset**

Tutkimuksen johtopäätökset ovat linjassa monen muun scrumista tehdyn tutkimuksen kanssa: scrum on toimiva ketterä menetelmä, joka vapauttaa kehittäjät raskaasta dokumentaation kirjoittamisesta, samalla tehostaen heidän toimintaansa tekemällä tiimeistä itseorganisoituvia. Kun kehittäjät saavat itse olla enemmän vastuussa omasta työstään, he kokevat tekemänsä asiat tarkoituksenmukaisemmiksi, jolloin motivaatio ja tulokset paranee. Lisäksi projektinhallinnan prosessi on selvästi nähtävissä ja kontrolloituessa, jolloin ongelmakohtiin tarttuminen on nopeampaa ja tehokkaampaa.

Risto Salo
Rules and tools for software evolution planning and management


Johdanto


Tulokset

Tässä tutkimuksessa tutkittiin kaudeksaa tietojärjestelmän evolutionlakia. Vuosien aikana nämä laki on huomattu antavan tärkeää tietoa tietojärjestelmän prosessin ymmärtämiseen. On huomattu, että kuusi (1,2,3,5,6,8) laki ovat yhdenmukaisia prosesseihin, jotka heijastavat dataa. Lakeja on paljon kritisoitu heti alusta alkaen. Tutkimuksessa tutkittiin tietojärjestelmiä tapauskohtaisesti ja tuloksista pääteltiin parhaat tavat miten evolutionlakien kanssa tuleet toimia.

Ensimmäinen laki on järjestelmän jatkuva muutos. Jotta E-tyyppineen tietojärjestelmä pysyy ulkopuolisen maaillman mukaan, tulee sen pystyä muuttumaan. Ajan myötä käyttäjäkokemus lisääntyy, käyttäjien tarpeet ja odotukset muuttuvat, uusia tarpeita, käyttötapoja ja rajoituksia ilmaantuu järjestelmän käyttöympäristöön jne. Tämän takia loputun ylläpitäminen alkaa. Seuraavaksi on listattu joitain seuraavia jotka pitää ottaa huomioon jatkuvasti muuttuvalle E-tyyppin systeemeille: (a) laaja dokumentaatio, jota päivitetään kun muutoksia tapahtuu; (b) järjestelmän ikää ja muutostasoa määrittelevä
malli, joka tarjoaa turvallisen rajan muutoksien määrään per julkaisu; (c) On tärkeää määritellä lisäyksien ja muutoksien määrä vaatimusmäärittelyissä per julkaisu, jotta voidaan esimerkiksi löytää järjestelmän kohdat jotka ovat uudelleen jäsentelyn tarpeessa.


Kolmas laki on tietojärjestelmän itseohjautuvuus. Globaalit E-tyyppin systeemien prosessit ovat itseohjautuvia. Jotta pystytään tunnistamaan palaute mekaanisi Sekä käyttämään sitä hyväksi tulevaisuuden suunnittelussa, johtamisessa ja prosessin parantamisessa kannattaa seuraavat kohdat ottaa huomioon: (a) Käytä mittauskaavioita ja mallintamisessa tekniikoita, jotka määrättelevät  yleiset mallit, suuntaukset, kasvuvauhdin sekä muutoksien toteuttamismääränsä organisation sisällä; (b) Perusta tyypilliset arvot prosessin kasvulle, vioille, muutoksille ja yksiköiden muutoksille, lisäämiselle sekä poistamiselle jne.

Kolmas laki on järjestelmän itseohjautuvuus. Globaalit E-tyyppin systeemien prosessit ovat itseohjautuvia. Jotta pystytään tunnistamaan palaute mekaanisesta ja hallinta, joilla on rooli itsevakauttamisessa sekä käyttämään sitä hyväksi tulevaisuuden suunnittelussa, johtamisessa ja prosessin parantamisessa kannattaa seuraavat kohdat ottaa huomioon: (a) Käytä mittauksessa ja mallintamisessa tekniikoita, jotka määrättelevät yleiset mallit, suuntaukset, kasvuvauhdin sekä muutoksien toteuttamismäärän organisation sisällä; (b) Perusta tyypilliset arvot prosessin kasvulle, vioille, muutoksille ja yksiköiden muutoksille, lisäämiselle sekä poistamiselle jne.

Neljäs laki on onganduran säilyttäminen. Keskimääränyys aktiiviteettitaso (elementtien muuttuminen, käsitteleminen ja käsitellyt per julkaisu) E-tyyppin prosesseilla pysyy tyyppillisesti vakiona järjestelmän eliniän aikana. Neljäs laki johtaa ainakin seuraavaksi suosituksen: (a) Tulee käyttää aktivisuustaso metriä, kuten elementtejä käsiteltävät per julkaisu, jotta voidaan käyttää tunnusomaisia prosessin suoritus yhdessä muutoksen vaikutus analyysityökalujen kanssa ja hankkia perusta työn suunnittelumalleille.

Yleensä keskimääräinen yleistäin kasvu E-tyyppin järjestelmillä on tapana laskea. Viides tietojärjestelmän yleistungilaki on samankaltaisuuden säilyttäminen. Pitkän ajan laskeminen kasvussa on yleistä julkaisu-perustaisissa järjestelmissä. Potentiaalista kasvu hidasta järjestelmän suunnittelussa muutoksien vahvistumisen takia, jossa tehdään vain tärkeitä korjauksia tai suljetaan järjestelmä. Hidastuvalle kasvulle on annettu suurimmaksi syyksi järjestelmän lisäantuviä monimutkaisuus. Seuraavat olleet kannattaa ottaa huomioon päättäessä julkaisu: (a) Kerää ja mallinna järjestelmän kasvu ja muutosdataa ajan tai julkaisun sektoreihin merkittyä, jotta voidaan päätellä tietojärjestelmän suunnittelumalleille. (b) Suunnittelee tasaisin vähäisiä järjestelmän sisävoimien, uudelleen jäsentely ja uudelleen ohjelmointi ja muut vastaavat toimenpiteet, jotka hillitsevät järjestelmän kasvun hidastumistusta; (c) Kehitä automaattisia työkaluja
Keräämään, mallintamaan ja tulkitsemaan tietoa, jota saadaan kun tutkitaan järjestelmän toimintaa.

Kuudes laki on jatkuva kasvaminen. E-tyyppin järjestelmissä funktionaalista kykyä täytyy olla jatkuvasti kasvattamaan, jotta voidaan ylläpitää käyttäjien tyytyväisyys koko järjestelmän elinkaaren aikana. Järjestelmän tulee kehittyä tyydyttävästi, jotta se voi tukea uusia tilanteita ja olosuhteita. Mitä siistimpi arkkitehtuuri ja rakenne järjestelmällä on, sitä todennäköisempää on, että lisäyksiä voidaan lisätä siististi palomuurnin avulla, joka hyväksyy vain oikeanlaisen tiedon siirtymisen vanhojen ja uusien osien välille. Tämä ei kuitenkaan aina ole mahdollista, varsinkin silloin kun järjestelmää ei ole suunniteltu uusien komponenttien lisäämiseen. Tavallisesti on järkevää, että vaihtelee järjestelmän julkaisuissa päivitykset niin, että julkaisee järjestelmän korjaukset ja uusien ominaisuuksien lisääminen erillään.

Seitsemäs laki on heikkenevä laatu. E-tyyppisten järjestelmien laatu heikkenee niiden kehittyessä, ellei sitä yritetä perusteellisesti muokata ottamaan kehitystä huomioon. Seitsemäs laki seuraa suoraan ensimmäistä ja kuudetta lakiä. Järjestelmän evoluution myötä suorituskyky huonenee ja potentiaaliset virheet lisääntyvät. Jotta laatu ei kärsisi náattaa seurata näitä ohjeita: (a) Suunnittele muutokset ja lisäykset järjestelmään niin, että käytetään samoja periaatteita koko järjestelmässä; (b) Käytä ressurseja järjestelmän monimutkaisuuden vähentämiseen.

E-Tyyppin evoluutioprosessit ovat monikerroksisia, monisilmukkaisia, monitoimisia palautejärjestelmiä. Kahdeksas laki on palaute systeemi. Perusidea tässä lassa on se, ettei päälliköt saa olla vapaita valitsemaan sopivaa toimintaa ilman tarkkaa liiketoiminnan tai jonkin muun vastaavan näkökulman mielipidettä. Havainnot ovat paljastaneet, että kannattaa noudattaa seuraavia ohjeita: (a) Tunnista sellaiset informaatiokanavat, jotka eivät ole osa virallista johtoa, mutta joista voi olla hyötyä järjestelmän kehityksessä; (b) Keskitä järjestelmän kehitys niihin toimintoihin, jotka ovat kaikista hyödyllisimpiä verraten alkuperäisiin tavoitteisiin.

Yhteenveto

Näitä suosituksia ei ole tarkoitus orjallisesti noudattaa vaan soveltaa tietojärjestelmään sopivaksi. Suosituksia voi olla hankalaa tai kalliista lisätä tietojärjestelmään, mutta potentiaalinen hyöty koskien tuottavuutta ja ennustettavuutta, prosessin tehokkuutta ja järjestelmän ylläpidettävyyttä, tekee sen kannattavaksi. ”Suositusten valitseminen ja soveltaminen muuttaa johtamisen päättämisen asiaksi, jotta niihin liittyvät suhteellinen tehokkuus, kustannus–hyöty-arviointi ja liiketoiminnalliset mahdollisuudet toteutuisivat” (s.39).

Hanne Korhonen