Software Development Governance Challenges of a Middle-Sized Company in Agile Transition

Ilkka Lehto
Helsinki University of Technology
Software Business and Engineering Institute
P.O. Box 9210 FI-02150 TKK
firstname.lastname@tkk.fi
+358 41 501 6167

Kristian Rautiainen
Helsinki University of Technology
Software Business and Engineering Institute
P.O. Box 9210 FI-02150 TKK
firstname.lastname@tkk.fi

Abstract

We studied how a middle-sized Finnish company employing agile methods governs its software product development. Through observations and interviews we followed the trace from strategic plans in the form of roadmaps to various backlogs and all the way to daily work. The governance roles, responsibilities and deliverables seemed to be in place on different organizational levels. However, closer inspection revealed challenges in the practical implementation. There were too many roles and hierarchy levels with information consistency problems in between. Prioritization of the high-level goals was unclear and made it difficult to plan and organize development work based on business value. The trace from high-level goals to more detailed plans was easily corrupted due to poor planning practices. Progress monitoring of daily work was poorly done and not linked to high-level plans. Consequently, the required feedback loops were inadequate, making it impossible for management to take corrective actions in time.

1. Introduction

For a software company it is crucial that the outputs of its processes meet the company's strategic goals. Ensuring that this happens is the very essence of software development governance (SDG). SDG reaches its goals by defining organizational structures through establishing chains of roles, responsibilities and communication [1, 2]. An important part of communication is feedback, which people need for successfully carrying out their responsibilities.

Agile methods, e.g. Scrum [3] and XP [4], provide in their own way a predefined SDG model, but originally assume a simple situation where a single team is developing one product at a time. However, when a larger organization wants to be agile and use agile processes, little guidance is provided for modifying the simple, basic "agile" SDG model to suit a more complex situation while still preserving the agile principles. Some literature has emerged in recent years to address agile in the large enterprise context, e.g. [5-8], but empirical research is still scarce on this subject.

In this paper we present our initial findings on how software development is governed in a middle-sized software company that is going through a transition phase to adopt agile methods. We also present challenges we have identified so far during our case study.

Section 2 presents the used research methodology. A framework for depicting product planning and development in an agile context that we used to analyze the case company is presented in Section 3. The initial findings of our work in progress are presented in Section 4. Section 5 concludes the paper with discussion and suggestions for future work.

2. Methodology

Our ongoing case study [9] of a middle-sized Finnish software company started with a current state analysis in February 2008. The case company employs roughly 700 people. Our case study is conducted in a research and development organization, which has roughly 300 employees of which about half are software engineers. The company offers products and services in more than 20 language versions to consumer and enterprise markets. The products are based on common components and platforms. Typically there are more than 20 concurrent projects performed by teams on 4 sites in 3 countries.

The company's research and development organization started a transition to agile software development in 2003. This was done as process push by a manage-

rial decision. The change was motivated by a need to release products in a timely manner, increase the ability of reacting to change and to get customer feedback earlier and more often.

The research method used can be characterized as participative action research [10]. Our cooperation with the company is coordinated by setting short-term goals in monthly meetings, where findings are also discussed.

Our observations and interviews focused on the research and development organization's largest developer pool consisting of 5 collocated teams averaging 9 people per team. The observations and interviews were digitally recorded and a case diary was kept.

3. Agile product planning and development framework

We constructed a framework (Figure 1) to depict agile product planning and development. The framework is based on the Cycles of Control framework [11, 12], which is an extension of Scrum, which in turn is the basis for the product development process in the case company. We have used the framework as a support tool in our analysis of the case company by studying how roles, responsibilities, communication, and metrics relate to the objects in the framework.

The framework is divided into four levels. The three topmost levels – *product*, *release*, and *iteration* – represent the different stages of product planning and monitoring. The lowest level, *heartbeat*, represents the development work done on a daily basis.

Everything starts with a *product* a company is developing. High-level plans for the product can be expressed, e.g., as business or technology goals, which should be prioritized and recorded as *backlog items* in a *product backlog*. The high-level and long-term plans for the product are summarized and communicated in a *roadmap*.

When planning goes into more detail, the high-level backlog items can be split into more detailed backlog items, forming a hierarchical structure. The hierarchy of backlog items helps in preserving the "big picture" of the product plans and forms the basis for progress monitoring.

The target of software product development is making and selling a *release* of a product to customers. One or several backlog items from the item hierarchy should be chosen as *release goals* to guide the more detailed release and iteration planning.

A release is developed incrementally in multiple *iterations*. As a continuous planning practice release goals or other backlog items are split until they are small enough to be implemented in an iteration. The backlog items that are selected for implementation in a specific iteration are the *iteration goals* in the *iteration backlog*.

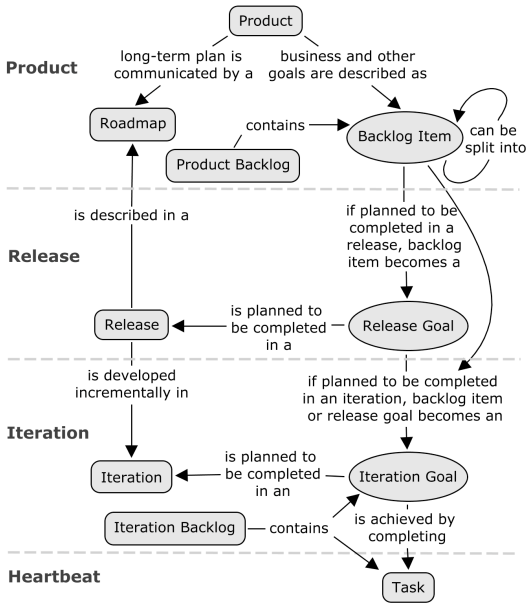

Figure 1. Agile product planning and development framework

In iteration planning, the *tasks* needed for achieving the iteration goals are planned and added to the iteration backlog. When tasks get done during the iteration, progress can be monitored and traced back through the backlog item hierarchy all the way to the high-level plans.

4. Findings

4.1. Software development governance

A *solution manager* is responsible for yearly updating a product's roadmap for a three-year span. The roadmap presents the schedule for releases and high-level business goals, which are called business themes. The roadmap planning sets the boundaries for planning the release projects. The *executive team* is a managerial board that inspects and approves the roadmaps.

The solution manager, *resource owner* and *project manager*, together with other stakeholders, prepare a release project proposal. The resource owner is the "line manager" of a developer pool. Her responsibility is to plan projects' resource allocation and balance teams' workloads. The *product council* is a managerial board responsible for inspecting and approving all release project proposals and ensuring that the project

portfolio is in line with the boundaries set by the roadmaps.

The product council appoints a *project steering group* for each approved release project. The project steering group is responsible for ensuring that the release project reaches its goals – expressed by the business theme(s) – with the allocated resources. The project steering group makes scoping decisions within these boundaries based on progress monitoring in monthly iteration demonstrations. If the release project is compromised, the issue is escalated to the product council.

As the starting point of more detailed release and iteration planning, the solution manager and project manager explicate the business themes as more detailed solution backlog items. Typically these are one-liners in a spreadsheet called solution backlog.

Releases are typically composed of multiple software components. Each component has an appointed technical expert called *product owner*. Each product owner maintains a product backlog, which contains more detailed and technically oriented product backlog items for the component.

Prior to iteration planning, the product owners copy the top priority product backlog items to a platform backlog, which is owned by the resource owner. Items are prioritized and modified to remove duplicates by the resource owner and the product owners.

In iteration planning the top priority product backlog items are assigned to the development teams and put in to team backlogs. In a later phase of iteration planning the teams plan the necessary tasks needed to accomplish the backlog items.

4.2. Challenges in communication

We observed two major challenges in communication: lack of communication between product owners and teams; and lack of feedback loops.

During our observations we noticed that in the largest developer pool there were times when all five teams were working on the same component. In this situation all teams needed time from the same product owner to clarify the team backlog items. This was difficult to schedule due to all the other product owner's responsibilities. Also, the opposite situation could be observed. Sometimes multiple product owners were requesting work from a single team, which disrupted the team's work by too many meetings.

The teams we observed had abandoned the task-planning phase of iteration planning. Instead, they used product backlog items as such. We learned in an interview that in the early phases of the company's agile transition the agreed practice was to try to make the product backlog items as equal in estimated effort size as possible. These were then used as such and the velocity of a team was calculated based on the number of completed backlog items. This was used as a progress metric. However, this "size equalization" is no longer done. Likewise, proper effort estimation and continuous re-estimation is not done. Therefore, development progress on the heartbeat level (Figure 1) could not be monitored. Consequently, progress monitoring on the other levels was not based on true progress information, making it impossible for management to take corrective actions in time.

Different people on many organizational levels used separate spreadsheets for managing backlogs as a part of release planning. While links existed between backlog items in different spreadsheets, the links were not dynamic, which resulted in information inconsistency. Changes in one spreadsheet were not automatically reflected in other spreadsheets. This would have made tracing development progress back to business themes challenging even if the teams' iteration planning practices had been working properly.

4.3. Challenges in roles and responsibilities

We observed three major challenges related to roles and responsibilities: team structure that conflicted with agile principles; applying product owner role in a large and complex context; and lack of business theme priorities.

Each team was composed of specialists of a certain aspect for the company's products, e.g. user interface team. Integration and coordination of the work of all teams was needed to build the product. This caused handovers and communication overhead. Therefore additional coordination effort was needed from the project manager and the resource owner. Also, the simple backlog practices of Scrum did not work and caused workarounds, such as the hierarchy of backlogs described in Section 4.1. Another challenge related to this was that work could not proceed in priority order because the teams could only pick backlog items they had the skills to perform.

The role of product owner as described in Scrum was considered to be too much for one person in the company. They had decided to split the responsibilities in three parts: the solution manager has commercial responsibility; the product owner has technical responsibility; and the resource owner has resource responsibility. Although the plan looks good in principle, in practice it resulted in a coordination and communication chaos manifesting itself as a seemingly endless amount of meetings, which frustrated everybody. Also, the teams were at a loss who to turn to when they needed specific information regarding a backlog item.

The solution managers are responsible for clarifying the roadmap's business themes in the release project proposal. However, we observed that no priorities between the themes were set. This contributed to difficulties in planning and organizing development work. As there were many approved business themes and no apparent prioritization between them, one result of this seemed to be that development work on most of the themes was done in parallel.

5. Discussion and future work

When we studied the case company's software development governance, we discovered many challenges. Two of them related directly to the functionality of the SDG model: lack of feedback loops and lack of business theme prioritization. The rest were related to the company's transition to agile methods.

The teams had abandoned the task-planning phase of iteration planning. This made it impossible to monitor progress on a detailed level. Also, the depth of backlog hierarchy and lack of dynamic linking between the backlog items made it impossible to link development progress to high-level plans. This, combined with the excessive parallel work caused by the lack of business theme prioritization, increases the risk of release project failure.

The prolonged agile transition seems to be the result of two decisions that contradict agile principles. The first decision was not to break the existing team structure. Some of the challenges reported in Section 4 are the result of workarounds to support and coordinate the work of the teams. According to agile literature cross-functional, self-organized feature teams are essential in agile software development, because they enable the lean principle of one-piece-flow [13], provide flexibility in assigning work, and reduce the need for coordination. For an unknown reason, feature teams were not introduced until October 2008.

The second decision was to appoint product owners per component. Since the teams we observed were neither feature teams nor component teams, the teams could not have a dedicated product owner. This caused challenges in the communication and coordination between teams and product owners. This probably resulted in splitting the product owner responsibilities to three roles, which it seems to us made the communication and coordination challenges even worse. In general, the product owner role has received very little attention in agile literature. We strongly urge that more research on this demanding role should be conducted.

Next we will start working on the two challenges that hamper the functionality of the company's SDG. Feedback loops need to be established. This includes proper task planning in the iteration planning phase and making the links from high-level plans to tasks dynamic. Excessive parallel development should be reduced to make the work more focused and effective. To provide a basis for this, business theme prioritization needs to be made.

In February 2009 the case company initiated an organizational change that will affect the product owner role. We will assist the company in the practical implementation by providing our insights through monitoring the resulting effects and giving feedback for continuous improvement. In the same way we will closely observe how the new feature team concept works. All in all, we hope to gain insight on scaling agile practices in the large enterprise context.

6. References

[1] S. Chulani, C. Williams and A. Yaeli, "Software development governance and its concerns," in *SDG '08: Proceedings of the 1st International Workshop on Software Development Governance,* 2008, pp. 3-6.

[2] S. W. Ambler and P. Kroll. (2007, Nov). Lean development governance. IBM Corporation, USA.

[3] K. Schwaber and M. Beedle, *Agile Software Development with Scrum.* Upper Saddle River: Prentice Hall, 2002, pp. 158.

[4] K. Beck, *EXtreme Programming eXplained.* Addison-Wesley, 2000, pp. 224.

[5] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises.* USA: Addison-Wesley, 2007, pp. 384.

[6] G. Larman and B. Vodde, *Scaling Lean & Agile Development: Thinkingand Organizational Tools for Large-Scale Scrum.* USA: Addison-Wesley, 2009, pp. 348.

[7] K. Schwaber, *The Enterprise and Scrum.* USA: Microsoft Press, 2007, pp. 176.

[8] S. Ambler, "Agile Software Development at Scale," *Lecture Notes in Computer Science,* 2008.

[9] R. K. Yin, *Case Study Research: Design and Methods.* ,Second ed., vol. 5, London: SAGE Publications, 1994, pp. 192.

[10] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," *ASQ,* vol. 23, pp. 582-603, December. 1978.

[11] K. Rautiainen, C. Lassenius and R. Sulonen, "4CC: A Framework for Managing Software Product Development," *Eng. Manage. J.,* vol. 14, pp. 27-32, 2002.

[12] K. Rautiainen, L. Vuornos and C. Lassenius, "An experience in integrating strategic product planning and agile software development practices," in *Proceedings of 2003 International Symposium on Empirical Software Engineering,* 2003, pp. 28-37.

[13] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit.* Boston: Addison-Wesley, 2003, pp. 240.