# Controlling and Monitoring Agile Software Development in Three Dutch Product Software Companies

Tjan-Hien Cheng[1], Slinger Jansen[1], Marc Remmers[2]

Utrecht University[1]
Information and Computer Sciences
Padualaan 14
3584 CH Utrecht, the Netherlands
Tcheng@cs.uu.nl, S.jansen@cs.uu.nl

Business Base[2]
Kobaltweg 48
3542 CE Utrecht, the Netherlands
Marc.remmers@businessbase.com

## Abstract

*Agile software development governance depends on successful monitoring and control mechanisms. Software development managers generally are unaware of the complete set of monitoring and control mechanisms that are available to them, leading to uninformed decisions with unsuccessful outcomes. This paper presents a list of key process indicators and a list of interventions that software development managers require for successful governance of agile development processes. The list is based on three case studies of product software companies that have successfully developed and sold software for several decades.*

## 1. Agile Development Governance

Software development managers within a product software company (PSC) struggle with controlling and managing the development process such that a product is finished and delivered within budget and on time. Traditionally, software developers would release their software infrequently, work in large increments, according to well-defined structures and documentation. Technological, market or customer requirement changes are detrimental to the success of a project when developing software this traditional way. Adopting agile software development methods enables a software developer to be more flexible and responsive to changing environments and customer demands. Numerous agile software development methods used in practice are empirically reviewed and the agility of the methods has been measured [1, 4, 5].

Agile methods proved to have a far higher agility and flexibility than traditional software development [1]. Agile processes of various agile methods are described in both scientific and practical literature [5, 6]. Many of these methods are described dogmatically, i.e., the described method is considered the best and only method. In earlier research [1] was concluded that software vendors generally do not accept such a dogmatic approach but use best of breed approaches to their development processes. This research focuses on the aspects that unite agile software development practices, through the concept of software development governance. This paper in particular describes what needs to be measured and what actions can be taken to steer the development process successfully. Successful steering is reached by monitoring and controlling agile governance in PSCs by the development of Key Performance Indicators (KPI) & Interventions.

The following section continues with a discussion of the position of KPIs and interventions in development governance frameworks. Section 3 describes the approach used in the case studies that explore the KPIs and the interventions. Section 4 provides a list and a description of the encountered KPIs and section 5 does the same for the interventions. Finally, section 6 concludes with lessons learned from this research.

## 2. Monitoring and Controlling Governance

Governance models generally focus on IT. IT Governance focuses on aligning the IT function with the business, based on three perspectives: structures, processes and relational mechanisms [6]. The aim of IT governance is to sustain and extend the strategy and objectives of an organization regarding to the IT function [8, 9].

Different types of governance based on IT Governance recently emerged, being '*agile governance*' [2] and '*software development governance*' [7]. Qumer's agile governance model


978-1-4244-3736-8/09/$25.00 

focuses on maximizing business value by the business alignment and application of agile software development methods. Software Development Governance (SDG) helps software development organizations to achieve their strategic goals by establishing the structural component and measurement component of governance. The structural component is setting up the responsibilities and authorities of people. The measurement component is concerned with establishing measurement and control mechanisms which let people execute their roles and responsibilities. The definition of agile governance we follow is "*the accountability and responsibility of management, adopting agile software development methods, and establishing measurement and control mechanisms in an agile environment*".

Development managers of PSCs are responsible and accountable for managing, measuring and controlling the software development process. This research aims to provide software development managers with KPIs and Interventions that can be employed across different agile development methods. KPIs are numbers or values that indicate the performance of a process or activity. These KPIs show whether organizational goals are going to be reached with the current methods. Provided with KPIs management can take action whenever the KPIs go out of bounds or when goal satisfaction is in danger. These actions are termed interventions. Figure 1 shows the relation between the concepts described in the work of Chulani et al [7]. This research focuses on Chulani's concept '*Control and measurement mechanisms*' in agile environments. Furthermore, KPIs are part of the arrow '*Control and monitor*' and interventions are part of the arrow '*Process controls*'.

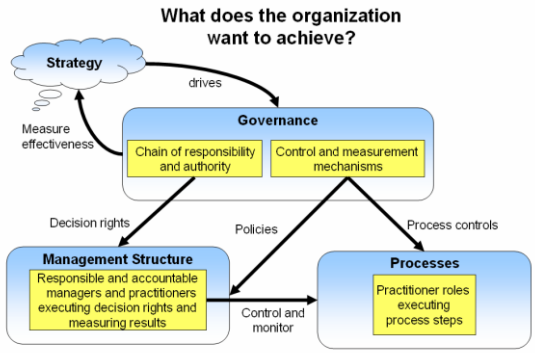


**Figure 1. Governance, Management and Processes**

## 3. Research Method and Case Studies

The field of agile software development regarding KPIs and Intervention is not yet explored. Hence it is almost impossible to create a comprehensive list that will be complete. Creation and design of KPIs and Interventions require constant adjustments in order to be practical and useful. The agile world assumes a changing and dynamic environment [1, 6]. KPIs and interventions should also be dynamic and constantly refined. The design cycle [3] enables us to solve the problem through an iterative logical process. Solving a single problem by the construction of concepts and choosing the best concepts that fit best. During the design process, evaluation and refinement of the concepts consequently increasing the suitability and usability of KPIs and interventions. First, a literature study is performed to gather information and explore the field of agile software development, governance, KPIs and interventions.

A case study at a PSC was performed to have a first impression of an adopted agile software development method in practice. The company has adopted SCRUM since 2004 and therefore has many years of experience with agile software development. Interviews and document reviews were used to gather information about their agile development process. The goal was to uncover KPIs and interventions used within this company to manage the process. Scientific literature also served as the knowledge base for deriving KPIs and interventions.

A list of KPIs and a list of Interventions were constructed. These lists are validated with three case studies. The case study subjects are three PSCs. Two of them adopt the agile development method Scrum and the third company follows the "best of breed" agile software development method. Managers, project leaders and Scrum masters (team leader of development team) were interviewed. Background information about the three case study companies can be found in table 1.

Reliability of the study refers to obtaining the same results, should the study be repeated. Internal validity prescribes that the research variable has a causal relationship with the dependent variable. To acquire a complete list of KPIs and Interventions, interviews are divided in two parts. The first part of the interview starts with discovering used KPIs and Interventions within the case study subject. The second part of the interview is the presentation of the standard list of KPIs and Interventions. The interviewee validates the lists and judges usefulness of the KPIs and interventions in practice. Dividing the interview this way, prevents the

influence of the lists on the interviewees. Interviewing the persons individually is another measure to get unbiased data.

External validity is the generalization of the results to the population, which in this research is achieved by validating the list at several companies through expert interviews and case studies. The case studies are performed at Dutch PSCs. These companies have adopted an agile software development method. They have several years of experience with developing agile and are thus suitable for this research. Used KPIs and interventions within these companies are therefore valid and can be generalized.

**Table 1. Case study information**

| | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| **Name** | **ERPPSC** | **GDPSC** | **FMPSC** |
| **Software Product** | ERP | Graphic design plug-in | Facility management |
| **Dev. Method** | Best of breed | Scrum | Scrum |
| **Years of experience** | 0,5 years | 1,5 years | 4 years |
| **Company size** | 240 | 70 | 325 |
| **# teams** | 9 | 7 | 5 |
| **# devs in team** | 2-10 | 2-7 | 4-8 |

After validation, the list of KPIs is adjusted to avoid ambiguous meaning. KPIs and interventions that were unclear or not marked as relevant by any of the interviewees were removed. The lists are also extended with new KPIs and interventions, suggested by the interviewees, to provide development managers a comprehensive list. We cannot guarantee completeness of the lists, since new items were added to the list up until the last case study. The list items are divided into four categories; Team, Person, Task and Quality. Examples of the case studies are used to clarify these aspects.

## 4. KPIs

Measuring several aspects of the software development process enables software development managers to control, manage and steer the development process. The aspects are divided in team, people, task and quality. The validated list of KPIs is shown in table 2. Team aspects focus on the capabilities and productivity of software development teams. Knowing the availability of teams enables development managers to do accurate planning. As time goes by, more experience is gained and better insight in the team's capability is acquired. In practice, a person is not spending their whole working time on planned tasks. Part of the working time is used for socializing, meetings, writing emails and so on. Thus, the real effective availability is a factor to take into account. By using this factor, more accurate estimations can be made and provides a better view of reality. The person aspect focuses around individuals. Getting insight of each person's work and capability enhances problem detection and makes task assignments easier. When someone is underperforming compared to their usual productivity, he or she might have problems. The development manager can immediately detect this by looking at the KPIs and can try to find out the problem.

**Table 2. List of KPIs**

| *Aspect* | *Key Performance Indicator* |
|---|---|
| **Team** | Team total available hours |
| | Team total effective available hours |
| | Team effort remaining |
| | Team effectiveness |
| | Team velocity |
| | Total available capacity |
| **Person** | Individual available hours |
| | Individual effective available hours |
| | Individual effectiveness |
| | Individual effort remaining |
| | Weekly working hours of individual |
| **Task** | Number of completed tasks |
| | Number of remaining tasks |
| | Remaining task effort |
| | Hours spent on task |
| | Working software delivery success rate |
| **Quality** | Total reported bugs |
| | Number of critical bugs |
| | Outstanding bugs |
| | Fixed/solved bugs |
| | Bugs coming from previous release |
| | Hours spent on bug |
| | Test success rate |
| | Test failure rate |

KPIs related to the task aspects show the progress of planned tasks and how much time is being spent. Continuously measuring progress provides software development managers with an overview of business goal achievability and in-development requirements planning. Within agile software development, software is developed and delivered in iterations. Measuring the rate of working software delivery indicates whether the planning was realizable, if estimations were right and if development teams were working as intended. Spending more time than estimated can be a waste of time if the task's business value is low.

Quality aspects focus on software quality. To improve software quality, bugs have to be tracked and fixed. Especially critical bugs send by customers, have to be fixed in time or the company might lose them. The customers are the main source of providing and encountering bugs. Each of the case study subjects has a test team to enhance software quality. Error and faults in the software can be found and fixed before they are released. One of the case study companies even has monitors of the tests and bugs in the hallway, among traditional other overviews, so that everyone can see them. The number of bugs per development team is shown and in this way motivates development teams to reduce the number of bugs.

## 5. Interventions

Measuring aspects of the development process only indicates if goals can be reached and alerts if goals are in danger. Whenever KPIs indicate the goals are in danger, the management should intervene. Interventions are related to KPIs in such a way that they are triggered by the KPIs. Furthermore, interventions can by organized into the same categories as KPIs. A validated list of interventions is presented in table 3.

Interventions in the team category focus on the control and improvement of team productivity, collaboration and capability. For example, when KPIs indicate that the productivity of team A is not high enough to finish tasks, another team with spare capacity can be assigned to help. Another intervention is to delete some tasks of team A or let them work overtime to finish the tasks on time. Meetings can improve productivity and capabilities of teams. Meetings are useful to discuss progress, encountered problems of team members and evaluation of past iteration/project.

The second and third case study subjects have daily Scrum meetings and an evaluation meeting at the end of iteration. During daily meetings, a team comes together to talk about their progress and encountered problems. Problems are solved together and mostly solved fast, because of daily encounters. As a result, time is saved and thus improving the productivity. At the end of iteration, an evaluation meeting is held to discuss the past iteration. Members discuss the good and bad things with the aim of improvements for next iterations, resulting into increase of team's capability. Organizations can arrange trips to enhance collaboration between team members and managers. The first case study subject organizes a yearly trip for the whole organization for teambuilding. Hence improves the relationships and communication between people.

**Table 3. List of interventions**

| *Aspect* | *Intervention* |
|---|---|
| **Team** | Assign extra team |
| | Delete task of team |
| | Add task to team |
| | Hold a meeting |
| | Hold evaluation meeting of past project |
| | Let team work overtime |
| | Organize day-out/trip |
| **Person** | Add task to individual |
| | Delete task of individual |
| | Remove team member |
| | Order individual to take days off |
| | Send individual on training course |
| | Assign person with more experience and expertise for support |
| | Arrange replacement |
| | Provide additional guidance throughout the process |
| | Weekly visit |
| **Task** | Move task to next sprint/iteration |
| | Stop tasks temporary |
| | Delete task |
| | Delete current task and start over |
| | Assign more tasks than estimated |
| | End the project |
| | Report to higher management for planning and decision making |
| **Quality** | Set criteria for working software |
| | Let customers test the software |
| | Make visible chart for whole organization |

Person aspects focus on an individual's welfare and capability. When KPIs indicate a developer has too many tasks, is working long hours and shows signs of stress or burnout, their welfare is in jeopardy. An employee of ERPPSC was working long hours and also showed signs of stress. The stress of this person influenced the team negatively and a manager ordered the employee to take some days off to reload.

If KPIs indicate there are still many tasks with many hours to complete at the end of iteration, this might indicate that employees are lacking knowledge or skill. They can be sent on training courses to improve their capabilities. Another person with more expertise and knowledge can also be assigned to them as a support. Managers can also provide guidance if needed. ERPPSC is focused around the people and is oriented around the development of their capabilities. A person with more experience and expertise is assigned to support and guide an employee who just started. An interviewee of the company mentioned he often visits his team and talk to them. In this way you get involved with their work, know what they are doing and help them if needed. This interviewee was really important to the team he was leading according to another interviewee. "The team will do everything for that man" others said.

The task category concerns tasks management. For example, unfinished tasks within a sprint or iteration are moved to the next one. Tasks that are too complex or impossible to realize are stopped or restarted. In FMPSC, tasks were temporarily stopped when critical bugs arose. These bugs were so critical for customers that they needed to be solved immediately or the company might lose the customer. Tasks were resumed when these bugs were solved.

Creating awareness of the quality of software is the goal of interventions in the quality category. By making a chart visible to the whole organization showing the success or failure rate of tests and number of reported bugs, everyone knows the quality of the software. One interviewee highlighted the fact that some teams do not want to have bad results, because everyone in the organization will see them. They will do their best to avoid problems in the future and solve problems fast. Another way to improve quality is by getting a small group of customers to test the software first, before releasing it to a wider public. Case study subject one has a room where customers can test the software. Setting criteria for working software will let the teams know what is needed, before it is presented and released to customers. One of the interviewees issued four criteria for working software. Teams must satisfy all four criteria before they are allowed to deliver and present the software.

## 6. Conclusions and Lessons Learned

This section presents the conclusions and lessons learned about KPIs and Interventions. In this paper we present a list of measurement and control aspects in agile governance and validate this with three case studies. Measurements should be set up around teams and people. They are responsible for developing the software and are the most important aspect to monitor. Knowing the capabilities and capacities of the teams leads to more accurate estimations and accurate planning.

A lesson learned is that KPIs should not only be visible for the management, but also available for the whole organization. For example, the total number of reported bugs and the number of fixed/solved bugs. The reporting of bugs tells the organization that the software is of a certain quality level. ERPPSC, for instance, has a briefing with all departments to inform them about the software development process in the upcoming period. Test plans are already created before the software is developed. This was required in order to preserve the quality of software.

Furthermore, during a sprint or iteration the number of completed tasks and tasks effort remaining indicates if planned tasks can be realized on time. All departments will know the progress of the current project and can properly give the right information to the customers. Promising certain functionalities that can not be finished on time will damage the relationship between the company and its customers. FMPSC avoids this problem by preparing the sales, marketing, support and training departments, before the release of software. These departments are trained and updated with the relevant information about the release, so that they will be able to serve and provide customers with the right information.

The second lesson is that having a fixed composition of software development manager and team, who are specialized and accountable in a certain area of software. In this way expertise and experience is build up. All case study subjects are organized in this way. FMPSC used to have a different set up. There were no fixed combination of teams and managers. Furthermore, they were not specialized in a specific area or module of the software. A team can be developing pieces of software for module A and B in one iteration and developing for module C in another. Before any iteration, a kind of auction system was used for assigning tasks/requirements. Teams could bid on

the requirements they wanted to develop. FMPSC emphasized this was not working, due to no specialization leading to lack of expertise and responsibility. For example, when there are bugs reported by customers that originate in relatively old code, it could be a problem if the organization does not know which team is responsible. When a development team does have a fixed set up of team and manager their relationship will strengthen, because of facing problems together and the tight collaboration. They will know each other's capabilities, knowledge and personalities. Assignment of tasks will be easier and making the planning will be more accurate.

The third lesson is holding short daily meetings. The two case study companies that follow the Scrum method have daily Scrum meetings. The progress, performed tasks and problems of the team members are discussed. During Scrum meetings the cause of the problem becomes clear and a look at KPIs is taken. By monitoring measurements every day, software development managers can intervene on time. One interviewee points out the fact that you face problems every day and will be confronted with the reality. You are forced to take action. For example when measurements indicate the effort remaining for tasks is unchanged for the past week, something might be wrong and management can take action to do something about it. Some tasks may be too complex and should be moved to the next sprint/iteration or even stopped and deleted. Another possibility is the lack of knowledge and skills of an individual. This person can be sent on a training course and in some extreme measures removed. This is the case with people who do not fit in. One of the case study subjects had a case where someone really was not fitting within the company. There was a personality conflict and was unable to collaborate with others. This was negatively affecting the motivation and spirit of the team. In the end this person was relocated. Consequently, daily meetings allow management follow daily progress and intervene when necessary.

## 7. Future Work

The model of Chulani et al. [7] (Figure 1) encompasses monitoring (through KPIs) and providing intervention mechanisms to the management and enable control of the software development process into SDG. This paper suggests a further detailed model, where KPI *boundaries* and *combinations* are established to trigger (combinations of) interventions. We coin the *intervention trigger model* to provide lists of possible interventions when KPIs outrun preset boundaries. The intervention trigger model requires the PSC's strategy and goals, KPIs and their values, and all possible and acceptable interventions as inputs. The intervention trigger model can be used at any point in the software development process to establish which interventions must be executed to correct any extreme KPI values or value combinations. We consider further development of the intervention trigger model as future work, since this paper does not claim to provide a complete set of interventions and KPIs, nor does it provide a method for establishing KPI boundaries and trigger values.

More work is required on the extension and validation of the list of KPIs and interventions to achieve comprehensiveness and completeness. For instance, soon we noticed that there exist relationships between negative and positive interventions (fire/hire a person). Furthermore, relationships exist between specific interventions and KPIs. A limitation of this research is the fact that two of the three case study subjects adopt the Scrum agile development method. Future PSCs adopting other agile methods such as XP, Feature-Driven Development, Dynamic Systems Development Method, and Crystal Methods are required to further enhance the validity of the lists. Further research is needed on general aspects of agile governance regarding to the relation of agile methods with measurement and control mechanisms and study of the relationship between KPIs and interventions.

## 10. References


[1] A. Qumer, and B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice", *Journal of Software Systems*, *81*(11), Elsevier Sciencedirect, November 2008, pp. 1899-1919.

[2] A. Qumer, "Defining an Integrated Agile Governance for Large Agile Software Development Environments", *Lecture notes in Computer Science, Agile Processes in Software Engineering and Extreme Programming*, Springer Berlin / Heidelberg, 2007, pp. 157-160.

[3] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa, "Modeling Design Processes", *American Association for Artificial Intelligence,* volume 11 issue 4, AI Magazine, 1990, pp. 37-48.

[4] K. Conboy, and B. Fitzgerald, "Toward a Conceptual Framework of Agile Methods: a study of agility in different disciplines", *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*, ACM, Newport Beach CA USA, 2004, pp. 37-44.

[5] P. Abrahamsson, J. Warsta, T. Siponen, and R. Jussi, "New directions on agile methods: A comparative analysis", *Proceedings of the 25th International Conference on Software Engineering (ICSE'03),* IEEE Computer Society, May 2003, pp. 244-254.

[6] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis", *VTT Publications 478*, Elsevier Sciencedirect, Finland, September 2002, pp. 1-107.

[7] S. Chulani, C. Williams, and A. Yaeli, "Software development governance and its concerns", *Proceedings of the 1st international workshop on Software development governance,* ACM, Leipzig Germany, May 2008, pp. 3-6.

[8] S.D. Haes, and W.V. Grembergen, "Analyzing the relationship between IT governance and business it alignment maturity", *Proceedings of the 41st Hawaii International Conference on System Sciences,* IEEE Computer Society, January 2008, pp. 428.

[9] W.V. Grembergen, S.D. Haes, and E. Guldentops, "Structures, Processes and Relational mechanisms for IT Governance", *Strategies for IT*, 2004, pp. 1-36.