FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Evaluation of Lean-Agile Multi-Project Management in a Medium-sized Development Environment

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Dipl.-Ing. Raoul Vallon
Matrikelnummer 0525496

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Thomas Grechenig

Wien, 31.01.2012

(Unterschrift Verfasser)	(Unterschrift Betreuer)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Evaluation of Lean-Agile Multi-Project Management in a Medium-sized Development Environment

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Dipl.-Ing. Raoul Vallon
Registration Number 0525496

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Thomas Grechenig

Vienna, 31.01.2012

(Signature of Author) (Signature of Advisor)

Evaluation of Lean-Agile Multi-Project Management in a Medium-sized Development Environment

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Dipl.-Ing. Raoul Vallon

0525496

ausgeführt am
Institut für Rechnergestützte Automation
Forschungsgruppe Industrial Software
der Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig

Wien, 31.01.2012

Declaration

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

I hereby declare that I have written this thesis independently, without the use of documents and aids other than those stated and that I have marked all literally or analogously taken statements as such.

Wien, am 31.1.2012

Raoul Vallon

Acknowledgement

I would like to thank Thomas Grechenig for supervising this thesis and his support in setting up the case study. Furthermore, I am deeply grateful to Michael Müller-Wernhart for his feedback and advice, making available his support in a number of ways. I would also like to show my gratitude to everyone I worked with during the case study, which has been a great enrichment to this thesis.

Last but not least, I would like to thank my family and my girlfriend Christiane for their support.

Abstract

Agile and lean software development practices have been developed in early 2000 and continuously gained popularity every since. Concepts for multi-project management in the field, on the other hand, have only started to evolve very recently. Currently established agile and lean literature does not really focus on this issue. Publications tend to deal with very specific approaches such as e.g. organization-wide agile and lean transformations. This observation is the motivation for a further investigation and evaluation of agile and lean values in a multi-project management environment.

The thesis begins with an analysis of known traditional process frameworks to draw conclusions for lean and agile practices. Then known lean-agile multi-project management approaches in literature and research are examined. Furthermore, a case study is conducted covering three projects in a distributed development environment. The lean-agile practice in use is then analyzed, evaluated and improved with regard to the theoretic concepts. In conclusion, the adoption of lean-agile values in traditionally grown and run organizations is discussed in light of findings from the case study and the literature review.

Results show that almost all regular traditional multi-project management goals also have a lean-agile counterpart. However, the ways in which they are achieved are completely different, e.g. regarding hierarchies and styles of management. The application in traditional organizations imposes several constraints, e.g. former project managers may not want to let go of established hierarchies and handicap the self-management of teams as fake Scrum Masters.

The review of the lean-agile approaches revealed that the shared highest goal is to provide transparency across the value stream, which is especially important in a multi-project environment. Moreover, simple communication is the preferred means to deal with impediments, in the Daily Scrum, Scrum of Scrums or an optional third-layer Meta Scrum. Apart from communication and interaction, transparency is achieved via incremental planning and prioritization in all the examined approaches.

Finally, a best practice lean-agile multi-management approach is suggested combining several ideas.

Keywords: *agile, lean, Scrum, Kanban, multi-project management, distributed development*

Kurzfassung

Konzepte für lean-agiles Multiprojektmanagement haben sich erst kürzlich begonnen zu entwickeln. Die aktuell vorhandene Literatur beschäftigt sich primär mit sehr speziellen Herangehensweisen, wie z.B. organisationsweite agile oder schlanke Transformationen. Diese Beobachtung stellt die Motivation für eine nähere Untersuchung und Evaluierung agiler und schlanker Methoden im Multiprojektmanagement dar.

Die Diplomarbeit beginnt mit einer Analyse bekannter traditioneller Prozessstandards, um Rückschlüsse für lean/agile Methoden ziehen zu können. Es folgt eine Untersuchung lean-agiler Praktiken aus der gegenwärtigen Literatur und Forschung. Zusätzlich wird im Zuge eines Fallbeispiels der lean-agile Multiprojektmanagement-Ansatz für drei Projekte in einer verteilten Entwicklungsumgebung untersucht, evaluiert und in Bezug zu den vorangehenden theoretischen Konzepten verbessert. Abschließend wird die Einführung lean-agiler Werte in traditionell gewachsenen und geführten Organisationen diskutiert. Als Grundlage dienen die in dieser Arbeit gewonnenen Erkenntnisse.

Die Ergebnisse zeigen, dass fast alle traditionellen Ziele im Multiprojektmanagement auch ein lean-agiles Gegenstück haben. Die Herangehensweisen zur Erreichung dieser Ziele sind jedoch häufig in krassem Gegensatz zueinander, z.B. in Bezug auf Hierarchien oder Managementstile. Die Anwendung lean-agiler Ansätze in traditionellen Organisationen wirft häufig Probleme auf: so wollen frühere Projektmanager oft ihren Status und festgefahrene Hierarchien nicht einfach so aufgeben und behindern daher die Selbstorganisation der Teams als Pseudo-Scrum Master.

Die Untersuchung verschiedenster lean/agiler Praktiken hat verdeutlicht, dass das gemeinsame oberste Ziel die Erreichung von Transparenz über die gesamte Wertschöpfungskette darstellt. Dieser Punkt gewinnt im Multiprojektmanagement noch mehr an Bedeutung. Überdies haben sich schlanke Kommunikationswege als bevorzugtes Mittel herausgestellt, um Hindernisse aus dem Weg zu räumen: im Daily Scrum, Scrum of Scrums und in der optionalen dritten Ebene, dem Meta Scrum. Abgesehen von Kommunikation und Interaktion wird Transparenz in allen analysierten Ansätzen mit Hilfe von inkrementeller Planung und Priorisierung erreicht.

Als Abschluss der Diplomarbeit wird ein „Best Practice"-Konzept für lean-agiles Multiprojektmanagement präsentiert, das die Ideen mehrere Ansätze kombiniert.

Keywords: *agil, lean, Scrum, Kanban, Multiprojektmanagement, Verteilte Entwicklung*

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement and Motivation

Multi-project management (MPM) describes the practice of managing related projects and the need of sharing resources between them, e.g. team members working on several projects. This often also involves prioritizing projects according to business goals.

Lean Management (also known as *Just-in-Time Production*) has been introduced by the Japanese automobile company Toyota in 1978 (Japanese: [Ohno78], English translation: [Ohno88]) and has found its way into software engineering in 2003 with [Ande03] and [PoPo04], where lean principles have first been applied to IT development.

Agile software development has gained increasing popularity since [Beck00] in year 2000. Both agile and lean practices can be seen as lightweight development methods that share common values. [KnSk10], for example, has pointed out many similarities in his comparison of lean follower Kanban and agile follower Scrum. [Vall11] examined a possible combination of Kanban and Scrum by developing a hybrid process.

However, multi-project management has only started to evolve in this field (cf. chapter 4) compared to traditional approaches (cf. chapter 3). Therefore more light will be shed on the issue during the course of this thesis, i.e. on existing approaches in traditional process frameworks and lean or agile ones.

The author's personal motivation is furthermore to examine a lean-agile multi-project management in the case study, which is conducted in an Austrian IT company[1] (cf. chapter 1.4) that has incorporated agile principles to daily software engineering. The approach will be analyzed and evaluated. Moreover, the experience gained will be used to elaborate a lean-agile best practice with respect to the existing concepts.

This thesis is not about large-scale or *Distributed Software Development*[2]. It solely focuses on lean and agile procedures to manage several small to medium-sized projects with shared resources simultaneously. A medium-sized project, as understood in this thesis, may consist of up to ten Scrum teams with ten being the exception rather than the rule. A Scrum team typically consists of five to nine members [Pich08].

[1] The company will remain anonymous in this thesis, as agreed upon by all parties

[2] Although the case study touches the subject, it is not one of the main concerns of this thesis

1.2 Objectives

The aim of this thesis is to improve and emphasize multi-project management also in medium-sized lean and agile environments, as it is part of everyday software engineering.

This is achieved by answering the following research questions:

1. *Which traditional Multi-Project Management approaches are the most accomplished?*

First of all, traditional approaches are examined since they are more experienced than agile and lean practices and thus conclusions can possibly be drawn.

2. *What has to be changed in traditional Multi-Project Management to adjust it to Agile or Lean project environments?*

It will be reviewed which parts of traditional approaches can be used in lean and agile environments, i.e. their consistency with the Agile Manifesto (cf. [BeBe01]) is analyzed.

3. *What has to be changed in known practices of Lean and Agile Multi-Project Management to achieve an improvement?*

The answer to the last and most important research questions is a conclusion from the two precedent questions and the case study.

The scientific accomplishment of this master thesis is an improvement of multi-project management in lean and agile software development. This is achieved by looking at the best practices for traditional approaches (research question 1) and examining how they need to be modified in order to work in lean and agile environments (research question 2). After this research phase a case study in an Austrian IT company will be conducted to get hands-on experience on multi-project management in a medium-sized environment and evaluate the answers gained so far. Afterwards a concluding answer to main research question number 3 is presented. The detailed procedure can be found in the following chapter.

1.3 Methodology

Research is divided into the following phases:

- **Phase I: Analysis of Theoretical Background**
  In this phase the theoretical background of this thesis is analyzed, i.e. multi-project management in general, agile software development with the follower Scrum and lean software development with the follower Kanban. Concluding to this phase, related work in the field is presented.

- **Phase II: Traditional MPM to Lean-Agile MPM**
  In this phase extensive literature review is conducted to identify best practices for traditional approaches. Furthermore, it is examined which conclusions of traditional multi-project management (MPM) hold true for lean and agile MPM. Additionally, already existing approaches for Kanban (lean) and Scrum (agile) are examined in this phase as well. This phase provides answers to research questions 1 and 2.

- **Phase III: Case Study**
  During the case study a lean-agile MPM approach is analyzed and evaluated in an Austrian IT company. Additionally, improvements are suggested in light of the outputs of phases I and II.

- **Phase IV: Discussion & Conclusion**
  Identified problems in theory and practice and improvements for lean-agile MPM are discussed in consideration of the three precedent phases. This phase provides the answer to the main research question (number 3). The phase ends with an overall conclusion to this thesis and an outlook to future research opportunities.

1.4 Partner for Case Study

The case study of phase III (cf. chapter 1.3) involves medium-scale Scrum of about 30 people working on three different products that share the same codebase in a multi-team and multi-site development environment. As a consequence, several organizational tools for multi-project management have been established that will be analyzed and evaluated. Problems and strengths of the approaches in use will be identified and improvements suggested. In conclusion, a critical view on the case study's findings with regard to known multi-project management approaches is presented.

The case study partner will remain anonymous in this thesis. All of the documents, experiences and findings gained during the case study will be generalized to protect the partner’s identity.

1.5 Thesis Outline

Chapter 2 provides definitions and a theoretical background to the research questions, i.e. it gives a short view on multi-project management and explains lean and agile software development and its most famous followers Kanban and Scrum. Concluding to this chapter, related publications are commented.

Traditional multi-project management (MPM) is the focus of chapter 3. Possible conclusions for lean and agile MPM are analyzed.

Chapter 4 presents known practices of lean and agile multi-project management in literature and research respectively.

A case study is conducted in chapters 5 and 6. The lean-agile multi-project management approach in use in an Austrian IT company will be analyzed in chapter 5 and evaluated in chapter 6 alongside improvements and lessons learned.

Chapter 7 discusses results and provides a concluding answer to the research questions.

In chapter 8 the conclusion to this master thesis as well as an outlook to further research possibilities in the field are presented.

2 Definitions

This chapter provides a short overview of multi-project management, as it is used during the course of this thesis. Furthermore, agile and lean software development is presented and its most successful followers Scrum and Kanban, respectively (based on [Vall11]). The chapter ends with comments on related work.

2.1 Multi-Project Management (MPM)

Multi-project management (or MPM in short) is understood as *Program Management* (or *Programme Management*[3]), as defined by the Project Management Institute (PMI) in the following citation:

"Program management focuses on the project interdependencies and helps to determine the optimal approach for managing them." (p.10, [Pmbok08])

[Pmbok08] also speaks of *"centralized coordinated (program) management"* (p.10), which may not be suitable for lean and agile project environments. This is also one of the concerns of this thesis (see chapter 3). Since program management in literature mostly refers to the definition given in [Pmbok08], this thesis uses the more general term *Multi-Project Management* to stress the difference to traditional program management. In denotations of PMI however, program management represents multi-project management the closest.

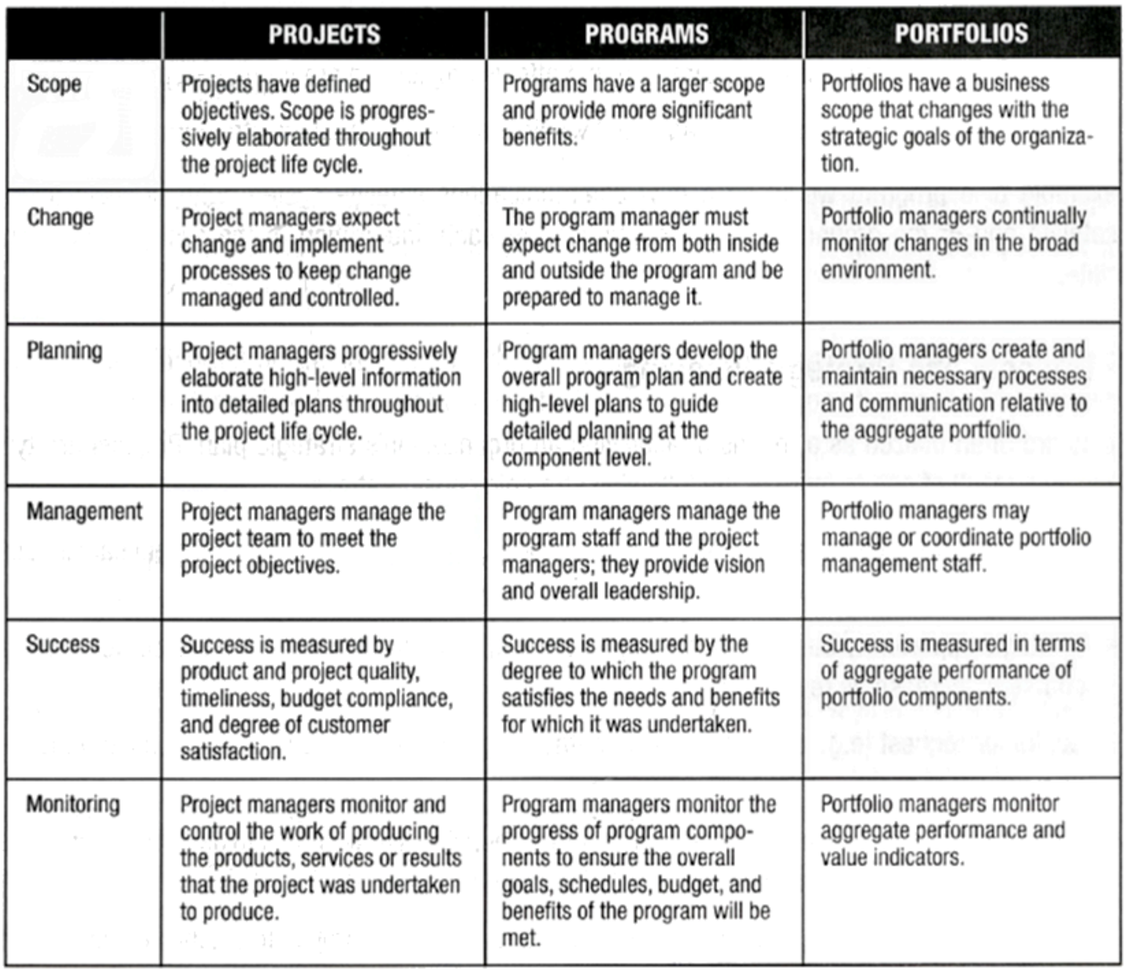
Robert Prieto defines the difference between project management and program management in [Prie08] as follows:

"Program management is not just the sum of all project management activities but also includes management of the risks, opportunities and activities that occur 'in the white space' between projects." [Prie08]

So multi-project management introduces new tasks to enable the management of the whole program rather than individual projects.

Another term not to be confused with multi-project management or program management is *Project Portfolio Management* (PPM). [Pmbok08] makes the distinction as shown in Table 1.

[3] British English



	PROJECTS	PROGRAMS	PORTFOLIOS
Scope	Projects have defined objectives. Scope is progressively elaborated throughout the project life cycle.	Programs have a larger scope and provide more significant benefits.	Portfolios have a business scope that changes with the strategic goals of the organization.
Change	Project managers expect change and implement processes to keep change managed and controlled.	The program manager must expect change from both inside and outside the program and be prepared to manage it.	Portfolio managers continually monitor changes in the broad environment.
Planning	Project managers progressively elaborate high-level information into detailed plans throughout the project life cycle.	Program managers develop the overall program plan and create high-level plans to guide detailed planning at the component level.	Portfolio managers create and maintain necessary processes and communication relative to the aggregate portfolio.
Management	Project managers manage the project team to meet the project objectives.	Program managers manage the program staff and the project managers; they provide vision and overall leadership.	Portfolio managers may manage or coordinate portfolio management staff.
Success	Success is measured by product and project quality, timeliness, budget compliance, and degree of customer satisfaction.	Success is measured by the degree to which the program satisfies the needs and benefits for which it was undertaken.	Success is measured in terms of aggregate performance of portfolio components.
Monitoring	Project managers monitor and control the work of producing the products, services or results that the project was undertaken to produce.	Program managers monitor the progress of program components to ensure the overall goals, schedules, budget, and benefits of the program will be met.	Portfolio managers monitor aggregate performance and value indicators.

Table 1: Comparative Overview of Project, Program and Portfolio Management [Pmbok08]

To stress the difference further, the following figure represents a possible arrangement of projects, programs and portfolios in a typical organizational environment (adapted from [Opm308]):

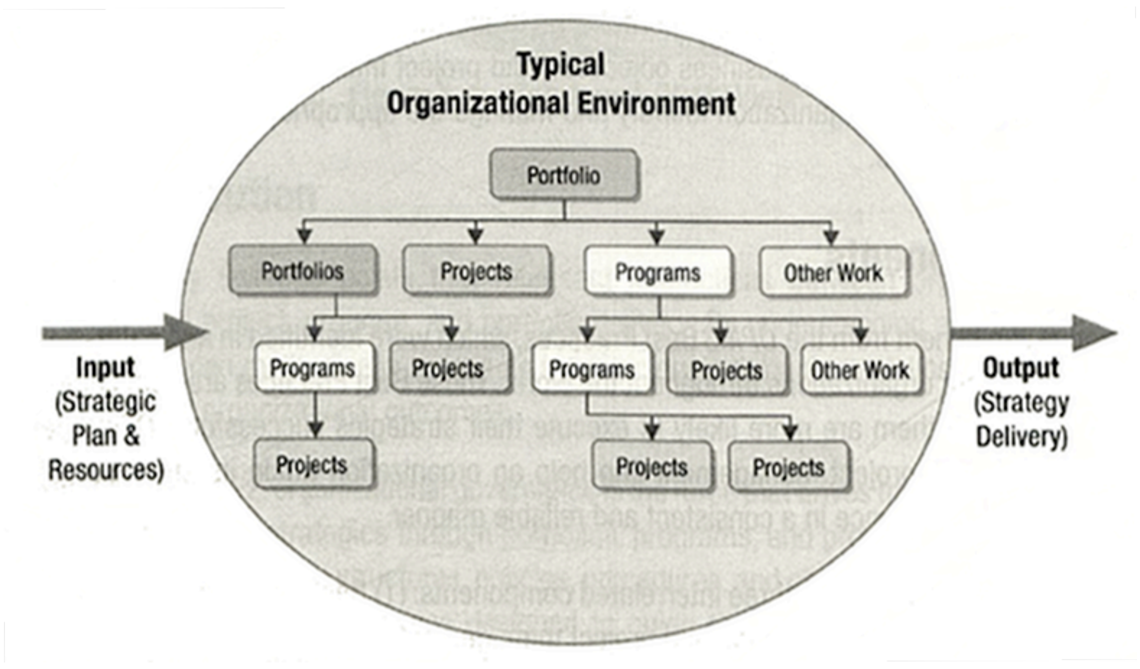


Figure 1: Typical Organizational Environment (cf. [Opm308])

Both Table 1 and Figure 1 point out that programs are the links between projects and portfolios from an organizational point of view.

Another term, which needs to be distinguished from multi-project management, is *IT Governance*. According to [Meye03] the IT governance summarizes principles, methods and actions to assure that business goals are met, resources are spent responsibly and that risks are being monitored appropriately. [TjKa05] argue that every organization should have IT governance mechanisms applied to some extent.

The relation between IT governance and portfolio management is as follows: *"Many organizations, including governments, are looking to enterprise IT governance to support portfolio management, closer business-IT alignment, [and] prioritization across projects and across agencies." (p.5, [PaBu09])*. However, IT governance as well as project portfolio management is beyond the scope of this thesis. Thus, also the two most widely used (cf. [Forr05]) IT governance frameworks ITIL and COBIT (cf. [Itil11] and [Cobit11] respectively) are not subject to review in chapter 3.1 (traditional MPM approaches) of this thesis.

The definitions, similarities and distinctions provided in this chapter describe how the term *Multi-Project Management* is understood and used in this thesis.

2.2 Agile Software Development

The following description of agile software development and its follower Scrum (see subchapters) is based on [Vall11].

Agile software development is regarded as an answer to the problem that even with exhaustive planning, the resulting software is seldom of high quality. One of the reasons is constantly changing requirements that are part of most of today's projects [DoKl05]. Hence agile processes try to use a more lightweight approach in planning to cope with changing requirements. Furthermore, to establish a common ground for all agile followers, the *Agile Manifesto* has been negotiated among representatives of various agile process flavors in 2001 [BeBe01].

The Agile Manifesto contains the following principles (cf. [BeBe01]):

„Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more."

The Agile Manifesto points out that individuals and interactions are one of the key issues of software engineering. This also implies that hierarchies are loosened compared to more traditional approaches (e.g. Rational Unified Process, cf. [Kruc00]). Moreover, planning needs to be done on a regular basis instead of following a strict plan (in e.g. still to be found *Waterfall Model*, cf. [Royc70]). Moreover, flexibility and customer collaboration is increased and documentation kept to a minimum [ScBi10], because it should not be used as a substitute for interaction [High04].

Among the most widely used agile methodologies are Scrum (to be explained in detail in the following chapter), XP (*Extreme Programming*, cf. [Beck99]) and *Crystal* (cf. [Cock05]). However, it is important to adopt agile principles rather than strictly following one of the agile methodologies or it will lead to *"constant struggles and many other old school problems*" (p.938, [FrRi06]). Agile methodologies can be used in all kinds of software projects; it has even been applied successfully to disaster management after a terrorist attack (cf. [NaZu09]).

Scrum

The description of Scrum in this chapter is mostly based on [Pich08] and [Vall11].

The name Scrum originates from *“the strategy used in rugby for getting an out-of-play ball back into play”* (p.1, [ScBe02]). It has been chosen because *“both [the game rugby and Scrum] are adaptive, quick, self-organizing, and have few rests.”* (p.1, [ScBe02]). The most important properties of Scrum can be derived from these similarities: high productivity, adaptivity, low risk and uncertainty and resulting increased comfort for practitioners [GrBe10].

Figure 2 provides a rough overview of Scrum that is explained in short below. A more comprehensive description of Scrum follows in the subchapters.

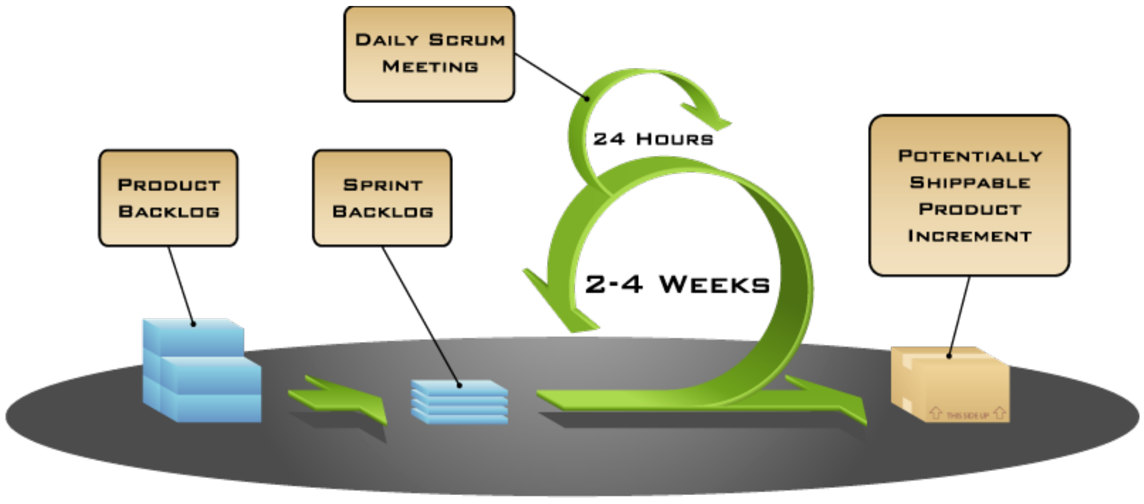


Figure 2: Scrum Overview (cf. [Mgs11])

Requirements in Scrum are defined as *user stories*. A user story is a requirement written from the user’s point of view (in a specific role), e.g. “As standard user I want to be able to create a new meeting.” [Pich08]. Phrasing user stories correctly is a very important issue as books have been written focusing solely on the matter (cf. [Cohn10] and [Wird09]).

The *Product Backlog* is the collection of all user stories of the product to be developed. Scrum works with *Sprints*, i.e. iterations of typically 30 days. At the beginning of each Sprint, user stories of the Product Backlog are pulled into the *Sprint Backlog* by the team (*Sprint Planning Meeting*), which commits to developing all the user stories within the Sprint. Furthermore, there is the *Daily Scrum Meeting* in which current progress and impediments are discussed in a very short manner.

At the end of each Sprint a product increment has to be created including all the user stories that have been developed during the Sprint. The team and stakeholders will then reflect on the current product increment (*Sprint Review Meeting*) as well as on the

Scrum process itself (*Sprint Retrospective*), which shall help improve the next Sprint iteration.

The following table shows how the Agile Manifesto (cf. [BeBe01]) is implemented in Scrum:

Principles of the Agile Manifesto	Implementation in Scrum
Individuals and Interactions	Team defines Sprint Backlog Team is self-organizing Daily Scrums
Working Software	Shippable Product Increment each Sprint
Customer Collaboration	Product Owner Sprint Planning Meeting Sprint Review Meeting
Responding to Change	Daily Scrums Sprint Planning Meeting Sprint Retrospective

Table 2: The Agile Manifesto and Scrum (cf. [Vall11])

The following subchapters provide more detail on the different roles, meetings and artifacts of Scrum.

Roles

Scrum defines three types of roles:

- Product Owner
- Scrum Master
- Team

However, none of the three is a project manager in a traditional sense. The question is whether or not managers are needed and, if so, what they can or should do in agile environments. These and other questions are discussed in [AnAl03]. Mary Poppendieck provides the following answer: *"Teams appreciate good leadership, because it helps them be successful. I distinguish management tasks - getting the maximum value from the dollar - from leadership tasks - helping people to excel. Leaders are required. Managers are optional."* (p. 276, [AnAl03]). So managers are definitely needed in agile environments, but their tasks may change. Ward Cunningham concludes that *"a manager [in agile methods] does more oversight than day-to-day 'managing' of the programming activities"* (p.276, [AnAl03]), which leads to the observation that agile managers have in fact more time to focus on the important administrative matters due to self-organizing development teams.

The following description shows how management tasks are divided among the three roles.

Product Owner
The Product Owner takes over some of the traditional management tasks but without leading the team (cf. [Pich08]):

- Definition and Management of Requirements
- Release Management and Return on Investment
- Close Collaboration with the Team
- Stakeholder Management

He also serves as the link between the team and the customer, i.e. he communicates with the development team and represents the customers' interests, e.g. when defining user stories of the Product Backlog and setting their priorities. Regular meetings with the customer are absolutely necessary. The Product Owner works with the team during the whole project, i.e. requirements are constantly being refined and product increments are being reviewed at the end of each Sprint.

Yahoo! states that the Product Owner is the *"single wringable neck"* [Pich08], so he is solely responsible for the success or failure of a project. Thus the Product Owner needs to take many different interests into account (customer, marketing, service, etc.) and update the Product Backlog accordingly, i.e. add, refine or delete requirements during the course of the whole project, not just in the beginning.

In bigger projects with multiple Scrum teams the Product Owner role can be very complex and demanding. [Pich08] advises to have one Product Owner for one team. The various product owners then form another team: the *Product Owner Team* that may also include marketing, service or other representatives and one *Chief Product Owner*. [Glog11] also states that in complex project situations several product owners are possible and that coordination among them needs to be done one level above the team(s).

Scrum Master
While the Product Owner is responsible for the success of the project, the Scrum Master is responsible for a working Scrum process [ScBe02].

[Pich08] identifies the following tasks for a Scrum Master:
- Establish Scrum as the Process Model in the Team
- Support the Team
- Ensure Direct Collaboration between Product Owner and Team
- Remove Impediments
- Help improve Development Methods
- Lead by Serving

[Gree02] characterizes a *servant leader*, i.e. a leader without authority that supports the team. [Glog11] defines the Scrum Master as a powerless change manager, because he does not have any authority, yet needs to create and sustain a working Scrum process. This difficult task must be taken seriously because *"A dead Scrum Master is a useless Scrum Master!"* (Ken Schwaber, p.26, [Glog11]). Furthermore, the Scrum Master has to assure that the team does not trade quality for productivity [Schw08].

Summing up, the Scrum Master has influence, but no power or authority regarding the team's organization [Pich08].

Team

The team is fully self-organizing. While the Product Owner prioritizes the user stories, the team itself selects the user stories that it can commit to in the next Sprint (following the Product Owner's prioritization). The team needs to be interdisciplinary (from architecture to testing) and work autonomously, i.e. it needs to be able to reach the Sprint objectives without major external dependencies.

If a user story is not ready for deployment at the end of the Sprint, then it is neither the developer's nor the tester's fault. The whole team is held responsible. [Lenc10] argues that in good teams its members need to call each other to account and hence show their mutual respect. Further research of [HoNo10] shows that team members tend to adopt one or more of the following six roles in the team: *Mentor*, *Co-ordinator*, *Translator*, *Champion*, *Promoter* and/or *Terminator*.

Meetings

Scrum prescribes a variety of meetings that reflect the agile character of "individuals and interactions" (cf. Agile Manifesto [BeBe01]). A short overview of the various meeting types is provided in this chapter (based on [DoKl05] and [Glog11]).

Project Planning

At the beginning of each Scrum project there is a project planning session where the vision, project staff and conventions (e.g. programming language, tools, etc.) are set.

The Product Owner defines the first version of the Product Backlog and sets priorities for the user stories.

Estimation
The Product Owner needs to have an updated and estimated backlog for the Sprint Planning Meeting. The Estimation Meeting should be held at least once each Sprint and should not exceed a total length of 90 minutes. The Product Owner can also use this meeting to present new backlog items to the team [Glog11].

In contrast to traditional approaches, estimation is done on a team level [Wird09], i.e. "What can the team accomplish in one Sprint?" instead of "How much can developer X implement or tester Y test in one Sprint?". Moreover, estimations are conducted in reference to other user stories and by using *Story Points*, which is an abstract unit that the agile community has agreed upon [Glog11]. The most widely used agile estimation method is [Gren02]'s *Planning Poker*.

[Glog11] also introduced a new estimation method called *Magic Estimation*, because he argues that planning poker does not work well with bigger teams and backlogs. More on agile estimation can be found in Mike Cohn's highly acclaimed *Agile Estimation and Planning* (cf. [Cohn05]).

Sprint Planning
The Sprint Planning Meeting takes place at the beginning of each Sprint to decide on the Sprint goal, i.e. which user stories from the Product Backlog will be put into the Sprint Backlog to be developed within this Sprint. The team discusses possible ways of implementation, which improves the teams' understanding of the user stories similar to a requirements workshop [Glog11]. The defined Sprint goals are then kept in the Sprint Backlog.

Daily Scrum
The Daily Scrum is a daily *stand-up meeting* that should take no longer than 15 minutes. The meeting is held standing up to enforce the short nature of the meeting. Every team member should state his status and problems (if any) shortly.

More precisely, the Scrum Master will ask each team member the following three simple questions (cf. [DoKl05]):

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

The Scrum Master takes notes of impediments (see *Impediment Chart* in the following chapter) and will try to eliminate them as quickly as possible [Pich08]. In bigger Scrum teams it may help to focus the questions on individual user stories rather than individual team members to keep the meeting short and to the point [DaSe10].

Sprint Review

In the Sprint Review meeting at the end of each Sprint the developed product increment is presented. Participants of this meeting should be members of the management, the customer, user(s) and the Product Owner [ScBe02].

The Scrum Master moderates the meeting. However, it is an informal meeting and therefore it is not allowed to prepare presentations. The product increment is the main issue. Participants should collect information for the next Sprint Planning Meeting by identifying strengths and weaknesses of the current product increment. Furthermore, it is reviewed, which user stories have been implemented during this Sprint (ideally, all that have been selected for the Sprint Backlog). The Product Owner decides if they have been implemented adequately.

Sprint Retrospective

Without a retrospective, teams would make the same mistakes over and over again [Knib07]. The Sprint Retrospective should be held immediately after the Sprint Review Meeting. It is dedicated to reflecting on the Scrum process itself during the last Sprint, e.g. level of collaboration within the team and room for improvement in general [Schw04]. The objective is to increase productivity and efficiency of the team as well as overall software quality [DeLa06].

After the Sprint Retrospective the Sprint is formally completed.

Artifacts

The following artifacts support the Scrum process. Product Backlog and Sprint Backlog are the core components of every Scrum implementation. Additionally, the Burndown Chart is an important tool to track progress within a Sprint. The Impediment Chart and End-of-Sprint Chart are also valuable (optional) utilities (cf. [Pich08]).

Product Backlog

The Product Backlog is created during project planning at the beginning of a Scrum project. It is a list of user stories that should become part of the product. However, in contrast to traditional product specifications, it is intentionally kept incomplete [ScBe02]. This is part of agile thinking because the Product Backlog needs to be under constant development and refinement by the Product Owner. New user stories can be added or old ones deleted in every phase of the project. User stories are usually not only written by the Product Owner (but also by the team itself or other stakeholders e.g.), especially in bigger projects [Glog11].

It is very important that the Product Owner keeps an updated prioritization of the items (i.e. user stories) in the Product Backlog at all times.

Sprint Backlog
At the beginning of each Sprint the team selects user stories from the Product Backlog to be added to the Sprint Backlog. The team then divides the user stories in tasks that will be worked on during the Sprint. The Sprint Backlog may not change during the Sprint with exception of new subtasks being added [ScBe02], because the team has committed to completing all tasks in the Sprint Backlog within the Sprint's duration.

Burndown Chart
The Burndown Chart is a tool to keep track of activities within a Sprint [DoKl05]. More precisely, it shows the day-to-day progress of the Sprint [ScBe02] by comparing estimated and actual effort over time. Figure 3 shows an exemplary Burndown Chart.

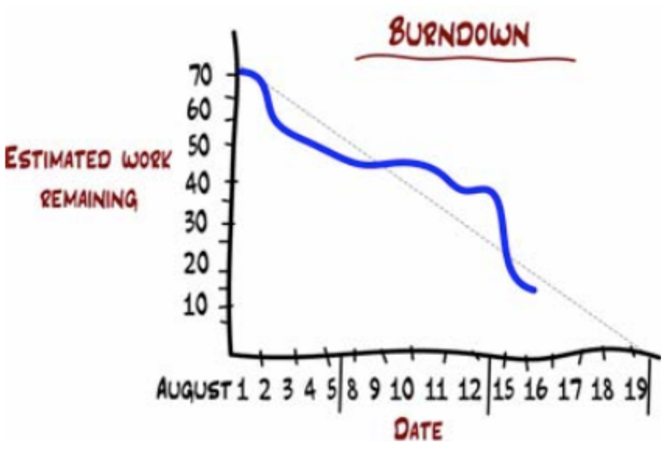


Figure 3: Burndown Chart [KnSk10][4]

The ideal burndown describes the ideal Sprint progress [Pich08]. By summing up efforts (see y axis of Figure 3), it is possible to check if the actual burndown of work is above or below the estimated burndown that is shown as a straight line in Figure 3. This way actual progress (burndown) is compared to the estimated one.

Impediment Chart
The Impediment Chart contains a short description of impediments as well as the date of first occurrence and removal [Pich08]. The Scrum Master should update it at the end

[4] Image is taken from the free online version of [KnSk10] on http://www.crisp.se/kanban

of each Daily Scrum meeting. It is the Scrum Master's duty to deal with the removal of impediments.

End-of-Sprint Chart

The End-of-Sprint Chart is used to document the key data of a Sprint, e.g. how many user stories have been developed successfully and accepted by the Product Owner [Cohn05]. Hence, also third parties (e.g. management) get the possibility to learn about the quality of the Sprint [Pich08].

An exemplary End-of-Sprint chart may have the following content and structure (cf. [Pich08]):

- Sprint Data (Begin, End, Team Members, Planned Availability, Actual Availability)
- Review (Planned Requirements Done/Accepted/Evaluated, Velocity, …)
- Metrics (Sprint Burndown, Impediment Chart, Quality Metrics)

2.3 Lean Software Development

The following description of lean software development and its follower Kanban (see subchapters) is based on [Vall11].

To understand lean software development one first needs to know where *lean* comes from. *Lean Production* (also *Lean Management* or *Just-in-Time*) originates from automobile industry, namely Taiichi Ohno's Toyota Production System in 1978 (cf. [Ohno78]). Toyota developed the new production model after World War II in the 50ies. The new lean production approach stood in sharp contrast to commonly found *Mass Production* among competitors in automobile industry that started to stagnate in both the US and Europe at the time [WoJo92].

The central assumption of lean production is to align production to what provides value for the customer and in this way avoid rework. Everything that does not provide value to the customer is waste[5] and needs to be eliminated [Ohno88, WoJo96].

Toyota observed that costs per unit were lower at smaller production levels than in bigger ones, which provided a competitive edge over mass production [WoJo92]:

- Enormous inventory levels of mass production were diminished (beginning of Just-in-Time production)
- Defects could be spotted more easily at smaller production levels and be dealt with immediately

[5] Sometimes referred to as "muda" which means waste in Japanese

However, this new lean approach demanded highly qualified and motivated personnel, since the anticipation and correction of defects requires one's initiative *before* a blockage in the workflow occurred. Otherwise the whole production flow may stall [WoJo92].

The lean production approach that revolutionized automobile industry can also be applied to software development. [Ande03] and [PoPo04] pioneered in the field in 2003 by putting great effort into introducing lean principles to software development.

The following table shows the principles of lean software development as identified by [PoPo04] and their origin in the Toyota Production System (cf. [Ohno09]).

Lean Software Development [PoPo04]	Toyota Production System [Ohno09]
Eliminate Waste	Extensive analysis of waste
Amplify Learning	The Significance of Understanding
Decide as Late as Possible	Just-in-Time
Deliver as Fast as Possible	Just-in-Time
Empower the Team	Power of individual and team's ability Ability to hand over competence
Build Integrity In	Installation of a production flow
See the Whole	Kanban-Board

Table 3: Comparison Lean Software Development and Toyota Production System (cf. [Vall11])

However, the goal is not to bring manufacturing or the Toyota Production System to software engineering: *"The Toyota Production System is Lean, but Lean is not the Toyota Production System. We are not trying to make software development look more like manufacturing, because Lean is not about manufacturing. Lean is about value streams."* (p.13, [Lada08]).

Donald Reinertson also agrees on the issue: *"However, these [Toyota Production System] approaches are not optimal when we must deal with non-repetitive, unpredictable jobs; with different delay costs; and different task durations – exactly what we must deal with in product development."* (Foreword, [Ande10]).

So it is not about the Toyota Production System, but about the underlying lean principles (according to [WoJo96]):

1. Precisely specify **value** by specific product.
2. Identify the **value stream** for each product.
3. Make value **flow** without interruptions.
4. Let the customer **pull** value from the producer.
5. Pursue **perfection**.

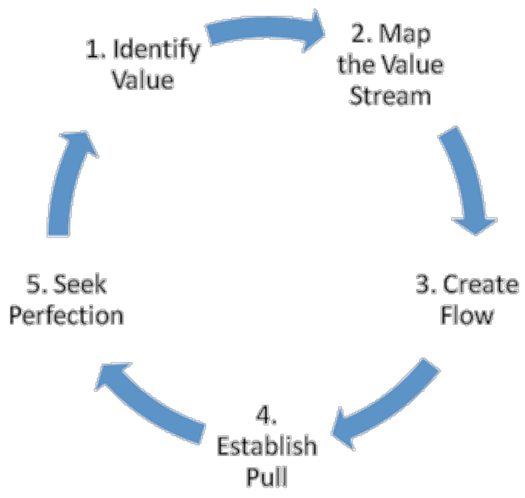


Figure 4: Five Principles of Lean [Lean11]

These principles (cf. Figure 4) have been incorporated into software engineering by means of the process *Kanban* (see following chapter). Still, Kanban is not the only lean software development process. [JaSu09] points out that two other known approaches to software development are also based on the lean *pull* principle (in contrast to traditional *push* approaches):

- *Test Driven Development* by [Beck03], i.e. writing test cases before code
- *Goal-driven Software Development* by [ScPi06], i.e. defining goals before setting requirements and using these goals to *pull* requirements and their priorities

Kanban

The word *Kanban* is Japanese for signal card and originates from the Toyota Production System (cf. [Ohno88]), in which Kanban cards have been used to signal demand in the production flow.

The task of Kanban in software development is very similar:

"Within software development, kanbans are used to 'pull' user stories into development. By limiting the amount of kanbans that are available one can limit the amount of user stories currently developed, i.e., the 'work-in-progress' or in other words, the amount of code that is not finished yet." (p.2, [JaSu09])

That is the Kanban process in a nutshell. A more precise definition follows.

David J. Anderson has introduced Kanban to software development in 2007 at the Lean New Product Development conference [Ande10]. One year later, already six presentations have been held at the Agile 2008 conference [Ande10], which shows the growing interest and need for Kanban in software development.

Kanban's *Recipe for Success* includes six steps [Ande10]:
- Focus on Quality
- Reduce Work-in-Progress
- Deliver Often
- Balance Demand against Throughput
- Prioritize
- Attack Sources of Variability to Improve Predictability

Nevertheless Kanban can be broken down into only three simple rules. The following description is based on [KnSk10] and [Vall11].

Visualize the Workflow
Work is split into pieces and each item is written on a card and put on a Kanban board that is divided into named columns to illustrate the workflow [KnSk10]. The Kanban board is used to visualize the workflow that each item has to run through [Vall11]. There are both physical and electronic Kanban boards. [Ande10] argues that both have their right to exist, i.e. physical boards provide a better psychological effect while electronically ones simplify the creation of metrics and reports. The single work pieces (i.e. cards) may be referred to as *kanban* (lower case "k") [Ande10]. An example Kanban board and workflow is shown in Figure 5.

Limit Work In Progress (WIP)
Limiting the work in progress or WIP means to *"assign explicit limits to how many items may be in progress at each workflow state"* (p.4, [KnSk10]). By setting WIP limits, bottlenecks as well as idle time within the workflow can be identified and visualized on the Kanban board. Furthermore, one has to actively work on solutions for staying

within WIP limits and thus try to anticipate future blockages to improve the overall flow rate (i.e. lead time) [Vall11]. The usage of WIP Limits is demonstrated in Figure 5.

Measure the Lead Time

The *Lead Time* (also *Cycle Time*) is the average time to complete one work item [KnSk10], i.e. it is the time that is needed for one item to complete all steps of the workflow. It is the most important metric in Kanban, because the goal is *"to optimize the process to make lead time as small and predictable as possible"* (p.5, [KnSk10]).

Figure 5 illustrates the application of all three rules by means of a Kanban Board that visualizes the following simple workflow:

Backlog → Selected → Develop (Ongoing/Done) → Deploy → Live

Each work item (illustrated as kanban cards) runs through the whole workflow from *Backlog* to *Live* in a certain time. The average time of all completed kanban cards, i.e. work items, is the Lead Time.

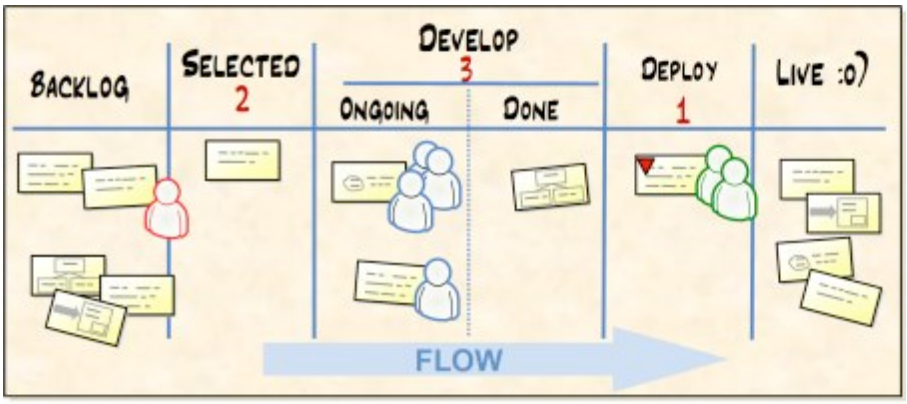


Figure 5: Example Kanban Board [KnSk10][6]

The numbers in Figure 5 are WIP Limits. The number "2" in the *Selected* column denotes that only two work items at a time may be pulled into the workflow. In case of *Develop* we have a shared column and thus shared WIP Limit of "3" for *Ongoing* and *Done*. This forces developers to care for deployment of developed items because otherwise they cannot start with new items due to the WIP limit. Thus deployment on a regular basis comes naturally, which usually helps to improve the overall quality of the product.

[6] Image is taken from the free online version of [KnSk10] on http://www.crisp.se/kanban

Roles, Meetings and Artifacts

Kanban does not specify any roles, meetings or document types. This does not mean that Kanban works without any roles but that it remains free to add whatever roles seem to fit. However, *“the general mindset in both Scrum and Kanban is ‘less is more’. So when in doubt, start with less.”* (p.11, [KnSk10]). Scrum’s daily standup meeting may also be found in Kanban teams, although it is not obligatory [Vall11].

One instrument that deserves to be mentioned is the Cumulative Flow Diagram (CFD). It is also not prescribed by Kanban, but can help to present the correlation of WIP limits and Lead Time. An example is provided in Figure 6.

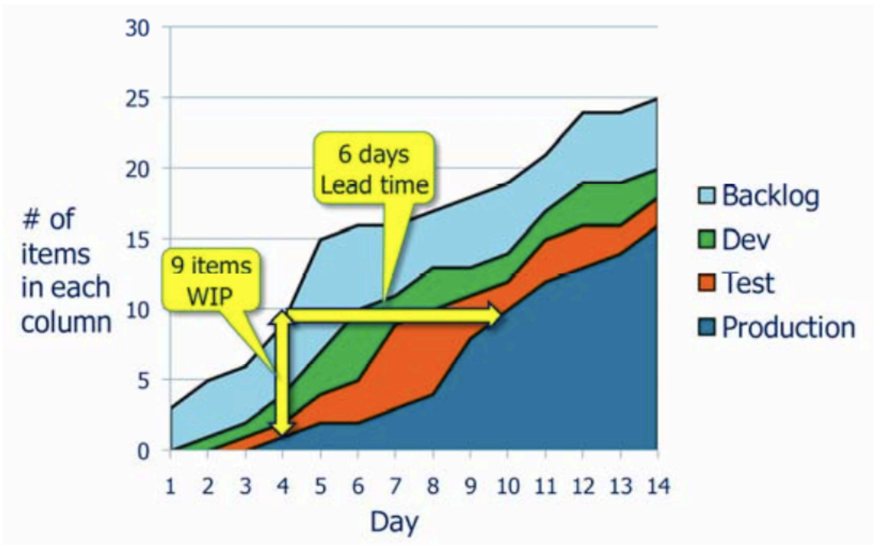


Figure 6: Cumulative Flow Diagram [KnSk10][7]

The horizontal yellow arrow in Figure 6 shows the amount of time that the work item needed in each phase of the workflow, i.e. each column of the Kanban Board, until it reached production.

The vertical yellow arrow in Figure 6 shows the number of work items in each phase. Thus the correlation of WIP and Lead Time can be identified at any given moment in time using a Cumulative Flow Diagram, which may help to find the right WIP limits in order to reduce Lead Time.

The only artifact that really needs to be part of every Kanban process is the Kanban Board, because it is the very central component.

[7] Image is taken from the free online version of [KnSk10] on http://www.crisp.se/kanban

2.4 Comments on Related Publications

This chapter provides an overview of different approaches to multi-project management in publications. Approaches range from more traditional ones, like [Rung10], to lean and agile ones. At this point, only a short overview is given in order to be able to determine which approaches are examined more closely in chapters 3.1 (traditional approaches) and 4.2 (lean and agile ones).

[Rung10] focuses very strongly on how to handle interdependency of projects in "Management of Interdependency in Project Portfolio". While this thesis remains at the program management level (in contrast to portfolio management, see chapter 2.1), the methods presented in [Rung10] to deal with interdependency may be applied to program management as well. A more extensive examination is conducted in chapter 3.1.

[Gree10] shares his insight on an enterprise-level Scrum in "Enterprise Scrum: Scaling Scrum to the Executive Level". He is using a single top-down enterprise Scrum such that even non-engineering departments have to adopt Scrum. Also new methods have been introduced to manage Scrum at an enterprise level:

- Quarterly Sprints
- Assigning whole teams to projects
- Possibility for Kanban methods to deal with shared resource constraints
- Modification (or rather enhancement) of Scrum Master and Product Owner roles

This is a very interesting approach to agile multi-project management and is analyzed more closely in chapter 4.2.

[TaDu09] apply a full lifecycle governance model to agile projects in their work "Governance of an Agile Software Project". Its focus is a single iteration of a large-scale agile software project, which is a very small base to draw conclusions for multi-project management.

[KtLé09] criticize in their paper "Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects" that the business perspective is not sufficiently taken into consideration in large-scale agile development projects. They suggest introducing a *B-Scrum* (Business Scrum), which includes a steering committee to support the decision making process and provide visibility on business expectations. A steering committee might also be an interesting approach for multi-project management, which is examined in chapter 4.2.

[VäRa08] also addresses the need for a better link between product and business planning in agile software development in their work "Towards a Conceptual Framework and Tool Support for Linking Long-term Product and Business Planning with Agile

Software Development". They propose using the tool *Agilefant*[8] that evolves around the conceptual framework that the authors are presenting, which also includes a portfolio level.

[BuJe09] introduces a UX[9] Scrum team that serves several development Scrum teams and works ahead of them in time. The coordination scheme presented is analyzed in chapter 4.2.

[PoOl08] examines requirements prioritization strategies in "Using Simulation to Investigate Requirements Prioritization Strategies". Agile and lean projects both focus heavily on breaking down requirements to manageable levels. So proper prioritization methods are one of the most important tasks. However, the strategies presented do not consider multi-project management.

[AuCu06] describes the use of a lean-agile Project Management Office (PMO). They propose Scrum on the project level and a lean-agile approach for portfolio management including a PMO. The concepts provided are extensive and thus presented in chapter 4.2 for further review.

In conclusion, work on how to handle multi-project management in a lean or agile project environment is rather scarce. This observation is also supported by [VäRa08]: *"proper practical guidance to what it all entails in multi-team, multi-project environments has only started to emerge".* The thesis at hand wants to contribute to this emerging research field by elaborating a best practice in lean-agile multi-project management.

[8] http://www.agilefant.org/

[9] User Experience

3 Comparison of Traditional and Lean-Agile Multi-Project Management

The primary issue of this chapter is to find answers to the following two research questions raised in this thesis:

1. *Which traditional Multi-Project Management approaches are the most accomplished?*

2. *What has to be changed in traditional Multi-Project Management to adjust it to Agile or Lean Project Environments?*

Research question one is examined in chapter 3.1, where a number of state of the art as well as best practice traditional multi-project management approaches are presented.

In chapter 3.2, research question two is analyzed in a theoretical manner in order to draw conclusions for lean-agile multi-project management. This is done by examining their alignment to the *Agile Manifesto* (cf. [BeBe01]).

3.1 Traditional Multi-Project Management

There are various traditional[10] multi-project management approaches that have emerged as best practices:

3.1.1 Project Management Body of Knowledge (PMBOK)
3.1.2 Projects in Controlled Environments (PRINCE 2)
3.1.3 IBM Rational Program Management Method
3.1.4 Doctoral thesis on the management of project interdependencies
3.1.5 Applied Prioritization in Multi-Project Management

3.1.1 PMBOK

PMBOK stands for *Project Management Body of Knowledge* and is an acknowledged standard for the project management profession [Pmbok08].

According to [Pmbok08], *“a program is defined as a group of related projects managed in a coordinated way to obtain benefits and control not available from managing them individually”* (p.9, [Pmbok08]).

[10] as opposed to lean or agile (cf. chapter 4.1)

Program management may include the following actions (cf. [Pmbok08]):

- Resolving resource constraints and/or conflicts that affect multiple projects within the program
- Aligning organizational/strategic direction that affects project and program goals and objectives
- Resolving issues and change management within a shared governance structure

[Pmi08] identifies five *Program Management Process Groups*:

Initiating. Defines and authorizes the program or project within the program and produces the program benefits statement for the program.

Planning. Plans the best alternative courses of action to deliver the benefits and scope that the program was undertaken to address.

Executing. Integrates projects, people and other resources to carry out the plan for the program and deliver the program's benefits.

Monitoring and Controlling. Requires that the program and its component projects be monitored against the benefit delivery expectations and that their progress be regularly measured, to identify variances from the program management plan. This Process Group also coordinates corrective actions to be taken when necessary to achieve program benefits.

Closing. Formalizes acceptance of a product, service or benefit/result and brings the program or program component (e.g. project) to an orderly end.

These Process Groups are not executed in a linear fashion, i.e. they overlap, and one or more processes from each Process Group will normally be executed at least once in every phase of a program life cycle [Pmi08].

[Pmbok08] advises to establish a *Project Management Office (PMO)*, which is *"an organizational body or entity assigned various responsibilities related to the centralized and coordinated management of those projects under its domain"* (p.11, [Pmbok08]). However, the projects supported or administered by the PMO do not need to be related [Pmbok08].

The primary functions of a PMO are defined as follows (cf. [Pmbok08]):

- Managing shared resources across all projects administered by the PMO
- Identifying and developing project management methodology, best practices and standards
- Coaching, mentoring, training and oversight
- Monitoring compliance with project management standards, policies, procedures and templates via project audits
- Developing and managing related project policies, procedures, templates and other shared documentation (organizational process assets)
- Coordinating communication across projects

Although both the project manager and the PMO are aligned with strategic needs of the organization, they are driven by different requirements, which include (cf. [Pmbok08]):

- The project manager focuses on the specified project objectives, while the PMO manages major program scope changes, which may be seen as potential opportunities to better achieve business objectives.
- The project manager controls the assigned project resources to best meet project objectives while the PMO optimizes the use of shared organizational resources across all projects.
- The project manager manages the constraints (scope, schedule, cost, quality, etc.) of the individual projects while the PMO manages the methodologies, standards, overall risk/opportunity and interdependencies among projects at the enterprise level.

3.1.2 PRINCE 2

The following description of PRINCE 2 is based on [Köhl06].

PRINCE 2 (**Pr**ojects **in** **C**ontrolled **E**nvironments) has been developed in the United Kingdom in 1996 and is endorsed by the government as de facto standard. It is a process-oriented IT project management framework. It does, however, not cover the development of hard- or software in detail, because these tasks are beyond the scope of the framework, but it provides best practice concepts to choose from and be applied and customized to individual needs.

Program management in PRINCE 2 is conducted as *Management by Exception*, i.e. management knows about the status of the project, but it only interferes if there are deviations from the original project plan. As long as the project stays in time and budget, the project manager may lead the project as he pleases. [Köhl06] argues that this is one of the major advantages of PRINCE 2: The program management is forced to set parameters for each project individually and within these parameters the project manager is allowed to steer the project.

The program management tasks are:

- Appoint the Executive and the Project Manager
- Capture previous lessons
- Design and appoint the project management team
- Prepare the outline Business Case
- Select the project approach and assemble the Project Brief
- Plan the initiation stage

Furthermore, the *Project Board* may request corporate advice or decisions from the program management.

Figure 7 provides an overview of the PRINCE 2 process model (without the delivery stage). A detailed description is beyond the scope of this thesis, but can be found in [Prince211].

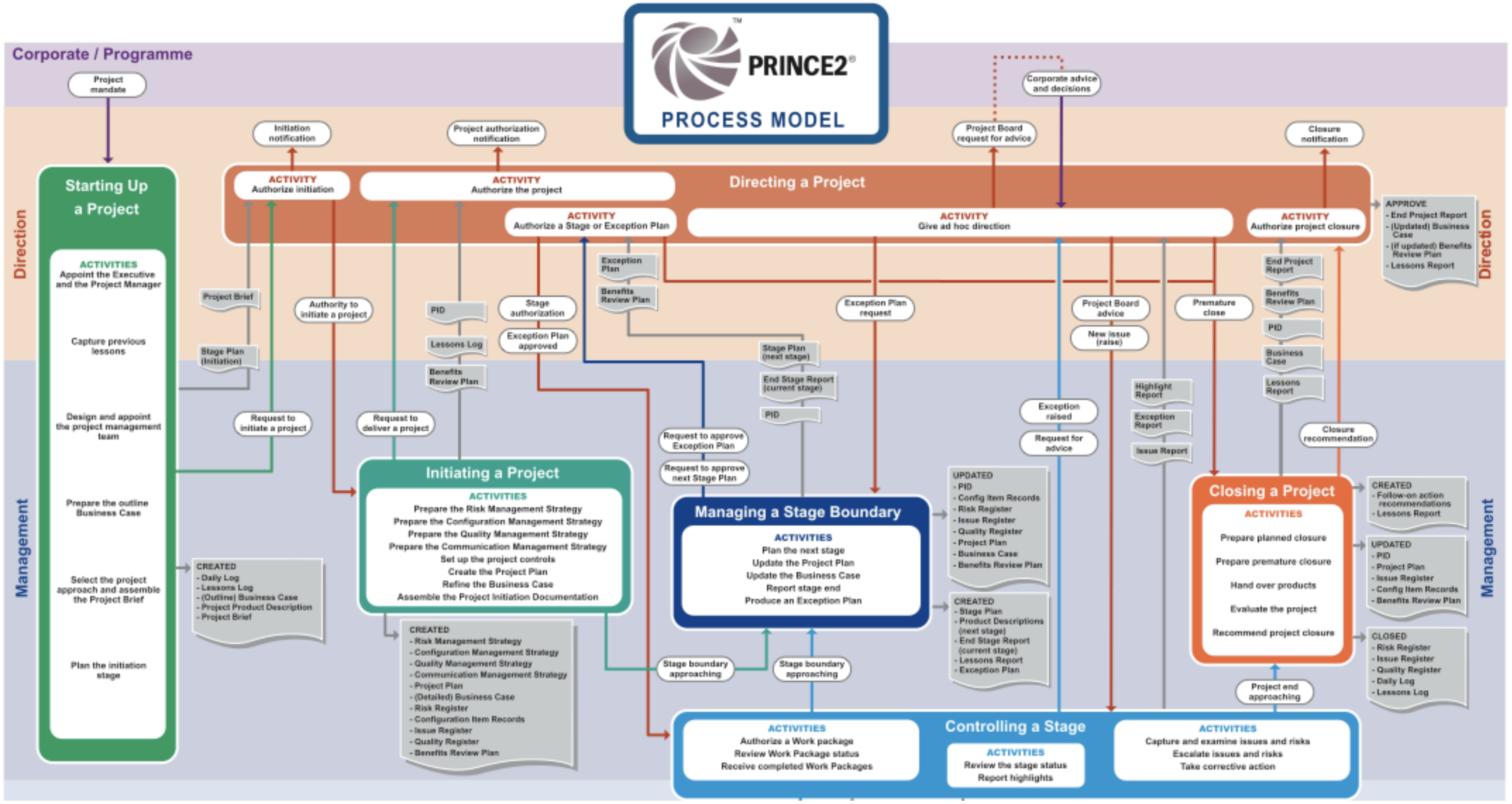

PRINCE2®
PROCESS MODEL
Corporate / Programme
Project mandate
Direction
Management
Starting Up a Project
ACTIVITIES
Appoint the Executive and the Project Manager
Capture previous lessons
Design and appoint the project management team
Prepare the outline Business Case
Select the project approach and assemble the Project Brief
Plan the initiation stage
CREATED - Daily Log - Lessons Log - (Outline) Business Case - Project Product Description - Project Brief
Initiation notification
Project authorization notification
Directing a Project
ACTIVITY Authorize initiation
ACTIVITY Authorize the project
ACTIVITY Authorize a Stage or Exception Plan
ACTIVITY Give ad hoc direction
ACTIVITY Authorize project closure
Corporate advice and decisions
Project Board request for advice
Closure notification
APPROVE - End Project Report - (Updated) Business Case - (If updated) Benefits Review Plan - Lessons Report
Project Brief
Stage Plan (Initiation)
Authority to initiate a project
Request to initiate a project
Request to deliver a project
PID
Lessons Log
Benefits Review Plan
Stage authorization
Exception Plan approved
Exception Plan
Benefits Review Plan
Initiating a Project
ACTIVITIES
Prepare the Risk Management Strategy
Prepare the Configuration Management Strategy
Prepare the Quality Management Strategy
Prepare the Communication Management Strategy
Set up the project controls
Create the Project Plan
Refine the Business Case
Assemble the Project Initiation Documentation
CREATED - Risk Management Strategy - Configuration Management Strategy - Quality Management Strategy - Communication Management Strategy - Project Plan - (Detailed) Business Case - Risk Register - Configuration Item Records - Issue Register - Quality Register - Benefits Review Plan
Stage Plan (next stage)
End Stage Report (current stage)
PID
Request to approve Exception Plan
Request to approve next Stage Plan
Exception Plan request
Managing a Stage Boundary
ACTIVITIES
Plan the next stage
Update the Project Plan
Update the Business Case
Report stage end
Produce an Exception Plan
UPDATED - PID - Config Item Records - Risk Register - Issue Register - Quality Register - Project Plan - Business Case - Benefits Review Plan
CREATED - Stage Plan - Product Descriptions (next stage) - End Stage Report (current stage) - Lessons Report - Exception Plan
Stage boundary approaching
Stage boundary approaching
Project Board advice
New issue (raise)
Exception raised
Request for advice
Premature close
Highlight Report
Exception Report
Issue Report
End Project Report
Benefits Review Plan
PID
Business Case
Lessons Report
Closure recommendation
Closing a Project
ACTIVITIES
Prepare planned closure
Prepare premature closure
Hand over products
Evaluate the project
Recommend project closure
CREATED - Follow-on action recommendations - Lessons Report
UPDATED - PID - Project Plan - Issue Register - Config Item Records - Benefits Review Plan
CLOSED - Risk Register - Issue Register - Quality Register - Daily Log - Lessons Log
Project end approaching
Controlling a Stage
ACTIVITIES
Authorize a Work package
Review Work Package status
Receive completed Work Packages
ACTIVITIES
Review the stage status
Report highlights
ACTIVITIES
Capture and examine issues and risks
Escalate issues and risks
Take corrective action

Figure 7: PRINCE 2 Process Model (cf. [Prince211])

3.1.3 IBM Rational Program Management Method

[Hanf04] states that in the *IBM Rational Program Management Method* a program has three levels of hierarchy:

1. **Bottom:** The *Project Managers* are assigned to the various projects within the overall program.
2. **Middle:** The *Program Manager/Director* ensures that the work effort achieves the outcome specified in the business and IT strategies.
3. **Top:** The *Program Sponsor(s)* and *Program Steering Committee* own and oversee the implementation of the program's underlying business and IT strategies and define the program's connection to the enterprise's overall business plan(s) and direction.

Responsibilities of the middle level *Program Manager* also involve (cf. [Hanf04]):

- Is accountable to executive sponsors for schedule, budget, and quality of all program elements.
- Leads high-level sessions for program plan and schedule development.
- Reviews/approves project plans for conformance to program strategy and program plan and schedule.
- Acts as the communications conduit to executive sponsors and program steering committee and conducts periodic briefings/status updates.
- Escalates decisions to executive sponsors as necessary.

Further top level tasks are (cf. [Hanf04]):

- Providing and interpreting policy
- Creating an environment that fosters sustainable momentum for the program (i.e., removing barriers both inside and outside the enterprise)
- Periodically reviewing program progress and interim results to ensure alignment with the overall strategic vision

Similar to the PMBOK's *Project Management Office* (cf. chapter 3.1.1), [Hanf04] discusses the establishment of a *Program Management Office (PMO)*, but he focuses on PMOs that support single programs (in contrast to a permanent *Enterprise PMO*). As such, [Hanf04] defines the following roles in a PMO:

- Program office management
- Resources coordination
- Budget administration and procurement
- Risk assessment
- Work products tracking and review
- Facilities administration
- Contracts administration

- Technical support liaison
- Training coordination
- Methodology and process support
- Issues management
- Communications management
- Status reporting management

The PMO serves the program manager/director and essential PMO roles are filled by a group of senior specialists [Hanf04].

[Hanf05] also expresses problems regarding financial aspects of program management within an organization: *"However, larger efforts [of multiple projects in program management] may have multiple funding sources, each wanting a say about where and how money is spent. (...) Because of their significant cost, programs are typically an enterprise concern rather than the concern of a single business unit. Funding often comes from multiple business segments and is overseen by the program governance structure."*

Finally, Figure 8 provides an overview of the five modules of the IBM Rational Program Management Method.

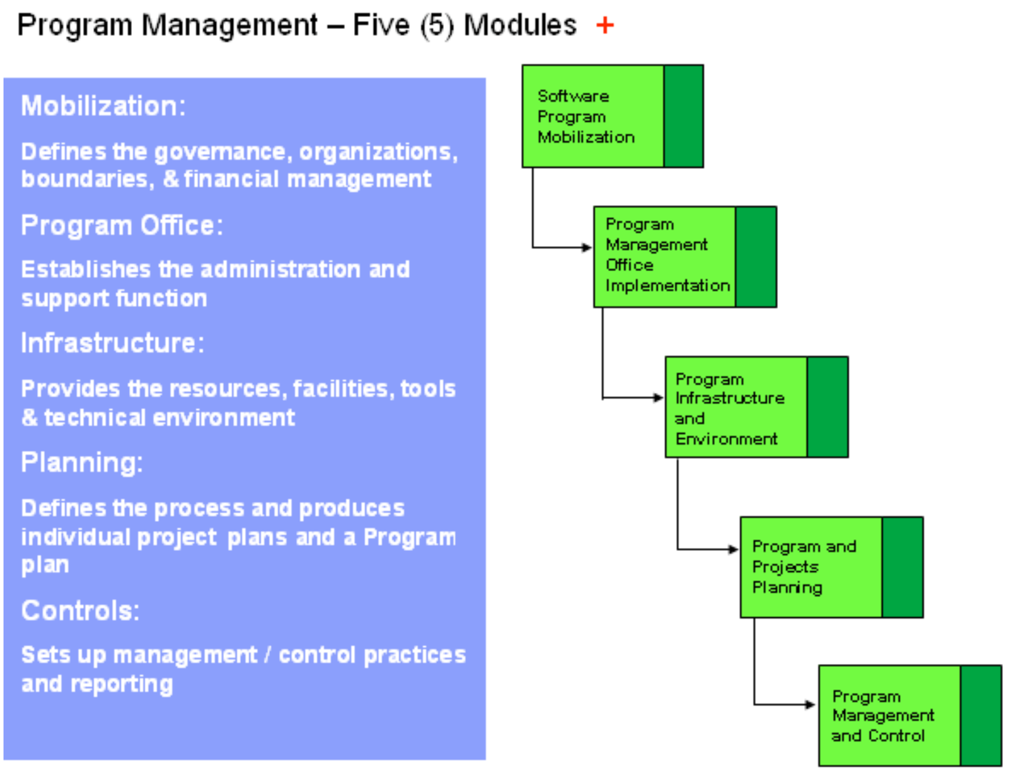

Program Management – Five (5) Modules +
Mobilization:
Defines the governance, organizations, boundaries, & financial management
Program Office:
Establishes the administration and support function
Infrastructure:
Provides the resources, facilities, tools & technical environment
Planning:
Defines the process and produces individual project plans and a Program plan
Controls:
Sets up management / control practices and reporting
Software Program Mobilization
Program Management Office Implementation
Program Infrastructure and Environment
Program and Projects Planning
Program Management and Control

Figure 8: IBM Rational Program Management Method [IBM05]

3.1.4 Management of Project Interdependencies

[Rung10a] has devoted the research of his doctoral thesis “Management of Interdependency in Project Portfolio” to the management of project interdependencies.

In one of his associated publications [Rung10b], he provides an overview of interdependency techniques. The various groups that emerge are (see also Figure 9):

- Informal Methods
- Mathematical Optimization Methods
- Scoring/Ranking Models
- Dependency Matrix
- Visual Methods

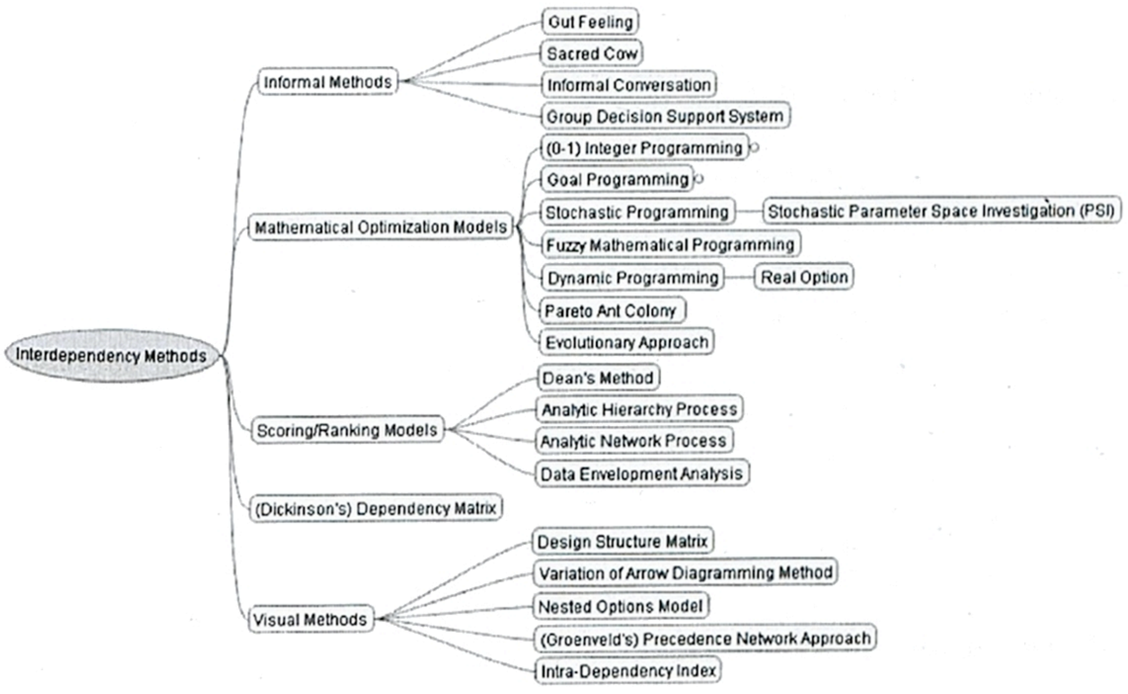


Figure 9: Project Interdependency Techniques [Rung10b]

However, [Rung10a] states that *“in the literature* [cf. CoEd99, MeMa99]*, mostly mathematical techniques are recommended, but decision makers do not want to use them”* (p.90, [Rung10a]). Informal approaches prevail, e.g. by using meetings instead of mathematical techniques, but *“no specific meetings for* (…) *interdependencies* [exist]*, these aspects are indirectly managed by several* [general] *meetings”* (p.1510, [Rung09] and p.90, [Rung10a]).

Concerning human resources, understanding is easy, correct assignment is not [Rung10a], e.g. *“it is not reasonable to assign one person for many projects simultaneously* (…) *[Rung08]”* (p.91, [Rung10a]). [Rung09] argues that in case of sharing a person between projects, monetary incentives or changing the working place within the company are not always useful to deal with motivational problems. During case studies, career path and parties were reported as relieving solutions [Rung09].

Technical interdependencies are the responsibility of project managers and technicians, not of the top management [Rung10a]. However, the top management needs to take care of knowledge diffusion [Rung10a], which occurs mostly in technical meetings and review meetings [Rung08].

The main drivers for managing interdependencies on a daily basis were (cf. [Rung10a]):

- Achieve overview of resources
- Establish good knowledge diffusion across the company
- Awareness of situations
- Get an overview of all projects (to see all connections between them)

[Rung10a] concludes that according to his multiple-case study most companies discuss interdependency-related issues in several general meetings (among other unrelated subjects), because *"a more effective, full-scale approach to interdependency management would require more concentrated and formalized PM and PPM processes"* (p.93, [Rung10a]).

3.1.5 Applied Prioritization in Multi-Project Management

[HiAl11] provides many strategies for multi-project management and describes its application to real-life working environments. One of the core components discussed, is the prioritization of projects in multi-project management. Petra Leyendecker defines the following rules for prioritization (among others) in [HiAl11]:

Prioritize according to Resource Constraints

The assignment of top priority among resources needs to be limited because *"if everything is high priority, then nothing is"* (p.163, [Lada08]). A possible approach would be that only projects get top priority that can be completed with full resource support. In conclusion, the assignment of top priority would be restricted due to the limited amount of available employees, equipment and budget.

Prioritize according to Urgency

Urgency is determined by deadlines or by asking the following question: *"What happens if we postpone the project?"*. The higher the strategic and economic negative impact is, the sooner the project should be started. By freeing resources from less urgent projects, necessary resources can be assigned to complete urgent projects in time.

Take Line Organization into Account

Line assignments play an important part in prioritization of resources because they have their own deadlines and priorities. In case of a conflict, a mutual agreement between projects and line organization needs to be reached. To achieve this outcome, line organization needs to regard the support of project as one of its tasks, e.g. through target agreements with executives.

Prioritize with Project Clusters

Prioritization within a group of projects is only reasonable if projects compete for the same resources. Projects with different resource needs do not require prioritization in relation to one another. The solution is the establishment of *project clusters* according to resource settings and prioritization within these clusters. Project clusters also facilitate the alignment to line organization's goals. Table 4 provides an example of functional project clusters.

Resource Consumers		**Resource Suppliers**				
Project Clusters	**Projects and Priorities**	R&D[11]	Production	Business Organization	Application Development	HR[12] Development
Product Development	P-1 P-2 P-3	**x**	**x**			
IT Projects	I-1 I-2 I-3 I-4			**x**	**x**	
Organizational Projects	O-1 O-2			**x**	**x**	**x**

Table 4: Prioritization with Functional Project Clusters (cf. [HiAl11])

Prioritize with Categories

Projects should be categorized, e.g. as *A*-, *B*- or *C-Projects*. Each category should hold the same amount of projects. If all projects are category A projects, then prioritization is useless. The categories can be associated with their resource capacity, i.e.:

[11] Research & Development

[12] Human Resource

- Priority A: Resource needs are fully allocated.
- Priority B: At least 50% of needed resources are allocated.
  Deviations from original plan are expected.
- Priority C: At least 25% of needed resources are allocated.
  Considerable deviations from original plan are expected.

The classification in categories can be applied to the whole project environment or to project clusters. Within a category or a project cluster, projects need to be prioritized further.

Establish a Regular Prioritization Process
Project evaluation, selection and prioritization need to be conducted for the construction and revision of project portfolios in a *Project- and Portfolio Review* that e.g. can be held quarterly. During this review, changes in the business environment (market, strategy, laws and regulations, etc.) and their impact on project evaluations and priorities need to be examined.

Communicate Priorities and their Significance
It is very important that all involved parties share a common understanding of priorities. Prioritization can only work if it is supported and understood by everyone because priorities indicate the significance of a project. Hence, reasons for changes in prioritization also need to be communicated.

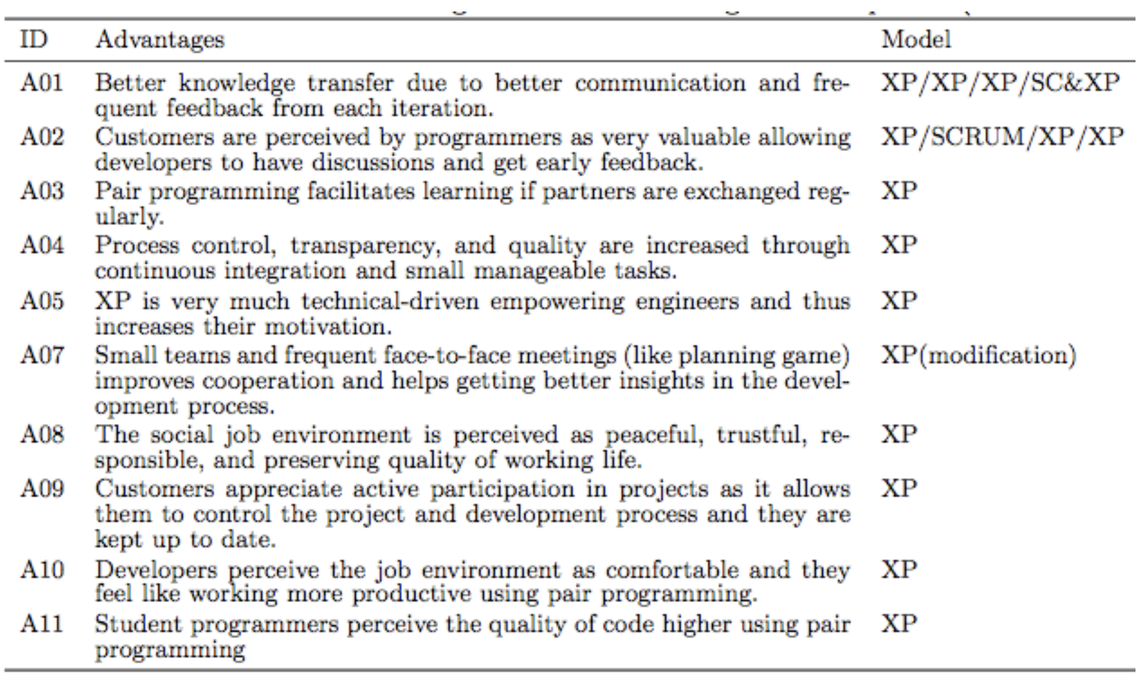
Identify Limitations of Priorities
There is no ideal approach to prioritization. It needs to be developed by each company individually to meet constraints of different organizational structures and processes. Prioritization also poses risks: Low priorities may lead to underestimate a project's significance among participants. It may also decrease motivation and cause contrary personal priorities and thus slow down the progress of the project. Thus the amount of projects within a portfolio or project cluster should be limited, because with e.g. one project ranked 20 and another project ranked 21 in a project cluster, the difference in priority has lost its practical value.

3.2 Conclusions for Lean/Agile Multi-Project Management

The approaches from chapter 3.1 are examined here to answer research question two:

2. *What has to be changed in traditional Multi-Project Management to adjust it to Agile or Lean Project Environments?*

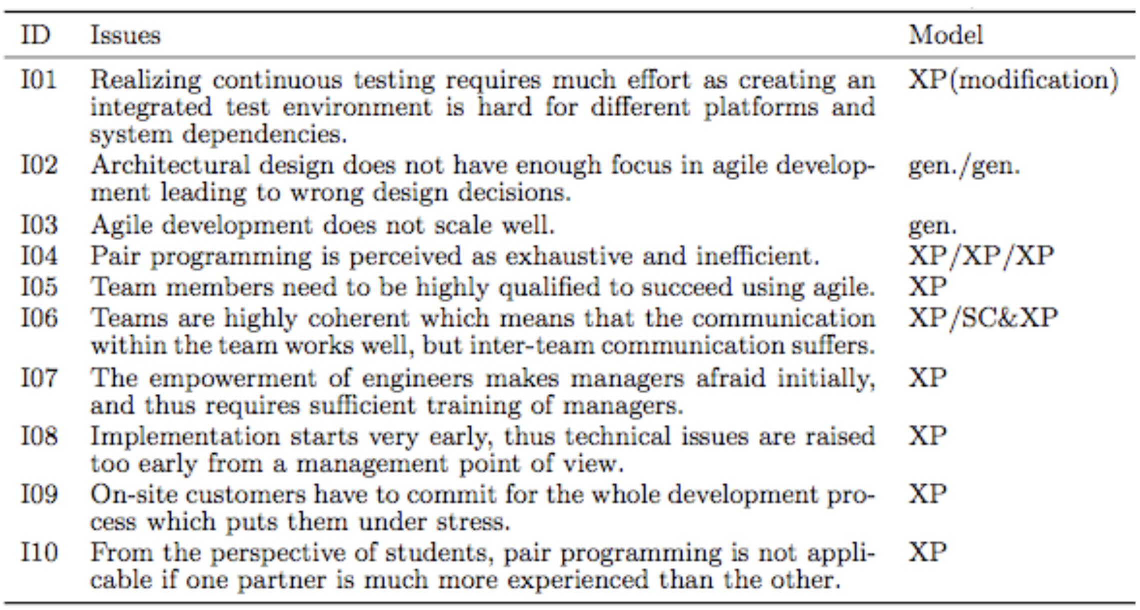
Both the *Agile Manifesto* (cf. [BeBe01]) and the *issues and advantages in agile and incremental development* (cf. [PeWo09], Figures 10 and 11) shall be taken into consideration when drawing conclusions from the traditional multi-project management approaches that have been presented in chapter 3.1.



ID	Advantages	Model
A01	Better knowledge transfer due to better communication and frequent feedback from each iteration.	XP/XP/XP/SC&XP
A02	Customers are perceived by programmers as very valuable allowing developers to have discussions and get early feedback.	XP/SCRUM/XP/XP
A03	Pair programming facilitates learning if partners are exchanged regularly.	XP
A04	Process control, transparency, and quality are increased through continuous integration and small manageable tasks.	XP
A05	XP is very much technical-driven empowering engineers and thus increases their motivation.	XP
A07	Small teams and frequent face-to-face meetings (like planning game) improves cooperation and helps getting better insights in the development process.	XP(modification)
A08	The social job environment is perceived as peaceful, trustful, responsible, and preserving quality of working life.	XP
A09	Customers appreciate active participation in projects as it allows them to control the project and development process and they are kept up to date.	XP
A10	Developers perceive the job environment as comfortable and they feel like working more productive using pair programming.	XP
A11	Student programmers perceive the quality of code higher using pair programming	XP

Figure 10: Advantages of Agile and Incremental Practices [PeWo09]

Moreover, [Ambl09] mentions that agile practices have greater visibility and opportunity to steer for stakeholders.



ID	Issues	Model
I01	Realizing continuous testing requires much effort as creating an integrated test environment is hard for different platforms and system dependencies.	XP(modification)
I02	Architectural design does not have enough focus in agile development leading to wrong design decisions.	gen./gen.
I03	Agile development does not scale well.	gen.
I04	Pair programming is perceived as exhaustive and inefficient.	XP/XP/XP
I05	Team members need to be highly qualified to succeed using agile.	XP
I06	Teams are highly coherent which means that the communication within the team works well, but inter-team communication suffers.	XP/SC&XP
I07	The empowerment of engineers makes managers afraid initially, and thus requires sufficient training of managers.	XP
I08	Implementation starts very early, thus technical issues are raised too early from a management point of view.	XP
I09	On-site customers have to commit for the whole development process which puts them under stress.	XP
I10	From the perspective of students, pair programming is not applicable if one partner is much more experienced than the other.	XP

Figure 11: Issues of Agile and Incremental Practices [PeWo09]

3.2.1 Project Management Office (PMO)

Both the *PMBOK guide* (cf. chapter 3.1.1) and the *IBM Rational Program Management Method* (cf. chapter 3.1.3) suggest introducing a *Project Management Office (PMO)* to cope with multi-project management affairs. The PMBOK's PMO has six defined tasks. IBM defines a lot more for its PMO: 13 tasks that need to be managed by a group of senior specialists [Hanf04]. A comparison of PMBOK's and IBM's tasks is conducted in chapter 3.2.2.

Agile practices always incorporate a *less is more* kind of approach: *"when in doubt, start with less"* (p.11, [KnSk10]). So a *lean or agile PMO* should also follow this advice. [Finc10] discusses the introduction of an agile PMO in his article "Is an agile PMO Possible?". He argues that the PMO *"can also help facilitate communication between developers, project managers and executives"* (p.1, [Finc10]). In other words, the PMO can help to keep the rest of the company informed about scope changes, delays or quality issues [Finc10]. The benefit is better risk management through the PMO and greater levels of transparency through agile teams [Finc10]. However, for an agile PMO to work, compromises have to be made: *"'Don't be pure PMI or pure agile'. Rather, find ways to get each team to give a little ground."* (p.2, [Finc10]).

Another approach to a lean and agile PMO is presented in chapter 4.2.4.

3.2.2 Tasks for Multi-Project Management

Table 5 consolidates all multi-project management tasks that have been identified in chapter 3.1. It can be referred to as a checklist for lean and agile multi-project management in a traditional sense. The marks in Table 5 show where the tasks originated from during research for this thesis. However, this does not imply that these tasks are missing in another framework that is not marked.

Multi-Project Management Task	PMBOK	PRINCE 2	IBM
Monitoring and Controlling	x		x
Managing Shared Resources	x		x
Resolving Issues	x		x
Coaching, Mentoring and Training	x		x
Identifying Common Methodology, Best Practices and Standards	x		x
Developing Shared Process Assets (Templates etc.)	x		x
Coordinating Communication across Projects	x		x
Aligning Projects to Organizational/Strategic Direction	x	x	x
Capture Previous Lessons		x	
Appoint Project Manager and Team	x	x	x
Select Project Approach and Plan Initiation	x	x	x
Risk Assessment			x
Technical Support			x
Contracts Administration			x
Facilities Administration			x
Status Reporting Management			x

Table 5: Consolidation of Traditional Multi-Project Management Tasks

Since PRINCE 2 follows *Management by Exception*, its tasks are very different from PMBOK and IBM, which share more common ground. IBM defined the most tasks as it has introduced many roles and hierarchy levels for program management (cf. chapter 3.1.3).

3.2.3 Management by Exception

Multi-project management in PRINCE 2 (cf. chapter 3.1.2) is driven by the idea of *Management by Exception*, i.e. if the project manager navigates the project within pre-defined boundaries, the program management is not going to interfere.

In general, the PRINCE 2 framework seems too heavyweight to draw conclusions for agile practices, but the paradigm of *Management by Exception* deserves careful consideration.

As far as agile practices are concerned, *Management by Exception* can be found partially. It can be found in a Scrum team's commitment during a Sprint iteration. The Scrum Master and team are allowed to work without interference and self-organizing for the duration of a Sprint. At the end of the Sprint, the Product Owner (that comes closest to a traditional project manager in Scrum's roles) reviews the current product increment and provides some direction for the next Sprint.

However, *Management by Exception* does not really meet the essence of agile thinking since Stakeholder involvement and collaboration are one of the core principles of the Agile Manifesto. Due to the violation of the Agile Manifesto, *Management by Exception* should not be part of lean-agile multi-project management.

3.2.4 Multi-Project Management Hierarchy

The *IBM Rational Program Management Method* (cf. chapter 3.1.3) works with strict levels of hierarchy: Project Managers (Bottom), Program Manager (Middle) and Program Sponsors/Committee (Top).

Agile practices usually try to avoid strict hierarchies. The Scrum Master, e.g., is regarded as a *servant leader* [Gree02]. Scrum's Product Owner is not leading either, although he is the one responsible for the success of the project. In consequence, the hierarchy in lean and agile multi-project management should be kept as flat as possible to emphasize *individuals and interactions* (cf. [BeBe01]).

Chapter 4.2.4 shows how [AuCu06] combine agile practices and a lean-agile PMO (Project Management Office).

3.2.5 Project Interdependency Meetings

[Rung10a] finishes his doctoral thesis on project interdependencies (cf. chapter 3.1.4) with the conclusion that in practice interdependency issues are mostly discussed in general meetings, because many companies lack *"more concentrated and formalized PM and PPM processes"* (p.93, [Rung10a]). Another problem is that decision makers do not want to use complex mathematical techniques [Rung10a].

However, solving interdependency issues in meetings aligns well with the Agile Manifesto's principle "*individuals and interactions over processes and tools"* (cf. [BeBe01]). This lightweight approach relying strongly on meetings and communication is suitable for lean-agile multi-project management in addition to other components such as continuous prioritization (cf. chapter 3.2.6).

3.2.6 Prioritization

Prioritization as described by [HiAl11] (see chapter 3.1.5) is an important issue in multi-project management for agile and lean projects as well. The main rules were:

- Prioritize according to resource constraints
- Prioritize according to urgency
- Take line organization into account
- Prioritize using project clusters
- Prioritize in categories
- Establish a regular prioritization process
- Communicate priorities and their significance
- Identify limitations of priorities

Most of these also apply to lean and agile environments, where periodic prioritization plays an important role. The prioritization with project clusters can be achieved using *shared Product Backlogs* for projects that require similar resources (cf. Table 4). Prioritization with categories has also been suggested by Corey Ladas in [Lada08] in a similar fashion. The method is named *Progressive Priority Filtering* and was developed for lean follower *Kanban* (cf. chapter 2.3). The method has been developed for prioritization on requirements' level but can be applied to the project level as well (similar to [HiAl11]).

Communication is a central component of lean and agile practices (cf. Agile Manifesto [BeBe01]). So quarterly meetings (as suggested by [HiAl11]) for the evaluation of project clusters and portfolio are easy to establish. Even shorter time spans for these meetings are possible.

Dealing with line organization is often neglected in agile practices, but many companies do not use agile practices for all of their projects. Even if all projects were agile, projects would still need steering to be aligned with business goals. Hence, line organization has to be considered.

In conclusion, frequent prioritization is part of most agile as well as traditional practices: *"For one, they are both* [agile and PMBOK] *interested in prioritising projects to ensure that the organisation is investing in the right ones."* (p.2, [Finc10]). Thus periodical and continuous prioritization is also an important issue in lean-agile multi-project environments.

3.2.7 The Adoption of Agile Elements in Traditional Practices

[SpVo10] provides an interesting discussion on the adoption of agile elements (or rather lack thereof) in the PMBOK (cf. chapter 3.1.1). The main arguments in [SpVo10] are summarized here to gain insight on the relationship of traditional and agile practices.

Thinking and planning are the two most important virtues in agile projects as well as in traditional ones. Scrum (cf. chapter 2.2) has very specific definitions for iterations, artifacts and phases of a project. The PMBOK (cf. chapter 3.1.1), as a process framework, on the other hand is rather vague. Hence, a comparison between the two is rather difficult because the result would be that most of Scrum could be described using the PMBOK guide one way or another. The real question is if this comparison would be reasonable because the PMBOK guide completely lacks the agile context. The latest edition [Pmbok08] does not mention the term "agile" even once [SpVo10].

Agile processes follow a different approach than traditional ones in the PMBOK: The full extent of the project is not set initially and then gradually implemented, but planned from one Sprint to another [BlWo08]. But agile methods cannot be applied to all projects, e.g. to a fixed price project: The customer willingly hands over the risk to the supplier such that each change in requirements needs to be paid for (by means of change requests). This also violates one of the principles of the Agile Manifesto *"customer collaboration over contract negotiation"* (cf. [BeBe01]). [SpVo10] argue that also in agile projects contracts are indispensable. For instance, the stakeholder on the customer's side may change from an agile supporter to an agile adversary. In this case the authors suggest starting with a traditional approach and establishing change requests by means of a simple *Product Backlog* (cf. chapter 2.2).

To sum up, PMBOK and agile processes do not contradict each other, but have not been united on a larger scale either. [SpVo10] conclude that leading agile projects requires a profound methodical background as well. [BoMi11] also provides interesting insight on the matter stating that being *agile* demands discipline as well: *"Process does not contradict agility; in fact, we talk about agile processes! To borrow from Barry Boehm and Richard Turner's book title* [cf. BoTu03]*, finding the right dosage between agility and discipline is a balancing act."* (p.545, [BoMi11]).

4 Lean-Agile Multi-Project Management

This chapter examines known practices for lean and agile multi-project management in literature (chapter 4.1) and scientific publications (chapter 4.2). Both are essential for the practical evaluation of lean-agile multi-project management during the case study of this thesis (cf. chapters 5 and 6).

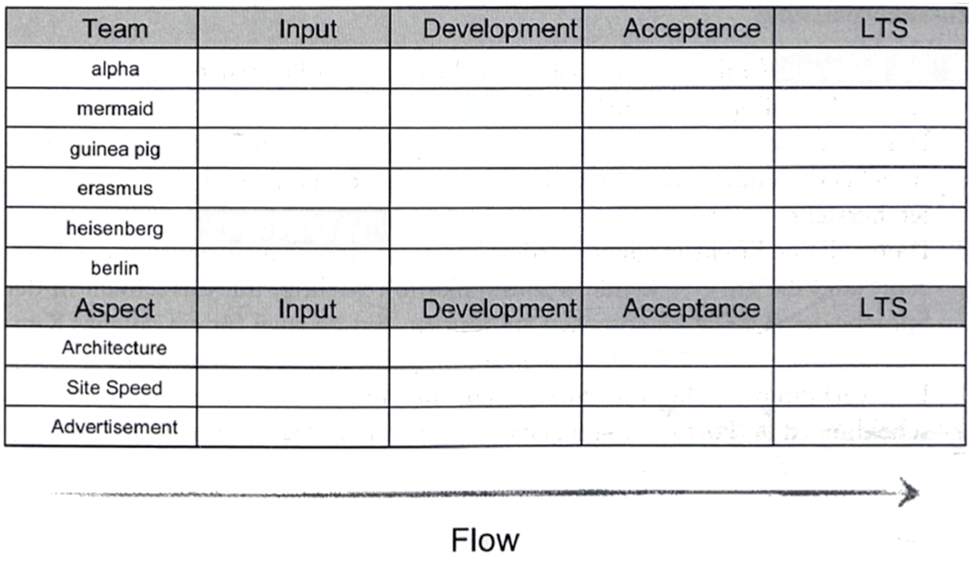
4.1 Known Practices in Literature

There are few approaches to multi-project management in literature, which shows that there are not many established methods to deal with this complex issue in lean and agile project environments. This claim is supported by [VäRa08]: *"proper practical guidance to what it all entails in multi-team, multi-project environments has only started to emerge"*. This subchapter provides an overview of the approaches in literature.

4.1.1 Kanban Swim Lanes

In the German edition [Ande11] of "Kanban" by David J. Anderson (cf. [Ande10]), a field report of mobile.international GmbH, which operates Europe's biggest online car auction, has been added. It describes how Kanban can be used to coordinate different teams between projects. The approach is team-based rather than project-based by introducing a *swim lane* for each team.

Figure 12 shows an example with swim lanes from *Input* to *LTS (Live To Site)*. The workflow is very simple so teams have the possibility to work according to their own process, which is either Scrum or a Kanban implementation. However, it is important that the global board (Figure 12) stays synchronized with the team-level boards.



Team	Input	Development	Acceptance	LTS
alpha				
mermaid				
guinea pig				
erasmus				
heisenberg				
berlin				
Aspect	Input	Development	Acceptance	LTS
Architecture				
Site Speed				
Advertisement				

Figure 12: Team-based Swim Lanes [Ande11]

The aspect part of the board in Figure 12 was added to the board in a second version to deal with matters that go beyond regular project work and affect all teams. Exclusively assigned short-term teams or virtual teams consisting of regular team members were created to work on these aspects. Having technical aspects on the global Kanban board allows the prioritization in relation to (regular) business projects.

The flexibility this approach provides has been greatly appreciated, but it also had the following side effects: teams do not "own" code or parts of the product due to changing assignments. In response stakeholders came to the compromise that continuity for teams is granted whenever priorities allow (e.g. assigning second highest priority instead of highest if it allows the team to continue in the same or a similar field).

[KnSk10] also mentions the use of swim lanes, but here swim lanes are assigned to products instead of teams (Figure 13), so that each product is tracked in one horizontal swim lane. Hence, the focus is now set on products instead of teams (cf. Figure 12).

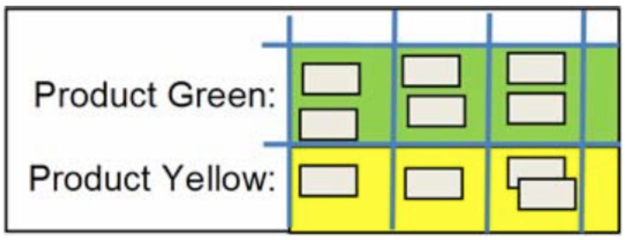


Figure 13: Product-based Swim Lanes [KnSk10]

[Lada08] prefers the product-based approach to the team-based one as it *"dedicates capacity to each work product type, without actually dedicating people"* (p.123, [Lada08]).

4.1.2 Scrum Program Management

The description of Scrum Program Management in this chapter is based on [Pich08].

Every team needs its own Scrum Master and Product Owner, even in a single multi-team project [Pich08]. Furthermore also Product Owner teams are needed, headed by a *Chief Product Owner*. These product owner teams may also include the chief architect as well as several representatives, such as marketing and sales. The Product Owner team itself may use Scrum to operate and have a (part time) Scrum Master.

To introduce program management to Scrum, the hierarchy can be laid out as pictured in Figure 14:

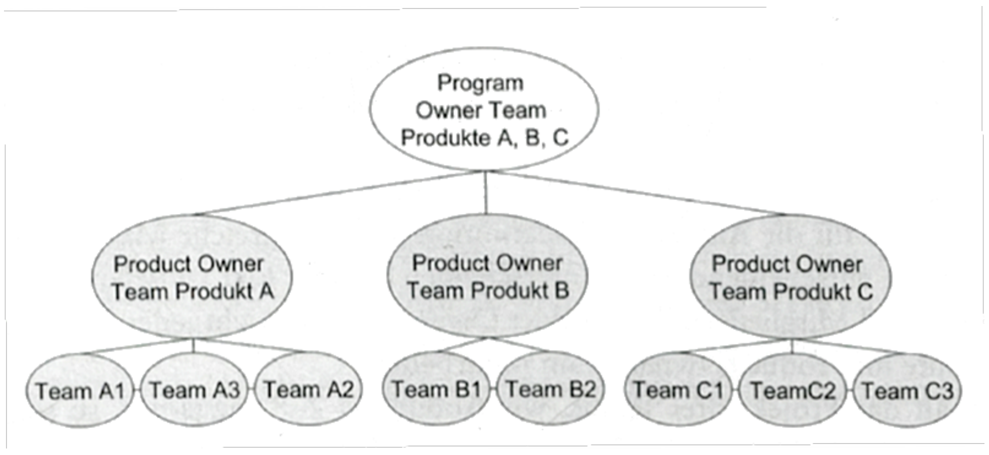

Program Owner Team Produkte A, B, C
Product Owner Team Produkt A
Product Owner Team Produkt B
Product Owner Team Produkt C
Team A1
Team A3
Team A2
Team B1
Team B2
Team C1
TeamC2
Team C3

Figure 14: Scrum Program Management [Pich08]

Dependencies between teams should be as small as possible [Pich08].

Component and Feature Teams

Teams may be organized as either *component teams* or *feature teams* [Pich08]. Component teams take responsibility for a subsystem, e.g. presentation, logic or the data layer. The dependencies are minimal but no component team can implement a feature on its own, as integration with other layers is needed.

Feature teams, on the other hand, are built vertically, i.e. they can work on all layers and are able to implement features on their own. The focus remains on end user requirements and thus on adding customer value. However, it is harder to maintain integrity and consistency of the software architecture with feature teams.

Most teams should be feature teams and component teams kept to a minimum for customer-centric development [Pich08].

(A)Synchronous Sprints

It is best to have synchronous Sprints within a project because coordination and planning is easier [Pich08]. But if one project is the supplier for another project, then it may be sensible to have asynchronous Sprints, i.e. have the supplier's Sprint start earlier. Hence the Product Owner of the supplied project can be part of the Sprint review of the supplying project to get an overview and plan his own Sprint accordingly.

Estimation

All teams should use the same base for estimation, e.g. *story points* (cf. [Cohn05]), and also agree on the same scale and semantics of points.

Lookahead Planning
Lookahead Planning has been developed by [Cohn05]. It describes the planning activities that arise when multiple Scrum teams are involved in a project. It assures that every team is working at full capacity by identifying dependencies between the teams [Pich08]. Lookahead Planning involves four steps (cf. [Pich08]):

1. Analyze requirements.
2. Anticipate dependencies between teams when working on different requirements (e.g. between component and feature teams).
3. Identify conflicts between feature teams and resolve them, e.g. by changing the order in which requirements are implemented.
4. Analyze each team's workload that has been generated by the precedent steps.

If one team is above its capacity after step 4, adjustments to the planning have to be made. Depending on size and complexity of the project, the steps may need to be executed several times until planning is optimal [Pich08].

Pipelining
Pipelining describes the creation of a product increment over the course of several Sprints [Larm04]. However, pipelining should only be used if absolutely necessary [Pich08]. Figure 15 shows the exemplary implementation of pipelining. The numbers denote user stories that are not finished within one Sprint. Instead, they are picked up by other teams (A and B in Figure 15) to be finished in the consecutive Sprint.

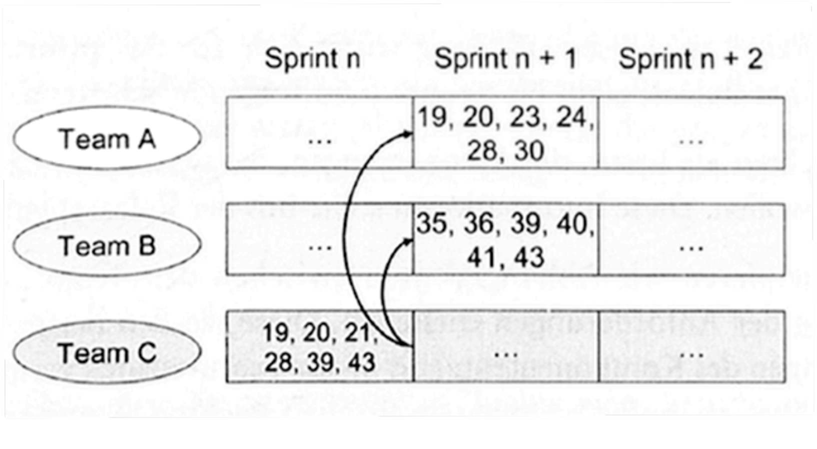


Figure 15: Scrum Pipelining [Pich08]

Pipelining can also be suitable to schedule test activities [Pich08], e.g. functional tests in the first Sprint and integration tests in the next one.

Meta Scrum
Meta Scrum helps to coordinate projects in a program that uses Scrum (cf. [Suth05]). It uses the concept of the *Scrum of Scrums*, which is a project-wide standup meeting that improves communication and coordination among several teams in big Scrum projects.

Each team sends one team member to the Scrum of Scrums meeting. However, the Scrum Master should not be sent there too often in order for the team to remain self-organizing [LaVo09].

The following questions need to be answered by each participant of the Scrum of Scrums (cf. [Pich08]):

1. What did your Scrum team do since the last Scrum of Scrums?
2. What will your Scrum team do until the next Scrum of Scrums?
3. Are there any impediments in your Scrum team's way?

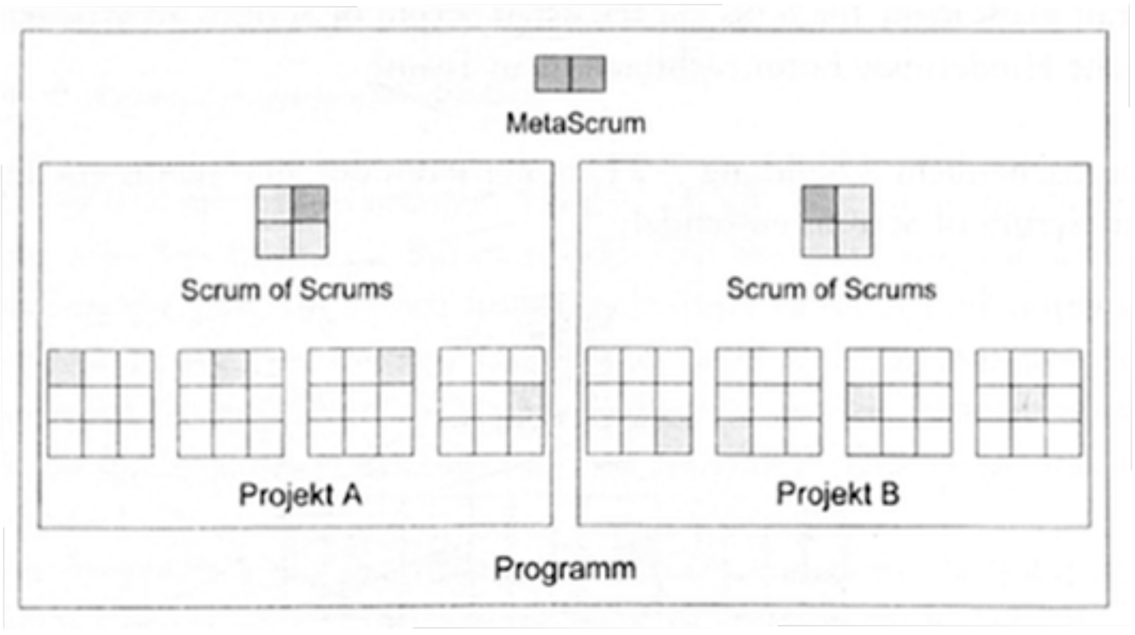


Figure 16: Meta Scrum [Pich08]

Figure 16 illustrates the Scrum of Scrums of projects A and B. It also shows the implementation of the Meta Scrum: Each Scrum of Scrums sends one participant to the Meta Scrum [Pich08]. Hence, the Meta Scrum adds the possibility for program (or inter-project) management. In contrast to the Scrum of Scrums, the Meta Scrum is usually not held daily [Pich08].

4.1.3 Scrum in a Multi-Project Environment

The description is based on [Glog11].

Neither Scrum nor traditional approaches were designed to deal with multi-project chaos [Glog11]. Scrum does not help to control project chaos. It only increases the transparency of the individual projects and thus facilitates the enterprise's decision-making and prioritization process, which is one of the most important issues according to Google CEO Larry Page:

“There are basically no companies that have good slow decisions. There are only companies that have good fast decisions.” (Google CEO Larry Page at Google Zeitgeist, Sept. 2011)

According to [Glog11] a company needs to maintain a *Project Backlog* that includes all projects. The CEO needs to appoint a full-time Product Owner for this Project Backlog to prioritize and update it. The teams then work on projects according to the enterprise-wide prioritization in the Project Backlog. Figure 17 illustrates the idea: High priority projects are on top of the Project Backlog.

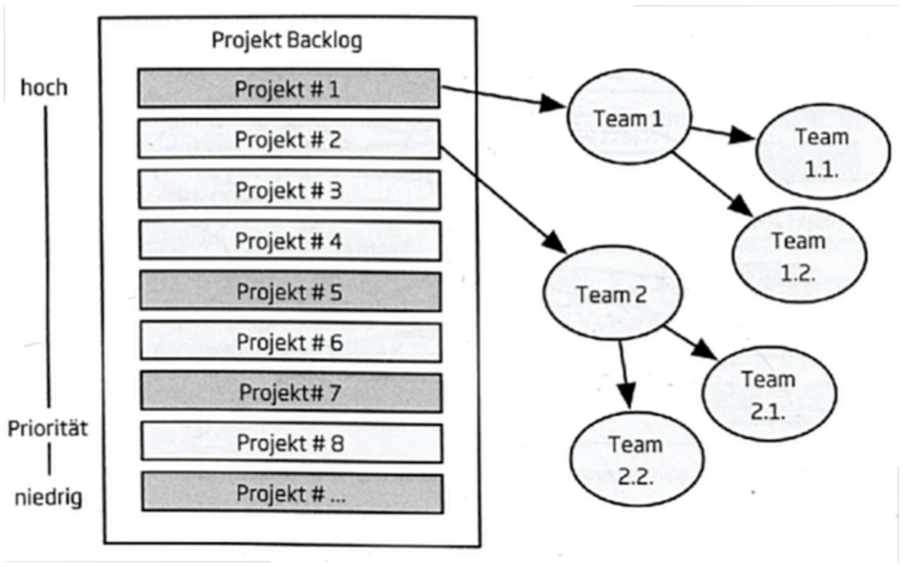


Figure 17: Project Backlog [Glog11]

Every company needs to find its own strategy for prioritizing this backlog. [Glog11] shares the following examples, among others:

- Ricardo Semler from Semco[13] uses democratic voting
- Apple’s Steve Jobs drastically narrowed apple’s product line to focus on a small number of products

The kind of strategy in use is of secondary importance because they all lead to the single most important concern: *focus* [Glog11]. Focus is the critical success factor in a multi-project environment. [Glog11] explains that with Scrum in a multi-project environment, focus is achieved in the following ways:

- Teams work on projects according to a prioritized enterprise-wide backlog.
- Prioritization of Enterprise Product Backlog is done by decision makers, e.g. by a Product Owner appointed by the CEO.
- As few projects as possible should be worked on simultaneously.

[13] http://www.semco.com.br/en/

Team Backlog

If teams need to work on several projects, [Glog11] suggests using a *Team Backlog.* Every Team has a Team Backlog that contains the team's work items (from several projects, see Figure 18). Hence the team's work is not steered by the Product Backlog any longer but by the Team Backlog.

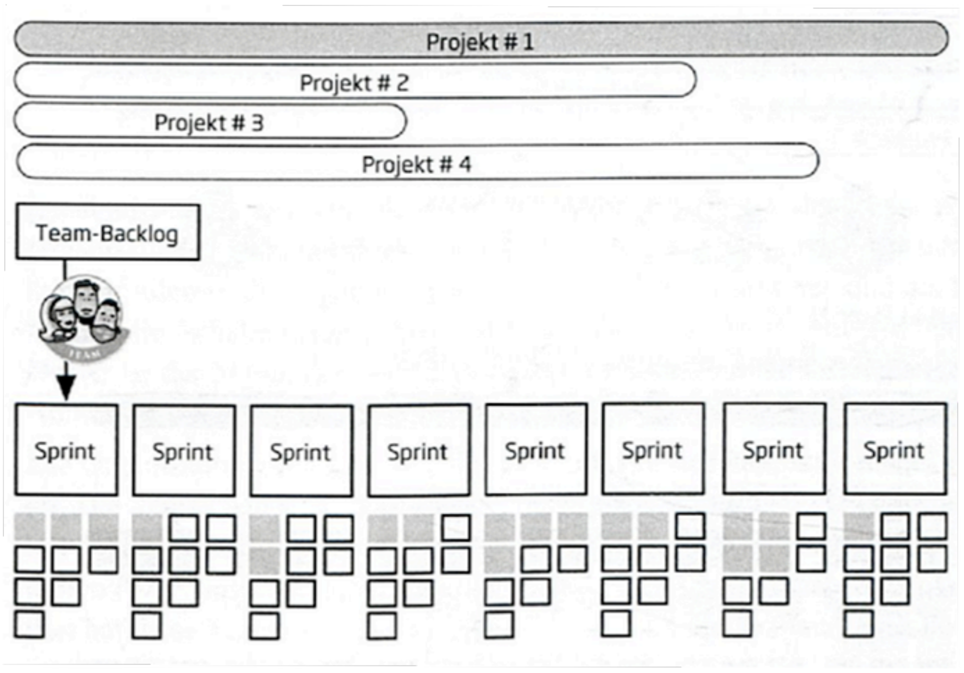


Figure 18: Team Backlog [Glog11]

However, this does not solve the problem of lacking focus [Glog11]. Eventually, a Project Backlog should be introduced to prioritize projects and have teams only work on one or two projects at the same time [Glog11].

4.1.4 Scaling Lean and Agile

The case study (cf. chapters 5 and 6) involves a medium-sized multi-team and multi-site project environment implementing a large-scale Scrum. Hence, this subchapter explores some organizational tools to deal with these circumstances as described by [LaVo09].

Long-lived Teams

The authors stress that long-lived teams are very important to improve a team's process and thus increase performance. [Katz82] showed the relationship of R&D[14] team performance and team longevity in his research (Figure 19).

[14] Research and Development

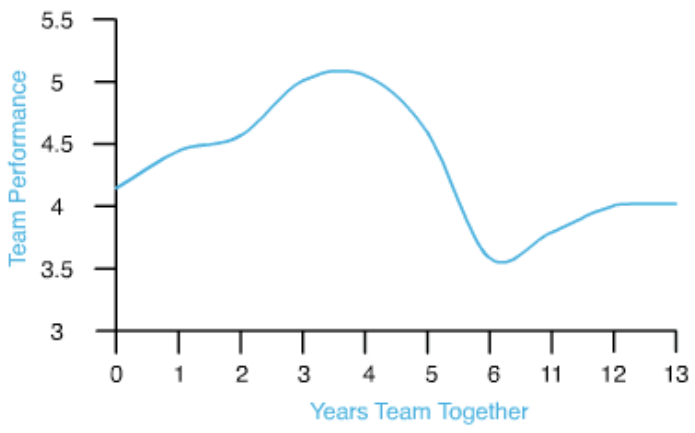


Figure 19: Team Performance over Time [LaVo09]

The research shows that a peak is reached after four years of working together [Katz82], followed by a drop in performance. Hence, team members should be rotated across teams after some time to create new insights [LaVo09].

Cross-Team Working Agreements

Multiple Teams need to agree on cross-team working agreements, including e.g. the definition of “done”. It is important that *all* teams own the working agreement so that it can be adapted as teams reflect, learn and improve.

Team Manages External Dependencies

MIT professor Deborah Ancona discovered in her research [AnBr07] that teams who manage across their boundaries, i.e. also have an external focus, outperform teams with solely an internal focus.

The organization needs to make clear that the teams themselves are responsible for co-ordinating their work with other teams [LaVo09].

Team Makes Decisions

Self-organizing teams make their own decisions. This can be hard for members used to a command-and-control environment. The result can be endless discussions without a decision. Therefore, the Scrum Master should help the team to learn how to make decisions, such as voting, consensus or “expert decides”. Most healthy teams apply a consensus-based method [KaLi07].

Overview: Teams in Large-scale Scrum

Teams need to be self-organizing, cross-functional, long-lived and dedicated. Figure 20 provides greater detail:

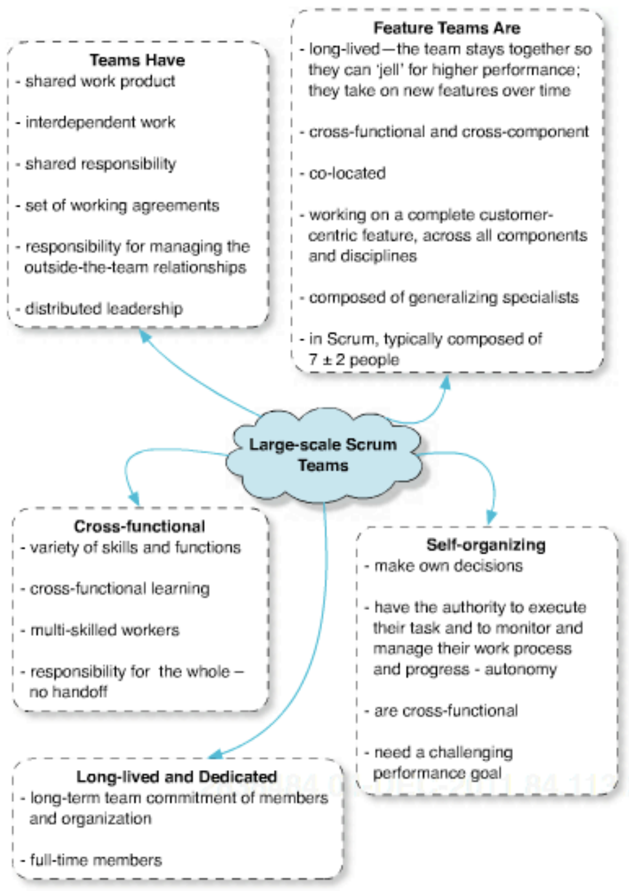

Teams Have
- shared work product
- interdependent work
- shared responsibility
- set of working agreements
- responsibility for managing the outside-the-team relationships
- distributed leadership
Feature Teams Are
- long-lived—the team stays together so they can 'jell' for higher performance; they take on new features over time
- cross-functional and cross-component
- co-located
- working on a complete customer-centric feature, across all components and disciplines
- composed of generalizing specialists
- in Scrum, typically composed of 7 ± 2 people
Large-scale Scrum Teams
Cross-functional
- variety of skills and functions
- cross-functional learning
- multi-skilled workers
- responsibility for the whole – no handoff
Self-organizing
- make own decisions
- have the authority to execute their task and to monitor and manage their work process and progress - autonomy
- are cross-functional
- need a challenging performance goal
Long-lived and Dedicated
- long-term team commitment of members and organization
- full-time members

Figure 20: Large-scale Scrum Teams [LaVo09]

Requirement Areas

[LaVo09] states that it is very difficult for the teams in a product with more than five to ten feature teams to work on the whole product and also for the Product Owner to work with so many teams. The solution presented is to work with *requirement areas*. Requirement areas are customer-centric categories of *Product Backlog items (PBI)* such that each PBI belongs to one requirement area. Figure 21 illustrates the idea:

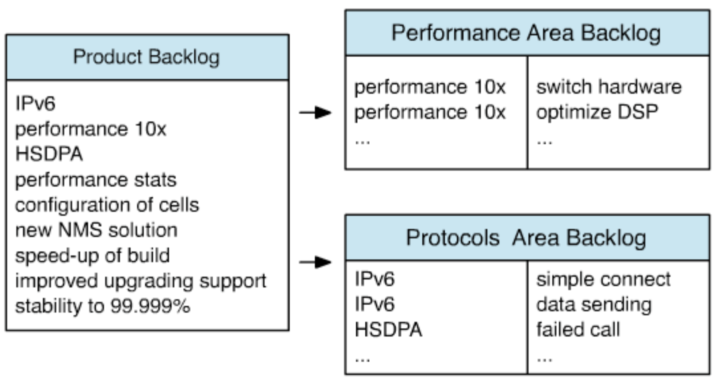


Figure 21: Requirement Area Backlogs [LaVo09]

In the example of Figure 21, the PBI "IPv6" belongs to the requirement area "Protocols", where it is divided into the finer-grained area backlog items "simple connect" and "data sending".

The advantage is that the Product Backlog remains manageable for the Product Owner because items are coarse-grained. Area Backlog items are prioritized by a separate person, the *Area Product Owner (APO)*. The APO acts as Product Owner in relation to the teams for that requirement area and is part of the *Product Owner Team* that is headed by the Product Owner. The APOs work as Product Owner with all teams in their requirement area, e.g. in iteration planning and reviews. Figure 22 gives an overview:

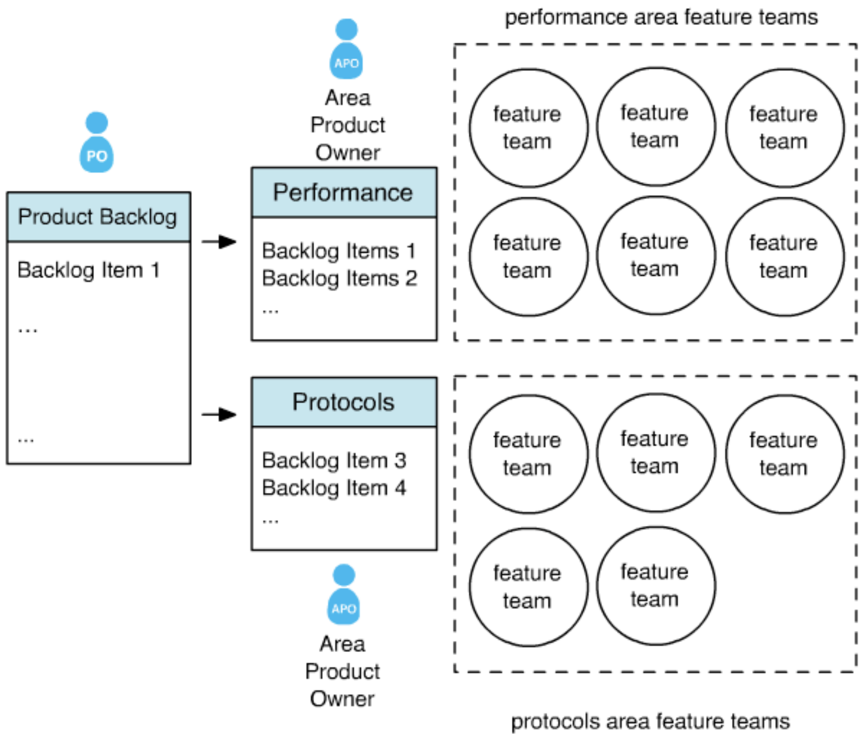


Figure 22: Feature Teams working on Requirement Areas [LaVo09]

Specialization in requirement areas is done from a customer's perspective (e.g. "network management area") instead of a component-based approach (e.g. "adaptations framework subsystem"). Teams now share the same vocabulary with customers and can be directly involved in the clarification of requirements. As a side effect, teams usually still specialize in a few subsystems because working in one requirement area usually involves working with the same family of subsystems. This also speeds up development without the restriction of component ownership.

As priorities change, requirement areas can be added or removed and *whole* teams be moved between requirement areas.

Merged Product Backlog for a Set of Products

Prioritization in a company is often based on products, not features. The problem with this point of view is described by [LaVo09]:

"Coarse-grained product prioritization leads to a local optimization in which the low-priority items of a high-priority product are implemented instead of the ***essential*** *items of a lower-priority product."* [15] (p.256, [LaVo09])

The solution provided by [LaVo09] is to merge the Product Backlogs of different products into one Product Backlog for the whole company. This works especially well when the products share a common codebase. The prioritization aspect in a company is then reduced to prioritizing one Scrum Product Backlog on a feature level. In bigger organizations scaling-up the Product Backlog with requirement areas may be an option, although the authors cannot back up this suggestions with an experience report.

If a project is too small for one Scrum feature team (approx. 7 people), then it is possible to merge several small products' backlogs into a combined Product Backlog. Again, this works best if the small products share a common codebase.

Large-Scale Scrum Framework

[LaVo09] presents two large-scale Scrum frameworks, but only one framework will be described here, which scales Scrum to up to ten Scrum teams and one Product Owner. The other framework scales Scrum to many hundreds or even thousands of people in a product group, which is beyond the scope of the case study (cf. chapters 5 and 6) and this thesis.

The challenge with large-scale Scrum is that the Product Owner cannot effectively interact with many teams nor can he grasp an overview of the entire product any longer [LaVo09]. Thus the burden of the Product Owner needs to be minimized by having the team interact with real customers. The Product Owner does not need to be involved in low-level details and should be able to focus on product management [LaVo09]. In any case there should be only one Product Owner, whether there are five or fifty Scrum teams in the product group.

A feature team should have little need to interact or coordinate with other feature teams except for the integration of code. Each team has its own Sprint Backlog. The product increment is not per team: all teams need to integrate their output into one increment [LaVo09]. If participants have a good understanding of Scrum, the Scrum Master may serve several teams instead of only one.

[15] Emphasis by authors of [LaVo09]

The Sprint Planning is now held with the Product Owner and all teams' members if they can fit in one room. Otherwise each team has to send representatives. Backlog items are now offered to whole teams instead of individuals. In turn whole teams can now volunteer for features. Afterwards each team holds its own *part two* of the Sprint Planning where it creates its Sprint Backlog.

The Daily Scrum remains the usual meeting, but sometimes it may be useful to know what other teams are doing by sending a scout to observe another team's Daily Scrum (if the meeting schedule allows).

Product Backlog Refinement should be done once each Sprint in a several hour workshop, so that all teams or representatives have the chance to get in touch with the Product Owner [LaVo09]. Properly executed, it will speed up the Sprint Planning session drastically [LaVo09].

The Sprint Review is the normal Scrum Event including all team members or team representatives, but the Retrospective is held individually (per team). Optionally, a Joint Retrospective meeting can be established with a few representatives of each team and the Scrum Masters to discuss systemic impediments to an effective Scrum process [LaVo09].

There are also alternatives for the *Scrum of Scrums* meeting as *"healthy self-managing teams are* ***themselves*** *responsible for their coordination and communication with other groups"* [16] (p.301, [LaVo09]) The need for a Scrum of Scrums may also indicate that the feature teams are not really cross-functional and cross-component, i.e. they cannot work independently or continuous integration is not done properly [LaVo09]. If however a Scrum of Scrums is in place, it is very important that team representatives to attend these meetings are rotated such that no artificial management layer can be established. Other types of coordination meetings may work better, such as the *Open Space Technology*[17] meeting or *Town Hall*[18] meetings [LaVo09].

Lastly, all teams need to share a common *definition of done (DoD)* that can be achieved by all teams in the group. Continuous Integration is also especially important for multi-team development and an absolute foundation for a successful large-scale Scrum [LaVo09].

[16] Emphasis by authors of [LaVo09]

[17] http://en.wikipedia.org/wiki/Open_Space_Technology, accessed 31/10/2011

[18] http://en.wikipedia.org/wiki/Town_hall_meeting, accessed 31/10/2011

4.2 Known Practices in Publications

The publications presented in this chapter have been selected in *2.4 Related Work* for closer examination.

4.2.1 Enterprise Scrum

[Gree10] introduces an enterprise-wide Scrum approach. The idea came up because Daniel Greening was not able to answer the following two questions by Jeff Sutherland[19] satisfactorily (cf. [Gree10]):

1. What is your current velocity?
2. What are the blockers impeding your progress, and what are you doing about them?

If a CEO is unable to answer the first question, he cannot balance features against effort [Gree10]. The second question shows if there is sufficient communication between engineering and top management [Gree10]. The problem identified was the following: *"Bottom-up techniques do not produce the "sticky-note" simplicity of Scrum at large scales."* (p.1, [Gree10]). So [Gree10] developed a new top-down approach, the *Enterprise Scrum*, with the following parameters:

- A quarterly Enterprise Scrum Sprint including planning, work, production of releasable products and a retrospective
- A weekly Standup meeting with Product Owners, Scrum Masters and the team leads for the discussion of progress, short-term planning and finding collaborative solutions for impediments

Architects and team leads were used to estimate large projects with *Enterprise Story Points* for the quarterly Enterprise Sprint. Individual Projects still use regular 1-4 week Sprints.

Figure 23 gives an overview of the components and their relations.

[19] One of the two inventors of Scrum

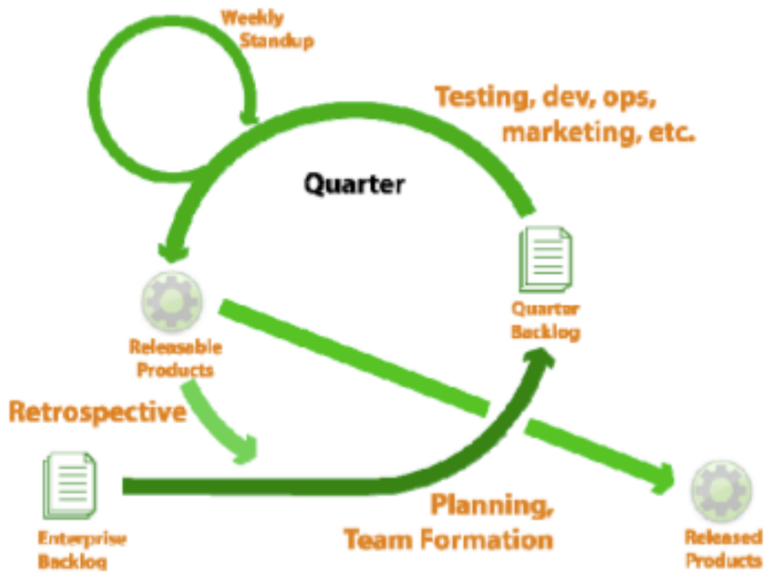


Figure 23: The Enterprise Scrum [Gree10]

The *Enterprise Scrum Team* members are (cf. [Gree10]):

- The Enterprise Product Owner (filled by the Director of Product Management in [Gree10]) is responsible for coordination, value research and prioritization.
- The Vice President of Engineering (no role name provided) manages engineering, including effort management, hiring, firing, budgeting and day-to-day operations.
- The Enterprise Scrum Master enforces the Enterprise Scrum process.
- Further members include several Product Managers, Project Scrum Masters, Tech Leads, the User Experience Manager and Operations Director.

[Gree10] also describes the use of an *Enterprise Backlog*: *"Each EBI [Enterprise Backlog Item] is a project that must be at least one Scrum team-month in effort (a Scrum team consists of the standard 5-9 people), and it must be feasible to finish an EBI in three months. In practice, this means an EBI must be between 1 and 9 team-months of effort (e.g. may require 3 teams for 3 months)."* (p.3, [Gree10]).

The bottom line is that top management has since been able to answer the two important questions: they know about the velocity the enterprise has each quarterly Sprint (in Enterprise Story Points) and about all the impediments and its solutions from the weekly Standup meeting. However, it is not possible to introduce this approach without the full support of the top management.

4.2.2 B-Scrum: Business Extension of the Scrum Framework

[KtLé09] argue that greater stakeholder's contribution to development efforts is needed. They advise to install a steering committee to *"maintain a common artifact that provides enough visibility on business expectations and sources of value"* (p.63, [KtLé09]).

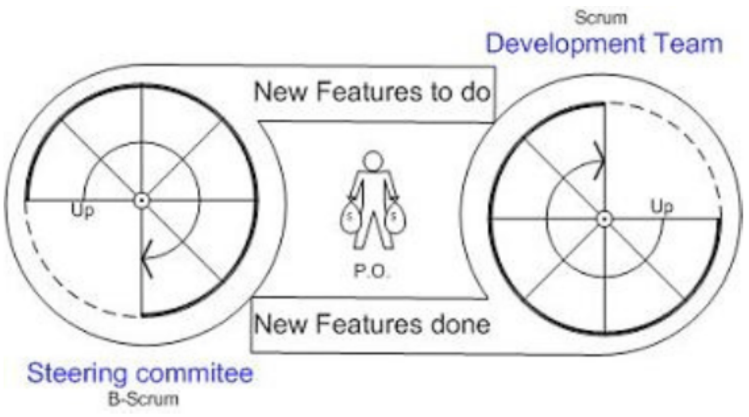


Figure 24: B-Scrum [KtLé09]

The Product Owner acts as the link between the steering committee and the development team (see Figure 24). Especially in large-scale development the Product Owner often lacks visibility to be properly involved in the development process.

The following reasons support the extension by a B-Scrum (Business Scrum) (cf. [KtLé09]):

1. Help the Product Owner understand the underpinning reasons behind the stakeholders' expectations and solve any conflict situations.
2. Help the Product Owner track the value progress and make decisions that reflect stakeholders' interests with regard to the evolving business context.
3. Help the Product Owner identify and report real business opportunities to higher management.
4. Help the Product Owner avoid chaotic software development due to scalability issues and miscommunication.

It becomes easier for the Product Owner to align prioritization with business goals. [KtLé09] advises to work with goals due to the following reason (cf. [KtLé09]):

- Stakeholders can express their expectations in terms of goals, i.e. a language that they are more familiar with
- Low level requirements such as user stories can be linked to strategic (business) goals
- With a goals map traceability, the development team can show progress in terms of business goal's satisfaction as well

- The stakeholders can prioritize the importance of business goals at different levels and think about possible IT solutions
- The most valuable goals are more likely to be developed first

In conclusion, the B-Scrum extension helps the Product Owner to prioritize according to stakeholder's expectations.

4.2.3 UX Component Team

[BuJe09] describes the installation of a separate User Experience (UX) Scrum team and its struggles and benefits. At first, it was not easy to set up a UX Team because the development team did not want to be separated from the UX [BuJe09]. However, the separate Product Backlog enabled the UX to work more efficiently. Moreover, the UX gained better transparency due to the UX Product Owner who could interact with the other parties and is the single point of contact of all UX issues. Additionally, UX team members attend the development team's daily meetings once or twice a week and a standing cross-functional meeting has also been set up at the same frequency to improve collaboration and communication.

Figure 25 illustrates the setup, stakeholders and their interactions.

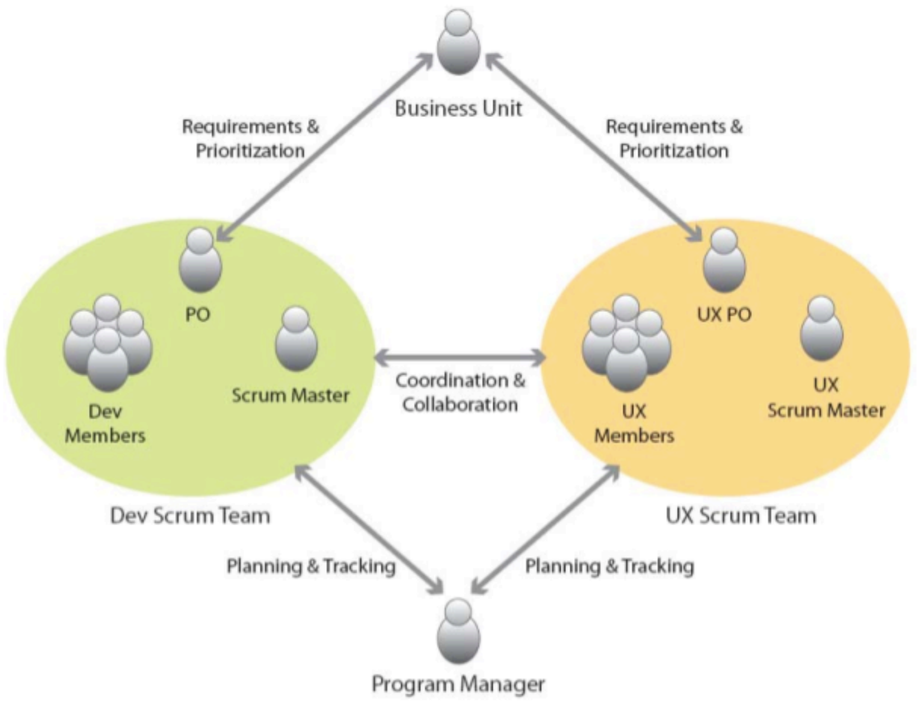


Figure 25: Scrum Setup including a UX Component Team [BuJe09]

This separate UX team can be regarded as a *component team* (see chapter 4.1.2). It should be noted that this case study has been conducted in the User Experience and Design unit of a big company. Hence, a separate UX team may not always be needed.

These are the lessons learned from [BuJe09]:

- Schedule UX teams to work one or two Sprints ahead of the development teams
- Incorporate quarterly design vision Sprints into the normal Sprint cycle
- Organize the UX team into a separate scrum team with its own product backlog and product owner
- Working one to two Sprints ahead of the development scrum teams has resulted in less churn and an improved work-life balance for the UX team members.
- Having a quarterly design vision Sprint has resulted in more cohesive and better quality designs.
- A UX product owner provides a primary point of contact to cross-functional teams and helps coordinate UX teams and deliverables.
- A separate UX product backlog helps the UX team assign resources to projects and track team capacity.
- A single UX team can support multiple development teams with less churn and better coordination.

As several authors have mentioned (cf. [Pich08], [LaVo09], among many others) component teams, such as this separate UX team, do not align well with Scrum as the approach is not customer-centric and many dependencies arise between teams. Feature teams are the favored approach because they are able to implement whole features by themselves with no or little inter-team dependencies.

4.2.4 Lean-Agile Project Management Office

[AuCu06] offers suggestions on how to combine agile project delivery at the project level with lean thinking at the portfolio level. One of the benefits is that agile projects can deliver financially meaningful interim results, which can impact the PMO's (Project Management Office) decision-making process.

Figure 26 shows the organizational structure proposed by [AuCu06]:

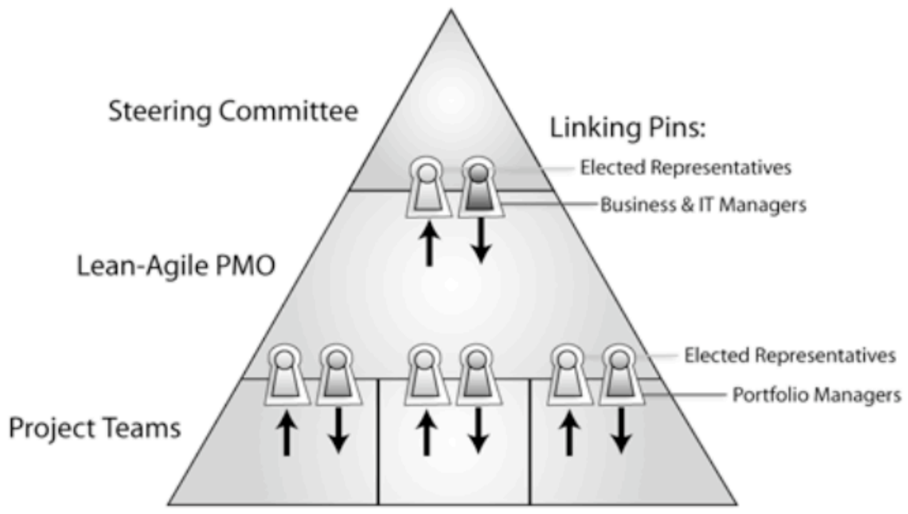


Figure 26: Lean-Agile PMO Organizational Structure [AuCu06]

The lean-agile PMO is connected via *linking pins* to the executive-level steering committee and project teams. Representatives are elected from lower to higher levels: project teams elect representatives to the PMO and the PMO elects its representatives to the steering committee. Hence, there is a combination of representatives that are elected and assigned top-down to ensure that the PMO stays self-organizing and adaptive to change.

Traditional Project Portfolio Management focuses on local optimization while lean thinking strives to manage the whole and optimize globally across the value stream [AuCu06]. [LaVo09] agrees on the paradox of local optimization: A party, making the best decision for local optimization, sub-optimizes the overall system throughput.

The lean-agile PMO has the following operation principles (cf. [AuCu06]):

1. Align continuously
    - Communicate strategic intent
    - Ideate against strategy
    - Measure business results
    - Make ranking and selection open and visible
    - Reprioritize regularly
2. Manage project throughput
    - Practice Lean project scheduling
    - Reduce project inventory/work in process
3. Manage system constraints
    - Identify organizational and process constraints
    - Optimize and elevate the constraints

Figure 27 illustrates *Lean Process Scheduling*. The idea behind is that [AuCu06] advises agile core team member not to work on more than one project. Therefore, the lean-agile PMO can only start as many projects as there are available teams.

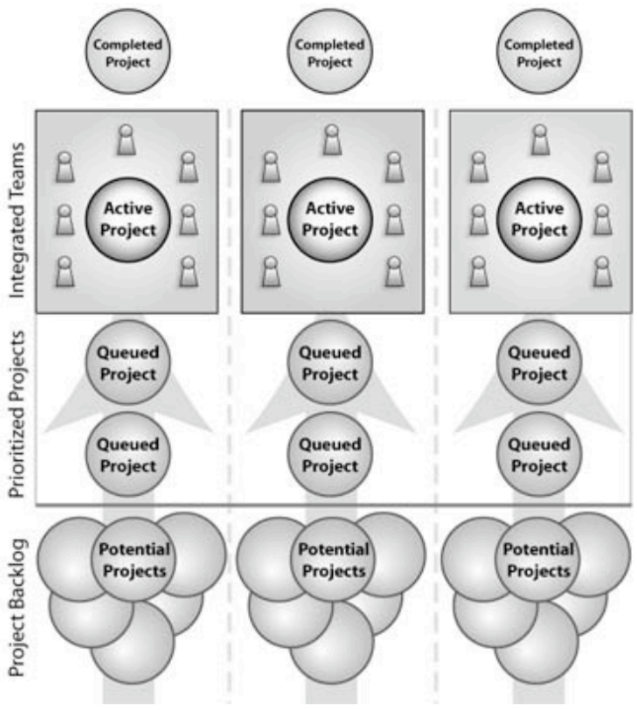


Figure 27: Lean-Agile Project Flow [AuCu06]

As Figure 28 shows, projects can then be visualized and tracked using a regular Kanban board with WiP[20] limits and lead time (cf. chapter 2.3), but on a portfolio level. Bigger projects should be split into MMFs (minimum marketable feature, cf. [DeCl03]), which is the minimum aggregation of product features that provides value to the customer [AuCu06].

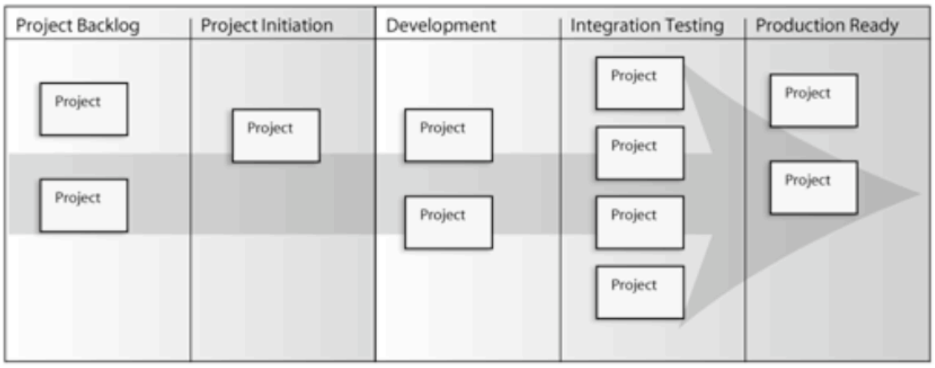


Figure 28: Lean-Agile Project Kanban Board [AuCu06]

[20] Work in Progress

As far as the operation of the lean-agile PMO is concerned, it should use a Scrum process as well, including month-long Sprints with review and planning meetings, daily standups, the maintenance of a prioritized long-term backlog for release planning meetings and a short-term one for Sprint planning and review sessions [AuCu06]. Regular progress tracking and monitoring via e.g. burndown charts is also suggested.

With the new lean-agile PMO approach, [AuCu06] tries to avoid a structure that will lead to traditional top-down and command-and-control management by combining elected representatives with top-down assigned ones in the hierarchy. The application of lean principles at the portfolio management level, such as managing throughput by reducing project inventory and improving the project completion rate, help to globally optimize the financial performance [AuCu06].

However, [LaVo09] criticize this and similar ideas strongly because neither a traditional PMO nor an “agile” PMO is needed. With Scrum most project management responsibilities are moved into the teams and that all other responsibilities (releases, schedules, etc.) should be moved to *Product Owner Teams* (see chapter 4.1.2) [LaVo09].

5 Case Study: Lean-Agile Multi-Project Management in Practice

During the case study an approach to lean-agile multi-project management is analyzed (in this chapter), evaluated (cf. chapter 6) and improved (cf. chapter 6). Chapter 5.1 describes the project environment and the setting that the case study is conducted in, followed by known strengths (chapter 5.2) and problems (chapter 5.3), as identified by employees during interviews or meetings.

5.1 Status Quo

The case study is set in a spatially separated development environment, which involves five parties:

- 1 Main Supplier
- 1 Additional Supplier
- 3 Customers

Figure 29 illustrates the interaction of the parties. The customers only interact with the main supplier. The main supplier had to hire new personal to cope with customers' demands. The additional supplier does not directly interact with customers.

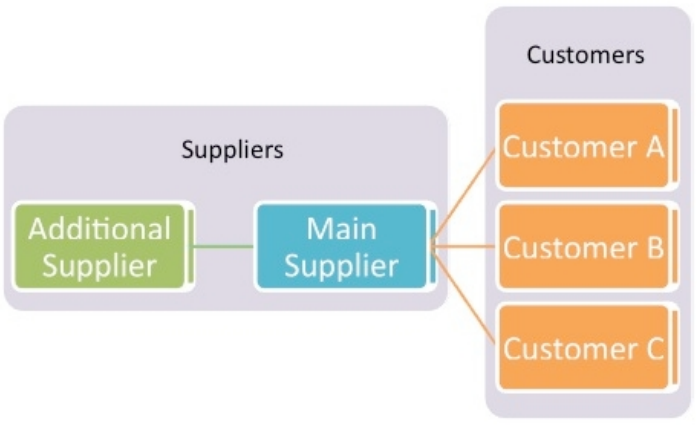


Figure 29: The Parties Involved

Multi-Site Environment

The product development takes place on two different sites. The distance between the main supplier and the additional supplier is about 300 kilometers. Also, the two suppliers are not part of the same company, so inter-company related issues also have to be expected. More information on the suppliers is provided in chapter 5.1.1.

Multi-Project Environment

The case study involves the development of software for three customers. Existing software is improved and adapted, resulting in three different products that share the same codebase. These products are treated as three different projects, hence the multi-project environment.

More information on the products and the projects environment is provided in chapter 5.1.2.

5.1.1 Suppliers

Both suppliers are using the agile process Scrum. Overall 30 people are working on the expansion of three products. Three inter-company Scrum teams have been formed based on shared requirement areas among all products. There is no communication between the three original customers and the additional supplier (cf. Figure 29). The Scrum implementation is described in detail in chapter 5.1.2.

Main Supplier

The main supplier's staff for these products includes 20 people, consisting of:

- 11 Software Developers
- 3 Software Testers
- 3 Scrum Masters/Developers
- 3 Product Owners

The main supplier has started development on the products in May 2011.

Additional Supplier

The additional supplier's staff for these products includes 10 people, consisting of:

- 6 Software Developers
- 2 Software Testers
- 2 Scrum Masters/Developers
- 0 Product Owners[21]

The additional supplier joined the main supplier for co-development in August 2011. Most weeks one of the Scrum Masters travels to the main supplier's premises for a couple of days per week to gain new insight and updates. The Scrum Masters noted here do not fully comply with the actual role in the truest sense because the official roles are assigned to staff at the main supplier. But they can be regarded as an extension of the "real" Scrum Master to support the process at the additional supplier's premises.

[21] All Product Owners are with the main supplier

In November a team-building event was held to get to know each other better.

5.1.2 Implementation of Scrum

As has been outlined in the beginning of chapter 5.1, software development is based in two different suppliers' locations with a distance of 300 kilometers. This poses several constraints to the Scrum process that need to be dealt with. A presentation of the status quo follows. Known strengths and problems are discussed in separate chapters (cf. chapters 5.2 and 5.3, respectively).

At the time of analysis in October 2011 development on all three products has started. Initially, development was conducted in one big Scrum Team consisting of all 30 people, placed in two different locations. The following meetings have been established:

- Separate internal Daily Scrums held at the main supplier and the additional supplier in the morning
- Joint Daily Scrums via (video) telephone conference calls
- Two-tiered Sprint Planning
- Joint Sprint Review
- Joint monthly Retrospective

The Sprint length is set to two weeks. Planning is done for whole months, i.e. two Sprints. The main and the additional supplier work together in the same Sprint.

In November 2011, the big team has been split into three smaller ones, consisting of:

- 1 Scrum Master
- 2-3 Developers of the Main Supplier
- 2-3 Developers of the Additional Supplier
- 1-2 Software Tester of either one of the suppliers

Figure 30 illustrates the internal division of the two suppliers within one Scrum team, i.e. the main supplier on the left side and the additional supplier on the right side.

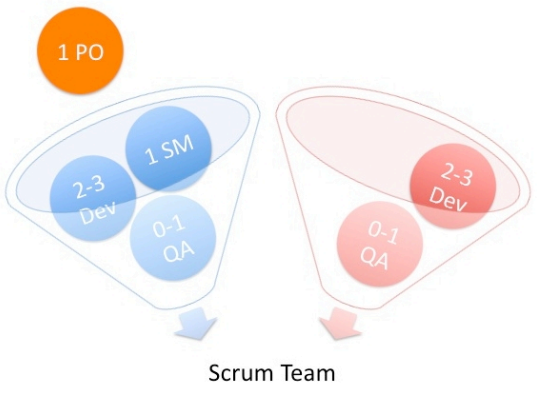


Figure 30: Composition of a Scrum Team

The three Scrum teams have been formed according to logical units, i.e. requirement areas, across all three products (in contrast to one team per product/customer). All Scrum Masters and Product Owners are based on the main supplier's side. The meetings have also been adjusted and are described in the following in greater detail.

Joint Daily Scrum

In October 2011 the Daily Scrums have been organized as pictured in Figure 31. Before that the Joint Daily Scrum has been one meeting per day, but it had been split into three separate meetings because one meeting with 30 people on two separate locations was simply not manageable. Two of them were roughly formed around requirement areas, the third one was dedicated to quality assurance. The meeting schedule was as follows:

10:30 Internal Dailies
13:00 Requirement Area A Daily
13:15 Requirement Area B Daily
13:45 Quality Assurance Daily

The internal Daily Scrums are held in the morning before the joined ones with all team members, i.e. 20 people on the main supplier's side and 10 people on the additional supplier's side respectively.

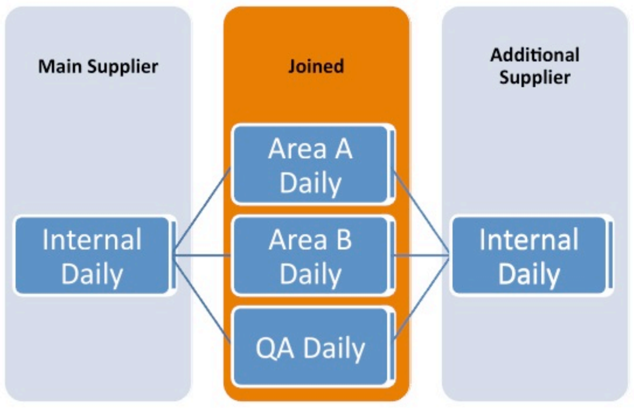


Figure 31: Former Division of Daily Scrums

Usually a team member is only participating in *one* of the joint dailies, except for quality assurance members who attend the requirement area daily and the QA daily as well.

This setting changed once the big Scrum team has been split into three smaller teams in November 2011. Since then, each of the three Scrum teams has its own Daily Scrum meeting. Additionally the software testers of all three teams meet in a separate *QA Scrum* after the usual Daily Scrum to coordinate quality assurance issues. The division of three Product Owners for three products/customers remains intact.

Scrum of Scrums

With the split into three Scrum teams in November 2011, a daily *Scrum of Scrums* meeting (cf. chapter 4.1.2) has been established as well. As Figure 32 shows, all teams send their Scrum Master (SM) to the Scrum of Scrums. Additionally the three Product Owners (PO) participate in the meeting to keep track of the progress and coordination between the teams. The additional supplier is not part of the Scrum of Scrums by choice due to the argument that coordination is easier to do in person (i.e. at the main supplier's premises).

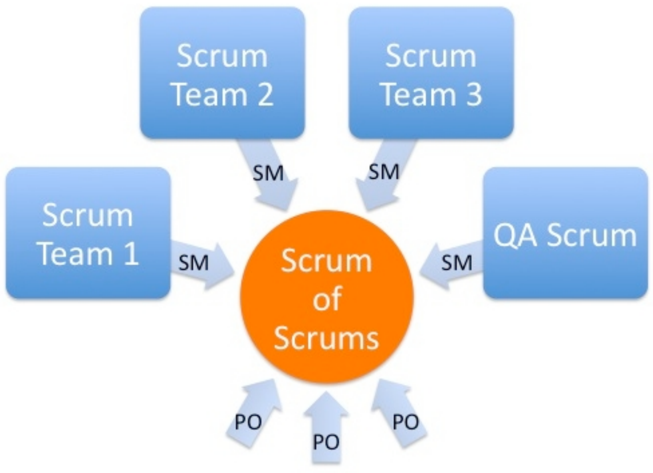


Figure 32: Scrum of Scrums

Two-tiered Sprint Planning

Planning covers one month, i.e. two Sprints. Then, more or less 50% of the planned user stories get assigned to each Sprint.

It is a two-tiered process. At first planning is done at the main supplier's side with one of the two Scrum Masters of the additional supplier present. The Scrum teams decide, which of the prioritized user stories in the Product Backlogs they want to implement in the next two Sprints. Additionally, user stories are assigned to the additional supplier. Estimation is then done using *Planning Poker* (cf. [Gren02]).

The second level planning continues at the additional supplier's side: The Scrum Master returns from the main supplier with user stories for the additional supplier. At this point, the main supplier has already estimated the user stories. The user stories are then put on the white board (cf. Figure 33) and divided into several tasks during the additional supplier's planning meeting. The team members then volunteer for a certain task until all tasks are assigned. When a team member accepts a task, he adjusts the original estimation of the main supplier to his own.

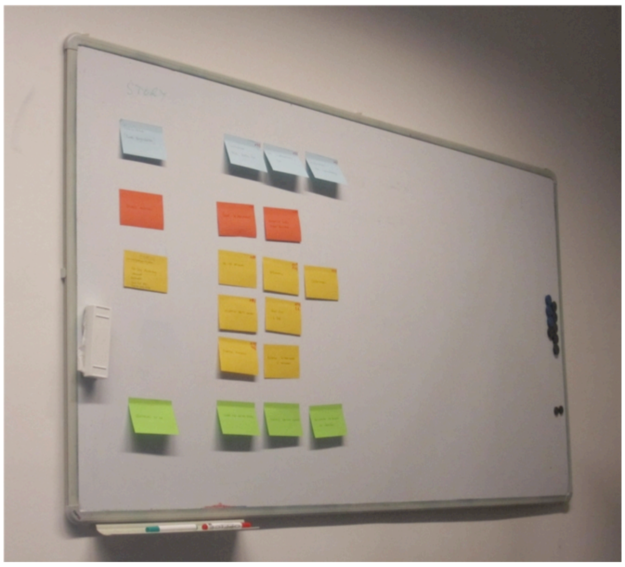


Figure 33: Sprint Planning Whiteboard

The outcome of the meeting, i.e. the updated estimations and task assignments are then kept in a planning spreadsheet that is shared with the main supplier. Estimations are done in hours on both sides.

Joint Sprint Review

The Sprint Review is held jointly after each Sprint. It is, however, primarily held at the main supplier, but the additional supplier joins via a video conference call. Figure 34 shows part of the setup for this call, which consists of a monitor, a webcam, speakers and a microphone. Additionally, one of the Scrum Masters is present at the main supplier's premises to serve as a communication hub. The rest of the additional supplier's team is mainly observing the review, but can raise questions and/or concerns when necessary.

The review consists of demonstrations and discussions about different areas of the current product increment and takes about two hours. All three Scrum teams attend this meeting, i.e. about 20 people on the main supplier's side (plus one Scrum Master from the additional supplier) and 9 people from the additional supplier joining in via a video conference call.

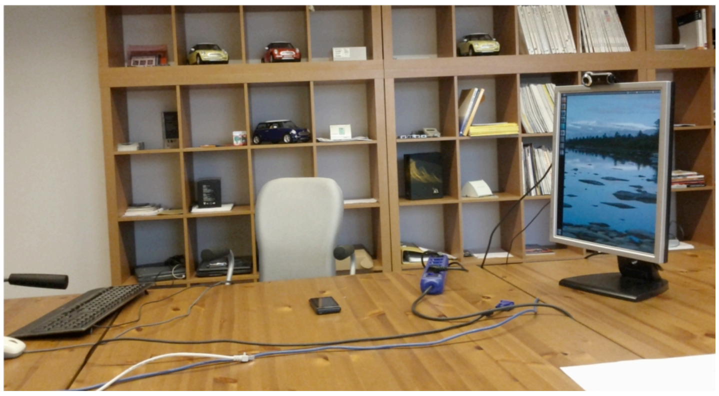


Figure 34: Video Conference Setup

Joint Sprint Retrospective

The Sprint Retrospective is held monthly, i.e. after two Sprints, with the same setup (cf. Figure 34): the additional suppliers' team joins the meeting via a video conference call and one Scrum Master is present at the main suppliers' office.

The Retrospective is divided into six steps:

1. Individual, overall evaluation of the last Sprint from good to bad on a 15-part scale (cf. Figure 37 right flipchart).
2. Evaluation and discussion of the measures taken since the last Retrospective.
3. Every participant writes three concerns (either positive or negative) on paper and puts them on the flipchart, shortly presenting each while doing so (cf. Figure 37 left flipchart).
4. The individual concerns from step 3 are clustered to topics.
5. Every participant assigns three points to one or more of the clustered topics according to his personal weighting.
6. Measures for the top three topics are discussed that will be implemented during the next Sprint.

Product/Sprint Backlog

Each Product Owner maintains a Product Backlog on the main supplier's side for his product. The additional supplier currently does not have access to the Product Backlog, but works with the Sprint Backlog only, which is the planning spreadsheet (also see "Two-tiered Sprint Planning" earlier in this chapter).

Scrum Board

Both the main supplier and the additional supplier are using paper Scrum boards. Each Scrum team is using one board. Since the two suppliers are based at different locations, six boards would be needed, but the additional supplier currently only uses one for all three teams, i.e. four boards are in use in total. Figure 35 shows the Scrum board of the additional supplier. The board is divided into the following columns:

User Stories ⇒ TO DOs ⇒ In Progress ⇒ Review ⇒ Done

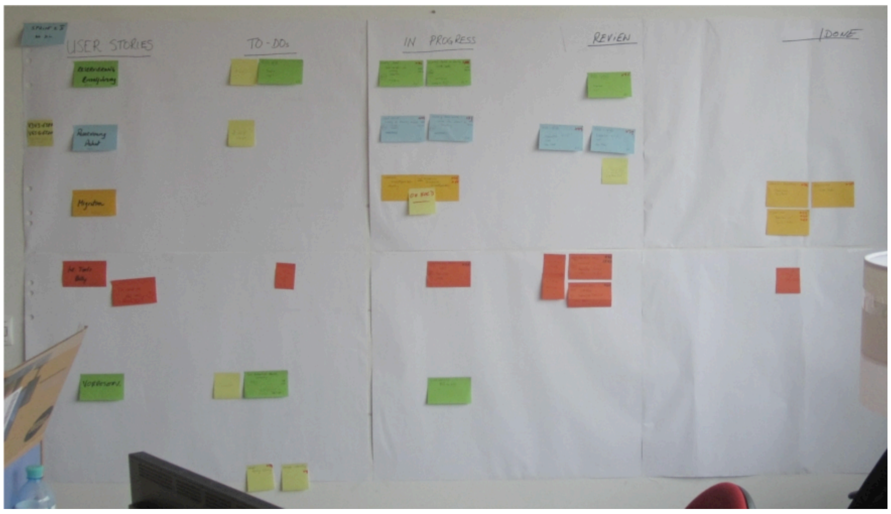


Figure 35: Paper Scrum Board

The first column *User Stories* contains user stories from the backlog. Sticky notes of the same color are then used to break the user stories into smaller tasks that run through the workflow *To Do*, *In Progress*, *Review* and *Done*. The column *Review* denotes the tasks being reviewed by a colleague (at any supplier's side), whereas tasks in *Done* are really closed and done.

Tasks on the Scrum board are also marked with the issue tracking number of the electronic tool that is in use (also see end of this chapter "Means of Communication"). The Scrum boards of both suppliers are synchronized every day during the Daily Scrums (for each team).

Burndown Chart

The Burndown charts are drawn and updated on paper at the main supplier's side only (one per team). The additional supplier does not operate a Burndown chart on its own, but the main supplier includes the additional suppliers' tasks into its Burndown.

The Burndown chart is not based on hours, but solely on tasks (without any kind of complexity neither in story points nor in hours). The argumentation for this procedure in one of the interviews was that tasks amount to the same hours on average and hence "burning down" tasks (instead of hours or story points) is sufficient.

Behavior Driven Development (BDD)
The suppliers develop software using *Behavior Driven Development* (cf. [Nort06]), which is an extension to *Test Driven Development* (cf. [Beck03]). The workflow applied in the case study is pictured in Figure 36.

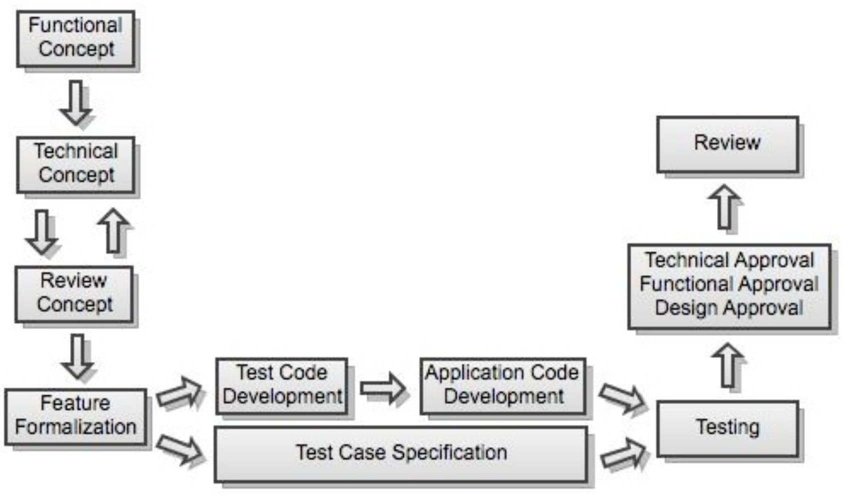


Figure 36: BDD Workflow

This has been the workflow of the main supplier in past projects and the additional supplier is adapting to it. The additional supplier acts as an extension of the main supplier's Scrum team and as such is involved in all steps of the workflow except for the various approvals and the review at the end, for which the Product Owner is needed.

Means of Communication
The main means of communication between the two suppliers are joint meetings (via video conference calls mostly), telephone calls, screen sharing sessions (to discuss individual concerns) as well as emails. Lastly, one of the Scrum Masters travels to the main supplier each week to serve as a communication hub and clarify issues in person.

Burndown charts and Scrum boards are drawn on paper and thus hard to share, but the following electronic tools are in use:

- Bug/Issue Tracking System
- Time Management Tool
- Project Management Tool
- Software Versioning and Revision Control System
- Planning Spreadsheets

The collaboration between tools works as follows: Every user story is divided into smaller tasks, which are then entered into the time management tool, so that team members can book working hours on these tasks. They are also entered into the bug/issue tracking system and are assigned to the responsible person. The task number is the main identifier in the time management tool, the bug/issue tracking system and also on the paper Scrum board.

The project management tool can issue reports, but has no real compatibility for Scrum. Developed software is checked into a remote repository using a software versioning and revision control system that both suppliers can access.

The main supplier also runs a database for requirements, which is solely used for the interaction between the main supplier and the customers. The main supplier then extracts parts of the requirements into the planning spreadsheet that is later updated by the additional supplier (cf. "Two-tiered Sprint Planning" earlier in this chapter).

5.2 Known Strengths

This chapter presents known strengths of the Scrum implementation, as employees have mentioned them during interviews or meetings, especially during the Retrospective.

The board on the right-hand side of Figure 37 shows the general evaluation of the previous months May to October 2011. During the beginning of each Retrospective, each team member puts one point on the board's scale from good (laughing smiley face) to bad (sad smiley face), i.e. one point equals the evaluation of one individual.

Figure 37 indicates that team members were very disappointed with the September Sprint, but that the measures taken improved the overall satisfaction in the consecutive October Sprint:

- Communication between the main supplier and the additional one in general
- Pointing out changes of use cases to the additional supplier
- Fewer broken builds checked into the remote repository

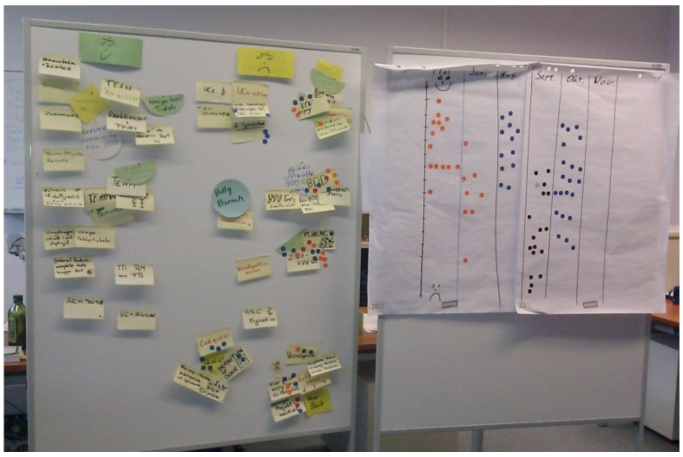


Figure 37: Sprint Evaluation in Retrospective

Team members have identified the following strengths for the month October.

Improved Communication and Collaboration

This point addresses recent problems with finding proper means of communication between the main supplier and the additional one that complicated the collaboration of the two parties. According to the Retrospective, these issues have been improved (cf. Figure 37), yet not completely resolved.

More precisely, the following improvements have been noted regarding the collaboration of the two suppliers:

- Improved coordination and time management
- Good reaction time in both directions
- Frequent visits of the Scrum Masters to the main supplier's premises facilitate communication

Continuous Improvement

The additional supplier joined the main supplier in August 2011 for co-development. Hence the Scrum process is adapted continuously to cope with issues that arise in this multi-site and multi-product environment. This continuous improvement is still at its early stages, but the process is improved with each Sprint.

The cornerstones for the continuous improvement are (as identified during Retrospective):

- Willingness and commitment to change and improve
- Good working atmosphere and employee attitude
- Highly motivated people
- Team work

The willingness to change and improve is also shown by the fact that the main supplier has hired consultants to accompany a new Scrum project. The consultants also glanced at the current project situation and proposed two changes:

- Divide the big Scrum team into several smaller ones
- Do not estimate tasks in man hours

Both changes have been put into practice with November 2011.

Requirement Areas

The original Scrum of 30 people has been split into three smaller ones that work on the same logical unit. These units can roughly be regarded as *requirement areas* that are proposed by [LaVo09] (cf. chapter 4.1.4). This is an important step in the transformation from one large team to several regular-sized feature teams (cf. chapter 4.1.2).

Physical Board
The physical board is in principle a good instrument, because it is a critical aspect of lean visual management that is diminished if an electronic tool is used [LaVo09]. It has higher psychological effect than an electronic one but it lacks automated reports and charts [Ande10]. The multi-site environment also constrains the usability of the physical board (cf. chapter 5.3.2 Lacking Tool-Support).

5.3 Known Problems

This chapter presents known problems with the Scrum implementation, as employees have mentioned them during the Retrospective (5.3.1) and interviews (5.3.2).

5.3.1 Retrospective

The Retrospective is held once a month (each two Sprints) to reflect on the Scrum process itself and discuss improvements and measurements. As such, it is a very important source to identify problems for the analysis in this thesis.

The flipchart on the left-hand side of Figure 37 (chapter 5.2) shows the following clustered problem categories taken from October's Retrospective:

- Use Cases
- VPN Access for Additional Supplier
- BDD Workflow
- Planning
- Code Quality
- Two-Week Sprint

Each team member had three points to prioritize problem clusters. 21 persons participated in the meeting; 62 points[22] have been assigned (cf. Figure 37 in chapter 5.2). The problem clusters have been renamed to describe the problems more properly.

19 Points: Continuous Stress in Two-Week Sprint
Many team members complained that they suffered from constant stress due to the following reasons:

- Workload too high in relation to available staff
- Changing between projects takes time
- Planning delay in general and also between the two suppliers
- Too little time to follow BDD workflow in a two-week Sprint

[22] Instead of 63 points, i.e. one point is missing or has not been assigned intentionally

Mostly the two-week Sprint iteration was held responsible for the stress, because planning often took too long or was incomplete and then team members had to cope with a lot of pressure and stress to still meet Sprint goals.

This was especially the case for the additional supplier, as coordination was even more difficult due to the different locations. So he would just start implementing because specifications were not ready and there was nothing else to do and then have to deal with major changes once they were ready.

The main supplier has noted during the Retrospective that support and assistance for the additional supplier still needs attention, although it has already improved since the last Retrospectives (cf. chapter 5.2).

One idea was to have a “3+1” iteration, meaning three weeks of planned work and a one-week buffer for integration, release and planning. However, the outcome of the discussion was that the two-week Sprint would not be abandoned because it is important to release working software every two weeks. Furthermore every team member is willing to cope with stress to some extent because software needs to be delivered. Nevertheless, process flaws need to be worked on to reduce the impact on the level of stress.

16 Points: Late Planning
This issue is similar to the first one and can be regarded as the root cause for the general dissatisfaction with a two-week iteration. Together these two aligned problems were assigned 35 of 62 points. So the overall problem can be denoted “Planning Delay increased Stress” and is the major issue of this Retrospective.

The problem is that the Sprint starts late for development since planning is late up to three or four days. One proposed solution is to plan continuously, so that the impact of planning on the beginning of the Sprints is reduced.

Another problem is the coordination between the main supplier (who does most of the planning) and the additional supplier. One proposal during the Retrospective was to brief the additional supplier better, i.e. to discuss use cases and specifications in detail each Sprint.

Lastly, the *Behavior-Driven Development* workflow is under-estimated in terms of effort, which also increases pressure. Most team members have not worked with BDD before, so there is a learning curve that needs to be considered in planning.

Planning is also hard in general because two different companies are working together. Hence, it is difficult to measure the Sprint Velocity and thus allow decent planning. However, velocity should get more accurate with each Sprint.

The measures for the next Sprints are to start early with planning/backlog refinement and include the additional supplier more into the planning process. Additionally, "seniors", i.e. people with BDD experience, should estimate BDD efforts.

8 Points: BDD Workflow
The following problems have been noted with regard to the *Behavior Driven Development* workflow:

- BDD Test Cases "broken"
- General misunderstanding of the meaning of BDD
- Bad Estimations (too low)
- No one in charge of BDD Feature Files

The measures for the next Sprint are to train every team member on both suppliers' sides to follow the BDD workflow correctly and fully comprehend the concept. To enforce learning, every team member should write at least one BDD feature file. Lastly, great care needs to be taken when changing "central" parts of code, because BDD test cases probably will not work any longer and need to be adjusted. Hence, responsibilities need to be assigned for BDD files.

8 Points: Code Quality
The problem with code quality was that broken builds have been checked into the remote repository, causing problems for fellow developers. In the next Sprints more care should be taken not to break builds, i.e. to check in code that cannot be compiled. Assigning responsibilities for modules should support this goal.

Another problem is that the teams do not share a common *Definition of Done* (DoD, see Cross-Team Working Agreements in chapter 4.1.4), which has to be established in the near future.

6 Points: Remote Access for Additional Supplier
The additional supplier needs a remote access to main supplier via VPN to be able to access his test environment.

The measure taken is to finally set up the proper infrastructure for the next Sprint.

5 Points: Uses Cases
The uses cases cover most of the specification and the general consensus was to keep it this way, but the overall quality has to be improved. The following complaints have been raised:

- Some Use Case are too sloppy or vague
- Contradictions in Use Cases
- Mixed languages (German and English)
- Frequent and/or unnoted Changes

5.3.2 Interviews

The following problems have been noted during interviews (both formal and informal).

Communication and Coordination between the two Suppliers
This issue has improved since the beginning of the collaboration of the two suppliers (cf. chapter 5.2), but still a lot of concerns have been raised during interviews on both sides.

One of the important points, yet relatively simple to overcome, is that the quality of the video conferences for all meetings is not good enough, i.e. a better setup is needed in general (better microphone, beamer instead of one monitor, etc.).

Another difficulty to overcome is that the main supplier is not used to co-develop with other suppliers. Hence, the Scrum implementation focuses on the main supplier, but lately tries to involve the additional supplier more and more into the process.

It has also been noted that it is hard for new developers to enter the projects because documentation is near to nonexistent. Many important details are discussed in meetings or emails and are thus hard to keep track of for newcomers.

Lastly, communication with regard to planning and new work packages from the main supplier to the additional one has been criticized, but this issue has already been discussed in the two main points of the Retrospective (cf. chapter 5.3.2).

Lacking Tool-Support
The tools as well as the process is rather main supplier centric, which presents two problems to the additional supplier:

1. The main supplier uses a great variety of tools.
2. The main supplier does not have electronic support for Scrum.

The first point causes the additional supplier to adapt to these tools that are not integrated very well, e.g. for the time management tool and the issue tracking tool, task IDs have to be transferred manually. Secondly, the tools do not support the Scrum process. The Burndown chart, e.g., is drawn on paper and thus not accessible to the additional supplier.

This causes a lot of overhead in communication to synchronize the different tools and get up-to-date information (especially on the additional supplier’s side).

Two-Tiered Sprint Planning
During the planning meeting at the additional supplier’s premises, it turned out that information about new work packages was too little to be able to properly split the new work/user stories into smaller tasks and estimate them.

The root cause for this problem is again late planning (cf. chapter 5.3.1).

6 Case Study: Evaluation of Lean-Agile Multi-Project Management

Chapter 5 analyzed a lean-agile multi-management approach in a practical setting, i.e. the status quo as well as known strengths and problems of the process implementation as described by employees.

This chapter continues with an in-depth evaluation (cf. chapter 6.1) of chapter 5's case study with regard to the concepts presented in this thesis (cf. chapters 2 to 4) and subsequently names identified problems in chapter 6.2.

Furthermore, chapter 6.4 proposes solutions to all problems (cf. chapter 6.2) in the form of improvements to the case study's multi-management approach. A short résumé to the case study is given in chapter 6.5 "Lessons Learned".

6.1 In-Depth Evaluation

In addition to known problems that have been identified by employees in interviews or meetings, this chapter identifies strengths as well as new unknown or overlooked problems with regard to the concepts presented in this thesis.

6.1.1 Comparison to Lean Values

For lean values, the case study's approach is compared to Kanban's *Recipe for Success* by [Ande10], which contains the following steps:

Focus on Quality
Both suppliers work according to *Behavior Driven Development*, i.e. there is great emphasis on the "test first" mentality. Every artifact, from feature file to code, is peer-reviewed before the final approval and review.

As chapter 5.3.1 shows, there are problems with the correct implementation of the BDD workflow and hence also with code quality that need to be improved.

Reduce Work-in-Progress
This value is partially implemented by dedicating feature teams to requirement areas. The teams can focus on the most important features in their area and thus reduce the work in progress.

Since the process in use is Scrum, work in progress is only limited by Sprints. However, there was a lack of respect for the beginning and end of the iterations, as the problems

Continuous Stress in Two-Week Sprint and *Late Planning* have pointed out in chapter 5.3.1.

Deliver Often

This lean value helps to improve quality by keeping complexity, i.e. the concurrent work in progress, to a minimum with frequent releases. The two suppliers build a new release every two weeks at the end of each Sprint, which is a commonly used time span for frequent deliveries.

Balance Demand against Throughput

This value is not met properly as the problem *Continuous Stress in Two-Week Sprint* shows in chapter 5.3.1. Demand is clearly higher than throughput, which may not necessarily indicate too much demand, but rather point out that the case study's approach and the collaboration of the two suppliers is still in its early stages and thus has room for improvement (cf. chapter 6.4).

Prioritize

Each Product Owner prioritizes the items of his Product Backlog. Prioritization among the three backlogs is done according to the most pressing deadlines.

Predictability needs to be improved with deliveries in the case study. Before that there is little point in paying attention to prioritization [Ande10].

Attack Sources of Variability to Improve Predictability

Dealing with sources of variability and reducing them is an advanced topic [Ande10], which first requires the correct implementation of the five preceding steps. Since that has not been fully achieved, investigation of this value is currently pointless.

6.1.2 Comparison to Agile Values

This chapter evaluates the realization of agile values, i.e. the Agile Manifesto (cf. [BeBe01]) and the five Scrum values by [ScBe02], in the case study's multi-project management approach.

Individuals and Interactions over Processes and Tools

The interactions between teams and individuals are frequent, given the fact that the two suppliers are not co-located. Several video conference meetings and daily follow-up telephone calls confirm this observation.

However, as the problems *Communication and Coordination between the two Suppliers* and *Lacking Tool-Support* indicate in chapter 5.3.2, there is potential for improvement (cf. chapter 6.4). More elaboration on the problems is provided in chapter 6.2.

Working Software over Comprehensive Documentation
Working software is going to be achieved by implementing the *Behavior Driven Development* workflow, but the general understanding of the BDD concept among team members needs to be raised to improve code quality (cf. chapter 5.3.1).

There is little documentation, which makes it very hard for new developers to join the teams.

Customer Collaboration over Contract Negotiation
The main supplier and the customers share a database for requirements, which is updated by the customers themselves. The additional supplier does neither have direct contact with the customers nor access to the database.

Responding to Change over Following a Plan
The first part of this value can definitely be found in the process implementation of the case study. Since the process in use is Scrum and not the lean follower Kanban, the commitment within iterations needs to be respected and thus no changes in priorities can be accepted during an ongoing Sprint. However, currently planning seems to be too uncoordinated for the teams to successfully commit to Sprint goals.

After the four values of the Agile Manifesto, the text continues (cf. [BeBe01]), *"While there is value in the items on the right, we value the items on the left more"*. Therefore, the second part of this fourth value *following a plan* is also important and should not be ignored. Incremental planning is a central component of successful agile and lean process implementations.

Scrum Value 1: Commitment
As discussed in the precedent paragraphs, commitment is hard to achieve, when planning is late for the beginning of the next Sprint. With timely planning, commitment will be easier to achieve.

Scrum Value 2: Focus
Focus means not having to worry about anything else [ScBe02], which is currently not given due to missing commitment.

Scrum Value 3: Openness
The value openness talks about the transparency within the project, which poses a problem, as Scrum boards and Burndown charts are both paper-based. Furthermore, the additional supplier currently does not have access to the Product Backlogs and the project management tool.

Scrum Value 4: Respect
Respect is hard to evaluate from a consultant's point of view with no direct involvement in daily project work, but in general everybody has been respectful to each other.

Scrum Value 5: Courage
This value also involves the courage to say "no", e.g. to frequent changes of goals within Sprint iterations, in order to be able to fulfill the commitment. Since recent Sprints have often started late, this indicates that the value has not been met and that team members need to be encouraged to stand up for timely planning.

6.1.3 Comparison to Traditional MPM Approaches

Chapter 3.2 discussed conclusions from traditional multi-project management approaches for lean and agile ones. These are now checked against the MPM measures taken in the case study.

Project Management Office (PMO)
There is no Project Management Office in place, which is frequently found in traditional approaches. But there is also no *Product Owner Team* (cf. chapter 4.1.2) to replace a PMO. Hence, a more structured approach for higher-level responsibilities is missing in the case study.

Management by Exception
This paradigm of *PRINCE 2* is not used in the case study, since it does not work well with agile practices in general (cf. chapter 3.2.3).

Multi-Project Management Hierarchy
The MPM hierarchy is very flat in the case study: Scrum teams and Product Owners. Additionally, there is a Scrum of Scrums meeting, in which also the three Product Owners participate for coordination.

Project Interdependency Meetings
In most organizations interdependency issues are discussed in "general" meetings [Rung10a]. In the case study there is the Scrum of Scrums, which is especially designed for inter-project coordination. Moreover, it is frequently discussed during daily Scrum meetings, since the three projects share the same codebase.

Prioritization
The customers and their individual deadlines control the current prioritization process. In [HiAl11]'s terms, the Product Owners deal with the internal prioritization mostly *according to urgency* and *resource constraints*. Limitations and significances of priorities are often not communicated in detail, especially not to the additional supplier.

6.1.4 Comparison to Lean/Agile MPM Approaches

This chapter checks the case study's approach against various agile and/or lean MPM ones that have been presented in chapter 4.

Kanban Swim Lanes

The two suppliers in the case study do not use an integrated board that includes all teams. Nevertheless, the approach in use is *team-based*, i.e. each team has its own board including user stories of different projects.

Scrum Program Management

[Pich08] suggests the introduction of *Product Owner Teams* headed by a Chief Product Owner among several other tips for agile Multi-Project Management. In the case study the three Product Owners need to work together because the products share the same codebase. But it is neither formally regarded as a *Product Owner Team*, nor does it have a chief. There is also no *Program Owner Team* above.

All three Scrum teams are organized as feature teams that work in synchronous Sprints. Estimation is based on hours, not on story points, as suggested by [Pich08]. The steps of *lookahead planning* are not used, nor is *pipelining*. Lastly, *Meta Scrum* would cause a process overhead for the given project environment. So in the case study only the *Scrum of Scrums* has been implemented.

Scrum in a Multi-Project Environment

[Glog11] also discusses a team-based and project-based view and suggests to have a company-wide prioritized project backlog. Scaled to the case study's environment, the three projects need to be ranked according to changing priorities. Currently the prioritization is based on the customers' deadlines.

Scaling Lean and Agile

The authors of [LaVo09] note several key points of large-scale Scrum that will now be compared to the case study.

Long-lived teams. The teams in the case study have recently been newly formed, especially on the additional supplier's side. Interviews showed that longevity is not one of the main concerns as frequent rotations of team members are planned.

Cross-Team Working Agreements. As the Retrospective showed (cf. chapter 5.3.1), these have not been established satisfactorily and need improvement.

Team Manages External Dependencies. The Product Owners manage all dependencies to the customer. Regarding the two suppliers, the two “Scrum Masters” at the additional supplier handle most of the coordination.

Team Makes Decisions. Many team members are young and some do not have previous experience with Scrum. Hence, for establishing truly self-organizing teams there is both room and need for improvement in the case study.

Requirement Areas. The teams are formed around requirement areas, although on a smaller scale than presented by [LaVo09].

Merged Product Backlog for a Set of Products. Currently all three products have individual Product Backlogs.

Large-Scale Scrum Framework. This framework is too large-scale for the case study. However, one of the concerns is proper *Product Backlog Refinement*, which was an issue in the Retrospective (cf. chapter 5.3.1) as planning was done too late and user stories remained unclear to the additional supplier far into the Sprint.

Enterprise Scrum
Implementing [Gree10]’s *Enterprise Scrum* is not possible in the case study’s current environment, nor in the given time frame of this thesis.

B-Scrum: Business Extension of the Scrum Framework
The *Business Scrum* extension is aimed at large-scale development and thus not applicable to the case study.

UX Component Team
[BuJe09] describe the advantages of a separate user experience (UX) *component team* in their experience report. However, given the fact that only three teams are involved, feature teams are a better choice. In general, many authors prefer feature teams (e.g. [LaVo09] or [Pich08]) because they offer a customer-centric approach and encourage little dependencies between teams.

Lean-Agile Project Management Office
Having a lean-agile Project Management Office is an interesting, yet controversial (cf. LaVo09]) approach. However, in the case study no PMO is in place and introducing one would require major organizational changes far beyond the time frame of this thesis (and willingness of the stakeholders).

6.2 Identified Problems (Top-Down)

This chapter summarizes all problems that have been found in interviews, data, the chapters 6.1.1 to 6.1.4 and through on-site observations.

6.2.1 Distributed Development

The following problems have been identified regarding the collaboration of the two suppliers in a distributed development environment:

No Official Scrum Roles at the Additional Supplier
All three Scrum teams are staffed by members of both suppliers, yet all Product Owners and Scrum Masters are on the main supplier's side. Nevertheless, at the additional supplier two team members have emerged that do more coordination work than their colleagues. They care more for the Scrum process than others (Scrum Master) and travel to the main supplier to attend meetings and discuss user stories in person (Product Owner).

More precisely, these two team members are different to regular team members in the following ways:

- First point of contact to and for the main supplier (Project Manager/Product Owner)
- Moderating the Daily Scrum meetings on the additional supplier's side of the video conference (Scrum Master)
- Moderating the additional supplier's planning meeting (Product Owner)
- Handling and supervision of the planning spread sheet (Project Manager/Product Owner)

First Point of Contact. If the main supplier were regarded as customer to the main supplier, then this would be a Product Owner's rightful task. But since the main supplier is part of the Scrum team, this practice impedes the self-organization of the team and introduces a hierarchy into the team's structure, which does not have room in Scrum.

Moderating the Daily Meetings. The two members act as Scrum Master on the additional supplier's side of the daily meeting, although they are ordinary team members (according to official roles).

Moderating the Planning Meeting. The two members have the best knowledge and information about upcoming user stories and try to answer all questions of the team members. This is a typical Product Owner's task.

Handling and Supervision of the Planning Spreadsheet. The two members supervise the planning spreadsheet and as such are given responsibility that exceeds the one of regular team members. This would typically be a project manager's or a Product Owner's task if the spreadsheet was based on the team's performance and not on individual's working hours, which does not align with Scrum values in general.

However, it is not surprising that these problems have arisen. The team members on the additional supplier's side are 10 people scattered over three different Scrum teams. There is no Scrum Master and no Product Owner on their side. Hence, it is very hard to remain self-organizing and in compliance with the Scrum process, when the contact to the remaining team members is hard to establish and nobody is officially assigned to look after the process at the additional supplier's premises.

Joint Estimation and Planning
Usually every team should have their own planning meeting. But since every team has up to three Product Owners, it is more efficient to hold the meeting jointly. The teams estimate the user stories themselves, but there is a rough pre-estimation by Product Owners. This procedure does not meet Scrum values.

Furthermore, the additional supplier is holding a second follow up estimation meeting to adapt the pre-estimated values.

Inter-Company Distribution of Team Members
As has been noted in the beginning of this subchapter, it is hard for the additional supplier to work in teams that are distributed. The larger part of each team is at the main supplier's premises. Hence, they can work regularly with an on-site Scrum Master and Product Owner and have two to three remote developers that they need to stay in touch with at the additional supplier's side.

For the additional supplier, this poses a big problem, as these two or three developers are separated from the rest of the team. In consequence, the additional supplier has formed a kind of *virtual team* to manage his own resources. This manifests itself in a single paper Scrum board covering all three teams on the additional supplier's side (instead of three individual ones). The follow-up planning session is also done jointly on the additional supplier's side.

6.2.2 Transparency

Transparency is a big issue between the two suppliers due to the following reasons:

Suppliers not Co-located
The physical distance of 300 kilometers between the two suppliers is the reason for many problems. The whole process becomes more complicated and less transparent.

Communication Issues
As the Retrospective in chapter 5.3 has shown, means of communication between the two suppliers are still not good enough in terms of quality. The bad video and audio quality handicaps all of the meetings and prevents the additional supplier from getting more involved in discussions.

Little Documentation
The codebase, on which all three products are built, is poorly documented. This causes difficulties for new team members to join the projects. There is also no knowledge-sharing tool available, such as e.g. a simple web-based *wiki*.

No Overview over All Teams
The three teams each have a paper Scrum board at the main supplier's side. The additional supplier only operates a combined board for all three teams. The paper Burndown chart at the main supplier's side is also only team-based. Hence, no high-level progress of all three teams is available.

6.2.3 Commitment

The following problems regarding commitment have been identified:

Commitment Fails with Insufficient Planning
The teams cannot commit to Sprint goals when the user stories are not properly specified. In consequence, no reliable estimations are possible.

Commitment Fails with Late Planning
When the effective beginning of a Sprint is delayed a few days up to a full week due to late planning, the teams will not be able to commit to the original Sprint goals.

Commitment Fails with Frequent Changes
The Product Owner cannot change user stories or prioritization within an ongoing Sprint. The team needs to have the *courage* (one of five Scrum values, cf. chapter 6.1.2) to stand up to the Product Owner and explain that ad hoc changes within the iteration violate their commitment. If the Product Owner insisted on the changes, then the current iteration would have to be stopped.

All of these three problems have occurred in the case study. It needs to be made clear that it is the Product Owner's responsibility to have a specified and prioritized Product Backlog at the beginning of each Sprint. Otherwise the two Scrum values *commitment* and *focus* cannot be achieved (cf. chapter 6.1.2).

Little Respect for Iterations
Humans are probably more sensitive to time variation than scope variation [LaVo09]. One of the core principles of Scrum is to release every Sprint even if not all goals have been met. Teams commit to Sprint goals and need to respect the Sprint's ending. Even if not all user stories have been completed, the Sprint ends at its deadline. Otherwise the whole Scrum process is broken, because timeboxes and commitment are central to Scrum implementations.

One of the advantages of working with iterations is to have a regular schedule for meetings, e.g. Sprint Review on the last Friday of the iteration. But since boundaries are not fully respected in the case study, the beginning and ending of Sprints are subject to change, as are meetings.

6.2.4 Planning

Insufficient Planning is one of the biggest problems in the case study's environment. The following root causes have been identified:

Late Actual Beginning of Sprint
Planning for the current iteration has often not been finished until a few days into the already ongoing two-week Sprint, which caused major delays in the schedule.

Little Participation of Additional Supplier
The main supplier defines the schedule and assigns user stories with pre-estimated work hours to the additional supplier. Hence, the additional supplier is not adequately involved in the planning process apart from updating the estimations of the main supplier (for his own user stories only).

Little Information for Additional Supplier
On top of the reason above, the additional supplier also frequently lacks information to reasonably split work among its team members, which also causes delays until sufficient knowledge about the user stories is gathered (through numerous phone calls e.g.).

6.2.5 Estimation

The following problems regarding estimation have been identified:

User Story Estimation in Hours
Man-hours do not represent complexity well because different people need different amounts of time to work on a user story. This fact especially poses a problem when the main supplier estimates in place of the additional one based on *his* experience.

Expert Estimation

During Retrospective, a team member suggested to have experts do the estimations. This would be highly counterproductive to the functioning of Scrum. Estimations in Scrum represent average work hours/complexity, not expert ones. When experts estimate, other team members without the expertise will fail to meet the expected time frame.

Estimation for Additional Supplier

For the same reasons, team members of the additional supplier should estimate their user stories without prior estimation by the main supplier. It influences them and increases pressure. Both will lead to bad estimations.

6.2.6 Predictability

The following problems regarding predictability have been identified:

No Proper Sprint Velocity

Sprint velocity is currently not correctly measured. The main supplier runs a paper Burndown chart that is based solely on tasks (that have been derived from user stories). Hence, the only available ratio, *tasks per Sprint*, does not represent any complexity because it neither takes into account man-hours nor story points.

Further Impediments for Better Predictability

Apart from the first main problem, the following ones also prevent a more significant predictability (cf. chapter 5.3 “Known Problems”):

- Varying understanding of the Behavior-Driven Development workflow among team members
- Code Quality Issues
- Missing remote access for the additional supplier to the main supplier’s test environment

6.2.7 Self-Organizing Teams

The following problems impede truly self-organizing teams:

Tasks Assigned to Team Members

At the additional supplier’s side, the two *Scrum Master-like* members coordinate the team members and suggest tasks for them during the follow up planning session or during Sprints, as interviews have pointed out. In Scrum, team members should always volunteer for tasks or assign them autonomously according to team-based consensus.

Estimations Based on Individuals
The estimation is done on an individual level. With Scrum, the velocity of whole teams should be measured and taken as a basis for estimation. The focus on combined efforts rather than on individual ones supports the development of a self-organizing team.

Missing Courage to Stand up
Most of the additional supplier's team members do not have prior experience with Scrum, which may explain the missing courage (or awareness) for the need to address the first two problems.

Cross-Team Working Agreements
As the Retrospective (cf. chapter 5.3.1) has pointed out, cross-team working agreements, such as a common *definition of done*, need to be elaborated and agreed upon in this multi-team and multi-project environment.

Scrum Master at the Scrum of Scrums
The Scrum of Scrums is a good means to deal with interdependency issues of the teams in a self-organizing way. In the case study, the Scrum Master is the one to attend this meeting every day, although he should not because he will evolve to the manager of the team [Glog11].

Product Owner at the Scrum of Scrums
The Product Owners should not be a regular part of the Scrum of Scrums since they are not part of the teams. Self-organizing teams neither need Scrum Masters nor Product Owners to solve dependencies or conflicts.

6.2.8 Tools

The following problems regarding tools have been identified:

Tools Lack Scrum Compatibility
The electronic tools in use all lack Scrum support, which impedes a proper process implementation.

Limited Access for Additional Supplier
The main supplier is not granting access to all tools, i.e. the additional supplier cannot access the project management tool or the requirements database. This would be acceptable if the main supplier were the customer to the additional one. But since they co-develop in joined Scrum teams, this restricted access violates the Scrum value *Openness*.

Paper Scrum Board

There are currently four paper Scrum boards in use, three at the main supplier for each team and a combined one at the additional supplier. For obvious reasons, these are cumbersome to synchronize, which makes it hard to keep track of other teams' progress.

Paper Burndown Chart

The Burndown charts (one for each team) are also drawn on paper at the main supplier's side and updated by hand. In contrast to Sprint planning and estimation, the Burndown chart is not based on hours but simply on tasks, e.g. Sprint X has Y tasks to burn down. This is a very risky approach that completely hides the complexity of user stories and their tasks.

6.3 Problem Root Cause Analysis

Table 6 provides an overview of all identified problems and their root causes in the case study.

Identified Problems	Root Causes
Distributed Development	• No Official Scrum Roles at the Additional Supplier • Joint Estimation and Planning • Inter-Company Distribution of Team Members
Transparency	• Suppliers not Co-located • Communication Issues • Little Documentation • No Overview over All Teams
Commitment	• Commitment Fails with Insufficient Planning • Commitment Fails with Late Planning • Commitment Fails with Frequent Changes • Little Respect for Iterations
Planning	• Late Actual Beginning of Sprint • Little Participation of Additional Supplier • Little Information for Additional Supplier
Estimation	• User Story Estimation in Hours • Expert Estimation • Estimation for Additional Supplier
Predictability	• No Proper Sprint Velocity • Further Impediments for Better Predictability
Self-Organizing Teams	• Tasks Assigned to Team Members • Estimations Based on Individuals • Missing Courage to Stand up • Cross-Team Working Agreements • Scrum Master in the Scrum of Scrums • Product Owner in the Scrum of Scrums
Tools	• Tools Lack Scrum Compatibility • Limited Access for Additional Supplier • Paper Scrum Board • Paper Burndown Chart

Table 6: Identified Problems and Root Causes

Additionally, Figure 38 puts the problem categories in relation to each other as part of the root cause analysis.

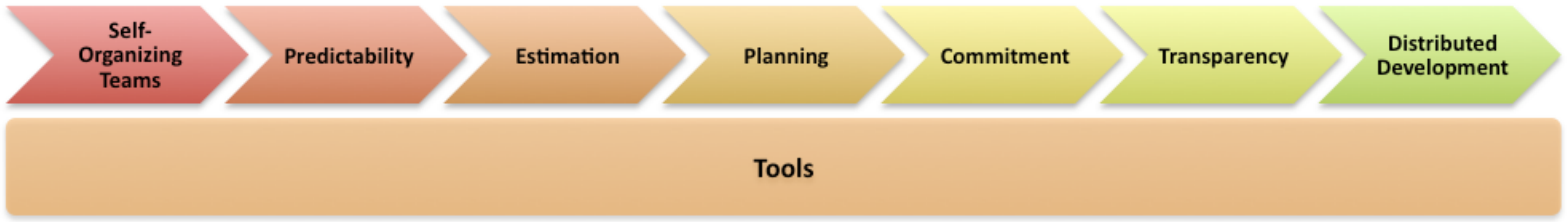


Figure 38: Problem Root Cause Analysis

Self-Organizing Teams are one of the most basic and central concepts of Scrum because it cannot work without them. Hence, this category has been put at the beginning in Figure 38, i.e. a problem category that needs to be addressed first.

Predictability evolves when long-lived self-organizing teams work in Sprint iterations without interference.

Estimation can only be accurate when a good predictability is achieved.

Planning relies on both estimations and predictability.

Commitment needs self-organizing teams that are willing to commit to Sprint goals and are allowed to work without disturbance. The case study shows that commitment cannot be achieved without sufficient planning.

Transparency is one of Scrum's highest goals, i.e. making impediments visible to everyone. All the precedent problem categories need to be solved, before transparency can be achieved.

Distributed Development is the central problem in this case study, since the suppliers are not co-located. Efficient collaboration can only be achieved by improving all the other problem categories first.

Choosing the right *tools* and artifacts is needed as support for solving all other problem categories.

The process in the case study needs to be improved bottom-up, i.e. solving root causes of the problem categories from left to right in Figure 38, which will eventually lead to an improved distributed development and thus a more productive multi-project environment. Solutions to all problem categories and their root causes are suggested in the following chapter.

6.4 Suggested Improvements (Bottom-Up)

The number one organizational impediment to a successful adoption of lean and agile principles is *silver bullet thinking and superficial adoption*, which often results in the opinion that “it does not work“ [LaVo09]. As chapter 6.3 pointed out, the biggest problem in the case study is that the two suppliers are not co-located, which poses a number of constraints to all problem categories. Additionally, some more “regular” problems arose such as the missing measurement of Sprint velocity or the respect for iterations.

This chapter will provide suggested solutions to all problems and their root causes in a bottom-up fashion (cf. Figure 38 in the precedent chapter). All previously known problems have been included in the identified ones, as Figure 39 illustrates.

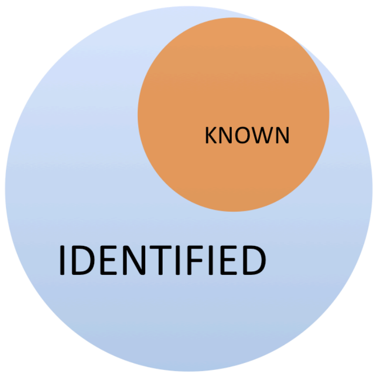


Figure 39: Relation of Known and Identified Problems

Hence, by suggesting solutions to the identified problems of chapter 6.2, all problems will be covered.

6.4.1 Tools

The following solutions are proposed to deal with root causes regarding tool problems:

Introduce Electronic Tool with Scrum Support
Since the two suppliers are not co-located, Scrum boards and Burndown charts on paper do not suffice because they are very cumbersome to synchronize and to keep track of. Therefore, an electronic tool with Scrum support should be set up to facilitate communication of the two suppliers.

Further suggestions on the Burndown chart can be found in chapter 6.4.4.

Introduce Knowledge-Sharing Tool
To have a minimum amount of documentation, a simple knowledge-sharing tool (such as a *wiki* e.g.) should be set up that is accessible to both suppliers.

Grant Full Access to Additional Supplier
The additional supplier needs to get the same access rights to tools as the main supplier, so that all team members can be equal partners in the distributed Scrum teams.

6.4.2 Self-Organizing Teams

The following solutions are proposed to deal with root causes regarding self-organizing teams:

Team Members Volunteer for Tasks
Instead of being assigned or proposed tasks, the team members need to decide on their own, how to split the tasks among them. This demands sufficient knowledge about new user stories, which has been a problem for the additional supplier in the past (cf. proposed solution “Product Backlog Refinement” in chapter 6.4.6).

Focus on Team, not Individuals
Scrum directs attention to teams, not individuals. This applies to estimation and planning as well as to the precedent suggestion. It is very important to respect teams and not to interfere with their self-organization.

Establish Cross-Team Working Agreements
As has been noted during the Retrospective (cf. chapter 5.3.1), the three teams do not share a common *definition of done*. This is very important to improve coordination among teams and needs to be adapted and refined on a regular basis [LaVo09] (cf. chapter 4.1.4).

Send Team Members to the Scrum of Scrums
The Scrum Master should never attend the Scrum of Scrums or he will evolve to the manager of the team [Glog11]. Instead, any team member may attend the Scrum of Scrums, preferably one that needs to discuss an impediment.

No Product Owner at the Scrum of Scrums

In the case study, Scrum Masters and Product Owners meet in the Scrum of Scrums. Neither should be part of this meeting because they do not manage the team. If a Product Owner wants to get in touch with the team, he can attend one of the regular Daily Scrums or also the Scrum of Scrums, but not on a regular basis.

Long-lived Teams

Teams should be allowed to stay together for as long as possible to increase productivity by building stable working relationships [LaVo09] (cf. chapter 4.1.4). Learning plays an important part for long-lived teams, as Figure 40 illustrates.

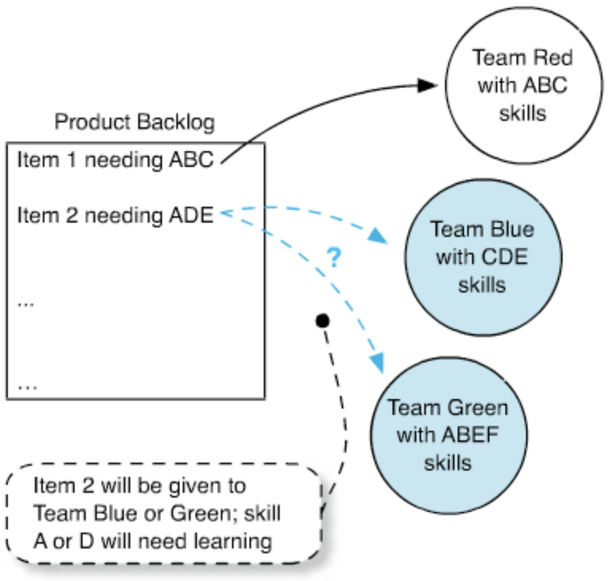

Product Backlog
Item 1 needing ABC
Item 2 needing ADE
...
...
Team Red with ABC skills
Team Blue with CDE skills
Team Green with ABEF skills
?
Item 2 will be given to Team Blue or Green; skill A or D will need learning

Figure 40: Long-lived Teams and Learning [LaVo09]

Sometimes not all skills required by work items can be covered. The solution is simple: learn. In consequence, long-lived teams will be able to work on a great variety of items.

6.4.3 Predictability

The following solutions are proposed to deal with root causes regarding predictability:

Measure Sprint Velocity with Story Points

Sprint velocity reveals the team's speed in implementing user stories within the iteration. In the case study it is measured in *tasks per Sprint*, which does not represent the complexity of tasks or their corresponding user story and is thus too imprecise. A better characteristic is *story points*, which will be presented in the following chapter.

Discuss Impediments with the Team
One of the key points in lean development is to find the most dominant constraint or bottleneck, try to reduce it until it is no longer dominant and then move on to the next one [LaVo09]. Consequently, the Product Owner should identify and discuss the main impediments with the teams to increase predictability. This also corresponds to the lean *Go See* management philosophy.

The other root causes are addressed in the following chapters:

- BDD workflow in chapter 6.4.7[23]
- Missing Remote Access in chapter 6.4.1

6.4.4 Estimation

The following solutions are proposed to deal with root causes regarding estimation:

Estimate in Story Points
Person hours do not represent the complexity of user stories because people are working at different speeds. Therefore, the introduction of *story points* is proposed (cf. [Cohn05]). Story points are an abstract unit to describe the expected complexity of a user story. Estimation is done *by analogy*, i.e. stories are estimated in relation to each other. Furthermore, it is important to use a decent scale for estimation, e.g. “1, 2, 3, 5, 8, 13, 20, 40, 100”, which helps to stick to rough estimations and not get lost in fussy arguments.

Velocity is measured in *Story Points per Sprint*. Burndown charts also work with story points instead of tasks (which are currently used in the case study).

Team Estimates
The secret of agile estimation is that team members, i.e. the ones who will develop the user stories, estimate themselves, which leads to more accurate results. Hence, there should be no pre-estimations, neither by Product Owners nor experts nor by the main supplier for the additional one. Otherwise estimations may be inaccurate and teams cannot commit to Sprint goals in good conscience. In the case study this fact often led to misestimations and prolonged Sprints.

Since the case study works with distributed Scrum teams, it is possible to use tools that allow using planning poker for estimation in such environments, e.g. the free *planning-poker.com*. Obviously, a prerequisite for this procedure would be that the additional supplier has sufficient knowledge about the user stories to be able to estimate beforehand. This can be established with:

[23] The remaining root cause *Code Quality Issues* is taken care of with a working BDD workflow as well

1. High Quality Video Conferences (cf. chapter 6.4.7)
2. Product Backlog Refinement (cf. chapter 6.4.5)

Include Research and Learning in Estimations
[LaVo09] presents an interesting approach on how to deal with research and learning of a team with regard to estimations. In addition to a team's regular work in a Sprint, i.e. implementing user stories, a time-boxed and effort-boxed research task is included in the Sprint estimation, coupled with a concrete goal such as an introductory report on the subject. An exemplary research task could be *"Introductory report on push and talk, maximum 30 person hours"* [LaVo09]. The Product Owner may then decide to invest more bounded effort of research into one of the next iterations [LaVo09].

6.4.5 Planning

The following solutions are proposed to deal with root causes regarding planning:

Plan In Time
Planning should ideally provide the development team with the best things to work on next, i.e. no more and no less [LaVo09]. In the case study, Sprints had to begin late because the specification was not completed in time and thus development could not start. In consequence, fulfilling commitment was impossible.

It is the Product Owners responsibility to maintain an updated and prioritized Product Backlog, so that Sprints can begin on time with sufficient information about the user stories for development.

Product Owner Team
Since the products share the same codebase, the Product Owners should form a *Product Owner Team* (cf. chapter 4.1.2) to discuss priorities and coordinate efforts. The chief technical architect or marketing/sales representatives can also be part of this team that may run Scrum as well (with Daily Scrum meetings, etc.).

Merged Product Backlogs
If the Product Owners improve their collaboration by working as a team, merging the Product Backlogs to a single one (for all three products) can lead to a better prioritization. In contrast to [Glog11]'s Project Backlog (cf. chapter 4.1.3) a merged Product Backlog demands the Product Owner's collaboration and prioritization on the requirement level instead of prioritizing whole products or projects. This allows the timely implementation of essential items of a lower priority product (i.e. one with later deadline). Coarse-grained product prioritization, on the other hand, would lead to a local optimization [LaVo09].

Include Additional Supplier in Planning

As has been noted in the precedent chapter, the additional supplier has to be part of the actual planning and estimation process. The members of the additional supplier should also be part of planning sessions and join in a high quality video conference (cf. chapter 6.4.7). This way, the additional supplier can ask the Product Owner(s) directly about unclear user stories and is not bound to second hand information of travelling “Scrum Masters”, which produced an overhead of communication and meetings.

6.4.6 Commitment

Insufficient and late planning has already been addressed in the precedent chapter. The following solutions are proposed to deal with the remaining root causes regarding commitment:

Respect Iterations

Working in Sprints implies releasing at the end of each iteration, even if not all Sprint goals have been met. The experience gained can be used to make more accurate estimations in the following Sprints. But the iterations have to be respected as well as the meeting schedules. This is important for learning and improvement and brings a *sustainable pace* to software development. Therefore, meeting schedules should be set and also followed, e.g. Review on the last Friday and Planning on the first Monday of a Sprint.

Commitment can only work if the iterations are respected (and do not get prolonged). Additionally, estimations will eventually become more accurate with a consistent pace.

No Changes Within a Sprint

Once a team commits to a certain number of user stories in a Sprint, no more changes to this selection are possible. When priorities change within a Sprint, the Product Owner needs to wait until the end of the current iteration to pass them on to the team.

Lookahead Planning for Commitment

As the identification of root causes for problems has shown (cf. chapter 6.2.3), sufficient and timely planning is a prerequisite for a working commitment. In the case study’s distributed multi-project environment many constraints and dependencies arise between teams. To minimize these dependencies and thus ensure working at full capacity, the introduction of *Lookahead Planning* by [Cohn05] is suggested. It is an enhanced planning method especially designed to deal with a multi-team environment working on the same codebase. The complete listing of steps can be found in chapter 4.1.2.

Product Backlog Refinement
A lesser known guideline of Scrum is that five or ten percent of each Sprint must be dedicated by the team to refining the Product Backlog, which includes (cf. [LaVo09]):

- Detailed Requirements Analysis
- Splitting Large Items into Smaller Ones
- Estimation of New Items
- Re-Estimation of Existing Items

In a two-week Sprint, five percent are equal to half a day. [LaVo09] suggests a focused workshop, but the case study's distributed environment puts several constraints to its realization, e.g. video conference quality needs to be greatly improved first (cf. chapter 6.4.7).

6.4.7 Transparency

The root cause *Suppliers not Co-located* will not be considered here. Obviously, the process would be much simpler if the suppliers were co-located, but this is not possible under the given circumstances. Therefore, the solutions proposed strive to improve the current distributed process situation.

The following solutions are proposed to deal with root causes regarding transparency:

High Quality Video Conferences
With the current situation, it is quite hard to communicate via video conference calls as both video and audio quality is not good enough. Improvement is needed in two parts:

1. Equipment (e.g. beamer, better microphone, etc.)
2. Granting VoIP[24] rights at the main supplier

The second point requires further explanation: The main supplier has blocked VoIP to all of its employees. So video conference calls are only possible, when one of the two additional supplier's "Scrum Masters" travels to the main supplier and enters the guest network (where VoIP is not blocked). This causes daily meetings to be held via simple cell phone conference calls, where quality is a big issue.

Both parts of this suggestion need to be addressed to improve video conference calls, which are essential to the communication of the two suppliers.

[24] Voice over IP

Meeting Protocols

A common misunderstanding is that agile means having no documentation at all. The Agile Manifesto (cf. [BeBe01]) only points out that *comprehensive* documentation is not encouraged.

This does not imply that a minimum amount is not needed in agile processes. Hence, meeting protocols of the Review and Retrospective by means of an *End-of-Sprint chart* e.g. (cf. [Pich08]) are important to keep track of progress and measures taken. The Scrum Masters should also operate an impediment list that gets updated in the Daily Scrum meetings.

Lastly, knowledge sharing can be supported by documentation as well in the form of a *wiki* e.g., as has been proposed in chapter 6.4.1.

Introduction of a Lean High-Level Product-based Board

Keeping track of three products may be difficult with team-based Scrum boards. [Ande11] presents a high-level team-based board; [Lada08] prefers the product-based point of view (cf. chapter 4.1.1).

It is advised to use a lean product-based board because it sets focus on what is most important: customer value, i.e. the progress of their products over the whole value stream. Figure 41 illustrates the realization of the idea for the case study's setting with three products and three Scrum teams (teams are portrayed as *orange*, *yellow* and *green*).

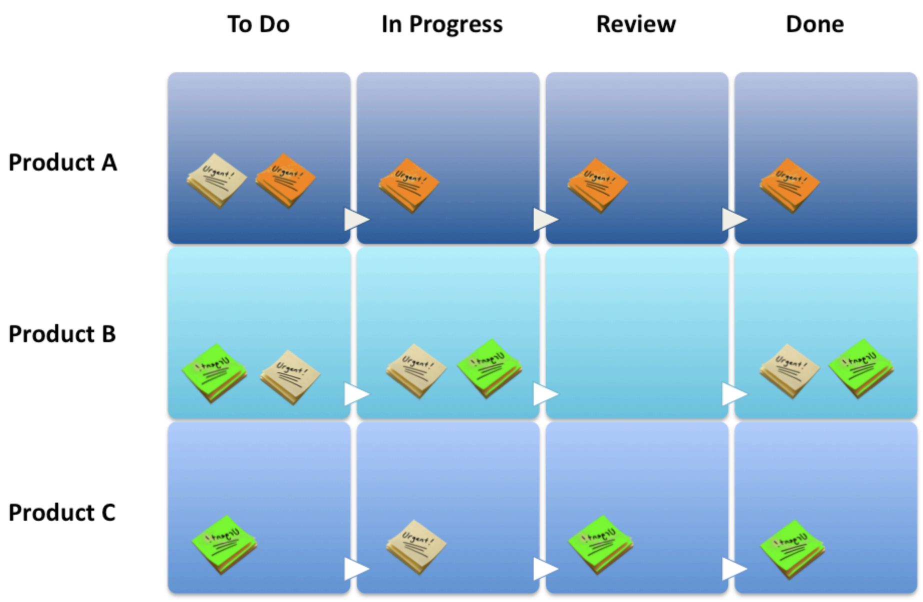


Figure 41: High-Level Product-based Board

The ideal practical solution would be to have an electronic tool for scrum boards (cf. proposal in chapter 6.4.1), which also allows filtering so that a product-based board can be built on the fly with pre-set filters. This view is especially interesting for Product Owners, but also for all others to get an idea of the current overall progress.

Realistic Lean Illustration of the Workflow
The current workflow on Scrum boards is:

To Do ⇒ In Progress ⇒ Review ⇒ Done

This is not an accurate representation of the *Behavior Driven Development* workflow. Especially the *In Progress* column on the boards hides a lot of complexity, such as *Feature File*, *Test Code*, *Application Code* and *Testing*. Mapping the value stream accurately is one of the central components of lean software development (cf. [PoPo04], “Eliminate Waste” and “See the Whole”).

With a truthful approximation to the actual workflow on the Scrum boards, bottlenecks and idle times become visible and thus overall transparency is greatly increased. Additionally, this proposal also supports a common understanding of the BDD workflow among team members, once it is visible to everyone on the Scrum boards.

6.4.8 Distributed Development

The following solutions are proposed to deal with root causes regarding the collaboration of the two suppliers in a distributed development environment:

New Composition of Scrum Teams
Figure 42 shows the current composition of Scrum teams: three teams consisting of team members from both suppliers. All Scrum roles, i.e. Product Owners and Scrum Masters, are assigned to the main supplier’s staff. Nevertheless two Scrum Masters have emerged at the additional supplier to take care of the process and also some of the Product Owner’s duties (cf. chapter 6.2.1).

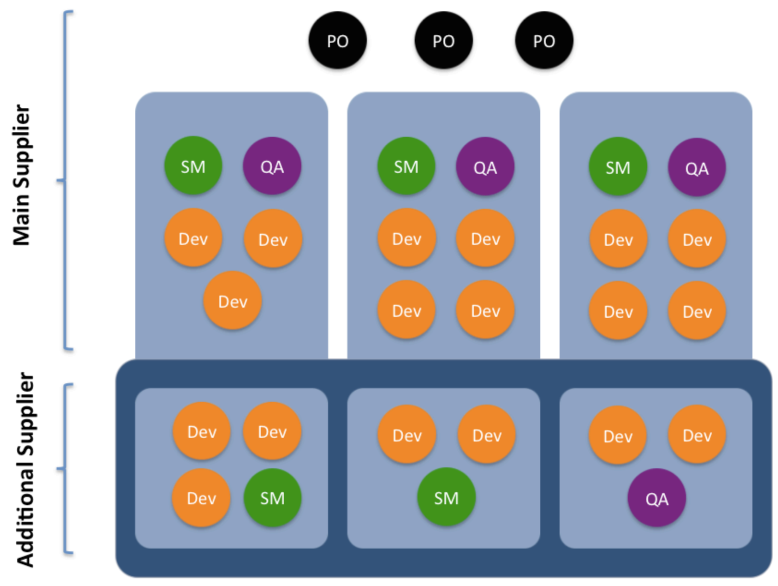


Figure 42: Current Composition of Scrum Teams

As has been pointed out during the Retrospective and identified problems (cf. chapter 6.2), it is a big effort to stay updated on the other team members' progress. Therefore, Figure 43 proposes a new composition of Scrum teams to deal with this issue.

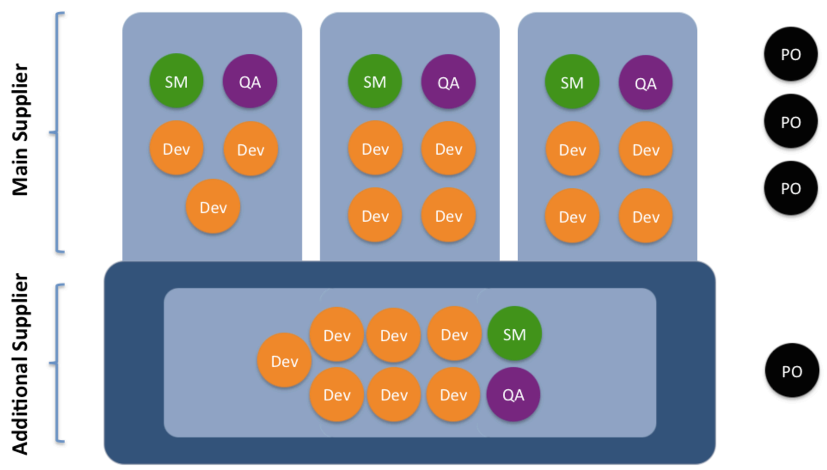


Figure 43: Proposed Composition of Scrum Teams

With new composition of teams, the in-team collaboration is disburdened because meetings are simply held on-site. The two *de facto Scrum Masters* now get official roles: a Product Owner and a Scrum Master. This new Product Owner is part of the Product Owner team (cf. suggestion in chapter 6.4.5), and travels to the main supplier frequently (as the two Scrum Masters already do now) and acts as PO to the additional supplier's Scrum team.

Moreover, one of the additional supplier's Scrum team also takes part in the Daily Scrum of Scrums, which now consists of one team member from each team (no Scrum Masters and no Product Owners, cf. chapter 6.4.2), to discuss inter-team coordination and dependency issues.

This new composition of Scrum teams also aligns with [LaVo09]'s definition of a feature team which states that its team members need to be co-located. Long-lived teams are also easier to accomplish with this new arrangement.

New Composition of Scrum of Scrums
If the suppliers were to keep the current distribution of team members (cf. Figure 42), then the Scrum of Scrums should also include a team member of the additional supplier, depending on which member has an impediment that needs clarification with other teams (cf. suggestions in chapter 6.4.2 for further insight on the correct implementation of the Scrum of Scrums).

Joint Sprint Planning
[LaVo09] offers the following procedure for Joint Sprint Planning:

- Part One: User stories are offered to or volunteered for by whole teams instead of individuals.
- If there is a Product Backlog Refinement meeting (cf. solution in chapter 6.4.6), then part one will be quick as there will be few questions to solve.
- Part two of the planning is not hold jointly but team-wise, where the Sprint Backlog is created for the team.

In conclusion, Table 7 provides an overview of all problem categories, their root causes and suggested solutions.

Problem Categories	Root Causes	Suggested Solutions
Tools	• Tools Lack Scrum Compatibility • Limited Access for Additional Supplier • Paper Scrum Board • Paper Burndown Chart	• Introduce Electronic Tool with Scrum Support • Introduce Knowledge-Sharing Tool • Grant Full Access to Additional Supplier
Self-Organizing Teams	• Tasks Assigned to Team Members • Estimations Based on Individuals • Missing Courage to Stand up • Cross-Team Working Agreements • Scrum Master in the Scrum of Scrums • Product Owner in the Scrum of Scrums	• Team Members Volunteer for Tasks • Focus on Team, not Individuals • Establish Cross-Team Working Agreements • Send Team Members to the Scrum of Scrums • No Product Owner at the Scrum of Scrums • Long-lived Teams
Predictability	• No Proper Sprint Velocity • Further Impediments for Better Predictability	• Measure Sprint Velocity with Story Points • Discuss Impediments with the Team
Estimation	• User Story Estimation in Hours • Expert Estimation • Estimation for Additional Supplier	• Estimate in Story Points • Team Estimates • Include Research and Learning in Estimations
Planning	• Late Actual Beginning of Sprint • Little Participation of Additional Supplier • Little Information for Additional Supplier	• Plan In Time • Product Owner Team • Merged Product Backlogs • Include Additional Supplier in Planning
Commitment	• Commitment Fails with Insufficient Planning • Commitment Fails with Late Planning • Commitment Fails with Frequent Changes • Little Respect for Iterations	• Respect Iterations • No Changes Within a Sprint • Lookahead Planning for Commitment • Product Backlog Refinement
Transparency	• Suppliers not Co-located • Communication Issues • Little Documentation • No Overview over All Teams	• High Quality Video Conferences • Meeting Protocols • Introduction of a Lean High-Level Product-based Board • Realistic Lean Illustration of the Workflow
Distributed Development	• No Official Scrum Roles at the Additional Supplier • Joint Estimation and Planning • Inter-Company Distribution of Team Members	• New Composition of Scrum Teams • New Composition of Scrum of Scrums • Joint Sprint Planning

Table 7: Root Causes and Suggested Solutions

6.5 General Lessons Learned

The discussion of results regarding lean-agile multi-project management follows in the next chapter. This chapter provides a short overview of general lessons that have been learned during the realization of the case study.

Distributed Development

The focus of the case study was multi-project management, but in retrospective the distribution of development over two suppliers also complicated software development. This observation did not come unexpected, as many authors warned to avoid distributed development by all means possible during the literature review for this thesis. The process gets a lot more complex and the level (and willingness) of cooperation between team members determines success or failure. In the case study the additional supplier has not been treated as an equal partner, which resulted in communication problems and decreased transparency.

Organizational Change

The larger the organization, the harder it is to introduce changes. This could especially be observed with the main supplier, where minimal changes are subject of lengthy discussions and/or take very long to be implemented. The additional supplier was often happy to realize changes to the process, but compromises had to be made frequently to deal with organizational impediments at the main supplier's side.

Superficial Adoption of Scrum

The number one organizational impediment to a successful adoption of lean and agile principles is *silver bullet thinking and superficial adoption* [LaVo09], which could also be observed in the case study. Although distributed development caused many problems, underneath many normal Scrum values have not been met, such as self-organizing teams and the respect for iterations, which revealed that initially the adoption of Scrum has been very superficial. Improvements to the process implementation come slowly, but gradually.

7 Discussion of Results

The discussion of results begins in chapter 7.1 with a critical view on the general relation of theoretic results and their practicability with regard to the case study. Chapter 7.2 provides answers to the three research questions of this thesis with emphasis on the strengths and limitations of lean-agile multi-project management.

7.1 Theory vs. Practice

The case study showed that theoretic concepts are not easy to put into practice. The main problem is that many organizations do not want to change or do not feel the need to. Frequently, Japanese *kaizen*, i.e. continuous improvement and learning, is not appreciated, although it is part of all agile and lean processes. The essence is simple:

"If you have always done it that way, it is probably wrong."
(19th/20th Century Inventor Charles F. Kettering)

But even if an organization is aware that continuous improvement and thus change is needed, there are still a lot of obstacles to overcome. Especially when trying to introduce lean or agile processes, former project managers often do not want to let go of established hierarchies and try to prevent self-organizing teams - actively or passively. This can manifest itself in *fake Scrum Masters*, who do not support the teams as a *servant leader* (cf. [Gree02]), but rather try to actively manage the teams and overtake responsibilities.

[LaVo09] provides a rigorous answer to the issue: *"Better to teach people and risk they leave, than not and risk they stay"* (p. 149, [LaVo09]). In the case study of [Vall11], the organization also had to let go of one former project manager, who was not willing to adjust to Scrum. An explanation is provided by Atsushi Niimi, Toyota North America president, who said that the greatest challenge in teaching the Toyota Way to foreign managers was, "They want to be managers, not teachers" [LiMe07].

In general, when implementing concepts, compromises will almost always have to be made due to various practical constraints. The sole focus on the practices rather than on the core values can even undermine the agile value system [FrRi06]. Hence, it is all the more important that underlying agile and lean principles are understood.

7.2 Concluding Answers to Research Questions

This chapter provides concluding answers to the three research questions of this thesis (cf. chapter 1.2) with regard to the results from chapters two to six.

The first research question was:

1. *Which traditional Multi-Project Management approaches are the most accomplished?*

This has been addressed in chapter 3.1, where two of the main traditional process frameworks, the *PMBOK* and *PRINCE 2*, have been examined. Additionally, the *Program Management Method* by IBM Rational, the doctoral thesis [Rung10a] on the interdependency of projects and the prioritization techniques of [HiAl11] have been taken into consideration.

This analysis was the foundation for the second research question:

2. *What has to be changed in traditional Multi-Project Management to adjust it to Agile or Lean project environments?*

This issue has been addressed in chapter 3.2. The analysis showed that project management offices interfere with agile and lean values, especially with the teams' self-management and the Product Owner's duties and responsibilities. Although [AuCu06] presents a lean-agile PMO (cf. chapter 4.2.4), *Product Owner Teams* are a more lightweight approach that can handle the same responsibilities. The co-existence of a PMO and Product Owner Teams is not advised, as heavy interference among the two will be inevitable.

Furthermore, PRINCE 2's *Management by Exception* is not applicable to lean and agile processes. At first glance, it reminds of the agile value *focus*, i.e. that the team is self-organizing and allowed to work without disturbance, but on closer examination two reasons come up that contradict the approach:

1. There is no management of self-organizing teams, not even by exception.
2. Stakeholder involvement (e.g. Product Owner, customer) is essential and not the exception.

Table 5 in chapter 3.2.2 consolidated many traditional multi-project management tasks originating from the PMBOK, PRINCE 2 or IBM Rational. To investigate the relation further, Table 8 shows the alignments of traditional MPM tasks to lean/agile ones.

Traditional MPM	Lean-Agile MPM
Monitoring and Controlling	• Product Owner (Team) • Sprint Velocity • Sprint Review • Burndown Chart • Lean Team-Based or Product-Based Boards
Managing Shared Resources	• Daily Scrum • Scrum of Scrums • Meta Scrum
Resolving Issues	• Daily Scrum • Scrum of Scrums • Meta Scrum • (Joint) Retrospective • Scrum Master
Coaching, Mentoring and Training	• Scrum Master • Pair Programming
Identifying Common Methodology, Best Practices and Standards	Cross-Team Working Agreements
Developing Shared Process Assets (Templates etc.)	Scrum Master
Coordinating Communication across Projects	• Scrum of Scrums • Meta Scrum • Joint Retrospective
Aligning Projects to Organizational/Strategic Direction	• Product Owner (Team)
Capture Previous Lessons	(Joint) Retrospective
Appoint Project Manager and Team	-
Select Project Approach and Plan Initiation	Product Owner (Team)
Risk Assessment	Product Owner (Team)
Technical Support	Team Members
Contracts Administration	• Product Owner (Team)
Facilities Administration	• Self-organizing Teams • Scrum of Scrums • Meta Scrum
Status Reporting Management	• Self-organizing Teams • Sprint Review • Burndown Chart • Lean Team-Based or Product-Based Boards

Table 8: Traditional MPM in Relation to Lean-Agile MPM

As Table 8 demonstrates, most of the traditional tasks have an agile or lean counterpart, albeit a more lightweight one. This leads to the assumption that they share many similarities, which is partly true. Agile and lean practices also require a high level of discipline. In fact, they demand a level of planning, estimating, engineering and process-improvement far beyond many traditional approaches, since *agility* is not easily attained and maintained [LaVo09].

Therefore, the differences do not necessarily lie in the tasks that are needed in a multi-project environment because these are roughly the same (cf. Table 8). The PMBOK guide e.g. could be used to describe Scrum one way or another, although [PMBOK08] does not mention the term "agile" even once [SpVo10]. They do, however, greatly differ in *how* the multi-project management tasks or goals are accomplished. One of the major differences is hierarchies/management styles. In contrast to Portfolio/Program/Project and even Sub-Project managers, agile and lean processes work with very flat hierarchies: Product Owner (Teams) and self-organizing teams. This agile characteristic alone opens up a completely new perspective on all the other tasks and goals that is not compatible with traditional views any longer.

Dealing with interdependency of projects and their priorities is important in both worlds, although in lean and agile implementations problems usually get addressed more quickly because transparency (through daily meetings and incremental planning/ prioritization e.g.) is one of their highest goals.

The discussion so far has already presented the fundamental differences between traditional and lean-agile MPM approaches. It remains to address the research question more concretely. The outcome of the examination of this research question is that the goals and tasks in multi-project management are roughly the same, but the approaches are completely different, i.e. they represent two different styles. Therefore, trying to change traditional approaches to agile or lean ones is not desirable.

The third and final research question was:

3. *What has to be changed in known practices of Lean and Agile Multi-Project Management to achieve an improvement?*

As the research in chapter 4 has shown, there are a great variety of approaches to both lean and agile multi-project management. The following two chapters discuss their strengths and limitations to find a concluding answer to this main research question.

7.2.1 Strengths

All practices have one great goal in common that is most important in lean-agile multi-project management: providing *transparency*. This is arguably also the ambition in traditional processes, but lean-agile methodologies provide very different approaches:

- The lean idea of mapping and controlling the value stream with *team-based boards* or *product-based ones* (cf. chapter 4.1.1 and [Ande11, KnSk10])
- *Product Owner Teams* (or even *Program Owner Teams*) take care of most of the typical tasks of project management offices in a more lightweight fashion (cf. chapter 4.1.2, Table 8 in previous chapter and [Pich08])

In the case study, no decent level of transparency could be reached. The boards were based on paper, which caused a lot of problems in the distributed development environment. A proposed switch to an electronic tool could not be carried out within the timeframe of this thesis due to resistance of the main supplier.

Prioritization and planning is also central to all lean-agile approaches as it allows teams to *focus* (one of the five Scrum values, cf. [ScBe02]). Various techniques have been suggested:

- *Lookahead Planning* by [Cohn05] involves several steps to solve team interdependencies (cf. chapter 4.1.2)
- Prioritization can be done in *team backlogs* or *project/product backlogs*, which both enforce a different point of view (cf. chapter 4.1.3 and [Glog11])

The *project/product backlogs* are preferred because they support the focus on the bigger picture and thus the customer value. At the end of the evaluation of the case study's approach, the product backlogs of all three products have been merged according to the suggested improvement in chapter 6.4.5. This allows for a better prioritization on the requirement level and leads to a global optimization across all products. This can be regarded as an enhancement of the *project/product backlog* approach by [Glog11].

Communication is essential to all lean and agile practices. In multi-project management it can be scaled to the following layout (cf. chapter 4.1.2 and [Pich08]):

- Daily Scrum
- Scrum of Scrums
- Meta Scrum

The *Daily Scrum* is the regular team status meeting. On top of that a *Scrum of Scrums* is introduced that is attended by team members with impediments that need clarification with other teams. Additionally, the *Meta Scrum* as a third layer is also possible for inter-project coordination (depending on the overall size of the lean-agile process implementation).

The experience gained in the case study confirms that communication is fundamental to multi-project management. The distributed development environment severely handicapped communication, which led to many problems such as continuous stress, miscommunication and code quality issues (cf. chapter 5.3). The result was a planning delay and a significant decrease in transparency.

Figure 44 summarizes the identified shared values among all of the lean-agile multi-project management approaches, their relation and possible implementations (on the right-hand side).

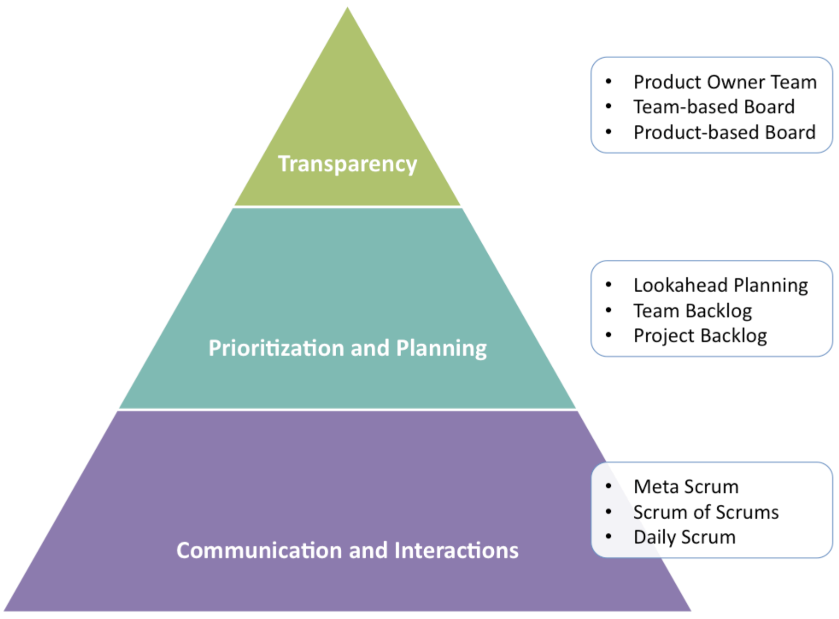


Figure 44: Shared Values of Lean-Agile Multi-Project Management

7.2.2 Limitations

In theory, the lean-agile approaches do not have significant limitations[25]. The reality is that they are mostly applied in traditionally grown companies, which imposes severe constraints on the concepts, e.g. they have to deal with traditional line organization that may disturb the compliance with some or all of the agile values (cf. [ScBe02]):

- Commitment
- Focus
- Openness
- Respect
- Courage

[25] Some shortcomings and preferences have already been stated in the precedent chapter

Many organizations have *Project Management Offices* that take care of most of the Product Owner's duties and responsibilities. Hence, compromises will have to be made. On the other hand, there are also newer organizations that have grown with Scrum and are thus able to faithfully implement the lean and agile approaches to scaling and MPM.

In most cases reality will be different, as agile practices are rather new (Agile Manifesto in 2001, cf. [BeBe01]). Lean values, on the other hand, are long known in automobile industry (cf. [Ohno78]), but have also only recently found their way into software engineering in 2003 (cf. [PoPo04]). Hence, the majority of today's large IT organizations have grown without agile or lean practices.

Nevertheless, these relatively new and evolving approaches offer many benefits that are needed in today's fast-moving world in order to deal with constantly changing requirements. They can also be applied to multi-project management as the case study (chapters 5 and 6) has shown. Many other experience reports by the authors of the concepts presented in chapter 4, such as e.g. [Pich08], [LaVo09] and [Glog11], also support this claim.

Finally, to address the research question more concretely, an improved approach has been implicitly formed by the suggestions in chapter 7.2.1. However, the lean-agile multi-project management practices should not be changed. What needs improvement and change, are strategies for accomplishing the adaption of lean and agile approaches in traditionally grown organizations. These are the key points that deserve further attention, as research (cf. chapter 2.4) mostly focused on ad hoc organization-wide changes that lack practicability.

8 Conclusion

All lean and agile practices strive to provide transparency for the value stream, which gains even more in importance in a multi-project environment. The case study showed that they in fact disclose problems with the workflow very quickly and effectively. Transparency is best achieved via an electronic product-based value stream board, where the progress of all products can be followed in separate swim lanes.

Planning and prioritization also play a central role in all the examined approaches. The techniques were based on either the team's or the product's/project's point of view. The latter is suggested for future use because it enforces the focus on customer value. Prioritization is best achieved with a merged Product Backlog across several products. This allows for a global optimization by taking into account individual requirements rather than a coarse-grained all-or-nothing product prioritization. Furthermore, the case study indicates that regular Product Backlog refinement is also advised to increase understanding of user stories among team members. This minimizes rework and keeps planning meetings more compact.

The last shared value among all the analyzed MPM approaches is communication. It is the most important prerequisite for establishing transparency. All emerging impediments are best dealt with directly in the meetings: from the *Daily Scrum* and *Retrospective* to the *Scrum of Scrums/Meta Scrum*.

Ideally, the following lean-agile implementation is suggested to maximize customer value in a multi-project management environment:

- *Electronic product-based Value Stream Board* with swim lanes for each product for transparency and the maximization of customer value (lean)
- *Product Owner Teams* instead of Project Management Offices for a more lightweight approach, which does not interfere with the teams' self-organization (agile)
- *Merged Product Backlog* to be able to globally optimize the whole project portfolio down to the requirement level (agile/lean)
- *Scaled Meetings*: Daily Scrum, Scrum of Scrums and the optional third-layer Meta Scrum to deal with impediments (agile)

Results show that traditional (such as e.g. the PMBOK and PRINCE 2) and lean-agile multi-project management goals share many similarities. However, the ways in which these are accomplished are completely different. Lean-agile is much more flexible in general, as e.g. incremental planning and prioritization demonstrates. Moreover, since it works with very flat hierarchies and self-organizing teams, it stands in sharp contrast to most of the traditional approaches. Therefore, it is hard to draw conclusions from one to the other. As a result of this finding, trying to change traditional approaches to agile or lean ones is regarded as not reasonable.

In general, lean and agile multi-project management approaches do not have any major limitations. They have only recently been established[26] and are thus mostly applied in traditionally run organizations, which have grown without them. This imposes several constraints, e.g. a Project Management Office may handicap the adoption of agile and lean values. Furthermore, former project managers often find it hard to let go of established hierarchies and work in a self-organizing team. These often end up as *fake Scrum Masters* that do not support the team and process, but rather try to manage it and overtake responsibilities.

There is a great chance that traditionally grown organizations only achieve a superficial adoption of Scrum without proper guidance by experienced coaches. The case study showed that the introduction of agile practices increases transparency and discloses many formerly unknown or overseen problems. The transformation is not easy, as organizational change takes time and requires the involvement and continued dedication of the top management. However, lean and agile practices provide the incentive to include Japanese *kaizen*, i.e. continuous improvement and learning, into the organization and change it for the better in the long run.

It is not surprising that research in the field mainly focuses on business-wide agile transitions because it is an out-of-the-box approach. However, many lack practicability, like e.g. the *Enterprise Scrum*. Not many organizations will switch to full agile adoption in an instance, if ever. It is a slow and gradual transformation that may never spread organization-wide and it does not have to.

The goal is simply to provide a decent environment for a faithful adoption of lean and agile multi-project management. Future research may therefore focus on the benefits and constraints of agile and lean projects in traditionally grown and run organizations and strategies for accomplishing the adoption of core values.

In conclusion, it is up to today’s IT organizations to provide a proper environment for lean-agile multi-project management instead of simply realizing a superficial implementation. Nevertheless, experience shows that compromises will always have to be made in practice to deal with given organizational constraints. That is why it is all the more important to understand underlying agile and lean values for a faithful adoption.

[26] Agile Software Development in 2001, Lean Software Development in 2003

Bibliography

[Ambl09] S.W. Ambler: Scaling Agile Software Development Through Lean Governance. In SDG '09 Proceedings of the 2009 ICSE Workshop on Software Development Governance, May 2009.

[AnAl03] L. Anderson, G.B. Alleman, K. Beck et al.: Agile management - an oxymoron? who needs managers anyway? In OOPSLA '03 Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, October 2003.

[AnBr07] D. Ancona, H. Bresman: X-Teams: How to Build Teams That Lead, Innovate and Succeed. Harvard Business School Press, 2007.

[Ande03] D.J. Anderson: Agile Management for Software Engineering. Applying the Theory of Constraints for Business Results. Prentice Hall, 2003.

[Ande10] D.J. Anderson: Kanban. Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

[Ande11] D.J. Anderson: Kanban. Evolutionäres Change Management für IT-Organisationen. Dpunkt; German Edition, 2011.

[AuCu06] S. Augustine, R. Cuellar: The Lean-Agile PMO. Using Lean Thinking to Accelerate Agile Project Delivery. Agile Project Management Advisory Service; Executive Report Vol. 7 No. 10, 2006.

[BeBe01] K. Beck, M. Beedle, A. van Bennekum et al.: The Agile Manifesto. http://www.agilemanifesto.org, 2001; last accessed January 30th, 2012.

[Beck99] K. Beck: Extreme Programming Explained. Embrace Change. Addison-Wesley, 1999.

[Beck00] K. Beck: Extreme Programming. Das Manifest. Addison-Wesley; 2nd Edition, 2000.

[Beck03] K. Beck: Test Driven Development By Example. Addison-Wesley Professional, 2003.

[BlWo08] W.G. Bleek, H. Wolf: Agile Softwareentwicklung. Werte, Konzepte und Methoden. Dpunkt, 2008.

[BoMi11] J. Boyer, H. Mili: Agile Business Rule Development. Process, Architecture and JRules Examples. Springer, 2011.

[BoTu03] B. Boehm, R. Turner: Balancing Agility and Discipline. A Guide for the Perplexed. Addison-Wesley Professional, 2003.

[BuJe09] M. Budwig, S. Jeong, K. Kelkar: When User Experience Met Agile. A Case Study. In CHI '09 Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, April 2009.

[Cobit11] ISACA. http://www.isaca.org/Knowledge-Center/COBIT/; last accessed July 7th, 2011.

[Cock05] A. Cockburn: Crystal clear. A Human-powered Methodology for Small Teams. Addison-Wesley, 2005.

[CoEd99] R.G. Cooper, S.J. Edgett, E.J. Kleinschmidt: Portfolio Management in New Product Development. Lessons From the Leaders, Phase II. In Project Portfolio Management. Selecting and Prioritizing Projects for Competitive Advantage. Center of Business Practices, 1999.

[Cohn05] M. Cohn: Agile Estimation and Planning. Prentice Hall; 1st Edition, 2005.

[Cohn10] M. Cohn: User Stories. Für die agile Software-Entwicklung mit Scrum, XP u.a. Mitp; German Edition, 2010.

[DaSe10] R. Davies, L. Sedley: Agiles Coaching. Praxis-Handbuch für ScrumMaster, Teamleiter und Projektmanager in der agilen Software-Entwicklung. Mitp; German Edition, 2010.

[DeCl03] M. Denne, J. Cleland-Huang: Software by Numbers. Low-Risk, High-Return Development. Prentice Hall, 2003.

[DeLa06] E. Derby, D. Larsen: Agile Retrospectives: Making Good Teams Great. Pragmatic Programmers, 2006.

[DoKl05] C. Dogs, T. Klimmer: Agile Software-Entwicklung kompakt. Mitp-Verlag; 1st Edition, 2005.

[Finc10] C. Finch: Is an Agile PMO Possible? http://www.projectsmart.co.uk/pdf/is-an-agile-pmo-possible.pdf; last accessed January 30th, 2012.

[Forr05] Forrester Research, Craig Symons: IT Governance Survey Results. April 14th, 2005.

[FrRi06] S. Fraser, L. Rising, S. Ambler et al.: A Fishbowl with Piranhas. Coalescence, Convergence or Divergence? The Future of Agile Software Development Practices. Some Assembly Required! In OOPSLA '06 Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, October 2006.

[Glog11] B. Gloger: Scrum. Produkte zuverlässig und schnell entwickeln. Hanser; 3rd Edition, 2011.

[GrBe10] T. Grechenig, M. Bernhart, R. Breiteneder et al.: Softwaretechnik. Mit Fallbeispielen aus realen Entwicklungsprojekten. Pearson Studium, 2010.

[Gree02] R.K. Greenleaf: Servant Leadership. A Journey into the Nature of Legitimate Power and Greatness. Paulist Press; 25th Anniversary Edition, 2002.

[Gree10] D.R. Greening: Enterprise Scrum. Scaling Scrum to the Executive Level. In HICSS '10 Proceedings of the 2010 43rd Hawaii International Conference on System Sciences, January 2010.

[Gren02] J. Grenning: Planning Poker or How to Avoid Analysis Paralysis While Release Planning. 2002. http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf; last accessed January 30th, 2012.

[Hanf04] M.F. Hanford: Program management. Different from project management. 2004. http://www.ibm.com/developerworks/rational/library/4751.html; last accessed January 30th, 2012.

[Hanf05] M.F. Hanford: Software program management. Questions and answers from an enterprise workshop. 2005. http://www.ibm.com/developerworks/rational/library/jun05/hanford/index.html; last accessed August 11th, 2011.

[HiAl11] M. Hirzel, W. Alter, M. Sedlmayer: Projektportfolio-Management. Strategisches und operatives Multi-Projektmanagement in der Praxis. Gabler; 3rd Edition, 2011.

[High04] J. Highsmith: Agile Project Management. Creating Innovative Products. Addison-Wesley; 1st Edition, 2004.

[HoNo10] R. Hoda, J. Noble, S. Marshall: Organizing Self-Organizing Teams. In ICSE '10 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, May 2010.

[Itil11] IT Infrastructure Library. http://www.itil.org/; last accessed July 7th, 2011.

[JaSu09] A. Janes, G. Succi: To Pull or Not to Pull. In OOPSLA '09 Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and application, October 2009.

[KaLi07] S. Kaner, L. Lind, C. Toldi et al.: Facilitator's Guide to Participatory Decision-Making. Jossey-Bass, 2007.

[Katz82] R. Katz: The Effects of Group Longevity on Project Communication and Performance. Administrative Science Quarterly; Vol 27, March 1982.

[Knib07] H. Kniberg: Scrum and XP from the Trenches. How we do Scrum. InfoQ, 2007.

[KnSk10] H. Kniberg, M. Skarin: Kanban and Scrum. Making the Most of Both. C4Media, 2010.

[Köhl06] P.T. Köhler: PRINCE 2. Das Projektmanagement-Framework. Springer, 2006.

[Kruc00] P. Kruchten: The Rational Unified Process: An Introduction. Prentice Hall; English Edition, 2000.

[KtLé09] O. Ktata, G. Lévesque: Agile development. Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In C3S2E '09 Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering, May 2009.

[Lada08] C. Ladas: Scrumban. Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press, 2008.

[Larm04] C. Larman: Agile and Iterative Development. A Manager's Guide. Addison-Wesley, 2004.

[LaVo09] C. Larman, B. Vodde: Scaling Lean & Agile Development. Thinking and Organizational Tools for Large-Scale Scrum. Addison-Wesley, 2009.

[Lean11] Lean Enterprise Institute. http://www.lean.org/whatslean/principles.cfm; last accessed July 18th, 2011.

[Lenc10] P.M. Lencioni: Die fünf Dysfunktionen eines Teams. Wiley-VCH; German Edition, 2010.

[LiMe07] J. Liker, D. Meier: Toyota Talent. Developing Your People the Toyota Way. McGraw-Hill Professional, 2007.

[MeMa99] J.R. Meredith, S.J. Mantel: Project Selection. In Project Portfolio Management. Selecting and Prioritizing Projects for Competitive Advantage. Center of Business Practices, 1999.

[Meye03] M. Meyer: IT-Governance. Begriff, Status quo und Bedeutung. Wirtschaftsinformatik 45, 2003.

[Mgs11] Mountain Goat Software. http://www.mountaingoatsoftware.com/scrum/overview; last accessed July 19th, 2011.

[NaZu09] A.I. Nawaz, I.A. Zualkernan: The Role of Agile Practices in Disaster Management and Recovery. A Case Study. In CASCON '09 Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, November 2009.

[Nort06] D. North: Behavior Modification. The evolution of behavior-driven development. In Better Software Magazine; Issue March, 2006.

[Ohno78] T. Ohno: Toyota Seisan Houshiki. Diamond, 1978.

[Ohno88] T. Ohno: Toyota Production System. Beyond Large-scale Production. Productivity, 1988.

[Ohno09] T. Ohno: Das Toyota-Produktionssystem. Campus Verlag; New Edition, 2009.

[Opm308] Project Management Institute: Organizational Project Management Maturity Model (OPM3). ANSI/PMI 08-004-2008; 2nd Edition, 2008.

[PaBu09]: T.A. Pardo, G.B. Burke: IT Governance Capability: Laying the foundation for government interoperability. Albany, NY: Center for Technology in Government, 2009.

[PeWo09] K. Petersen, C. Wohlin: A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. In Journal of Systems and Software; Vol. 82, Issue 9, 2009.

[Pich08] R. Pichler: Scrum. Agiles Projektmanagement erfolgreich einsetzen. Dpunkt; 1st Edition, 2008.

[Pmbok08] Project Management Institue: A Guide To The Project Management Body Of Knowledge (PMBOK Guide). ANSI/PMI 99-001-2008, 4th Edition, 2008.

[Pmi08] Project Management Institute: The Standard for Program Management. ANSI/PMI 08-002-2008, 2nd Edition, 2008.

[PoOl08] D. Port, A. Olkov: Using Simulation to Investigate Requirements Prioritization Strategies. In ASE '08 Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, September 2008.

[PoPo04] M. Poppendieck, T. Poppendieck: Lean Software Development. An Agile Toolkit. Addison-Wesley; 4th Edition, 2004.

[Prie08] R. Prieto: Foundations, Frameworks and Lessons Learned in Program Management. 2008.
http://www.pmhut.com/?s=%22Foundations%2C+Frameworks+and+Les

sons+Learned+in+Program+Management%22; last accessed July 5th, 2011.

[Prince211] PRINCE2: Projects in Controlled Environments. http://www.prince2.com/; last accessed January 30st, 2012.

[Royc70] W.W. Royce: Managing the Development of Large Software Systems: Concepts and Techniques. In IEEE WESCON, August 1970.

[Rung08] M. Rungi: Effective project selection and portfolio management through relationships between projects. In IPMA '08 Proceedings of the 22nd International Project Management Association World Congress, November 2008.

[Rung09] M. Rungi: Managing Resource and Technology Interdependencies in Project Portfolio. A Case-Study Results. In IEEE '09 Proceedings of the 2009 IEEE Industrial Engineering and Engineering Management, December 2009.

[Rung10a] M. Rungi: Management of Interdependency in Project Portfolio. Doctoral Thesis; Lappeenranta University of Technology, Finland, 2010.

[Rung10b] M. Rungi: Relationships between projects in project portfolio management. A content analysis of techniques. In Proceedings of the Project Management Institute (PMI) Research and Education Conference, July 2010.

[ScBe02] K. Schwaber, M. Beedle: Agile Software Development with Scrum. Prentice Hall, 2002.

[ScBi10] A. Schatten, S. Biffl, M. Demolsky et. al.: Best Practice Software-Engineering. Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen. Spektrum Akademischer Verlag, 2010.

[Schw04] K. Schwaber: Agile Project Management with Scrum. Microsoft Press, 2004.

[Schw08] K. Schwaber: Scrum im Unternehmen. Microsoft Press, 2008.

[ScPi06] I. Schnabel, M. Pizka: Goal-Driven Software Development. IEEE Computer Society, 2006.

[SpVo10] N. Spitczok von Brisinski, G. Vollmer: Pragmatisches IT-Projektmanagement. Softwareentwicklungsprojekte auf Basis des PMBOK Guide führen. Dpunkt; 1st Edition, 2010.

[Suth05] J. Sutherland: Future of Scrum. Parallel Pipelining of Sprints in Complex Projects. In ADC '05 Proceedings of the Agile Development Conference, July 2005.

[TaDu09] D. Talby, Y. Dubinsky: Governance of an agile software project. In SDG '09 Proceedings of the 2009 ICSE Workshop on Software Development Governance, May 2009.

[TjKa05] A M. Tjoa, D. Karagiannis: IT Governance Definition, Standards & Zertifizierung. Österreichische Computer Gesellschaft; OCG Journal Edition 4, 2005.

[Vall11] R. Vallon: Lean and Agile Software Development. Planung und Realisierung einer Verbindung von Kanban und Scrum. Master Thesis; Vienna University of Technology, 2011.

[VäRa08] J. Vähäniitty, K. Rautiainen: Towards a Conceptual Framework and Tool Support for Linking Long-term Product and Business Planning with Agile Software Development. In SDG '08 Proceedings of the 1st international workshop on Software development governance, May 2008.

[Wird09] R. Wirdemann: Scrum mit User Stories. Hanser, 2009.

[WoJo92] J.P. Womack, D.T. Jones, D. Roos: Die zweite Revolution in der Autoindustrie. Konsequenzen aus der weltweiten Studie aus dem Massachusetts Institute of Technology. Campus Verlag; 5th Edition, 1992.

[WoJo96] J.P. Womack, D.T. Jones: Lean Thinking. Banish Waste and Create Wealth in Your Corporation. Productivity Press; 1st Edition, 1996.