

Towards a Methodology-Growing Framework

MASTER'S THESIS

Author

Richard Cornelissen
University of Twente
r.cornelissen@student.utwente.nl

Supervisors (University of Twente)

Lodewijk Bergmans
Christoph Bockisch

Supervisors (Topicus)

Tom Palsma
Wouter Pollmann

November 21, 2013

Abstract

Software development methodologies instruct teams how to collaborate on developing software for their project. Methodologies include, among others, procedures, tasks, techniques, guidelines, and tools. As each software project is unique, each requires its own, tailored methodology, or *way of working*. With an inadequate methodology, or one that remains unadopted by the team, the project could end in failure.

The methodology-growing technique defined by Cockburn allows to create a suitable methodology for a software project. This technique does not support existing projects, though. Furthermore, it does not provide suggestions and guidance what to change to the methodology.

This thesis proposes to extend the methodology-growing technique into a framework that is able to support organizations in attaining suitable and adopted methodologies for software projects. Properties of the software project are collected that influence decisions on its methodology. These are used to prepare and hold incremental reflection workshops. During these workshops, team members collaboratively improve the methodology, making decisions on their previous experiences and on the properties of the project.

We propose to define methodologies as a composition of software development practices. Practices can be documented, reused and incorporated into methodologies when they are applicable. We propose to extend documented practices with their usage criteria. These descriptions indicate when a practice is suitable and how it can be adjusted to the project.

The methodology-growing framework has been applied at a small-scale project to improve its methodology. Furthermore, surveys have been held with employees at the organization where the project is located. The case study and evaluation together show the framework is usable to improve the methodologies of software projects.

Acknowledgements

The following people have supervised the research and the writing of this thesis. This research took place in a project called Findesk at Topicus, an organization located mainly in Deventer, The Netherlands.

dr.ir. Lodewijk M.J. Bergmans

The first supervisor from the University of Twente, with whom I have discussed in length about methodologies, relevant literature, changing the methodology in practice, the design of the framework and the writing of this thesis.

dr.ing. Christoph M. Bockisch

The second supervisor as well as my mentor from the University of Twente, who has helped in structuring and improving this thesis.

Tom Palsma, MSc

Supervisor and project manager at Findesk, who has helped apply the designed framework together with his team. He has also provided guidance, feedback and support in writing this thesis.

Ir. Wouter Pollmann

Supervisor and director at Findesk, who has guided me in designing the framework as a whole and motivated me to use it in practice at Findesk. He has also introduced me to the idea of discussing the framework with the interviewees listed below.

Interviewees

The following people are employees at Topicus who are experienced in different aspects of software development. They have helped me in designing the framework through extended discussions. Furthermore, they have filled in surveys for evaluation and have provided detailed comments.

- Marco van de Haar – Senior software developer.
- Sander Hofstee – Senior software developer.

-
- Nicky Koenders – Product owner for a project using Scrum.
 - Sander van Loon – Process engineer.
 - Marco Scholten – Senior software developer.
 - Daan Verbree – Product manager and team leader.

Lastly, I would like to thank all employees at Topicus for providing a fun place to perform my final project, especially those who have helped me evaluate the methodology-growing framework.

Contents

1	Introduction & Motivation	1
1.1	Background	1
1.2	Research objectives	3
1.2.1	Main objective	3
1.2.2	Research questions	3
1.2.3	Definitions	4
1.3	Approach	5
1.4	Evaluation approach	6
2	Methodology Design	7
2.1	Conceptual terms	7
2.2	Methodology design principles	9
1.	Interactive, face-to-face communication is the cheapest and fastest channel for exchanging information	9
2.	Excess methodology weight is costly	9
3.	Larger teams need heavier methodologies	10
4.	Greater ceremony is appropriate for projects with greater crit- icality	10
5.	Increasing feedback and communication reduces the need for intermediate deliverables	11
6.	Discipline, skills and understanding counter process, formality and documentation	11
7.	Efficiency is expendable in non-bottleneck activities	12
3	Methodology Composition	15
3.1	SEMAT Kernel	16
3.1.1	Alphas	17
3.1.2	Activity spaces	18
3.2	Practice library	19
3.2.1	Purpose	21

3.2.2	Prerequisites	21
3.2.3	Necessary commitment	22
3.2.4	Tailoring	22
3.3	Documenting software development practices	23
4	Project Inventory	27
4.1	Project properties	28
4.2	Team member properties	32
4.3	Solution properties	33
4.4	Methodology properties	34
4.5	Project inventory interview	36
5	Methodology-Growing Framework	39
5.1	Composing the methodology	39
5.2	Applying the framework	44
	Step 1. Project inventory	44
	Step 2. First practice attempt	45
	Step 3. Preparing reflection workshops	45
	Step 4. Holding reflection workshops	48
	Step 5. Changing the methodology	50
	Step 6. Middle of the increment	51
5.3	Start of a new project	51
5.3.1	First proposal	52
5.3.2	First reflection workshop	52
5.3.3	First practice attempt	52
5.4	Summary	53
6	Case Description	55
6.1	Project inventory	55
6.1.1	Project properties	56
6.1.2	Team member properties	58
6.1.3	Solution properties	59
6.1.4	Methodology properties	60
6.1.5	Summary	62
6.2	First iteration	63
6.2.1	First practice attempt	63
6.2.2	Preparation of the reflection workshop	63
6.2.3	First reflection workshop	65

6.2.4	Changing the methodology	67
6.3	Second iteration	69
6.3.1	Preparation of the reflection workshop	69
6.3.2	Second reflection workshop	71
6.3.3	Changing the methodology	73
6.4	Reflection on the methodology	75
7	Evaluation	77
7.1	Employee survey	77
7.1.1	Survey results	78
7.1.2	Summary	80
7.2	Team member & experienced employee survey	81
7.2.1	Survey results	81
7.2.2	Additional survey results	86
7.2.3	Summary	87
8	Conclusion	89
8.1	Conclusions	89
8.2	Contribution	93
8.3	Business recommendations	94
8.4	Limitations and future work	95
8.5	Concluding remarks	97
	Bibliography	99
A	Experienced employee interviews	104
A.1	First interview	105
A.2	Second interview	107
B	Practice Library	109
B.1	Weekly Cycle	111
B.1.1	Usage criteria	112
B.1.2	Alphas (things to work with)	113
B.1.3	Work products (artifacts to maintain)	114
B.1.4	Activities (things to do)	115
B.2	Daily Standup	117
B.2.1	Usage criteria	118
B.2.2	Alphas (things to work with)	119

B.2.3	Activities (things to do)	120
B.3	MoSCoW Prioritization	121
B.3.1	Usage criteria	122
B.3.2	Alphas (things to work with)	123
B.3.3	Work products (artifacts to maintain)	124
B.3.4	Activities (things to do)	125
B.4	Pair Programming	127
B.4.1	Usage criteria	128
B.4.2	Alphas (things to work with)	129
B.4.3	Activities (things to do)	130
B.5	Seasons of the Day	131
B.5.1	Usage criteria	132
B.5.2	Alphas (things to work with)	133
B.6	Visualize Workflow	135
B.6.1	Usage criteria	136
B.6.2	Alphas (things to work with)	137
B.6.3	Work products (artifacts to maintain)	138
B.6.4	Activities (things to do)	139
C	Project inventory interview	143
D	Social adoption measurement form	147
E	Questionnaires	149
E.1	Evaluation by employees	150
E.2	Evaluation at Findesk	153
E.3	Evaluation with experienced employees	155
F	Survey results	157
F.1	Employee survey results	157
F.2	Findesk team member survey results	158
F.3	Experienced employee survey results	159

Chapter 1

Introduction & Motivation

This chapter introduces software development methodologies and states the goal of this thesis. The problem statement is defined and research questions are stated that have specified the focus of the performed research.

§

This thesis will propose a framework for attaining and improving methodologies within a project. It has been applied within a single case to improve the methodology of a two year old project in the financial domain.

1.1 Background

Definition of methodology

A software development methodology can be defined as a means to achieve the development of systems, based on an underlying philosophy. Methodologies can include, among others, phases, procedures, tasks, rules, techniques, guidelines and tools [4, 30].

Benefits

Methodologies are useful for a number of reasons [4, 12, 25, 30]:

- Methodologies can increase transparency of the development process, which facilitates management and control of the project and thereby reduces risks and uncertainty.
- They can increase productivity and quality as necessary resources can be predicted beforehand.
- Methodologies divide the complex process of software development into plausible, consistent steps.

Furthermore, according to Cockburn [8], methodologies can

- Introduce new people to their jobs.
- Delineate responsibilities.
- Show progress.
- Provide a curriculum for education.

Agile methodologies

The Agile Manifesto provides a guideline for a new class of methodologies that emerged in the mid-nineties: the agile methodologies [6]. The Manifesto includes twelve principles for agile methodologies, focusing on customer satisfaction, changing requirements, frequent delivery, and other aspects to software development. Agile methodologies have gained increasing acceptance in the IT industry, as a number of surveys show [1, 26, 28, 29, 32], with up to 80% of the respondents indicating they use some agile methodology.

Surveys show agile methodologies are adopted for the following reasons [26, 32]:

- They shorten time to market.
- They manage changing requirements.
- They increase productivity and quality.
- They increase predictability and better align business and IT.
- They reduce waste.

Adoption

Usage of methodologies is hard to measure because of bias and focus on specific methodologies [10]. Surveys and research indicate, though, that many organizations are not using any methodology [12, 13, 14]. Among other reasons, this is caused by the selection of methodologies by management, who see more benefits to their use than developers, resulting in an unadopted methodology [14].

Furthermore, using an inappropriate or inadequate methodology can result in development failures [33]. Every project calls for a different methodology that fits both the project and the problem [8]. Attaining a methodology that is appropriate and adequate for the project, as well as accepted and adopted by developers is therefore important.

Tailoring

Fitzgerald suggests that methodologies that are tailored to the organization are more likely to be adopted [11]. A methodology is tailored to the needs of the development environment, for which parts of the methodology can be omitted

or described in a broader sense. Instead of describing the exact way development should take place, the tailored methodology often specifies activities and objectives in less detail.

In order to tailor a methodology, Cockburn has designed a methodology-growing technique [8]. With it, teams can construct and tune their own methodology “*on-the-fly*”, beginning with a selected base methodology and iteratively tuning it to better fit the project and the team members. Tuning is achieved by holding reflection workshops. In these workshops, teams discuss what they have learned during an increment and decide what they will improve.

Problem statement

The methodology-growing technique provides a guideline for iteratively tailoring methodologies. In doing so, teams look into “*things to try*” for their methodology. The technique does not, however, include a manner of composing methodologies. It also does not provide suggestions of practices that can be applied within the project. If teams want to look into new or alternative practices, they are required to do their own research.

Cockburn’s technique also does not provide instructions for modifying the methodology of an existing project. Applying the technique to an existing project would require it to replace its entire methodology. According to Anderson, this would result in an unadopted methodology, as team members are likely to resist the change [3].

1.2 Research objectives

1.2.1 Main objective

This thesis will extend the methodology-growing technique to both support existing projects and to suggest new practices that can be applied within projects. The main objective of this thesis is:

Designing a framework able to support organizations in attaining adequate agile software development methodologies for new and existing software projects.

1.2.2 Research questions

The following research questions will be the guideline to design such a framework.

1. For a given software project, how can suitable software development practices be identified and incorporated into a coherent methodology?

To improve a methodology, Cockburn indicates teams look into possible improvements, but does not include guidelines toward identifying these [8]. The following subquestions aid in the answer of the main research question:

- a) How can software development practices be documented?
- b) How can usage criteria of documented practices be described?
- c) How can software development practices be incorporated into a coherent methodology?

2. How can software development practices be selected and adopted in both new and existing software projects?

Previous work in this area has not addressed how both new and existing projects can attain a suitable methodology. Many techniques only support newly starting projects or replacing the current methodology entirely, such as the methodology-growing technique by Cockburn and the decision model introduced by Vavpotič and Vasilecas [8, 31]. As explained earlier in this chapter, replacing the entire methodology is likely to result in an unadopted methodology.

The goal of this thesis is not to help projects achieve a theoretically perfectly suitable methodology, but rather to help them attain a methodology team members will truly adopt. The following subquestions aid in the answer of the main research question:

- a) Which properties of software projects are important for making decisions on a methodology?
- b) How do identified properties of software projects apply to new or existing software projects?
- c) How do identified properties of software projects affect decisions regarding a methodology?
- d) By what method can software development practices be adopted in a methodology?

1.2.3 Definitions

The main objective and research questions use terms that are defined next:

Software development practice A repeatable approach to doing something with a specific purpose in mind [16]. For software engineering, practices provide guidance to deal with some dimension of software development.

Software development methodology A means to achieve the development of systems, based on an underlying philosophy. Methodologies can include, amongst others, phases, procedures, tasks, rules, techniques, guidelines and tools [4, 30].

Software project A software project can be defined as an endeavor in which people collaborate on developing software towards a solution.

1.3 Approach

To answer the research questions and accomplish the stated goal, the following approach has been used:

- Existing design principles for creating and adjusting methodologies are described in Chapter 2.
- A meta-model for methodologies to decompose them in smaller, reusable parts has been selected and discussed in Section 3.1.
- A library of reusable practices, appended with a description of their purpose, prerequisites, necessary commitment, and how they can be tailored, is described in Section 3.2.
- A ways of documenting reusable practices has been defined in Section 3.3.
- Properties of software projects that are important for making decisions on a methodology have been identified and discussed in Chapter 4.
- Checklists for composing methodologies out of reusable practices are provided in Section 5.1.
- A framework extending the methodology-growing technique is discussed in Section 5.2.
- This framework has been applied in practice at a small-scale, two year old project, as discussed in Chapter 6.
- Surveys are held amongst employees at Topicus, with the team members of the project at which the framework was applied, and with a number of employees experienced with software development and methodologies. The surveys and the results are discussed in Chapter 7.

Finally, Chapter 8 answers the research questions and discusses the contributions and limitations of this thesis.

1.4 Evaluation approach

This thesis has led to the design of a framework for improving software development methodologies. It has been evaluated within Topicus, where the research took place.

Firstly, the framework has been applied in practice at a small-scale project in a growing organization. This case study is described in Chapter 6. Team members of the project were asked to complete a survey on their opinion of the framework.

Secondly, six experienced employees within the organization have aided in the design of the framework by discussing it in detail. The discussions were held as two semi-structured interviews. The used forms are included in Appendix A. During the discussions, the framework was explained in detail. Their comments and feedback have aided in the design of the framework. They have also been asked to complete a survey on their opinion of the framework. This survey was an extended version of the one completed by the team members.

Lastly, employees within the organization have been asked to fill in a short survey. This survey determined whether the framework is in line with how employees with different roles and responsibilities would want to improve their methodology.

Chapter 2

Methodology Design

This chapter discusses concepts and design principles for software development methodologies. These are used to discuss and make decisions on methodologies.

§

Conceptual terms that are used for describing methodologies, as defined by Cockburn [8], are first introduced in Section 2.1. These are also used in Section 2.2, where methodology design principles are discussed.

2.1 Conceptual terms

Cockburn discusses the design of methodologies using the conceptual terms described in this section [8]. These definitions are used in the methodology design principles described in Section 2.2. Some of the terms described by Cockburn have not been included in this section as they remain unused within the rest of this thesis.

Methodology size

The size of a methodology is defined as the number of control elements included in it. These include deliverables, standards, conventions, activities, quality measures and technique descriptions.

A larger methodology, with more control elements, is said to be more prescriptive, while a lighter methodology is more adaptive [21].

Ceremony

The ceremony of a methodology is the amount of precision and the tightness of tolerance, both explained in this section later on. The necessary amount of

ceremony of a methodology depends on the system criticality which is explained below.

The past experience of the author of the methodology also affects its included ceremony. Authors tend to include additional ceremony on everything they have seen gone wrong.

Methodology weight

The weight of a methodology is the product of its size and ceremony. This is a conceptual product, as it cannot be expressed as a number. The term methodology weight is used to indicate and compare the size and ceremony of methodologies.

Problem size

The problem size is defined as the number of elements in the problem and their cross-complexity. This indicates how hard it is to solve a problem. The problem size cannot be shown as an absolute measure, but the difficulty of different problems can often be compared.

Project size

The project size is the number of people whose efforts need to be coordinated. Depending on the project, the methodology might only need to coordinate development efforts, or might need to coordinate an entire department with different roles.

System criticality

The criticality of a system is the potential damage an undetected defect in the system can bring. Cockburn recognized four main classes [7].

- **Loss of comfort**

With a system failure, only comfort is lost, such as having to do what the system automatically does by hand. Purchase support systems fall in this category.

- **Loss of discretionary moneys**

System failure results in loss of money, but only in the range of discomfort, such as the failure of an invoicing system. The loss of money can be recovered.

- **Loss of irreplaceable moneys**

If the system would fail, any money lost cannot be recovered by hand or after system recovery. Bank account systems fall under this category.

- **Loss of life**

System failure will put human life will be at stake. A well-known example would be system failures at nuclear power plants.

Stability

Stability is defined as the likelihood that something will change. Cockburn identifies three stability states:

- Wildly fluctuating, in which there is great likelihood something will change. This is often the case when development has just started.
- Varying, in which the details are likely to change. The stability of development is often in this state in the middle of an increment.
- Relatively stable, where there is a limited amount of things that can still change. This is the case at the end of a successful increment.

2.2 Methodology design principles

Cockburn has documented seven principles for designing and evaluating methodologies [8].

1. Interactive, face-to-face communication is the cheapest and fastest channel for exchanging information

Creating software is easier and less expensive if the entire team is sitting together and has frequent and easy direct contact.

As the problem size increases, direct communication will become more difficult to arrange, as more people are needed to solve the problem (also see the third principle). This results in increased cost of communication, as well as decreased quality of communication and more difficulty in developing the software.

This principle does not entail all software can nor should be developed by a small group of people in a single room. Principle 3 describes that as the problem size increases, the project size needs to increase to cope.

2. Excess methodology weight is costly

Letting team members spend time producing artifacts, such as intermediate work products, charts, documents or plans, takes time away from development. Adding elements to the methodology means adding workload to the team, which

What size problem can a given number of people attack, using various methodology weights?

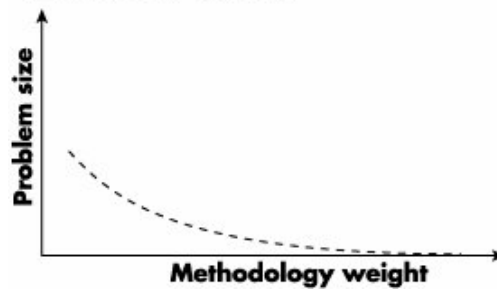


Figure 2.1: Problem size and methodology weight for a given project size [8].

decreases productivity. Adding a seemingly small amount of this methodology weight will add a large cost. Figure 2.1 shows the effect of adding methodology weight and shows that a fixed team size can often work on a larger problem with a lighter methodology.

Removing methodology elements can increase productivity of the team. But eventually, this approach can affect the elements that address (code) quality, in which case the removal of elements can backfire. Therefore, only *excess* weight should be removed, though what is excess is hard to determine. Cockburn suggests starting with a lighter methodology and adding elements where necessary.

3. Larger teams need heavier methodologies

As the project size increases, it becomes less clear what each team member is doing within the project. More coordination is required to ensure their work does not overlap or interfere with each other. In this case, heavier methodologies are necessary to offer this coordination. The relation between methodology weight and problem size, with a large number of people, is shown in Figure 2.2.

Principle 2 also applies to heavy methodologies, as shown in the right-most part of Figure 2.2.

4. Greater ceremony is appropriate for projects with greater criticality

The criticality of a project is an indication of the potential damage if the developed system were to fail. The ceremony of a methodology is the amount

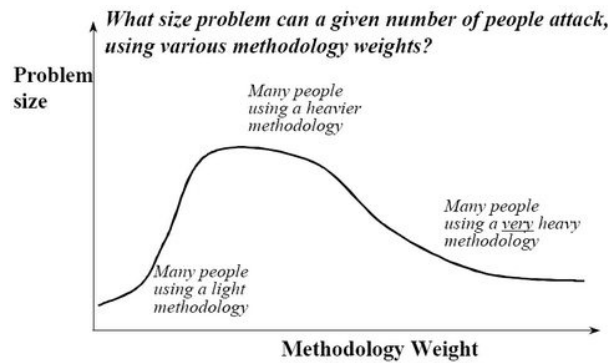


Figure 2.2: Problem size and methodology weight for a high project size [8].

of precision and the tightness of tolerance included in it. As the criticality increases, there is less tolerance for error. Additional costs to prevent defects in the system become justified and tighter control mechanisms within the methodology are necessary.

According to the second principle, additional methodology weight is costly. In projects with high criticality, however, this weight is not excess, it is added to prevent potential costs.

5. Increasing feedback and communication reduces the need for intermediate deliverables

Intermediate deliverables, such as refined requirements documents, are used *internally* (within the team) to facilitate decisions. There are two ways to reduce the need of these deliverables:

- Deliver a working increment of the system quickly enough for rapid feedback that will indicate whether the system is developing in the right direction.
- Reduce the team size so that everyone is close enough for direct communication. This way, no internal documents are necessary.

6. Discipline, skills and understanding counter process, formality and documentation

- **Documentation is not the same as understanding.** Most knowledge within a project is tacit knowledge, which cannot be documented. Only a small part of everything necessary to know within the project can be documented.

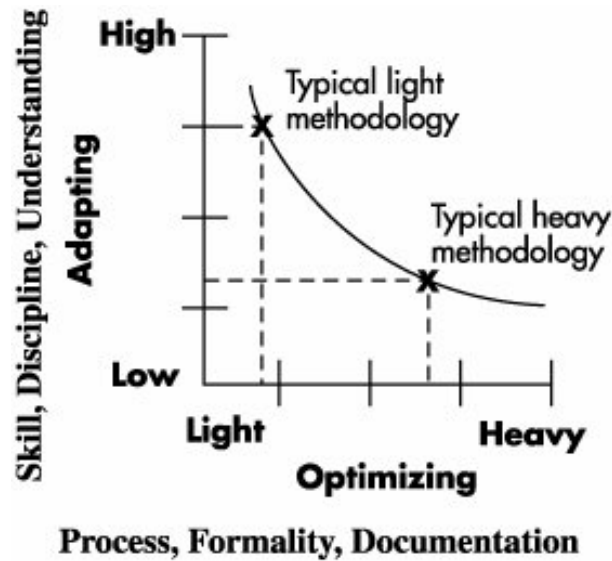


Figure 2.3: Discipline, skills and understanding counter process, formality and documentation [8].

- **Process is not the same as discipline.** In case of a process, people follow instructions that have been defined for them. Discipline means people have *chosen* to work in a consistent way. Therefore, process does not necessarily impart discipline.
- **Formality is not the same as skill.** Filling in forms cannot replace the skills of people. System architecture designers, for example, will apply use cases to create designs of the system and will rework this over time to improve it. No formality will be able to replace this skill.

Heavy methodologies often rely on process, formality and documentation, which *optimize* activities. These methodologies work better when the problem domain is well known and there is no need to adapt to changes. This gives the project the chance to optimize its costs. Lighter methodologies cling to discipline, skills and understanding, which can be labeled as *adapting*. This difference is displayed in Figure 2.3. Lighter methodologies are better when circumstances change and when the problem domain is not well known.

7. Efficiency is expendable in non-bottleneck activities

A bottleneck activity is one whose speed determines the speed of the entire project. Take, for instance, a project with a group of developers and a single

database administrator (DBA). All developers generate work for the DBA, but the DBA is unable to keep up with the generated workload.

Such a situation can be improved by getting work to a more complete and stable state before passing it to the bottleneck activity. In the example, developers could spend more time drawing designs so the DBA can understand what has to be done more easily. The bottleneck activity can then be done as efficiently as possible, with the least amount of rework. The activities before the bottleneck activity will contain more rework, thus working less efficient, but enabling the bottleneck to be efficient.

Chapter 3

Methodology Composition

This chapter proposes a way to decompose software development methodologies into smaller practices. In doing so, practices can be documented and extended with their usage criteria for reusability.

§

Methodologies are often seen as a composition of smaller parts into a whole. For instance, Cockburn describes methodologies as a structure of 13 types of related elements [8]. Another example is the Software & Systems Process Engineering Metamodel (SPEM), a meta-model for defining processes, such as entire methodologies [23]. Elements of a methodology can be added, replaced, discarded, altered, and adjusted to the project where they are applied.

In this thesis, we define methodologies as a composition of software development practices, as proposed by the SEMAT initiative [18]. Software Engineering Method and Theory (SEMAT) is an initiative started in 2009 by Jacobson, Meyer, and Soley “*with the aim of refounding software engineering as a rigorous discipline*” [17]. In contribution to this goal, a kernel in which to describe methodologies and practices has been defined so they can be composed, simulated, applied, compared, evaluated, measured, taught, and researched [16, 19, 24].

This chapter first describes the SEMAT Kernel in Section 3.1. Section 3.2 extends the documentation possible with the Kernel, proposing that by adding usage criteria of practices, they can be stored in a library and reused. Finally, Section 3.3 provides instructions on how practices can be documented, using the Weekly Cycle as an example [5].

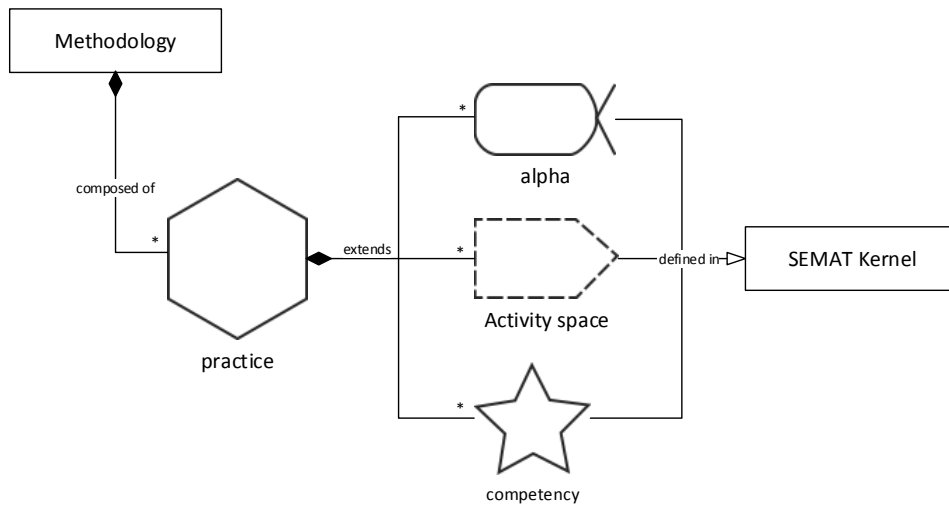


Figure 3.1: Methodology composition using the SEMAT kernel.

Definition of software development practice

SEMAT defines a practice as a repeatable approach that provides guidance to deal with some dimension of software development [16].

3.1 SEMAT Kernel

The SEMAT Kernel is divided into three areas of concerns: the *Customer*, the *Solution*, and the *Endeavor* [16]. Each area of concern contains alphas, activity spaces, and competencies, explained below.

Figure 3.1 depicts how methodologies are composed of practices, which can be documented using the Kernel. The symbols defined by SEMAT is used in combination with UML class diagram notation to show relationships. Practices, alphas, activity spaces, activities, and competencies all have their own symbols, which have been annotated with their name for clarification.

Note that SEMAT uses the composition notation also present in UML differently, for instance to indicate a practice extends an alpha from the Kernel [24]. In this chapter, only the symbols of SEMAT are used. In Appendix B, however, all diagrams are in SEMAT notation.

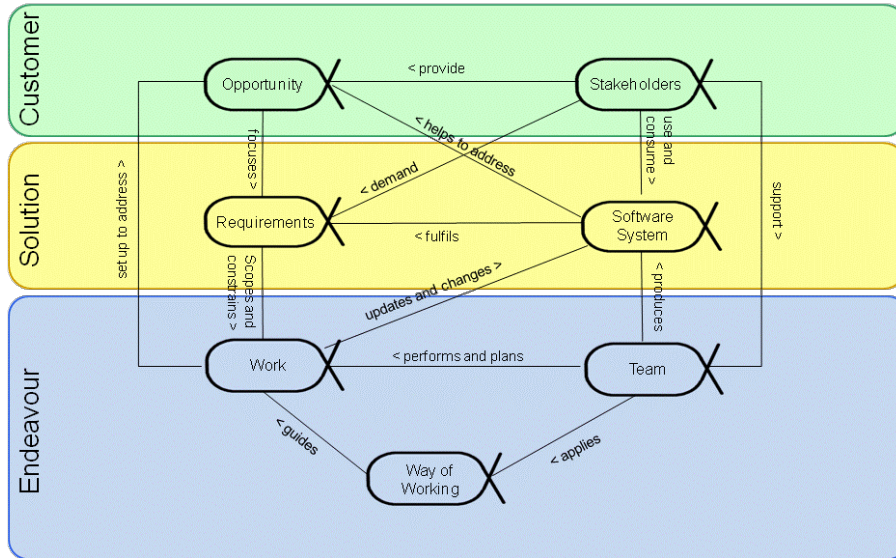


Figure 3.2: Alphas in the SEMAT Kernel [24].

3.1.1 Alphas

Alphas are representations of essential things to work with when developing software [16]. Figure 3.2 shows all alphas included in the Kernel and their relations. For instance, the Endeavor area of concern includes the *Team* that applies its *Way of Working*, which in turn guides the *Work* that the team performs. Alphas have states that reflect the current status of the development endeavor. The *Software System*, for instance, can be in states *Architecture Selected* through *Retired*.

Figure 3.3 shows the structure of alphas and sub-alphas that extend their parent alpha. States are defined which alphas can be in. An alpha enters a state when its defined checkpoints are achieved. The *Requirements* alpha, for instance, is in the *Addressed* state when the following checkpoints are achieved:

- Enough of the requirements are addressed for the resulting system to be acceptable to the stakeholders.
- The stakeholders accept the requirements as accurately reflecting what the system does and does not do.
- The set of requirement items implemented provide clear value to the stakeholders.
- The system implementing the requirements is accepted by the stakeholders as worth making operational.

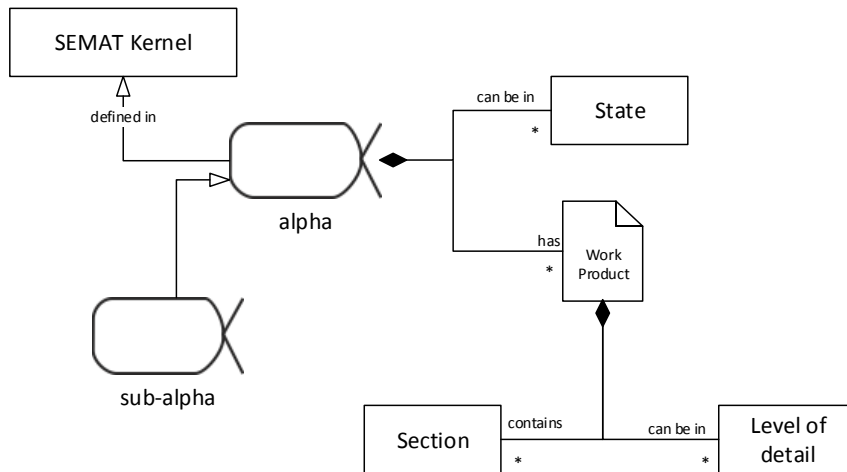


Figure 3.3: Structure of (sub-)alphas.

Adding a sub-alpha beneath an extended alpha implies the sub-alpha contributes to its parent alpha in some way. A multiplicity is added to this relation, for instance to show that the *Week* sub-alpha of the *Weekly Cycle* occurs multiple times under *Work*, as further explained in Section 3.3.

Alphas and sub-alphas can contain work products. A multiplicity is added to show the possible number of each work product type, for example “1..*”. The work products can be in defined levels of detail and can contain multiple sections describing its contents.

Practices can also add additional details to alphas they extend from the Kernel. For instance, the *Requirements* alpha has an additional state *Prioritized* with the *MoSCoW Prioritization* practice, described in Appendix B.3.

3.1.2 Activity spaces

Activity spaces contain essential things to do while developing software [16]. Examples of activity spaces are *Implement the System*, and *Explore Possibilities*. Note that these are not activities but that practices add activities to their corresponding activity space. See Figure 3.4 for all activity spaces included in the Kernel, grouped by area of concern.

Activities described in a practice are added under their corresponding activity spaces, which are extended by the practice. This is shown in Figure 3.5. Activities have any number of (sub-)alphas as input, on which some operation is

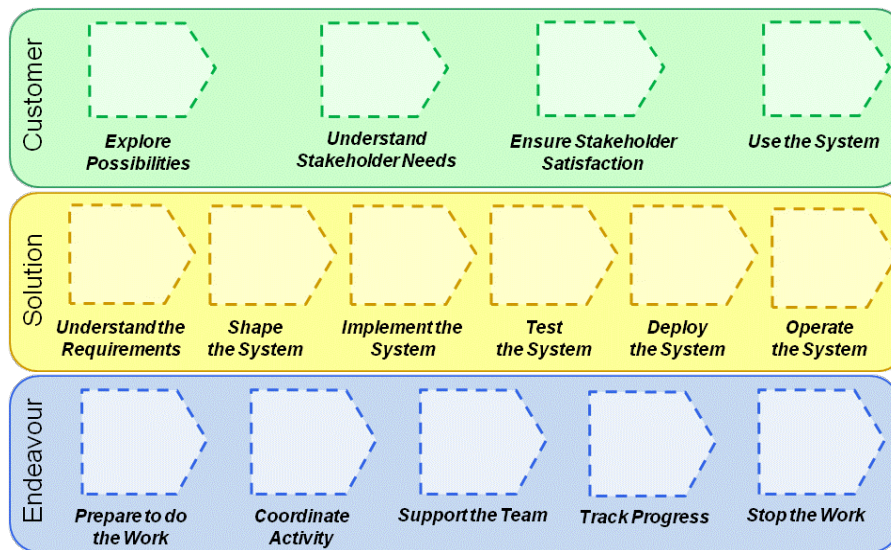


Figure 3.4: Activity spaces in the SEMAT Kernel [24].

performed.

Completion criteria are added to activities, which point to levels of detail (of work products) and states (of alphas). When an activity is performed, these levels and states are achieved for the corresponding work products and alphas. For instance, performing the Weekly Planning Meeting of the Weekly Cycle practice, used as an example in Section 3.3, means the Week Backlog work product reaches the Filled state.

Activities can also indicate the competency that is accountable for performing it, though not shown in the figure. Competencies are defined in the Kernel to represent the key competencies required to perform software engineering. Figure 3.6 lists all predefined competencies. Competencies are still a work in progress at the time of writing and therefore remain largely unused within this thesis [16].

3.2 Practice library

Using SEMAT, software development practices can be documented and stored in a library for reuse with other projects and methodologies. To incorporate practices into the methodology of a software project, we propose to extend the documentation of practices to include their purpose, prerequisites, necessary commitment, and possible ways of tailoring them.

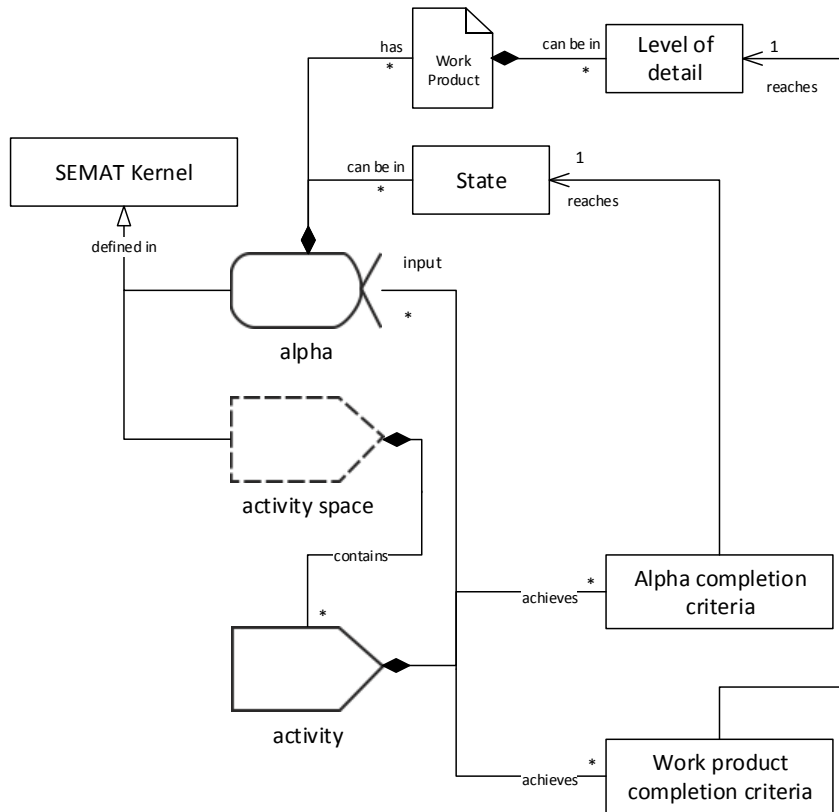


Figure 3.5: Relationship between activities, work products, and alphas.

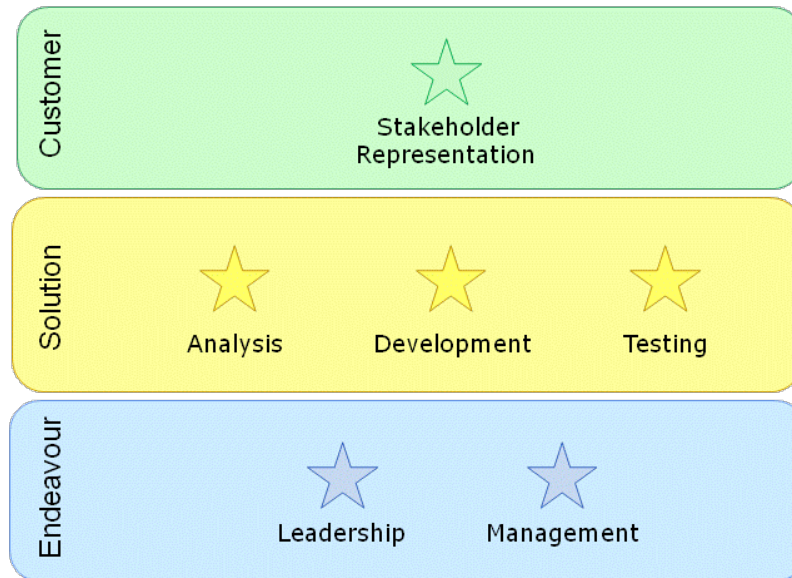


Figure 3.6: Competencies in the SEMAT Kernel [24].

3.2.1 Purpose

We propose to add a description of the purpose of each practice to its documentation. This description explains why a practice should be incorporated within the methodology of a project. When considering a practice, its purpose helps identify if it would be of use to the project and the team.

The purpose of a practice can touch many aspects of software development. For example, the User Stories practice would have the following purposes:

- Tasks are written as customer-visible functionality.
- Tasks are displayed on index cards on a card wall, making it possible to track their progress.

3.2.2 Prerequisites

Each practice has prerequisites before they can be applied within a project. Three types are recognized:

- Requirements that can be described using properties of the project. The Daily Standup, for instance, requires the team to be located close together.
- Dependencies on other practices. For example, the Weekly Cycle requires the use of User Stories so a customer representative can understand the tasks he chooses to be on the agenda for the next week.

- Other prerequisites, such as the necessity of a certain expertise. Visualize Workflow, for example, needs an outline of the workflow within the project before a card wall can be prepared and tasks can be tracked along this workflow.

We propose to add the prerequisites so they can be used as guidelines when considering if and how to incorporate a practice into a methodology. Using the description of how a practice can be tailored of Section 3.2.4, the practice can be adjusted to fulfill its prerequisites as well.

As an example, Scrum requires the availability of a domain expert to fill the role of Product Owner. This role can also be filled by a representative to which customers can relay requirements of the system under development.

Aside from tailoring a practice, the project can be changed to accommodate for the prerequisites of a practice as well. For example, a team that has been spread across two floors can be moved to a single floor, fulfilling the prerequisite of the Daily Standup.

3.2.3 Necessary commitment

Practices require certain work and activities to be carried out to apply it correctly. This is documented as the necessary commitment of a practice. Among others, include regular activities, meetings, and maintained work products. This description gives an idea of the additional methodology weight and of the workload demanded of the team. It helps in considering whether to incorporate a practice in the methodology of a project.

For example, Scrum requires two regular meetings: a planning meeting for scheduling the next increment, and a review meeting for demonstrating what has been achieved.

3.2.4 Tailoring

In many cases, practices will need to be tailored to a project before they can be applied. We propose to add a description of how practices can be tailored to the project, without damaging its purpose. Practices described in literature often already contain instructions for adjusting it to a project.

This description can be used to better fit the practice to the project and the team. Furthermore, the practice might need to be adjusted so its prerequisites are met.

For instance, Scrum can be used with different increment lengths [27].

3.3 Documenting software development practices

This section describes how software development practices can be documented with the SEMAT Kernel. The *EssWork Practice Workbench* has been used, which supports documenting practices with the Kernel [15].

The Weekly Cycle has been used as an example throughout this section [5]. Extreme Programming has combined this practice with User Stories and Test-First Programming, which are filtered out of the practice description. The entire documentation of the Weekly Cycle is available in Appendix B.1.

The following steps explain how practices can be documented.

1. Identify and extend alphas

The SEMAT Kernel predefines seven alphas, or “*things to work with*”. Go over each of these and decide whether the practice extends the alpha.

For instance, the Weekly Cycle extends the *Work* alpha, as it instructs the team that work is performed in weekly increments.

2. Add sub-alphas to alphas where appropriate

Adding a sub-alpha to an alpha implies the practice contributes to this alpha in some way.

For the Weekly Cycle, a sub-alpha *Week* is added under the *Work* alpha to denote work is divided in weeks. The *Week* sub-alpha is given the states Planned, In Progress, Concluded, and Closed, showing how a workweek progresses. As multiple weeks are held, the multiplicity between *Work* and *Week* is set to “1..*”.

Extend any additional alphas if new sub-alphas are discovered.

3. Add work products to alphas and sub-alphas

Add work products defined in the practice to their corresponding alphas or sub-alphas. Add levels of detail and sections that match the description in the practice.

For the Weekly Cycle, when workweeks are scheduled, a backlog is created containing the work items the team will complete the upcoming week. Correspondingly, the Week Backlog work product is added to the *Week* sub-alpha with Filled and Completed as levels of detail. No sections for the Week Backlog are

described. As a single Week Backlog is created every week, the multiplicity from *Week* to the Weekly Backlog work product is set to “1”.

If additional work products are discovered, extend the corresponding alphas if the practice does not do so yet.

The Weekly Cycle includes a Weekly Increment work product corresponding to the *Software System* alpha, which the practice will need to extend. The work product is added under the *Software System* alpha. The work product has Planned and Complete as levels of detail, which show that an increment to the system will first be scheduled and finally be completed. No sections are described. As the *Software System* is built by implementing multiple Weekly Increments, the multiplicity between these two is set to “1..*”.

4. Identify activities and extend corresponding activity spaces

The Kernel predefines fifteen activity spaces, or “*things to do*”. Look into the activities present in the practice and extend the activity spaces under which these belong.

The Weekly Cycle specifies two activities: a meeting held at the beginning of every week and deploying a new version of the system at the end of the week. The practice therefore extends the *Deploy the System* and *Prepare to do the Work* activity spaces.

5. Add activities to identified activity spaces

Describe the activities within the practice under the corresponding activity spaces.

For the Weekly Cycle, the *Weekly Planning Meeting* activity is added under *Prepare to do the Work*. The meeting takes the *Requirements* alpha from the Kernel as alpha input directly, as the backlog will be planned using these requirements. It has three completion criteria: the *Week* and *Weekly Increment* are Planned and the *Weekly Backlog* is Filled. The activity *Deploy Increment* is added under *Deploy the System* with three completion criteria: the *Weekly Increment* is Complete, the *Week* is Concluded, and the *Software System* is Operational.

Although competencies in the Kernel are still a work in progress, specifying an accountable competency makes it clear what role is necessary for completing the activity. For the *Weekly Planning Meeting*, this is the *Stakeholder Representative* present in the Kernel. For *Deploy Increment*, this is the *Development* competency.

Note that additional alphas or sub-alphas can be discovered at this point, as activities take alphas as input.

6. Specify usage criteria of the practice

Lastly, specify the usage criteria of the practice, as defined in Section 3.2. These can already be described in the literature where the practice is explained, but can become more apparent when the team gains experience with applying the practice.

For the Weekly Cycle, the following are identified from literature.

Purpose

- Focus to finish a deployable system every Friday.
- Have short iterations and thus receive feedback from customers often.
- Letting teams reflect on their progress often.
- Perform and reflect on experiments within their workplace, such as changing details to the methodology, weekly.

Prerequisites

- Availability of customers for selecting tasks.
- The User Stories practice or a similar practice that has task breakdown and estimation.

Necessary commitment

- Weekly meetings on Monday to schedule work.
- Reaching deployable software weekly.

Tailoring

- The Weekly Cycle can be combined with Test-First Programming, in which tests are written before work items are implemented.
- Some people start their week on Tuesdays or Wednesday. The weekly meeting can be moved as long as it will not pressure the team to work over the weekend.
- Reduce the time necessary for the planning meeting. When applying the Weekly Cycle initially, planning meetings can take hours.

Chapter 4

Project Inventory

Properties of software projects that influence decisions on a methodology are identified in this chapter. Furthermore, a way to identify the important properties of a project is proposed.

§

As each software project is unique, each requires its own methodology with which to perform the software development effort [3, 8]. Projects can differ, among others, in team size, team members, budget, culture, schedule, and risks. These properties of projects can influence how their methodologies are composed. This chapter proposes a project inventory which enables gaining insight in a software project by collecting these properties.

The identified project properties are summarized in Table 4.1. They are divided into four groups:

- Properties about the project itself, described in Section 4.1.
- Properties of the team members of the project, described in Section 4.2.
- Properties about the developed solution, described in Section 4.3.
- Properties of the current methodology of the project, described in Section 4.4.

The project properties are described as questions to ask interviewees. They include a description of their possible influence on the methodology and how they apply to newly starting projects and existing projects.

The project properties have been selected from existing literature [8, 31]. The following properties have been added to better support existing projects:

- The current backlog of the project.
- The maintainability of the current software system.
- The current methodology of the project.

Project properties	System and project history, team culture, criticality, priorities, budget, requirement stability, customer availability, planned milestones, environment, problem domain complexity, current backlog
Team member properties	People involved, problem domain experience, software development experience, methodology experience, willingness to change the methodology
Solution properties	Solution complexity, maintainability
Methodology properties	Current methodology, things to keep, things to discard or change, stakeholder requirements, standards and conventions, increment length, length of tasks, new requirements

Table 4.1: Identified project properties.

- The preferences of interviewed team members on their current methodology.
- The current length of increments, or time between releases if increments are not fixed.
- The length of lower-level tasks.
- The way new requirements become known within the project.

Section 4.5 describes how the properties of a software project are identified by interviewing key figures within the project. The project inventory is part of the methodology-growing framework further described in Chapter 5.

4.1 Project properties

This section lists the properties associated with software projects directly.

History

Ask for a short description of the history of the project, including changes in staff and emotional high and low points. This helps to gain insight in the size and type of the project [8], as well as to find other interesting questions to ask about the project. Furthermore, ask for the history of the system and whether there are any systems preceding the one under development. The goal of a project can be to provide maintenance on an existing system. The methodology should support this goal and not focus on developing new functionality, but rather on providing support. For newly starting projects, only the history of the system is applicable.

Team culture

Ask interviewees whether they experience the culture within the team as being autocratic or participative. In a participative culture, team members are more likely to contribute in improving their methodology. In most autocratic cultures, on the other hand, management makes all decisions for the team.

The team culture is yet to be determined for newly starting projects. The team can initially discuss how they plan to shape the team culture, for example by determining who will carry responsibility for what decisions.

Criticality

Ask interviewees about the criticality of the system under development, defined in Chapter 2. The criticality indicates the amount of damage a failure would cause. More critical systems will need more control elements in the methodology. This applies to both existing and newly starting projects.

Practices that enable code reviewing and extensive testing, for example, can be included in the methodology for such cases. Even with low criticality, correctness can be a priority, for instance when stakeholders require it.

Priorities

Ask interviewees what the priorities of the project are. Priorities can range between productivity and prevention of legal liability [31]. The former focuses on a shorter time to market, often to get feedback quickly. The latter entails that artifacts are traceable and work is documented and tracked. Additional possible priorities are low costs and correctness, but also ask whether there are other priorities within the project.

The priorities heavily influence the methodology. For example:

- For a short time to market, the methodology needs short increments, each of which resulting in a workable product.
- To keep costs low, the project might need to have as few as possible team members, with a correspondingly designed methodology.
- When correctness is important, practices that enable regression testing, for example, need to be included.
- For traceability, the methodology needs more control elements as well as clear guidelines on what needs to be documented.

This property applies to both existing and newly starting projects, though priorities of newly starting projects might still need to be fully defined.

Budget

Ask the interviewee to indicate the cost limitations within the project. A tight budget can influence decisions on the methodology. For instance, it means there is less room for excess methodology weight in the form of artifacts and documentation.

A budget too tight can have negative influence on the project's success. From the methodology's point of view, with a tight budget, less time should be spend creating artifacts and deliverables. Furthermore, letting the entire team sit in a single room will cost less than arranging for necessary communication.

For newly starting projects with a tight budget, the methodology should be composed light and later on extended if necessary. This will prevent excess weight. For existing projects, the team should look into which parts of the methodology are excess.

Requirement stability

Ask interviewees, especially those with a developer role, how stable and predictable requirements are. The stability and predictability of requirements can decide the length of increments and how soon feedback is necessary. If requirements are unstable, faster feedback is necessary and increments need to be short to allow this. Alternatively, include practices such as prototyping that allow for fast feedback.

This project property is of less influence to newly starting projects, as the overall stability of requirements is still unknown. Still, ask interviewees how stable they expect requirements to become.

Customer availability

Ask each interviewee how communication with customers or their representatives is arranged. In the best case, they are always available to answer questions and give feedback. Practices that enable quick feedback should be included in the methodology otherwise.

The level of customer cooperativeness influences the design of the methodology. For instance, lower cooperativeness means less feedback, so requirements might need to be made more concrete before they are moved towards development.

Projects can benefit from having a customer or customer representative in the team. This practice is also included in Scrum and Extreme Programming [5, 27]. Newly starting projects can decide to include this practice in their initial

methodology, if such a team member is available.

Planned milestones

Ask interviewees what they know of upcoming milestones within the project, such as minor or major features and version updates. It is possible that no milestones are scheduled, for instance when the system is in maintenance. Asking this helps to predict arrival of new requirements and workload. It is therefore related to the way new requirements become known, included in Section 4.4.

The planned milestones and workload influence the required efficiency of the project, especially when a large backlog of tasks is still in progress. For instance, when many milestones are defined, the methodology should focus on high productivity. This property applies the same to both new and existing projects.

Environment

Ask each interviewee about external projects, organizations, and other parties that are involved. Projects that develop a system integrating with the developed solution are an example of such a party. Also ask for the stability and predictability of each party in the external environment and whether there are any risks for the project.

For both new and existing projects, the way to deal with the environment needs to be included in the methodology. For instance, other parties might require some form of reporting, which is also included in Section 4.4. Furthermore, the way communication with external projects is arranged should be included in the methodology as well.

Problem domain complexity

Ask each interviewee their estimation of the problem domain complexity. This is also called the problem size by Cockburn [8]. No absolute metric can indicate this complexity, but it can be discussed in terms as how difficult it is to learn and to implement a system in the domain.

The problem size affects the necessary methodology weight and project size. A lower project size, with a low methodology weight, can often solve the same problem size as a greater number of people with a greater methodology weight. When the problem size becomes larger, though, the number of people needed to succeed will greatly increase.

For a newly starting project, the problem domain complexity can be unclear initially. When the project progresses and the team is more experienced with

the problem, team members can start making decisions based on this property.

Current backlog

Ask for the size of the current task backlog. If there is no backlog available, ask for an estimation of the number of work items and the time it would take to complete all of them. A large backlog can indicate the project is not running on schedule or if there are any unrealistic expectations towards the speed of development.

This property is not applicable to newly starting projects, which do not have a backlog of older tasks.

4.2 Team member properties

This section lists properties associated with the team members of software projects.

People involved and project size

Ask which people are involved with the project, including the interviewee. Determine how they are distributed amongst locations and what their roles and responsibilities are.

The number of people that are to be coordinated is an important factor for selecting and improving the methodology. As the number of people grows and their distribution becomes more spread, more communication elements must be included in the methodology for the project to succeed [8].

Problem domain experience

Ask interviewees how much experience they have with the problem domain of the project. If overall experience with the problem domain of the team is low, measures to prevent defects are necessary. For example, incorporating test-first programming in the methodology. Team members should also receive training to gain better understanding of the problem domain. This is especially important when the problem domain complexity is high, which is a project property described in Section 4.1.

For existing projects, it is likely there are experienced team members present. This does not have to hold true for new projects. In this case, it can be necessary to give team members training or to include a domain expert (such as a customer representative) in the team for quick feedback.

Software development experience

Ask how much experience the interviewee has with software development. If overall experience with developing software is low, try to improve the team, not enlarge it. Enlarging it will make it necessary to increase the methodology weight, which comes with a cost, as explained in Section 2.2. Also ask team members that are not in a developer role about their experience with software development to see if they require further training.

For existing projects, project-specific questions can be asked as well, such as experience with used programming languages or frameworks.

Methodology experience

Ask interviewees about their experience in using software development methodologies. Also ask which roles they have played within the methodologies of other projects. Employees with much experience are more likely to contribute in improving the used methodology. However, employees that do not have experience with methodologies are likely to need more guidance in changing their way of working.

Willingness to change the methodology

Ask interviewees to indicate how willing they are to change their current way of working. When interviewees are generally rigid, it is less likely major changes to their methodology will be adopted by the team.

4.3 Solution properties

Two properties of the developed system are identified to be important for making decisions on the methodology of a project: the complexity and the maintainability of the solution. There can be many other properties of the solution that can affect the methodology, such as its architecture, whether it uses a database and whether it includes support for legacy systems [31].

Solution complexity

Ask the interviewee on the complexity of the current version of the system. Ask whether specific metrics of the system are available, such as the number of components, cyclomatic complexity, or lines of code. Other metrics are useful as well, such as those indicating the (predicted) arrival of defects.

If no metrics are available, ask for an indication of the complexity of the system, for instance by asking how difficult it would be to add an entire new component. Especially weigh in the opinion of interviewed developers. If complexity is high, ask if there is excess complexity which can be resolved.

With high complexity, the methodology should include practices that prevent defects and faults, such as regression testing.

For newly starting projects that do not inherit an existing system, this property does not influence decisions on the methodology yet.

Maintainability

Discuss the maintainability of the existing system and whether and why fixing defects or adding functionality is difficult. Also discuss whether this has had any effect on the planning of new features or increments and if the effort for adding features has been estimated wrong.

For newly starting projects that do not inherit an existing system, this property is not applicable.

4.4 Methodology properties

This section describes properties of the project about the currently used methodology. These properties can still be useful to inventory for newly starting projects to find whether team members have preferences towards their methodology.

Current methodology

Ask for a description of the currently used methodology, for instance as a basic workflow. When discussing this, note any possible things to keep or discard. These are associated with the two properties below.

If interviewees do not know the entire methodology, they will probably only describe the elements they have experience with. These descriptions can be pieced together to get a full picture of the methodology. Also see if there are any differences in the given descriptions of the methodology. Differences can indicate a lack of cohesion of the methodology.

This project property is not applicable to newly starting projects, as they do not yet have a methodology defined.

Things to keep

Ask for things of the methodology to keep. The description given of the current methodology can already give some answers. Also ask which rituals, customs and other matters, such as monthly company drinks, are important to keep within the project.

For newly starting projects, the things to keep of the methodology are not applicable, but the rituals, customs, and other matters can already be part of the organization culture.

Things to discard or change

Similar to the things to keep, ask what not to keep or what to change within the methodology and project. Again, the description of the current methodology can give some answers.

For new projects, ask interviewees whether there are things to change within the culture of the organization, such as rituals and customs.

Stakeholder requirements

Ask for the requirements stakeholders have towards the project. Specifically ask for required work products, documentation, and other artifacts or deliverables. These are included in the methodology. Also ask whether there is a way to reduce the need of these deliverables, as they increase methodology weight.

This property applies to both new and existing projects, though for new projects, these requirements might not all be known yet.

Standards and conventions

Ask interviewees whether any standards or conventions, such as coding styles, use of certain languages, patterns, or frameworks, are set within the project. Also let them indicate how tight the standards and conventions are used and if they need to be improved or used more strictly. If there are none, ask interviewees whether they feel standards and conventions are desired. Any of these cases can indicate a possible improvement or refinement for the methodology.

This property only applies to existing projects. For newly starting projects, however, it can be useful to ask whether the interviewee has experience with predefined standards and conventions, and if these are recommendable for the new project.

Increment length

Ask for the length of increments, defined as the duration of an iteration. For example, Scrum uses 30-day Sprints in which a next version of the product is developed [27]. In the absence of increments or iterations, there might exist a default release interval, for instance every other week. The length of increments, if available, indicates how long it takes to develop new features for the system under development.

This property does not apply to newly starting projects, although interviewees can give indications as to what they expect the increment length will be.

Length of tasks

Discuss with interviewees what the length of the lower-level tasks is. Especially let them give an indication of the maximal length of these tasks. This gives insight in how larger features are split into smaller items of work. The task length has influence on the length of increments. If tasks are relatively large and cannot be split into smaller items of work, longer increments might be more appropriate.

This property does not apply to newly starting projects.

New requirements

Ask interviewees how and when requirements become known to the project, who announces them and where they come from. For instance, a higher number of requirements might arrive in a specific season. This is called the demand profile and can have influence on the workflow within the project [2].

The way new requirements enter the project and how they are prioritized and scheduled are part of the methodology. Therefore, it is useful for team members to realize how this takes place, as well as to discuss if it can be improved.

4.5 Project inventory interview

The project properties are identified by interviewing key figures within the project, such as stakeholders, project managers, business roles, experienced developers and graphical or interaction designers. For small projects all team members might be interviewed.

The project inventory interviews are individual and semi-structured to allow for flexibility [22]. A script is available, but it is incomplete and allows for the in-

interviewer to explore and look for surprises. As each project is different, each project can also have unique, additional properties, therefore, structured interviews, which do not allow for improvisation, would not have been appropriate.

The project inventory has been used in the case described in Chapter 6. Appendix C contains the interview that has been used for identifying all properties and additional topics within the case. Many of these properties and topics are closed, but during the interviews, additional notes have been taken where interesting opinions and facts were found.

For the experience with software development and methodologies, interviewees were asked to indicate their time on the current project, the number of years of experience with software development and methodologies, and an indication of how they evaluate their own competence on both. The perceived team culture, willingness to change their methodology, budget, maintainability, requirement stability, customer cooperativeness, and stability of the environment are all asked using a five-point likert-scale.

The distribution of people, project priorities, system criticality, tightness to standards and conventions, and planned milestones were asked as closed questions. For the priorities and milestones, interviewees were asked if there were additional options.

Chapter 5

Methodology-Growing Framework

This chapter proposes a framework that uses identified properties of a software project to incrementally improve a methodology. Reflection workshops are held with the entire team, during which decisions on how to improve the methodology are made collaboratively.

§

The methodology-growing technique is described as a set of five activities, starting with the adjusting of an existing methodology and improving it repeatedly [8]. The methodology-growing framework extends this technique by including the following:

- Support for existing projects that already have a methodology.
- Identifying project properties to make decisions on the methodology.
- A library of practices can be used to select possible improvements on the methodology.

Section 5.1 provides checklists for making decisions on the methodology to keep it coherent. Section 5.2 introduces the methodology-growing framework, written from the point of view of an existing project. Section 5.3 gives additional instructions for newly starting projects. Section 5.4 gives a summary of the framework and provides a high-level overview.

5.1 Composing the methodology

In this thesis, methodologies are defined as a composition of practices, as explained in Chapter 3. As each project is unique, each requires its own, tailored methodology. To compose a coherent methodology, this section provides checklists which help select and tailor practices. When applying the methodology-

growing framework, as described in Section 5.2, these checklists are used whenever decisions on the methodology are made.

The following set of questions can be used as a checklist when considering to incorporate a practice within the methodology.

1. Does the purpose of the chosen practice match with the goals of the team and the project?

Assess whether the purpose of the practice matches with the goals of the team. If there is uncertainty, briefly discuss with team members whether they feel the practice would be of value to them.

Also check whether the purpose of the practice matches with the goals and priorities of the project. In some cases, it is possible to adjust the practice. For instance, using a practice with two month increment lengths does not match a short time to market priority. This practice can possibly be tailored for shorter increments, though.

2. Does the purpose of the chosen practice not overlap with that of another selected practice?

See if there are no other practices selected that overlap in their purpose. If so, see if either or both of the practices can be adjusted or if they can be combined.

3. Are the prerequisites of the chosen practice met?

If the prerequisites are not met within the project, see if either the project or practice can be adjusted to accommodate for the them. The description of how the practice can be tailored can be used to make it fit the project.

4. Is the necessary commitment of the chosen practice worth its benefits to the project?

Each documented practice contains a description of its necessary commitment. This indicates its added methodology weight when incorporated. Consider whether this commitment is worth the benefits of incorporating the practice.

It can be possible to adjust the practice to add less weight to the methodology. If this is not enough, other practices probably better fit the project.

5. Will team members be willing to adopt the chosen practice?

Consider whether team members will be willing to perform the necessary commitment of the practice. If they would not be willing, trying to incorporate the practice can lead to it remaining unadopted.

Discuss the practice, its value, and its necessary commitment with a few team members if there is uncertainty. Do not propose or incorporate the

practice if team members feel it would remain unadopted.

6. **Does the chosen practice contain overlapping deliverables?**

See if there are any deliverables in the practice that provide similar value to the project or its stakeholders as other deliverables already in the methodology. Combine overlapping deliverables if possible to ensure they no longer overlap.

7. **Does the chosen practice cover the same topic as other practices in the methodology?**

Practices are documented using the SEMAT Kernel, which uses alphas to represent essential things software engineers work with. See if there are any practices that add work products and sub-alphas to the same alpha. Assess whether the practices are compatible and try to combine overlapping sub-alphas and work products. If practices are incompatible, they cannot be incorporated together. For instance, Scrum, which results in a working increment after every Sprint, is incompatible with traditional waterfall methodologies [27].

It is possible for practices covering the same topic to still be incorporated in one methodology. For example, Extreme Programming contains Weekly and Quarterly Cycle practices that would both extend the Work alpha in the Kernel [5].

8. **Does the chosen practice contain activities in the same activity space as other practices in the methodology?**

Documented practices group activities under activity spaces defined in the SEMAT Kernel. See if there are any practices already incorporated within the methodology that have activities in the same space. If so, ensure that the methodology will not contain activities that overlap or can result in rework or duplicate work. Combining, adjusting, or replacing these activities is often possible. If none of these options is possible, it is better not to incorporate the practice to prevent unnecessary rework.

9. **Does the chosen practice unnecessarily contain ways of indirect communication?**

Only when the team size grows will the methodology need to include forms of indirect communication. In case of a small team, if a practice uses intermediate deliverables as indirect communication, replace them with meetings between the necessary people.

In case of a large team size, indirect communication is necessary to prevent rework.

10. **Does the chosen practice represent excess weight for the methodology?**

The necessary commitment of a practice indicates the weight it adds to the methodology. Identifying *excess* weight is not easy, and removing too much weight can hurt quality of the produced software.

Consider whether the chosen practice would represent excess weight for the project and its current methodology. Note that some weight is necessary, especially with high criticality, i.e. how dire negative consequences are if the system fails. This is explained in the next checkpoint.

11. **Does the chosen practice increase the ceremony of the methodology?**

The criticality indicates the potential damage a failure of the system can bring. With greater criticality, practices should be incorporated that prevent defects, such as testing and validation techniques. These practices are often useful, but in projects with greater criticality, the weight they add to the methodology is less *excess*.

With lower criticality, the question is whether the chosen practice represents excess weight, which is discussed in the previous checkpoint.

12. **Does the chosen practice contain unnecessary process, formality, or documentation?**

Discipline, skills, and understanding work better than process, formality and documentation when there is need to respond to changing circumstances. As the team becomes more familiar with the problem domain and the environment and problem domain are stable, the methodology can start relying more on process, formality, and documentation. Until this point, prevent adding process, formality and documentation to the methodology.

Formality and documentation are often represented by having to maintain many work products, which can possibly be replaced with meetings between the necessary people. Process can be discarded by removing parts of practices that are too specific and thereby prevent the team from adapting to circumstances.

Additionally, the following questions represent a checklist for any decision on the methodology, including the incorporation and improvement of practices.

1. **Does the methodology weight correspond to the team size?**

Larger teams need practices that coordinate work to prevent overlap and rework. These often rely on indirect communication channels such as inter-

mediate deliverables. Therefore, a heavier methodology is necessary. Smaller teams, however, can use direct communication and keep their methodology light. It is often better to have a methodology that is too light and increase its weight as the project and its team matures.

Identifying excess weight to discard from the methodology is difficult. Furthermore, there is a risk of removing parts that ensure quality of the solution. When criticality is low, practices that increase ceremony can represent excess weight. Again, removing all such practices can affect software quality.

2. **Does the methodology contain enough ceremony for the criticality of the system?**

When criticality is high, the methodology needs to contain enough ceremony to prevent defects, as well as to prevent legal liability. A methodology with high ceremony and low criticality, however, contains excess weight.

To increase ceremony, practices that prevent defects can be incorporated, such as testing or validation techniques.

If the amount of ceremony is high for the criticality of the system, the methodology contains excess weight. This is explained in the previous checkpoint.

3. **Does the methodology facilitate enough feedback and communication?**

Feedback and communication reduce the need for intermediate deliverables for internal use. Increasing feedback can also prevent defects from being introduced in the system.

To discover if intermediate deliverables can be discarded, try to increase feedback and communication within the team and its stakeholders. If the artifacts are less useful, they can be removed to reduce bureaucratic burden.

4. **Can a bottleneck activity be identified?**

To improve a bottleneck situation, make the work that feeds the bottleneck activity more complete and stable before it is passed along. Practices in the methodology that influence the activities before the bottleneck need to be adjusted to make the produced work more stable and complete. Additional practices can also be selected to make the work more complete and stable. For instance, Test-First Programming can be used to bring development efforts in a more stable state.

5. **Does the methodology need adjustment to the project?**

The methodology needs to be adjusted to properties of the project. We have identified a number of properties of a software project that can influence

the design of its methodology. For instance, the priorities of the project can influence how often the solution is released to the customer. For each identified property, its possible influence on the methodology is described in Chapter 4.

Note that additional properties of the project can also make it necessary to adjust the methodology.

5.2 Applying the framework

This section explains how to apply the methodology-growing framework within a software project. The framework consists of six consecutive steps. The first two steps are applied once to identify the properties of the project and to see how well the team copes with changes to their methodology. The last four steps are held iteratively, following the same increment length as the project if possible.

Reflection workshops are prepared and held (the third and fourth steps), where teams reflect on what they have learned and decide collaboratively what they will improve on their methodology during the next iteration. The fifth step is to change the methodology gradually following the decisions made during the workshop. With the sixth step, the team can either make some final adjustments or try out improvements or entirely new practices in the middle of the increment.

Section 5.3 provides additional instructions for newly starting projects.

Step 1. Project inventory

First, hold inventory interviews with key figures within the project, as explained in Chapter 4. The project inventory will identify properties for the project that are important for making decisions. These properties are used when preparing and holding reflection workshops and during mid-incremental changes. Furthermore, Section 5.1 provides guidance towards composing a coherent methodology that use the identified properties of the project.

The project inventory also allows interviewees to indicate their preferences towards the methodology. For existing projects, interviewees are expected to indicate what they want to keep, change, or discard from their current methodology and project. For new projects, they are expected to indicate preferences on what they have seen in previous projects. These preferences should be considered whenever decisions on the methodology are made.

Step 2. First practice attempt

To assess how well team members cope with a change in their methodology, apply a practice that is easy to implement within the project. Preferably, select a practice that might be incorporated in the methodology. If the team has difficulty with implementing a single practice in their work process, changes to the methodology should be kept small. Alternatively, it might mean the weight of the current methodology is too large, requiring team members to keep themselves occupied with maintaining work products. According to the second methodology design principle, excess methodology weight is costly [8].

The practice library introduced in Section 3.2 can be used to select a practice as a first attempt. As this will be the first attempt of the team to change their methodology, select a practice that does not have any obstructive prerequisites. For example, Visualize Workflow requires creating a card wall. A practice that is easy to implement is the Daily Standup, available in Appendix B.2.

To measure how well team members adopt the selected practice within a few days, the social adoption of the practice can be measured, which is explained in Step 3. In the case study, the social adoption of the first applied practice has not been measured before the first reflection workshop.

Step 3. Preparing reflection workshops

The team reflects on their methodology during reflection workshops in step 4, where the team collaboratively decides how to adjust the methodology. These workshops are held periodically, after each increment. This step describes how to prepare reflection workshops and to explore new practices that can be applied within the project:

- Observing other projects to discover alternative practices.
- Creating a proposal containing changes and possible practices for the project.
- Measuring social adoption of changes and practices after a reflection workshop.
- Renewing the inventory to discover changes of the project and to discuss their impact.

Observing projects

To discover new or alternative practices, other projects within the organization can be observed, for instance by interviewing key figures of the project.

The methodology-growing technique contains the following interview to find the strengths and weaknesses of an organization [8]:

1. Ask for a sample of each produced work product.
2. Ask for a short history of the project.
3. Ask what should be changed next time.
4. Ask what should be repeated next time.
5. Ask what the priorities of the project were.
6. Ask if there is anything else important to hear about.

If new practices are discovered within another project they can be documented and added to the practice library, as explained in Section 3.3.

Proposal

Before holding a reflection workshop, a proposal can be created containing new or replacing practices to be used in the methodology, as well as other changes on the methodology, such as the removal or improvement of a practice.

When creating a proposal, it is important to keep resistance to change in regard. Trying to change too much within a single proposal can result in the changes being unadopted. Therefore, keep proposals small rather than large. Furthermore, ensure team members understand what the contents of the proposal will improve on the methodology. Prevent suggesting practices for which the team does not see any reason to apply. The results of the project inventory include preferences team members have towards the methodology that can be considered initially.

The practice library can be used to select practices for incorporation, based on their purpose, prerequisites, and necessary commitment. Section 5.1 provides checklists to validate whether a practice would fit the project and its team.

Measure social adoption

To find out how well changes are adopted, part of the measurement instrument introduced by Vavpotič & Bajec can be used [30]. This instrument concurrently measures the social and technical suitability of a methodology. In doing so, they define a methodology as a composition of interconnected elements. For each of these elements, questionnaires are held to measure the social adoption and technical efficiency and to find reasons for the measured levels. Case research by Vavpotic & Bajec indicated using this approach requires some team members to be experienced in using methodologies, as well as investing time in holding questionnaires with all team members. To limit the required investment of time

and the necessity of experienced team members, only the measurement of the level of social adoption can be performed.

The interview to measure the level of social adoption consists of the following close-ended questions with a seven-point Likert scale. Each question is named with an acronym given by Vavpotič & Bajec [30]. These questions are asked for each practice or change. Interview participants only for the practices they have applied in the project, excluding practices that have not yet been implemented or those not applicable to this specific interviewee.

1. **SIFU1:** Given the opportunity to use *the practice*, how often do you use it?
With the following points: never, in up to 20% of opportunities, in 20–40% of opportunities, in 40–60% of opportunities, in 60–80% of opportunities, in more than 80% of opportunities but not always, always.
If results of this question are low, the practice is overall unadopted and can be considered for replacement or removal.
2. **SIFU2:** Use of *the practice* is encouraged as a common activity within the team.
If team members indicate use of the practice is not encouraged, the team should discuss during the next reflection workshop why this is the case. Specifically look into impediments that hinder use of the practice.
3. **SIFU3:** Use of *the practice* is routine and is used at every opportunity.
Results of this question might indicate the practice is not applied completely within the project. Again, there might be impediments that hinder use of the practice, but the required use of the practice might also be too high for the project. The last can indicate possible improvements on the practice, better tailoring it to the project.
4. **SICU1:** When I use *the practice*, I follow the instructions defined within the team.
If this question gains low results, either team members did not fully discuss how the practice is to be applied in the project, or team members disagree on how to use the practice. In either case, this is to be discussed in the next reflection workshop.
5. **SICU2:** The way *the practice* should be used is clear.
If team members indicate the usage of the practice is unclear within the project, this can either be because of undefined instructions within the team, as is the case with the fourth question, or because team members are uncertain whether they apply the practice correctly.

Encourage the interviewees to give comments so to explain the reasons for their answers to find possible improvements or impediments.

Renew inventory

As the project progresses, the properties of the project that were identified using the project inventory of Chapter 4 can become obsolete. Although most changes will be recognized immediately, it is important the entire team knows about these changes so they can be discussed to see if the change might affect the methodology.

Some changes within the project might not be recognized or reported. To find and inventory these, a shorter, smaller-scale inventory can be held. Instead of holding the entire inventory interview, only inquire on the project properties.

Improvements during an increment

In the middle of the increment, team members can incorporate, replace, or change practices in their methodology to assess possible improvements. Feedback from these improvements can be used as input for a reflection workshop. This is further explained in the sixth step.

Step 4. Holding reflection workshops

Send an invitation to the entire team before the reflection workshop. If a proposal is made as a preparation to the workshop it should be included in an attached agenda. Give participants the opportunity to send in their own requests, additional practices, or other topics regarding the methodology and project in advance. Add all these requests to the agenda of the workshop.

It is also possible to give participants access to the practice library. This is especially useful if team members are experienced with methodologies.

Documentation

Prepare a board or flip chart to be used as a card wall. Practices and changes are written on cards so they can be put on the card wall. Outline the following columns on the board:

- *Deferred.* Practices and changes of which it is not yet certain whether they will be implemented are put here during the workshop.
- *Upcoming.* Practices and changes that are decided to be used are put in this column during the workshop. Furthermore, before the workshop, proposed practices and changes are put here as well.

- *Trying*. After the workshop, changes are gradually moved into this column when they are being implemented or tried out.
- *Keep*. Changes and practices are moved from Trying to this column during the workshop when the participants decide they are worth keeping.
- *Discard*. Changes and practices that will be removed from the methodology or project are moved here.

The board outlines how changes to the methodology are first tried out and then either kept, further refined, or discarded.

Reflection workshop

During reflection workshops, teams can make decisions on their methodology collaboratively. Section 5.1 can be used to assess whether these decisions will result in a coherent methodology, or whether decisions need to be altered.

Begin reflection workshops by looking at the cards in the *Trying* column. Discuss what the team has learned from applying these practices and changes. Conclude with the team which practices and changes will be kept, moving these to the *Keep* column. If the participants agree a practice or change should be kept, but needs to be adjusted, discuss what can be improved. Add a new card to the *Upcoming* column for these improvements. If the discussion goes on for too long, add it to the *Deferred* column and, if necessary, better prepare this topic the next reflection workshop. Move each practice or change that is to be removed from the methodology or project to the *Discard* column.

Look at the *Upcoming* column next and discuss for each card why it has remained in this state. Keep the discussion short and decide whether to move cards to the *Deferred* column, the *Discard* column, or to keep them in the *Upcoming* column. Next, discuss the results of any preparatory activities performed before the workshop, such as those of Section 5.2.

- If social adoption is measured for certain practices, present the gained results. Discuss them with the participants and decide how each measured practice can be improved. Add cards for each improvement to the *Upcoming* column.
- If new ideas for the methodology are discovered by observing other projects, present these and decide collaboratively whether to add cards to the *Upcoming* or *Deferred* columns or to disregard them. For any change or practice added to the *Upcoming* column, decide how to apply it within the project.
- If there has been a change in the properties of the project, discuss whether the methodology needs to be adjusted. Add changes that need to be incorporated

quickly to the *Upcoming* column. If it is unclear how and if the methodology needs to be adjusted, add a card to the *Deferred* column.

- If new practices or changes are suggested by team members, add these to the *Deferred* column. Discuss them with all participants and decide whether to keep them in the deferred column, applying them during the next increment, or to discard them. If participants agree to put practices or changes in the *upcoming* column, discuss how they should be applied within the project.
- If a proposal was prepared, add its contents to the *Upcoming* column. Discuss each of the proposed practices and changes with the participants and decide whether to keep them in the *Upcoming* column to be implemented within the next increment, or to move them to the *Deferred* column. For any card kept in the *upcoming* column, discuss how it should be applied within the project.

Lastly, ask if there are any other topics and problems to be discussed and if there is anything that can still be improved in the current methodology.

Keep reflection workshops short, preferably within an hour. This might require keeping some discussions short.

First reflection workshop

The first reflection workshop is held after applying the project inventory. Before beginning the workshop, explain to the participants that the goal of these workshops is to collaboratively decide how the methodology will be improved. Furthermore, present the results of the project inventory to all participants.

Step 5. Changing the methodology

While changing the methodology, there can be parts of the methodology that are missing or unclear. If possible, combine these changes into a mid-increment change, as described in the sixth step. However, it can be necessary to make decisions ad hoc, which need to be discussed in the next reflection workshop.

Changes to the methodology should be applied gradually rather than immediately. According to Jacobson, Ng and Spence, experience has shown that it is much more effective to transform the methodology one practice at a time [20]. Whenever a selected practice or change is ready for incorporation, move the corresponding card on the card wall to *Trying*.

First increment

During the increment directly after the first reflection workshop, assess whether there is a problem with the methodology that might result in a development failure. To do so, ask team members during individual or group interviews whether the expected goals will be achieved working with the improved methodology. According to Cockburn, if the methodology does not work, first consider reducing the scope of the increment, to rule out an overambitious schedule [8]. If this is not sufficient to ensure the success of the increment, look for structural impediments, such as bottlenecks or problems with understanding requirements. The goal is to deliver at least something, even if drastic measures are necessary, such as making emergency staff changes.

In the case of Chapter 6, this assessment was performed together with interviews for measuring social adoption of newly incorporated practices.

Step 6. Middle of the increment

Team members can try out additional changes to the methodology in the middle of each increment, as they have a working methodology to fall back on. They can choose to add, replace or refine practices within their methodology and see whether these fit the team and the project.

Refrain from changing the methodology during the increment after the first reflection workshop. At this point, changes to the methodology can result in not achieving set goals, as discussed in the fifth step.

During an increment, team members can observe what can still be improved on the methodology and can assess possible ways to perform those improvements.

Do not try to improve too much to avoid resistance to change. Also refrain from changing the methodology if the team is still busy with adopting the changes chosen during the last reflection workshop. Add all changes to the *Trying* column of the card wall so they can be discussed during the next reflection workshop.

5.3 Start of a new project

This section gives instructions when applying the methodology-growing framework on a newly starting project. As no methodology is yet in place, the first reflection workshop is to result in a methodology the team can apply immediately.

5.3.1 First proposal

For a newly starting project, the proposal consists of a methodology the team can start with. Cockburn suggests selecting an existing methodology, either from literature or one that is in use within the organization elsewhere [8]. This is still a viable option when using the methodology-growing framework. Boil a selected methodology down to its basic work flow and a draft of the conventions that will be used. This makes it easier to discuss. Use these work flow and conventions as a proposal that to be discussed in the first workshop.

Alternatively, the practice library explained in Section 3.2 can be used to select practices and combine them into a methodology. Use Section 5.1 to ensure the practices form a coherent methodology. Note that it is better to begin with a methodology that is too light, rather than one that is too heavy.

Observing projects

The third step of the methodology-growing framework describes a way to observe other projects to see how they develop software. At the start of a project, this can also be applied within the organization to see whether a methodology is already available that might fit the project.

5.3.2 First reflection workshop

The first reflection workshop for newly starting projects will primarily focus on the proposal, which contains a starter methodology. Discussing the methodology is enough content, so it might be necessary to keep other discussions short and defer them to the next workshop. The focus is on attaining a usable methodology for the first increment quickly.

5.3.3 First practice attempt

Newly starting project that have not yet started development would not benefit from attempting to adopt a first practice. This can still be done after proposing and adjusting the first methodology, but will only be useful if the team has sufficiently adopted the selected practices. Otherwise, continue using reflection workshops to iteratively improve the methodology.

5.4 Summary

In summary, the methodology-growing framework consists of the following six steps, also outlined in the flow diagram of Figure 5.1 on page 54.

1. Use the project inventory of Chapter 4 to identify properties of the project important for making decisions on its methodology.
2. Try a single practice to assess whether the team has difficulty changing their methodology.
3. Prepare reflection workshops to exploring possible improvements of the methodology.
4. Hold reflection workshops after every increment, keeping track of queued and tried improvements on the methodology.
5. After every reflection workshop, change the methodology according to the decisions made during the workshop.
6. In the middle of the increment, try new or improve practices. These can be reflected upon during the next workshop.

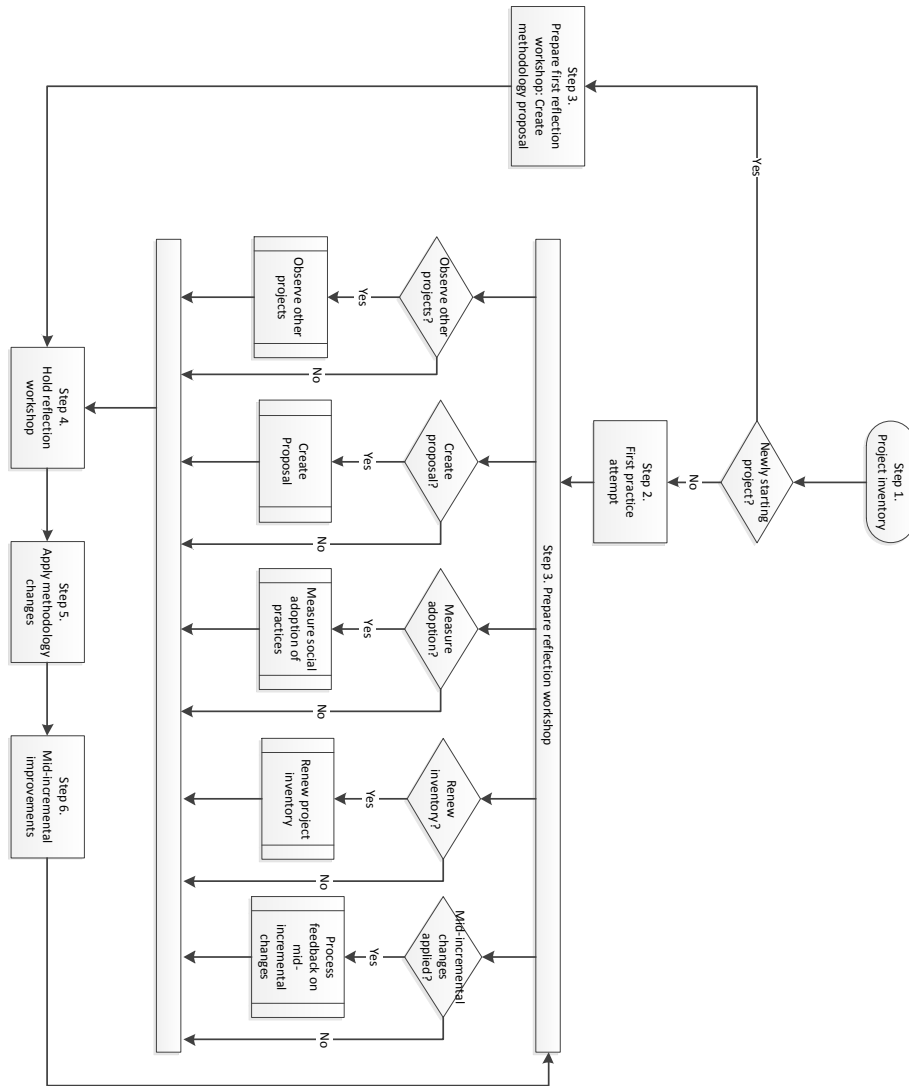


Figure 5.1: High-level overview of methodology-growing framework.

Chapter 6

Case Description

The methodology-growing framework has been applied on a small-scale project developing a product for mortgage advisors. This chapter discusses the results of identifying important properties of this project and the application of the framework within the project.

§

The methodology-growing framework has been designed and applied at a project called Findesk within an organization called Topicus located in Deventer, The Netherlands. Findesk is a two year old small-scale project developing software to assist mortgage advisors. The framework was applied to better fit the methodology to the growing number of team members.

In this chapter, first, the results of performing the project inventory are summarized in Section 6.1. The first iteration in improving the methodology is discussed in Section 6.2. A second iteration was performed and is discussed in Section 6.3. Lastly, Section 6.4 reflects on how the methodology of Findesk has been improved using the methodology-growing framework.

6.1 Project inventory

Project properties of Findesk have been identified by interviewing all of its team members. Initially, two separate interviews were created, one for team members with management or leadership roles and one for all other team members. However, as team members turned out to have multiple roles, the interview forms were combined. The combined form is available in Appendix C.

Below, the results for the inventory are summarized. Project properties are grouped in four categories:

- Properties about the project itself.
- Properties about the team members of the project.
- Properties about the solution developed within the project.
- Properties about the current methodology of the project.

6.1.1 Project properties

History

The system was initially implemented as a single-page web application that performed some financial calculations. The system has gradually grown and become more complex, allowing mortgage advisors to perform the entire advisory process with their customers. Eventually, it included functionality to send mortgage propositions digitally.

Team culture

The team culture is generally perceived as participative. One team member added that though the team has little influence on decisions about priority, the team does collaboratively decide how functionality is developed.

Criticality

The two directors both indicated the criticality of the system is in the *loss of discretionary monies* class. If the system were to fail or introduce a defect, mortgage advisors might have to delay servicing a customer or closing a mortgage.

The project manager perceives the criticality as *loss of comfort*, only requiring mortgage advisors to use other systems or doing some automated tasks by hand.

Priorities

A short time to market is a high priority for the project, as it enables the project to react to changes in the market quickly.

Budget

The directors have indicated that the project budget is more than sufficient. At the moment, the project is still backed financially by the organization. The goal is for the project to begin making profit within a year.

Requirement stability

Requirements within the project are generally unstable. This is agreed by all team members, including the project manager and directors.

Customer availability

There are two different views on this property. The directors indicate customers, specifically mortgage advisors and banking firms, are difficult to collaborate with. From the point of view of the project manager, the customer is the director/domain expert, whom is highly collaborative to work with.

Planned milestones

A maintenance period has been scheduled, in which team members improve parts of the system they feel should have been implemented differently. There is a continuous stream of minor upgrades which are released regularly. Furthermore, new major features are planned within the upcoming three to six months.

Environment

The following is a list of parties present in the environment of the case project.

- Banking firms at which mortgages are closed using the system.
- Insurance companies where insurances can be closed using the system.
- Another department within the organization that occasionally performs small side-projects.
- A platform for closing mortgages, insurances, and other financial products called HDN.

Overall, the environment is deemed unpredictable. Among other reasons, this is because the format defined by HDN tends to change often, banking firms and insurance companies are somewhat unreliable when it comes to sending mortgage and insurance applications, and closing contracts with banking firms and insurance companies can take time.

Problem domain complexity

Team members have between one and two years of experience with the problem domain and have indicated they do not have difficulty with the problem domain. A reason given by one interviewee is that the problem domain is specific to mortgages, making the problem size relatively small when compared to the entire financial domain.

Current backlog

The current backlog is not fully known by all team members. To the project manager, the current backlog would take around twelve weeks to complete.

The two directors have a different view of the backlog, also including tasks

and features they envision for the long-term. The domain expert/director, who focuses on the current software product, indicates the current backlog would take half a year to complete. The other director focuses also on side-projects which are performed by other departments of the organization, and indicates the backlog will take an entire year to complete.

6.1.2 Team member properties

People involved

The following people are involved with the project:

- Two directors responsible for creating new opportunities and prioritizing them. One of them is also a domain expert with the added responsibility of giving feedback to the team.
- A usability researcher responsible for receiving and processing feedback from users and is also responsible for the help desk.
- A business analyst responsible for communication with customers and the environment.
- A project manager also acting as an analyst and occasional developer. He is responsible for estimating and planning development efforts, as well as discuss requirements and their priority with the directors.
- Three full-time developers who perform the development effort, both front-end and back-end.
- A graphical interface designer and front-end developer who designs the user interface and consults with developers when they need advice on developing the front-end.
- Two part-time developers still in college, one of whom the author of this thesis.

The entire team is distributed close together over three adjacent rooms. All developers are located in a separate with the project manager.

A second graphical interface designer and a part-time front-end developer have been employed during the course of the case study. One of the directors has taken over the responsibilities of the leaving business analyst.

Problem domain experience

As the project is two years old, most team members have some experience with the problem domain. Two developers have gained additional experience with the financial domain at other departments of the organization.

Software development experience

The experience in developing software is split in two groups within the project. The team members not in a development role have relatively little experience with software projects, between 1 and 3 years. Three of the developers, including the project manager, have 5 to 8 years of experience. One developer was recently employed after finishing his education and thus has limited experience.

Methodology experience

Team members have had little experience in using software development methodologies. They did indicate to have knowledge of methodologies and have seen methodologies in other parts of the organization, where Scrum is used most often.

Willingness to change the methodology

Almost all team members have indicated they are willing to change their methodology. Only one of the developers indicated lower willingness to change the methodology, stating he would first want to agree changes are helpful.

6.1.3 Solution properties**Solution complexity**

There are no software metrics available of the complexity of the system. For team members, the source code contains a number of parts that are complex, mostly on the front-end and regarding the above mentioned party HDN. Team members have different opinions on whether or not it is difficult to add new features to the current solution, which indicates they have focused on different parts of the system.

Maintainability

Interviewees overall agree with the system being easy to maintain, as well as it being easy to add additional features to the system. Two interviewees, however, indicate they have more difficulty in maintaining the current system.

- The graphical interface designer indicates the system is very difficult to maintain. The designer is also responsible for web design, which includes the writing and structuring of cascading style sheets (CSS). The main reason for the maintainability being difficult is the badly structured and undocumented collection of CSS-files currently in the system.

- One developer, responsible for most of the back-end development, indicates the system is difficult to maintain as the current architecture is too simple, making it hard to add additional features without breaking the system.

6.1.4 Methodology properties

Current methodology

The current methodology of Findesk exists of the following:

- The entire team holds a morning meeting every Monday for about one hour. Here, they discuss what they have achieved the previous week and plan to achieve the upcoming week.
- Issues written by the project manager/analyst are stored and tracked using a issue tracking and project management tool called JIRA.
- The project manager assigns issues from JIRA to team members. These issues can represent small improvements which can be completed within a day to entire new functionality which take a longer period of time to complete.
- The director/domain expert regularly reviews new features before they are released to customers and indicates improvements if necessary.
- Software releases to customers of improvements happen every few days.
- Releases of new major features to customers happen every few weeks, though irregularly.

Although more conventions are in place at Findesk, the list above captures the normal flow of work within the project.

Things to keep

The interviewees indicated a number of things in the project and methodology to keep.

- The ambience and culture present within the project.
- The weekly lunch at the local market.
- The Friday afternoon beer.
- The company getaways.
- The currently used issue tracker, JIRA, as it is considered a major improvement to the previously used system.
- The short lines of communication.
- The Monday morning meeting, which is especially necessary when there is progress important to other members of the team.

- “*Doing it right*”, a loosely defined principle that tells developers to spend additional time to the quality of what is developed.
- The short code reviews and retrospective developers often have together.

Things to discard or change

Interviewees have not indicated any ritual or part of the methodology they feel should be discarded, but they have indicated a number of changes they would like to happen.

- A more complete backlog, so the team knows what they have to commit to.
- The help desk wants to know the way of working of the developers so they better know when they have time to ask questions.
- Passing along larger tasks to other team members is not efficient (this is a point for improvement reported by an interviewed part-timer).
- More meetings with the team, for instance daily.
- Better communication with external parties such as banking firms, but also customers.
- Team members would like to select their own tasks from the backlog, instead of asking the project manager what has priority.
- One of the directors indicated he wants the team to become more self-managing, specifically in an informal manner, such as arranging getaways and social as well as professional gatherings.

Stakeholder requirements

Only the organization requires some reporting on finances and revenue.

Standards and conventions

No standards and conventions are set, but the project manager has specified that code should be readable and follow a known naming scheme. He also specified standards and conventions would be useful to include in the methodology.

Increment length

Development is not performed in time-boxes or increments, but major releases happen every two to three weeks. Smaller releases, containing minor improvements and bug-fixes, occur more often, varying from every two days to weekly.

Length of tasks

Tasks assigned to developers usually take an hour to a full day for a single team member to complete.

New requirements

New requirements arrive at the project through the domain expert or through an external party. Small defects are often found by customers and reported over email or over the phone. External parties, such as insurance companies, can lead to new requirements, for instance when financial products are to be added to the software.

Changes in Dutch law also lead to new requirements as large as entire features.

6.1.5 Summary

In summary, the case project has a low criticality, a small number of people that are distributed in adjacent rooms, and a very light methodology they have not adjusted to the growth of the project size. Furthermore, a list of goals have been indicated by the directors and the project manager, as well as a number of points of improvement with regards to the current methodology.

- A better and longer roadmap, so the team can see the commitment required from them.
- Developers want the ability to take on tasks of their choosing.
- The help desk wants better communication with the development team to make it clearer how and when to ask for help.
- One part-timer indicated passing along tasks in the currently used methodology is unclear.
- More meetings with team members.

Additionally, more long-term changes to the methodology and project have been identified:

- Creating a better structure of the methodology and create a better role distinction.
- Creating an understanding of which team member has which knowledge on the project.
- Increasing predictability of development, shortening lead times, and improving quality.
- Attaining a methodology that is efficient for both team members and software development.
- Promoting self-management within the team, specifically on informal topics, such as social gatherings.
- Improving communication with external parties.

6.2 First iteration

This section describes the first iteration of using the methodology-growing framework within Findesk. An attempt at introducing a new practice was performed first. Based on how well the team adopted the new practice, the first reflection workshop was prepared with a proposal of new practices.

6.2.1 First practice attempt

In order to assess how well team members cope with a change in their methodology, the Daily Standup was added to their methodology. This practice also fits with one of the goals identified in the project inventory. Every Tuesday to Friday, team members gathered at around 9:00 PM to discuss what they had done the previous day, will do the current day, and whether they have any impediments that block their progress. To not further change the methodology, the daily standup meeting was not held on Mondays, which was reserved for the weekly team gathering.

Although no measurements of social adoption were made, the team appeared to have fully adopted the Daily Standup within a week. As such, the first reflection workshop was prepared by creating a proposal with changes to the methodology.

6.2.2 Preparation of the reflection workshop

The first reflection workshop is prepared with the composition of a proposal. The goals from the inventory that will be addressed by the proposal are “*more commitment*”, the “*ability to select own tasks*”, and “*clearer help desk procedures*”. The proposal was composed in collaboration with the project manager. As the project has low criticality and a low project size, with team members located in adjacent rooms, the focus is on keeping the methodology light. Below is a list of practices included in the first proposal, including the reasons for why they were selected. These practices are described in Appendix B.

Pair Programming

The summer months were scheduled for maintenance. In this period, team members would indicate parts of the system they felt could be improved. As these tasks were relatively small but difficult, Pair Programming was selected for use with each of these tasks. This allowed team members to collaborate on a single

task, brainstorm refinements, clarify ideas, and to take initiative when the other team member is stuck.

Visualize Workflow

Visualizing the workflow allows the project manager to queue upcoming work items, thus making the commitment the team has to take on in the short term more visible. Furthermore, the board enables team members to select their own tasks from the queue. The proposal also contained the following specifics for the board:

- The board will have swim-lanes for each feature and one only for issues and defects.
- The board will depict a workflow consisting of the columns “To do”, “Implementation”, “Testing”, and “Done”. This workflow was created while discussing the proposal.
- Cards on the board will contain a title, description, and an identifier for the issue tracking system.

MoSCoW Prioritization

This requirements prioritization scheme was selected to identify work items that must be implemented within a feature before a release can be made. The priority of requirements will be visible on the cards on the board.

Doing it Right

Doing it Right is a principle for allowing additional time to be spent on ensuring the quality of the product. During the inventory, Doing it Right was mentioned as a principle in use that should not be discarded. However, only a single team member indicated the practice being in place. It is therefore included in the proposal to discuss with the team whether they agree on applying this practice. In contrast to the rest of the proposal, Doing it Right is not a practice and is therefore not included in the practice library.

Daily Standup

Daily standup meetings were tried out as a first attempt at changing the methodology within the team. The practice was easily adopted and will be kept in order to better communication between team members. Therefore, it will be kept included in the methodology, but was not on the agenda of the first reflection workshop.

6.2.3 First reflection workshop

A coarse-grained version of the proposal was sent to all team members a week before the first reflection workshop. Team members were asked to also provide additional topics and possible practices they would like to be discussed.

Additional topics

The following additional topics, including a practice called Seasons of the Day, have been added to the agenda of the first workshop.

- Customer involvement. Presentations for customers about the planned milestones were being held at the moment the first workshop was being scheduled. During these, a discussion with end-users led to a number of possible improvements that have later been added to the system. This manner of customer involvement was considered useful by the team member holding the discussion and will therefore be discussed during the reflection workshop.
- Seasons of the Day. This is a practice (or principle) described by one of the directors where the morning is used for development and progress, without distractions. Meetings, customer communication, and other non-development tasks would strictly be performed in the afternoon.
- Workspaces within the organization. This topic was included because the number of team members is steadily growing and team members are preferably put in the same room as much as possible.
- Communication between the help desk and the development team. The help desk was responsible for accepting calls, but referred to the development team for more detailed questions and issues. This topic was added to the agenda to discuss better ways for communication, as they are divided in separate rooms.

Reflection workshop

Above, the prepared proposal and additional topics were described that together make up the agenda of the first reflection workshop. These have been put on the board visualized in Table 6.1. To avoid confusion with the board used with Visualize Workflow, this board will be referred to as the methodology board. The contents of the proposal were added to the *Upcoming* column and the additional topics were added to the *Deferred* column. As the Daily Standup was already in use at the time, it has been placed in the *Trying* column initially. One hour was scheduled for the reflection workshop, during which the following goals, attained from the project inventory, were presented first.

Deferred	Upcoming	Trying	Keep	Discard
<ul style="list-style-type: none"> • Customer involvement • Seasons of the Day • Workspaces • Help desk 	<ul style="list-style-type: none"> • Pair Programming • Visualize Workflow • MoSCoW Prioritization • Doing it Right 	<ul style="list-style-type: none"> • Daily Standup 		

Table 6.1: Methodology board at the beginning of the first reflection workshop.

- More visibility of commitment so team members can see where they are headed.
- The ability to take on tasks themselves, instead of continuously asking the project manager for new work.
- Better communications and agreements on tasks with the help desk.

After presenting the goals and the proposal, the team collaboratively decided how they wanted to apply the practices within their methodology. To do so, an open discussion was held with the entire team in a single room. The discussion was led together with the project manager to give everyone the opportunity to speak. Unfortunately, both directors and the business analyst were unable to attend.

The decisions made during the reflection workshop are visualized on the methodology board of Table 6.2. First, items in the *Trying* column were discussed, which contained the Daily Standup. Team members agreed to move the Daily Standup to the *Keep* column. They did add a card to shorten the standup meeting to the *Upcoming* column. These kinds of improvements on selected practices are displayed in italic.

Team members decided to place Doing it Right, Pair Programming, Seasons of the Day, and customer involvement in the *Deferred* column for the following reasons:

- Team members seemed unsure how to apply Doing it Right in practice, as it would affect many other parts of the methodology. One suggestion that everyone did agree on was that a “Definition of Done” must be made in order

to clear when something is actually “done right”. A Definition of Done is a check-list containing criteria for when an item of work is completed.

- Although all team members agreed on the usefulness of Pair Programming, for now, they agreed to postpone it as it was still uncertain what the maintenance period would entail.
- To apply Seasons of the Day, new agreements would have had to be made with the help desk on when to ask developers for help on questions of customers. As there were already many discussions on how to improve the situation, Seasons of the Day was deferred.
- The discussion on how to better involve the customer and how to find out their preferences remained inconclusive, as team members were uncertain how this would be applied in practice. Furthermore, it would be necessary to first find customers that would be willing to help with the development efforts. They did agree it was necessary to explore how the customer could give more input on development. Two ideas were named in particular:
 - Acceptance testing by customers of new features.
 - Letting customers review wireframes of new parts of the system.

Team members agreed on using the practices Visualize Workflow and MoSCoW prioritization, which were put in the *Upcoming* column to be tried during the next few weeks. Additionally, they wanted work items with database changes to be visibly marked on the methodology board, as they needed more attention before a release could be placed. A card was placed in the *Upcoming* column to do so.

One of the identified goals was to improve communication with the help desk and make clearer agreements on how tasks coming from the help desk would be performed or passed on. The discussion on this topic made clear that there were already a number of ideas, as well as a number of planned changes. For instance, they were planning on identifying the topics that were requested most often by customers and whether a manual of the product would help decreasing the workload of the help desk. As this discussion went on for too long, it was postponed for later. It was not added to the board, as it seemed unclear what could be changed at this moment.

6.2.4 Changing the methodology

After the first reflection workshop, agreed upon changes were implemented gradually. The shortened Daily Standup was affective immediately and was applied

Deferred	Upcoming	Trying	Keep	Discard
<ul style="list-style-type: none"> • Doing it Right • Pair Programming • Seasons of the Day • Customer involvement 	<ul style="list-style-type: none"> • Visualize Workflow • <i>Work items with database changes marked</i> • MoSCoW Prioritization 	<ul style="list-style-type: none"> • <i>Shorter Daily Standups</i> 	<ul style="list-style-type: none"> • Daily Standup 	

Table 6.2: Methodology board after the first reflection workshop.

by notifying the speaker when his story was becoming too long. Over time, the Daily Standup has been shortened to around 15 minutes, short enough to not feel time consuming.

The upcoming changes were limited to the Visualize Workflow and MoSCoW Prioritization practices. The workflow has been outlined on a blackboard that was already available in the workspace, containing the columns “To do”, “Implementation”, “Testing”, and “Done”.

Within two weeks, team members indicated the need for additional columns to better see the state of work items. The workflow was adjusted to the following successive states: “To do”, “Implementation”, “Testing”, “Review”, “Acceptance”, “Done”, and “Released”. The Review state indicates the Project Manager will look into the changesets associated with a work item and can test them if necessary. The Acceptance state is meant for acceptance testing by the domain expert, after which he can indicate possible improvements before new functionality is released. Finally, the Released state is used to indicate that work items are moved into production and are available to the customer. This state was added especially so team members can see whether found defects are already fixed in production. These changes were also added to the methodology board.

6.3 Second iteration

This section describes the second iteration of using the methodology-growing framework at Findesk. The second reflection workshop was prepared by measuring the social adoption of the Daily Standup and Visualize Workflow practices.

6.3.1 Preparation of the reflection workshop

To prepare for the second reflection workshop, the levels of social adoption of the Daily Standup and Visualize Workflow practices have been measured.

Team members are only interviewed on practices they have applied. For the Daily Standup, these were seven in total, including all developers, the project manager/analyst, and the graphical interface designer. Visualize Workflow was used mostly by the project manager, graphical interface designer, and three of the developers.

The MoSCoW prioritization practice remained unused at the time and was thus not included in the measurement of the level of social adoption. This was discussed during the reflection workshop.

Team members were asked to fill in the seven-point Likert scale questionnaires and give reasons for their answers. Furthermore, they were asked to indicate what they think could be improved on the practices. The results of the interviews are discussed below. The complete interview form is available in Appendix D.

Measured social adoption

Figure 6.1 shows the measured levels of social adoption for the Daily Standup and Visualize Workflow practices.

According to the results, the Daily Standup is held often, although not yet at every opportunity, as evident by questions SIFU1 and SIFU3. Furthermore, not all team members seem to regard it as a common activity at the moment of the interview (SIFU2). According to SICU1, the instructions are followed consistently, though some interviewees indicated no instructions have been provided. The results of SICU2 have been ignored, as the associated question was formulated incorrectly, insinuating the existence of a precise description of the Daily Standup, which was never provided.

The results of the questionnaire shows the board visualizing the workflow is used reasonably well, but can still be improved, as shown by SIFU1 and SIFU3.

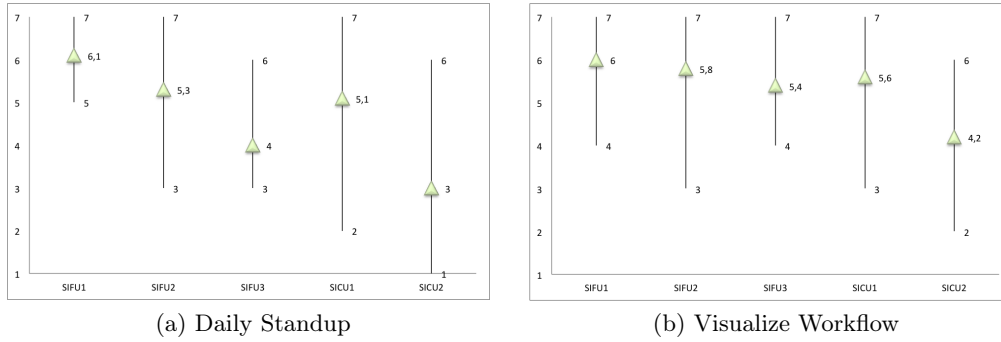


Figure 6.1: Measured levels of social adoption for the Daily Standup and Visualize Workflow.

SIFU2 shows the use of the board is indicated as a common activity, though the answers show some variability. According to SICU1, the instructions are followed consistently by most interviewees, though some indicate the instructions are not fully clear yet. The results of SICU2 have been ignored for Visualize Workflow as well.

Although the results for Visualize Workflow appear positive, the number of team members that indicated they use the board remains low. It is therefore possible the practice remains unadopted for the other team members.

Interviewees have indicated many possible points for improvements while answering the questionnaire:

- The workflow for work items from other principles (e.g. graphical interface design) has not been established and it could be possible to include this on the board.
- A project and issue tracking system called JIRA is available, but its role in combination with the board is not fully decided.
- Other team members should be encouraged to be present during the daily standup meeting. Often, only the developers and project manager are present.
- The board appears too crowded with work items, both in progress as queued.
- Usage of the board is often unclear. For example, it is unclear who is allowed to add work items to the board.
- Not everyone is using the board, which makes it difficult to see progress of all work items.
- The mix of coarse- and fine-grained work items, combined with no breakdown of the coarse-grained work items, make development speed obscure.
- The board can be used as guidance during stand-up meetings.

- Some work items arriving by mail, phone or after a discussion do not end up on the board.
- Descriptions of work items are not always understandable for everyone.
- The Acceptance column is a bottleneck and sometimes performed within the development team, rather than by upper management.
- People often enter the stand-up meeting halfway through.

Most of these points for improvement have been discussed with the team during the second reflection workshop.

6.3.2 Second reflection workshop

All team members were sent an invitation to attend the second reflection workshop, which included the following agenda:

- Improvements to Visualize Workshop that were indicated during the interviews.
- The items that were placed in the deferred column on the board: Doing it Right, Pair Programming, Seasons of the Day, and Customer involvement.
- Continuing the help desk discussion of the last reflection workshop.
- Additionally, standards and conventions for software development were added to the agenda. At this point, they were only set informally.

Reflection workshop

One hour was scheduled for the second reflection workshop, which was attended by the entire team except for one of the directors and a part-time developer. Table 6.3 shows the methodology board at the beginning of the workshop.

The results of the measurement of the level of social adoption were presented at the beginning of the workshop. After this, the items in the *Trying* column were discussed. Both Visualize Workflow and shorter Daily Standup meetings were moved to the *Keep* column. During the measurement of the level of social adoption, though, some team members indicated the standup meeting can be shortened still. The new review, acceptance and released columns were also moved to the *Keep* column.

Next, the items present in the *Upcoming* column were discussed.

- The MoSCoW Prioritization is moved to the *Discard* column as it would remain unused. The analyst indicated the work items that are queued on the board are almost always the highest priority tasks at that moment.

Deferred	Upcoming	Trying	Keep	Discard
<ul style="list-style-type: none"> • Doing it Right • Pair Programming • Seasons of the Day • Customer involvement 	<ul style="list-style-type: none"> • <i>Work items with database changes marked</i> • MoSCoW Prioritization 	<ul style="list-style-type: none"> • <i>Shorter Daily Standups</i> • Visualize Workflow • <i>New columns: review, acceptance, released</i> 	<ul style="list-style-type: none"> • Daily Standup 	

Table 6.3: Methodology board at the beginning of the second reflection workshop.

- The marking of database changes was not yet adopted. During the discussion, participants agreed to also include additional markings, such as highest priority and blocking tasks. The card *Work items with database changes marked* was replaced with *Marking properties of work items*.

The results of the measurement of the levels of social adoption were discussed next, which have led to additional items on the methodology board shown in Table 6.4.

- Bug fixes were blocked in the Acceptance column. Eventually, they were moved to the Done column without acceptance testing so they could be released. The team decided bug fixes would skip the Acceptance column so they could be released quickly. A card for this change was added to the *Trying* column immediately.
- Team members wanted to hold daily standup meetings near the board to easily discuss the tasks they were performing and planned to take on. This was added to the *Trying* column immediately.
- Seasons of the Day was moved to the *Upcoming* column. Team members decided they wanted the morning to focus on their work. Any help desk calls would be handled by developers in the afternoon if necessary.
- Customer involvement was moved to *Upcoming* and the usability researcher was made responsible for finding ways to receive feedback from customers. The director/domain expert decided it was necessary to receive more feed-

back from customers.

- The team discussed improvements on the Visualize Workflow practice, which led to the following cards being added to the *Upcoming* column:
 - The team wanted less active swimlanes.
 - Team members wanted to break coarse work items into smaller, finer tasks.
 - Two additional columns for specification and graphical design were to be added to the board.
 - Marking of additional properties on work items.
- The domain expert was made responsible of performing acceptance testing before functionality is released to customers, added to the *Upcoming* column.
- Standards and conventions were added to the *Upcoming* column as the need for which was indicated during the project inventory.
- At this point, there were already a number of unit tests available that were used as regression tests on the system. However, developers indicated new functionality is not always unit tested adequately. A card for this was added to the *Deferred* column, as team members wanted to further discuss unit testing later on.
- Doing it Right and Pair Programming remained *Deferred* as no time was left to discuss this in detail. Furthermore, team members did not see the need to discuss these topics at this point.

Table 6.4 shows the contents of the methodology board at the end of the reflection workshop. Changes in italic indicate improvements on existing practices.

6.3.3 Changing the methodology

Over the next few weeks, team members started applying the selected improvements. Immediately, the “Specification” and “Design” columns were added to the workflow and existing standards and conventions for writing code were selected and sent to all developers. The number of swimlanes on the board was decreased by removing those representing small or currently blocked features under development.

During this time, the focus of the author changed from facilitating the methodology changes to recording the results of the case study. The team became less focused on applying and improving the selected practices. Although not measured, the adoption of these changes remained low. The team did perform an

Deferred	Upcoming	Trying	Keep	Discard
<ul style="list-style-type: none"> • Doing it Right • Pair Programming • Unit testing 	<ul style="list-style-type: none"> • Seasons of the Day • Customer involvement • <i>Less swimlanes</i> • <i>Task breakdown</i> • <i>Columns for specification and graphical design</i> • <i>Work item types visible on board</i> • <i>Work item properties marked</i> • Standards and conventions • Acceptance testing by domain expert 	<ul style="list-style-type: none"> • <i>Daily Standup at board</i> • <i>Issues do not need acceptance column</i> 	<ul style="list-style-type: none"> • Daily Standup • <i>Shorter Daily Standups</i> • Visualize Workflow • <i>New columns: review, acceptance, released</i> 	<ul style="list-style-type: none"> • MoSCoW Prioritization

Table 6.4: Methodology board after the second reflection workshop.

intermediate reflection workshop on how to change the methodology, although it was unprepared.

6.4 Reflection on the methodology

The methodology at Findesk before applying the methodology-growing framework consisted of the description given in Section 6.1.4. There were weekly team gatherings to discuss what has been achieved the previous week and what is planned for the upcoming week. Furthermore, the project manager was responsible for assigning tasks to team members.

During the project inventory, team members indicated a number of possible improvements of the methodology. Amongst others, these include a better insight in the backlog, more regular team meetings and own selection of tasks. The full list is visible in Section 6.1.4.

Applying the methodology-growing framework has improved the methodology according to these preferences. Team members are now able to select their own tasks from the “To do” column of the board, which also indicates which tasks are scheduled for implementation. The team now holds daily standup meetings, where they discuss progress and impediments.

During the period after the second reflection workshop, the author of this thesis has stopped facilitating use of the framework, as described in Section 6.3.3. From this experience, a team member that facilitates use of the framework appears necessary. This team member would be responsible for keeping up the discipline required to improve the methodology and help the team improve its methodology.

Chapter 7

Evaluation

Three surveys have been held to evaluate the design of the proposed methodology-growing framework. The questionnaires used for these surveys, as well as the results, are discussed in this chapter.

§

Surveys were completed by three target groups to evaluate the methodology-growing framework:

- Employees at Topicus have completed a survey to confirm whether the design of the framework is in line with the way they want to adjust their methodology.
- The team members of Findesk, who have had experience in applying the framework, have completed a survey to confirm whether the framework allowed them to improve their methodology.
- A number of employees experienced with software projects have completed an extended version of the survey Findesk team members completed. This survey confirmed whether they feel the framework is usable in practice for improving methodologies of software projects.

Survey results are visualized in a diverging stacked bar chart. These charts show the percentage or number of participants that gave a certain answer. Positive answers are displayed in green on the left of the centre line, negative answers on the right and in red. This shows the spread of positive and negative opinions.

7.1 Employee survey

A survey was held at various departments of Topicus in order to confirm whether the goals of the proposed framework match with how employees prefer to attain

a methodology. Specifically, whether employees want to collaborate to attain an adequate methodology, and whether they find it worthwhile to collaborate on this.

The survey was kept short at six statements to ensure a high number of respondents. For each statement, participants were asked to give their opinion using a six-point Likert scale with strongly agree, agree, partly agree, partly disagree, disagree, and strongly disagree as possible answers. They were also asked to add comments if they want to clarify their answers. Below, each statement is first described, after which its results are discussed. A copy of the survey (in Dutch) as was handed to the participants is also added in Appendix E.1.

7.1.1 Survey results

The survey has been completed by 105 employees at Topicus. Figure 7.1 contains a diverging stacked bar chart showing the spread of positive and negative opinions. A table containing the results of this survey is available in Appendix F.1.

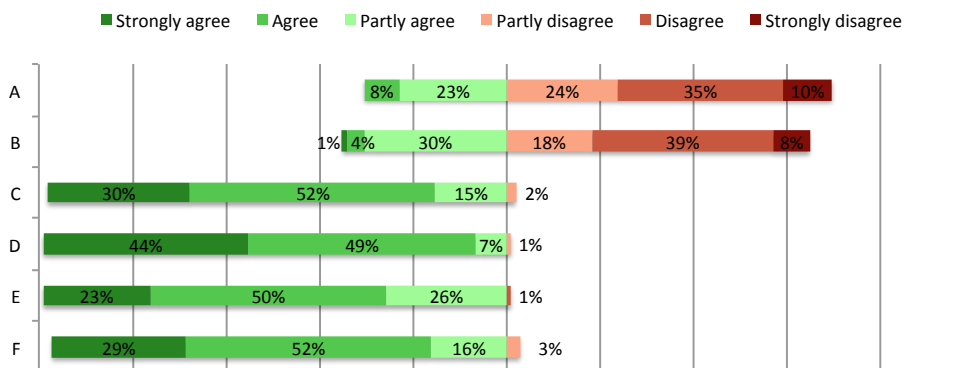


Figure 7.1: Chart of employee survey results.

Methodology selection

The first three statements assess how employees want to attain a methodology:

- A) I want management to select a methodology for me and my team.
- B) I want to determine my own methodology, independent of the rest of my team.
- C) I want to determine the methodology in collaboration with my team.

As shown in Figure 7.1, about 70% of the participants partially to strongly disagree with statement A. However, a significant number of participants (about 30%) have indicated they (partly) agree. Comments that were included indicate employees want management to select a coarse-grained methodology which is adopted and fine-tuned by the team itself. At the very least, employees want to participate with management on the selection of the methodology.

Around 65% of the participants indicate they partially to strongly disagree with having a methodology independent of the rest of the team. Around 30% of the respondents however indicate they partly agree with selecting their own way of working. Furthermore, 4% agrees and 1% strongly agrees with the statement. Comments indicate that some of them, such as graphical interface designers and senior developers, have different schedules and can have responsibilities on different projects. Their work does not necessarily follow the same iterations as development teams and they therefore need an independent way of working, at least partly.

Respondents overall agree with collaborating with their team on the methodology, as shown in Figure 7.1. Only 2% of the participants partially disagree.

Combined, the results yield three conclusions:

- The respondents who indicated they want management to select their methodology also want to collaborate with their team on the methodology.
- The respondents who have indicated they want to select their own methodology independently also want to collaborate with their team on the methodology.
- Almost all respondents prefer to collaborate with their team on how to shape their methodology.

D) I believe it is worthwhile to regularly reflect on how we work as a team and to improve our way of working

The methodology-growing framework uses periodically held reflection workshops with the entire team to improve their methodology. As this activity requires time and effort, employees are asked whether they find this regular collaborate reflection worthwhile.

From the results shown in Figure 7.1, respondents generally agree it is worthwhile to regularly reflect with their team on the methodology in order to improve it.

E) I believe it is worth the effort to actively adjust my methodology and to explore new practices

In applying the proposed framework, team members are required to actively adjust their methodology, as well as to explore new practices. Employees are asked whether they find it worthwhile to do so. Statement F confirms whether team members want to propose and introduce new practices, not only trying out new practices.

As shown in Figure 7.1, participants overall agree with it being worthwhile to actively adjust their methodology and to explore new practices.

F) I want to have the possibility to propose and introduce new practices within my team

The proposed framework allows team members to propose and introduce new practices within their team. Employees are asked for their opinion in order to confirm whether they would want this possibility.

Participants agree on wanting the possibility to propose and introduce new practices within their team. As can be seen in Figure 7.1, the majority (52%) agrees with the statement and 29% of the respondents even strongly agree.

7.1.2 Summary

The designed framework is in line with how participants envision improving the methodology. Participants generally agree on the following:

- Collaborating as a team to determine their methodology.
- Regular reflection being worthwhile to improve the methodology.
- Actively adjusting the methodology and exploring new practices being worthwhile to improve the methodology.
- They want to have the possibility to propose and introduce new practices within the team.

Respondents want to collaborate with their team to determine their methodology. However, some also indicate they would like management to select their methodology, at least partially, as well as wanting to select their own methodology independent of the rest of the team. Although some comments from participants indicate why they would want to do so, additional research would be necessary to conclude if and how the proposed framework would need to be adjusted.

7.2 Team member & experienced employee survey

Aside from the survey held at different departments of Topicus, two other surveys were held to evaluate the methodology-growing framework. Team members at Findesk, the case project described in Chapter 6, were asked for their opinion on the proposed methodology-growing framework and if it allowed them to improve their methodology. Furthermore, the experienced employees that also helped in the design of the framework, as explained in Section 6, have completed an extended version of the same survey.

The survey consisted of the statements below. For each statement, participants were asked to give their opinion using a six-point Likert scale with strongly agree, agree, partly agree, partly disagree, disagree, and strongly disagree as possible answers. They were also asked to add comments if they want to clarify their answers. Below, each statement is first described, after which its results are discussed. A copy of these surveys (in Dutch) as was handed to the participants is available in Appendixes E.2 and E.3.

7.2.1 Survey results

The survey has been completed by nine team members of Findesk and all six experienced employees. Figure 7.2 contains a diverging stacked bar chart showing the spread of positive and negative opinions for the survey completed by the team members of Findesk. Appendix F.2 contains a table of the results for the survey performed on the Findesk team members.

Figure 7.3 shows the same type of graph for the surveys completed by the experienced employees. Similarly, a table of the results for the survey completed by the experienced employees is included in Appendix F.3.

A) The framework enables teams to develop a methodology that suits their work

Team members from Findesk generally agree with the proposed framework allowing teams to develop a methodology that suits their work. Results displayed in Figure 7.2 show some doubts, as three respondents only partly agree and one partly disagrees. A comment from a partly disagreeing participant indicates uncertainty if the framework will result in fixed procedures and methods.

As shown in Figure 7.3, interviewed experienced employees respond more positive. Four of them agree and two strongly agree the framework allows teams to

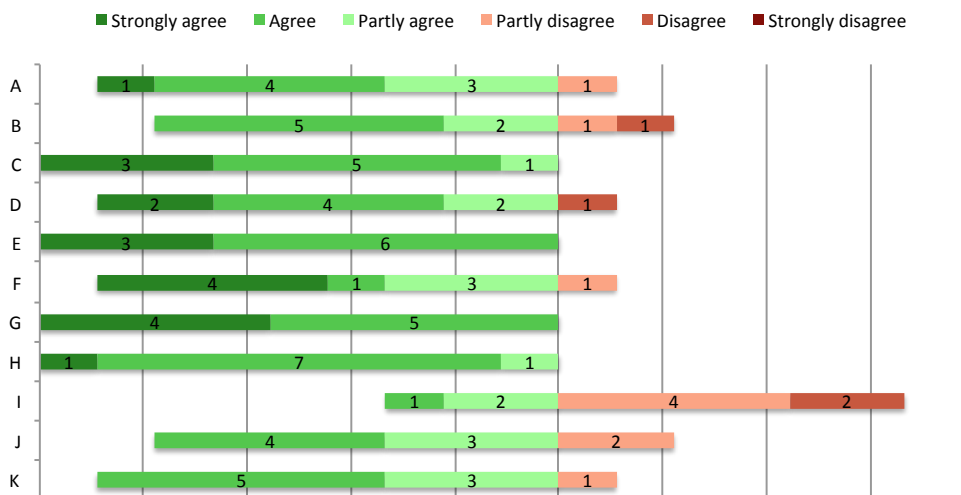


Figure 7.2: Chart of team member survey results.

develop a suitable methodology.

B) The framework enables team members to adjust the methodology to their goals

Seven of the nine team members agree or partially agree the framework allows them to adjust the methodology to their goals. A partially agreeing respondent indicates that because the framework requires a *group effort*, not all team members will ever get exactly what they want individually. The two negatively responding team members, unfortunately, have provided no explanation for their opinion.

All six interviewed experienced employees, however, agree the framework allows team members to adjust their methodology to their goals.

C) To attain a methodology that suits the project, iterative improvement of the methodology is essential

The methodology-growing framework uses iteratively held reflection workshops where the methodology is improved. Although team members are not fully positive on statements A and B, they do agree overall that iterative improvement is essential for attaining a suitable methodology. As shown in Figure 7.2, five respondents agree and three even fully agree with iterative improvement being essential.

Participating experienced employees generally agree with the necessity of itera-

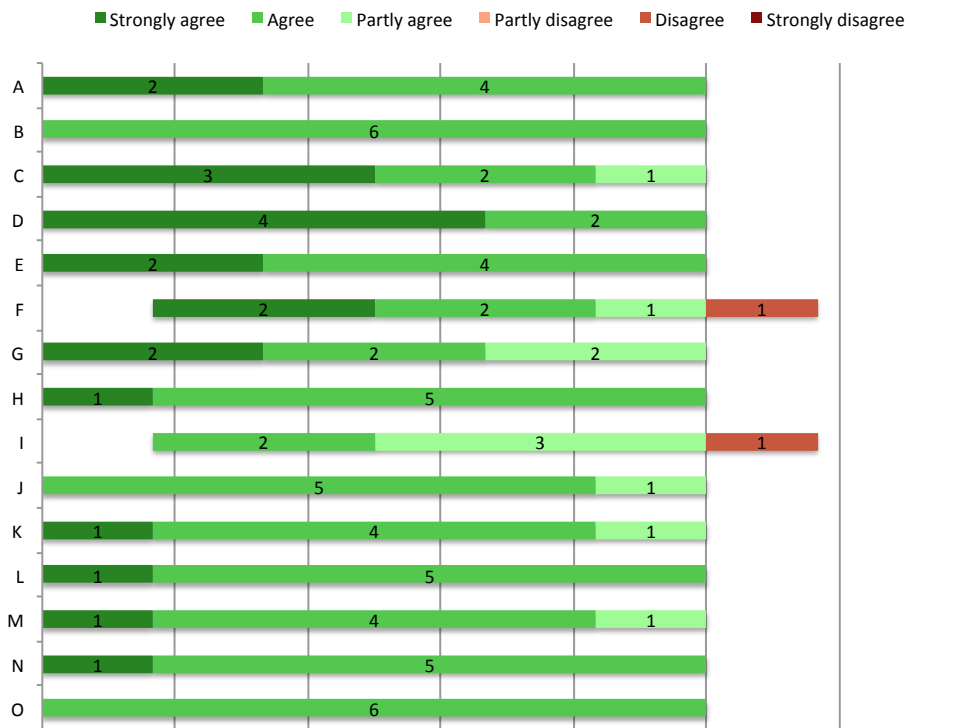


Figure 7.3: Chart of survey results of experienced employees.

tive improvement for attaining a suitable methodology. Figure 7.3 shows two of them agree, three strongly agree, and one partially agrees.

D) Workshops where decisions are made collaboratively are usable to iteratively improve methodologies

Statement C finds out whether team members find iterative improvement of the methodology important for tailoring it to the project. If so, this statement confirms whether the use of reflection workshops is a feasible approach to perform the iterative improvement.

Figure 7.2 shows team members agree overall with reflection workshops allowing for iterative improvement of the methodology. One respondent disagrees, but has left no comments to explain why. One comment of a strongly agreeing respondent indicates that it does require the entire team to be present, willing, and able to perform a reflection workshop. From the experienced employees, most strongly agree reflection workshops are a feasible way to iteratively improve the methodology of a project.

E) A proposal with changes to the methodology, with input from the team, is a good preparation for workshops

The first reflection workshop held in the case project described in Chapter 6 was prepared with the creation of a proposal. This proposal contained some changes that met the identified goals of team members, as well as a few discussion points they have entered themselves. Results show that a proposal with changes, including suggestions from the team, is considered a good preparation for reflection workshops. Six of the nine responding team members agree and the remainder even fully agrees with using proposals for reflection workshops. One comment says that a proposal results in a directed discussion.

The interviewed experienced employees agree a proposal of changes is a good preparation for reflection workshops, as visualized in Figure 7.3. One of the four agreeing respondents indicated using proposals can accelerate improvement of the methodology.

F) The workshops result in a better methodology than management selection

Four of the team members strongly agree that using reflection workshops results in a better methodology than selection by management. Only one team member partially disagrees, but has left no comment explaining his opinion. One strongly agreeing team member indicated he feels management should facilitate in selecting the methodology, and not instruct what it should be.

The experienced employees generally agree using reflection workshops results in a better methodology than selection by management. The one partially agreeing respondent indicated that though the *chance* of attaining a better methodology is higher using reflection workshops, it is not fully certain. One interviewee disagrees with this statement, commenting that vision by management is also important.

G) The workshops result in better adoption of a methodology than management selection

The opinions on statement F indicate there exists some doubts on whether workshops result in better methodologies, both by team members and experienced employees. The opinions on this statement, however, indicate both groups do expect the use of reflection workshops to increase adoption of the methodology within the team.

One of the partially agreeing experienced employees, the same interviewee who

partially agreed with statement F, again indicated the *chance* of improved adoption is higher than when management selects.

H) Holding workshops is worth the effort to attain a better methodology

As the results of Figures 7.2 and 7.3 show, the team members and experienced employees all agree reflection workshops are worth the effort to attain an improved methodology. One of the experienced employees commented reflection workshops are worthwhile until the methodology of the team has been crystallized, i.e. no further improvements are necessary.

I) Holding workshops is sufficient to improve methodologies collaboratively

Both groups find reflection workshops to improve the methodology and its adoption within a project (statements F and G). Furthermore, they also find reflection workshops worthwhile to hold (statement H). However, respondents also indicate the workshops alone are not sufficient to collaboratively improve the methodology. Two team members have commented there should be more focus on actually applying the methodology and that holding reflection workshops is only part of the process to improve the methodology.

Comments from experienced employees indicate only holding reflection workshops is not sufficient to improve a methodology. As shown in Figure 7.3, one of them disagrees, indicating knowledge on methodologies is also necessary for improving a methodology. One partially agreeing respondents indicates only holding workshops would be insufficient and indicates commitment is necessary.

J) Measuring social adoption of newly selected practices is sufficient input for further workshops

As shown in Figure 7.2, team members generally agree with this statement. Two team members indicate they partially disagree, but have not added comments to explain their opinion. Compared to their opinion on other statements, team members are less positive as none strongly agree with this statement.

Experienced employees, on the other hand, agree measuring adoption of newly selected practices is sufficient input for reflection workshops, as shown in Figure 7.3.

K) The framework is usable in practice to develop or improve the methodology of projects

Figure 7.2 shows team members find the framework is usable to be applied in practice. One team member partially disagrees with this statement, but has not

added an explanation for this opinion. Compared to other statements, team members are less positive as none strongly agree with this statement.

Experienced employees have a slightly more positive opinion on this statement. Most agree the proposed methodology-growing framework is usable in practice. However, only one respondent strongly agrees and one merely partially agrees with this statement. No respondent gave comments on their opinion.

7.2.2 Additional survey results

Below are statements that have been included only in the questionnaires of the experienced employees. These statements are about the project inventory and the practice library. As team members have not directly applied these parts of the framework in practice, their opinion has not been asked.

L) The project inventory collects useful properties of the project for improving the methodology

During interviews, the experienced employees have been explained what properties of projects are included and how interviews are held. Results show respondents all respondents agree the project inventory collects properties of projects that are useful for improving its methodology.

M) Performing the project inventory is worth the effort to attain a better methodology

The interviewed experienced employees are asked whether they find performing the project inventory is worthwhile to improve a methodology.

Results show respondents agree performing the project inventory is worth the effort to attain a better methodology.

N) The examples in the practice library are useful suggestions for improving the methodology

The respondents have been explained and have discussed on the practice library and how it can be used within the methodology-growing framework. Therefore, they are asked whether they find the available examples can be useful suggestions for improving the methodology.

Figure 7.3 shows positive opinions on the usefulness of the practice library.

O) Purposes, prerequisites, necessary commitment and tailoring of practices are a useful guideline for creating a proposal for workshops

Following statement N, interviewed respondents are asked whether they find the appended properties of practices in the library useful.

All interviewed experienced employees agree appending practices with their purpose, prerequisites, necessary commitment and tailoring of practices is useful for creating a proposal.

7.2.3 Summary

Team members are very positive on performing iterative improvements of methodologies. They also see reflection workshops useful for achieving this. The reflection workshops are not seen as sufficient, though, one comment indicating the need for more focus on actually applying the methodology. Creating a proposal with changes to the methodology and allowing team members to provide input is seen as a good preparation for these workshops. Measuring social adoption is shown to be at least useful for preparing reflection workshops. However, not every respondent agrees on only performing this measurement to be sufficient.

Interviewed experienced employees have an overall positive opinion on the methodology-growing framework. Figure 7.3 shows only statements F and I gain slightly less positive reactions. The comments on these statements indicate management does still require a role in the process of improving the methodology.

Both groups of respondents are positive on the usability of the methodology-growing framework in practice.

Chapter 8

Conclusion

This chapter concludes the research described in this thesis. The research questions are answered and the contribution of this thesis is discussed.

§

In this thesis, ways to improve methodologies of software projects have been explored. This has led to the design of a framework to support organizations in attaining adequate agile software development methodologies for both new and existing software projects.

The research questions that have directed the design of the proposed framework are answered in Section 8.1. The results of the performed research and the contributions of this thesis to software engineering are discussed in Section 8.2. Furthermore, we provide recommendations for organizations trying to attain a suitable methodology in Section 8.3 and discuss the limitations of this research in Section 8.4.

8.1 Conclusions

Section 1.2 proposes the two research questions that determine the focus of this thesis.

Identifying and incorporating practices

The first research question, “*For a given software project, how can suitable software development practices be identified and incorporated into a coherent methodology?*”, is answered by discussing its three subquestions.

- 1.a) *How can software development practices be documented?*

This thesis builds upon the SEMAT initiative, which defines methodologies as a composition of practices. The SEMAT Kernel provides a metamodel with which methodologies and practices can be composed, simulated, applied, compared, evaluated, measured, taught, and researched [16, 19, 24].

Software development practices can be identified from methodologies from literature and from existing projects. These practices can be documented and reused with other methodologies. When reused, documented practices can be tailored to the project, instead of tailoring an entire methodology.

1.b) *How can usage criteria of documented practices be described?*

To select practices for incorporation within a methodology, the criteria under which they can be applied needs to be known.

We propose to extend documented practices with the following descriptions:

- The purpose of the practice is described. The purpose of a practice should match with what the project and team is trying to improve or attain.
- The prerequisites of the practice indicate when a practice can be incorporated in the methodology and project. To meet the prerequisites of a practice, it can be necessary to either adjust the project or to tailor the practice.
- The necessary commitment of a practice describes what needs to be performed to correctly apply it. This also indicates the methodology weight that will be added when incorporating the practice.
- A description of how the practice can be tailored. This gives suggestions on what can be adjusted to make the practice better suitable to a project.

Adding these descriptions can aid both the selection and the tailoring of practices.

1.c) *How can software development practices be incorporated into a coherent methodology?*

The way practices can be incorporated into a methodology depends on the project. We have proposed a list of questions that can be used as a checklist to incorporate practices into a methodology. A second checklist is proposed to guide all decisions on the methodology. Amongst others, these questions assert the following:

- Whether practices are likely to be adopted and meet the goals of the team.
- Whether there is overlap between practices.
- Whether the ceremony and weight of the methodology fits the project.
- Whether there is a bottleneck situation that can be improved.

Project properties	System and project history, team culture, criticality, priorities, budget, requirement stability, customer availability, planned milestones, environment, problem domain complexity, current backlog
Team member properties	People involved, problem domain experience, software development experience, methodology experience, willingness to change the methodology
Solution properties	Solution complexity, maintainability
Methodology properties	Current methodology, things to keep, things to discard or change, stakeholder requirements, standards and conventions, increment length, length of tasks, new requirements

Table 8.1: Summary of project properties.

Section 5.1 contains the entire checklists.

Subquestion 2.d describes how practices can be adopted by the team by preparing and holding reflection workshops.

Selecting and adopting practices

The second research question, “*How can software development practices be selected and adopted in both new and existing software projects?*”, is answered by answering its subquestions.

2.a) *Which properties of software projects are important for making decisions on a methodology?*

In this thesis, we propose four groups of project properties that are important for making decisions on a methodology:

- Those about the project itself.
- Those about its team members.
- Those about the developed solution.
- Those about its current methodology.

The properties under these groups are shown in Table 8.1. Chapter 4 provides descriptions of all these properties.

2.b) *How do identified properties of software projects apply to new or existing software projects?*

The identified properties all influence decisions on the methodology differently and can apply to new or existing software projects differently. Each project

property includes a description of its possible influence in Chapter 4.

Many of the project properties, though, are still unknown at the beginning of a software project. The team can indicate what they expect for properties, but decisions upon them are likely to be made when they become more clear.

It is not yet known how unknown properties of new projects will affect the project and methodology, as the methodology-growing framework has only been applied with an existing project.

2.c) *How do identified properties of software projects affect decisions regarding a methodology?*

Documented practices are extended with their prerequisites. Prerequisites can point to project properties to indicate if a practice is suitable for the project. The identified properties of software projects thus influences if practices can be incorporated in a methodology.

In this thesis, we propose to use the methodology design principles defined by Cockburn as guidelines for making decisions on a methodology [8]. These are introduced in Chapter 2 and aid decisions on the methodology as a whole, not just on whether and how certain practices can be incorporated. The design principles also rely on the properties of the project where they are applied. For instance, according to the second design principle, excess methodology weight is costly. Whether methodology weight is excess depends on the problem domain complexity and the team size.

Chapter 4 describes the possible influence of each project property on the methodology.

2.d) *By what method can software development practices be adopted in a methodology?*

In this thesis, we propose a methodology-growing framework to support organizations in attaining a methodology for both new and existing projects. The major focus of this framework is on the adoption of an effective methodology by the team, not to attain a theoretically sound methodology.

The framework uses periodically held reflection workshops to collaboratively make decisions on the methodology. During the reflection workshops, the team decides whether and how to incorporate software development practices in the methodology. Agreed changes are to be applied gradually afterwards.

8.2 Contribution

This section summarizes the contribution of this thesis to software engineering. The methodology-growing technique does not cover the following:

- A manner of composing methodologies.
- Guidance and suggestions on what software development practices can be applied.
- Support for both newly starting and existing software projects.

This thesis proposes a framework able to support organizations in attaining adequate agile software development methodologies for new and existing software projects. In summary, this methodology-growing framework extends the original technique as follows:

Following the SEMAT initiative, we propose to define methodologies as a composition of software development practices. Practices are defined as a repeatable approach that provides guidance to deal with some dimension of software development. They can be documented and reused. With this approach, practices that are suitable for a project can be identified and incorporated in its methodology.

The incrementally held reflection workshops of the methodology-growing technique are also included in the framework. Selection and improvement of practices incorporated in the methodology can be performed either during while preparing these workshops. Furthermore, we propose to make an inventory of the properties of a software project to aid in making decisions on a methodology. This inventory is created by holding interviews with team members before holding the first reflection workshop.

Case study and evaluation

The methodology-growing framework has been applied at a small-scale, two year old software project at Topicus in Deventer, The Netherlands. A survey was held with its team members, as well as with a group of six experienced employees. Through the survey, both team members and experienced employees indicated the methodology-growing framework was usable in practice to improve methodologies of projects.

Another survey was held with 105 employees at Topicus. The results of this survey show the methodology-growing framework is in line with how employees would want to improve their methodology. Employees prefer to collaboratively

select the methodology above selection by management. They also find it worthwhile to regularly reflect on their way of working, as well as to actively adjust it and to explore new possibilities in improving it. Lastly, employees indicate they want the possibility to propose and introduce new practices within their team.

8.3 Business recommendations

In this thesis, ways to improve software development methodologies of projects are explored and combined in the methodology-growing framework. Whether the framework is applied completely or not, the following is a list of recommendations for projects trying to attain a suitable methodology.

Iterative and collaborative improvement

A methodology, or *way of working*, is often set at the start of a new project and is rarely consciously adjusted to changing situations. Furthermore, the methodology does not exist as a static entity, but manifests itself in the minds of the team members and in their habits, which can change either consciously or unconsciously. To respond to circumstances and to keep the way of working known to all team members, it is important to iteratively and collaboratively improve the methodology. In doing so, team members can synchronize the methodology and adjust it to new circumstances together.

Facilitator

Using a methodology, as well as adjusting it, requires discipline, which is hard to preserve when responsibility is divided between all team members. It is recommended to make a single team member responsible for arranging iterative improvements and performing decided adjustments on the methodology. Most likely, this responsibility can be given to a project management role.

This team member sees to it that the team applies the methodology. Furthermore, if team members have a problem with some parts of the methodology, they can discuss this with the facilitator. The author of this thesis performed the role of the facilitator for the case project of Chapter 6.

Reflect on the methodology

To improve the methodology of a project, it is important to let the team reflect on previous experiences. Reflect on what has been improved and tried out with the way of working thus far. In doing so, team members can indicate what

they feel should be kept and what should be changed or discarded. It is also important to assert whether there are any parts of the methodology that remain unadopted, to find out why, and to either alter or discard these parts.

Keep knowledge up to date

It is necessary to keep up knowledge on methodologies and on trends in software development. Doing so will enable team members to identify new and better practices to develop software, and to improve the practices they already apply.

Let management facilitate instead of decide

According to studies, methodologies can remain unadopted when management selects it [14]. Surveys held at Topicus also indicate employees want to collaborate on shaping their methodology, instead of letting management select one. Respondents do find management should have a role in selecting the methodology, but instead of making decisions, management should collaborate and facilitate.

It is worthwhile to improve the methodology

Lastly, both literature and held surveys agree it is worthwhile to improve the methodology within a project. Using an inappropriate or unadopted methodology can result in failure of the software project [33]. This can be prevented by iteratively improving the methodology in collaboration.

8.4 Limitations and future work

This section discusses the limitations of the research performed for this thesis.

First and foremost, this thesis only considers methodologies and practices that fall under the agile philosophy. Surveys show up to 80% of the IT industry is applying an agile methodology. Still, this does not entail other methodologies, often referred to as traditional methodologies, are not useful anymore.

The methodology-growing framework has only been applied at a single project. This project had a small project size, a low criticality, a small problem size and already had a very light methodology. The team members were overall experienced in software development and the problem domain, though they had low experience with methodologies. This can and will be a very different story for other projects. There will especially be a difference for newly starting projects, which do not yet have a methodology with which to work.

This thesis is also limited because it has only been applied at a single organization. The culture at Topicus is very collaborative and employees are open to ideas. Most of them have themselves graduated not so long ago. Furthermore, the size of the organization can also be of influence. With around 400 employees, Topicus is relatively small and still manages to use direct communication for most interactions.

The surveys held to evaluate the design of the methodology-growing framework have indicated a number of open issues.

- A survey held amongst employees indicates there exists a preference for management to select a coarse-grained methodology and for the team to work out the finer details collaboratively. Whether and how the framework should be adjusted to this result is yet unknown.
- The same survey indicates a significant number of employees at least partly want to make decisions on their way of working independently. Again, whether and how the framework should be adjusted to this result is not yet known. With a light methodology, however, team members have more space for making decisions on their way of working than with a heavy methodology.
- Surveys with team members and experience employees indicate that holding reflection workshops is not enough for improving the methodology. More focus can be necessary on actually applying the methodology and carrying out changes on it. In this thesis, we have also referred to the appointment of a facilitator role responsible for helping team members apply changes to their methodology. Further research on this task is necessary, though.
- The methodology-growing framework describes methods to prepare for reflection workshops. These preparations can discover ways to improve the methodology. Of these, only the composition of a proposal and the measurement of social adoption have been applied at Findesk, both only once. Of course, other methods for discovering methodology improvements can be added as well.

Finally, the timespan at which the methodology-growing framework has been applied at Findesk has been short. The long-term effects of preparing and performing reflection workshops are still unclear.

8.5 Concluding remarks

In this thesis, we have proposed a framework that enables team members to iteratively improve their methodology to better fit it to the project, its changing environment, and their own preferences. Writings by Alistair Cockburn and David J. Anderson have heavily influenced the creation of this framework. In our opinion, the idea of continuously improving the way of working to attain and keep a suitable methodology is not only useful, but even necessary. Only in rare occasions are the project and its environment stable enough to never have to adjust the way of working.

Bibliography

- [1] Ambler, S. W. (2006). Survey says: Agile works in practice. *Dr. Dobb's Journal*, 31(9):62–64.
- [2] Ambler, S. W. (2010). Introduction to Kanban. Available online at <http://www.drdoobs.com/architecture-and-design/introduction-to-kanban/225702051>.
- [3] Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change For Your Technology Business*. Blue Hole Press, Sequim, WA, USA.
- [4] Avison, D. E. and Fitzgerald, G. (2006). *Information system development: methodologies, techniques & tools*. McGraw-Hill, London, UK.
- [5] Beck, K. and Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, Reading, MA, USA, second edition.
- [6] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for Agile Software Development. Available online at <http://agilemanifesto.org/>.
- [7] Cockburn, A. (2000). Selecting a project's methodology. *IEEE Software*, 17(4):64–71.
- [8] Cockburn, A. (2006). *Agile Software Development: The Cooperative Game (2nd Edition) (Agile Software Development Series)*. Addison-Wesley Professional, Upper Saddle River, NJ, USA.
- [9] DSDM Consortium and others (2008). *DSDM Atern The Handbook*. DSDM Consortium, Ashford, UK.
- [10] Dybå, T., Moe, N. B., and Arisholm, E. (2005). Measuring software methodology usage: challenges of conceptualization and operationalization. In *2005 International Symposium on Empirical Software Engineering, 2005.*, number 7465, pages 432–442. IEEE.

- [11] Fitzgerald, B. (1997). The use of systems development methodologies in practice: a field study. *Information Systems Journal*, 7(3):201–212.
- [12] Fitzgerald, B. (1998). An empirical investigation into the adoption of systems development methodologies. *Information & Management*, 34(6):317–328.
- [13] Hardy, C. J., Thompson, J. B., and Edwards, H. M. (1995). The use, limitations and customization of structured systems development methods in the United Kingdom. *Information and Software Technology*, 37(9):467–477.
- [14] Huisman, M. and Iivari, J. (2006). Deployment of systems development methodologies: Perceptual congruence between IS managers and systems developers. *Information & Management*, 43(1):29–49.
- [15] Ivar Jacobson International (2012). EssWork Practice Workbench (Version 1.1.0). [Software]. Available online at http://www.ivarjacobson.com/Practice_Workbench_Download.
- [16] Jacobson, I. (2013). *The Essence of Software Engineering: Applying the SEMAT Kernel*.
- [17] Jacobson, I., Huang, S., Kajko-Mattsson, M., McMahon, P. E., and Seymour, E. (2012). Semat - Three Year Vision. *Programming and Computer Software*, 38(1):1–12.
- [18] Jacobson, I., Meyer, B., and Soley, R. (2009a). Call for action: the SEMAT initiative. *Dr. Dobb's Journal*, 10. Available online at http://semat.org/?page_id=2.
- [19] Jacobson, I., Meyer, B., and Soley, R. (2009b). Software Engineering Method and Theory - A Vision Statement. Available online at <http://blog.paluno.uni-due.de/semat.org/wp-content/uploads/2012/03/SEMAT-vision.pdf>.
- [20] Jacobson, I., Ng, P. W., and Spence, I. (2007). Enough of Processes - Lets do Practices. *Journal of Object Technology*, 6(6).
- [21] Kniberg, H. and Skarin, M. (2009). *Kanban and Scrum - Making the Most of Both*. C4Media, Los Angeles, CA, USA.
- [22] Myers, M. D. and Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, 17(1):2–26.
- [23] Object Management Group (2008). Software & Systems Process Engineering Meta-Model Specification (SPEM), Version 2.0.
- [24] Object Management Group (2013). Essence - Kernel and Language for Software Engineering Methods. Technical report.

- [25] Riemenschneider, C. K., Hardgrave, B. C., and Davis, F. D. (2002). Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models. *IEEE Transactions on Software Engineering*, 28(12):1135–1145.
- [26] Schwaber, C., Laganza, G., and D’Silva, D. (2007). The Truth About Agile Processes: Frank Answers to Frequently Asked Questions. *Forrester Report*.
- [27] Schwaber, K. and Beedle, M. (2008). *Agile Software Development with Scrum*. Prentice Hall, Upper Saddle River, NJ, USA.
- [28] Serena (2012). There is More to Agile than Development. Available online at <http://www.serena.com/index.php/en/solutions/app-dev-delivery/agile-development/agile-infographic>.
- [29] Shine Technologies (2003). Agile Methodologies: Survey results. Available online at <http://www.shinetech.com/download/attachments/98/ShineTechAgileSurvey2003-01-17.pdf>.
- [30] Vavpotič, D. and Bajec, M. (2009). An Approach for Concurrent Evaluation of Technical and Social Aspects of Software Development Methodologies. *Information and Software Technology*, 51(2):528–545.
- [31] Vavpotič, D. and Vasilecas, O. (2012). Selecting a Methodology for Business Information Systems Development: Decision Model and Tool Support. *Computer Science and Information Systems*, 9(1):135–164.
- [32] VersionOne (2013). 7th Annual State of Agile Development Survey. Available online at <http://www.versionone.com/state-of-agile-survey-results/>.
- [33] Wynekoop, J. L. and Russo, N. L. (1995). Systems Development Methodologies: Unanswered Questions. *Journal of Information Technology*, 10(2):65–73.

Appendixes

Appendix A

Experienced employee interviews

A.1 First interview

Expert review 1 – Project Inventory

Interviewee:	
Roles and responsibilities of interviewee:	
Years of experience in the field of software development:	
Years of experience with using software development methodologies:	
Opinion of these experiences with methodologies:	
Own evaluation of knowledge on methodologies:	
Very knowledgeable	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> No knowledge
Have sent documents been read?	<input type="radio"/> Yes <input type="radio"/> No
Changes on current inventory contents/missing in the inventory:	
Usefulness of development of the method:	
Very useful	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Not useful
Missing in the method:	

Would you like to be interviewed again?	<input type="radio"/> Yes <input type="radio"/> No
Would you be interested in using the method on your own project?	<input type="radio"/> Yes <input type="radio"/> No
Anything else:	

A.2 Second interview

Expert review 2 – Methodology-Growing Framework

Interviewee:	
Have sent documents been read?	<input type="radio"/> Yes <input type="radio"/> No
Creation of proposal:	
First workshop and documentation:	
Changing the work process:	
Future workshops and during increments:	

Opinion of usefulness of the framework: Very useful <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Not useful	
Missing in the framework:	
Would you like to be interviewed again?	<input type="radio"/> Yes <input type="radio"/> No
Would you be interested in using the method on your own project?	<input type="radio"/> Yes <input type="radio"/> No
Anything else:	

Appendix B

Practice Library

This appendix contains practices in text format which were documented using the *EssWork Practice Workbench* [15].

For each practice, an overview is added in SEMAT notation, displaying the alphas it extends and its additional sub-alphas and work products [24]. Furthermore, extended activity spaces and the activities defined for these spaces are displayed. Next, for each defined sub-alpha, its possible states are explained and, for each work product, its possible levels of detail.

For each activity in a practice, its accountable competency and alpha inputs are explained. The states of alphas and levels of details of work products after performing the activity are also listed.

Lastly, the usage criteria of each practice are described (see Section 3.2).

For both the overview and documentation, alphas, activity spaces, and competencies from the SEMAT Kernel are written in italic.

B.1 Weekly Cycle

With the Weekly Cycle, teams reflect on work done, plan work, and implement new functionality weekly. This practice is described in the Extreme Programming methodology [5].

The week is a widely shared time scale. By planning work each week, the team is focused on Friday. By Wednesday, if it is clear the selected tasks won't be completed and ready to deploy by Friday, the team still has the time to focus on completing only the most valuable functionality.

The weekly heartbeat gives a platform for team and individual experiments. For instance, when combined with Pair Programming, the team can try switching pairs every hour for a week and discuss their experience during the next Weekly Planning Meeting.

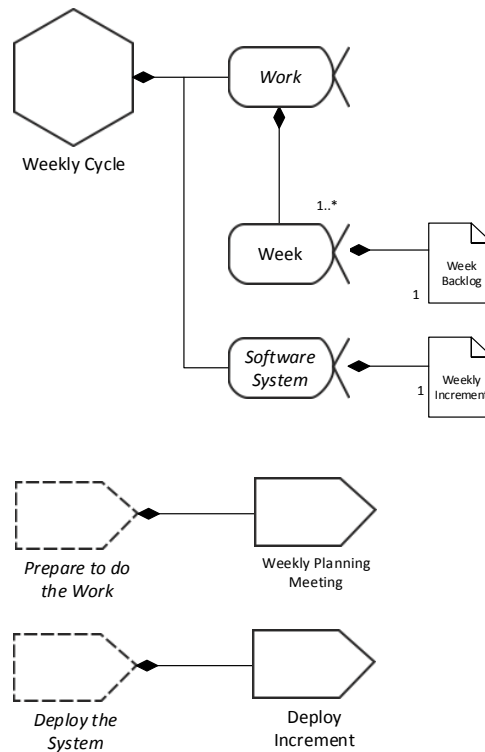


Figure B.1: Weekly Cycle in SEMAT notation.

B.1.1 Usage criteria

Purpose

- Focus to finish a deployable system every Friday.
- Have short iterations and thus receive feedback from customers often.
- Letting teams reflect on their progress often.
- Perform and reflect on experiments within their workplace, such as changing details to the methodology, weekly.

Prerequisites

- Availability of customers for selecting tasks.
- The User Stories practice or a similar practice that has task breakdown and estimation.

Necessary commitment

- Weekly meetings on Monday to schedule work.
- Reaching deployable software weekly.
- Creating Weekly Increments which divide developed functionality in smaller parts.
- Keeping track of a Week Backlog of tasks.

Tailoring

- The Weekly Cycle can be combined with Test-First Programming, in which tests are written before work items are implemented.
- The Weekly Cycle can be combined with acceptance testing, in which the customer tests the functionality of the completed Weekly Increment.
- Some people start their week on Tuesdays or Wednesday. The weekly meeting can be moved as long as it will not pressure the team to work over the weekend.
- Reduce the time necessary for the planning meeting. When applying the Weekly Cycle initially, planning meetings can take hours.

B.1.2 Alphas (things to work with)

Week (sub-alpha under *Work*)

The Weekly Cycle timeboxes development in iterations of a week. During this week, a set of tasks is selected for implementation. At the end of the week, a potentially deployable increment is completed. During the week, some tasks might be dropped to ensure a deployable product will be ready by the end of the week.

Possible states

A Week is first Planned, where the team reflects on previous week(s) and tasks are selected for the upcoming week. At the beginning of the next week, progress is reviewed again and the Week has been Closed.

Planned The Week is in this state when:

- The customer has selected tasks to be implemented.
- Current progress has been discussed and reviewed.

In Progress The Week is in this state when:

- The team has started implementing selected tasks.

Concluded The Week is in this state when:

- Planned tasks are implemented. Note that during the week, planning might have shifted.

Closed The Week is in this state when:

- Progress made in this week has been discussed and reviewed.

B.1.3 Work products (artifacts to maintain)

Weekly Increment (work product under *Software System*)

The Weekly Increment contains the completed functionality of the selected tasks in the Week Backlog of the same Week. The *Software System* is developed by completing Weekly Increments every week.

Possible levels of detail

The Weekly Increment can reach the following levels.

Planned The Weekly Increment has reached this level when:

- Tasks to be finished in the upcoming increment are known.

Complete The Weekly Increment has reached this level when:

- All planned tasks have been completed and their functionality has been added to the increment.

Week Backlog (work product under Week)

The Week Backlog contains all selected tasks to be implemented during the course of the week.

Possible levels of detail

The Week Backlog can reach the following levels.

Filled The Week Backlog has reached this level when:

- Tasks have been selected for the upcoming week.

Completed The Week Backlog has reached this level when:

- All selected tasks in the Week Backlog have been implemented (changes in the planning taken into account).

B.1.4 Activities (things to do)

Weekly Planning Meeting (activity under *Prepare to do the Work*)

The purpose of the Weekly Planning Meeting is to review progress and let the customer pick a week's worth of tasks for implementation. Team members sign up for tasks and estimate them. It is possible task breakdown takes happens during the planning meeting as well. The Weekly Planning Meeting is held every first working day of the week.

- **Accountable competency**

The *Stakeholder Representation* is accountable for performing this activity and responsible for selecting tasks to be implemented. The entire team is also present during the Weekly Planning Meeting.

- **Alpha inputs**

The Weekly Planning Meeting activity performs the following operation(s):

Requirements Tasks are specified from requirements.

- **Completion criteria**

This activity is complete when the team knows what it will be implementing for the upcoming week. This includes achieving the following:

- The Week sub-alpha is Planned.
- The Week Backlog work product is Filled.
- The Weekly Increment work product is Planned.

Deploy Increment (activity under *Deploy the System*)

The purpose of Deploy Increment is to deploy the functionality in the Weekly Increment. Preferably, the new functionality is deployed automatically.

- **Accountable competency**

Development is responsible for deployment and, if available, the automatic deployment.

- **Alpha inputs**

The Deploy Increment activity performs the following operations:

Software System The Software System is deployed.

- **Completion criteria**

This activity is complete when the functionality in the Weekly Increment has been deployed. This includes achieving the following:

- The *Software System* alpha becomes Operational.

- The Week sub-alpha is Concluded.
- The Weekly Increment work product is Complete.

B.2 Daily Standup

Daily Standup Meetings are 15-minute meetings held in the morning to quickly communicate what every team member has done, is going to do, and if there are impediments that block work. Daily team meetings help teams synchronize their work, check planned progress versus actual progress, plan the next working day, and resolve impediments before they impact progress. The Daily Standup meeting is present in the Scrum methodology, but is also mentioned in other methodologies such as the Kanban Method [3, 27].

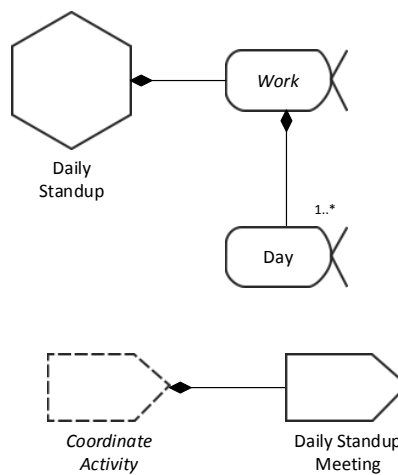


Figure B.2: Daily Standup in SEMAT notation.

B.2.1 Usage criteria

Purpose

- Synchronize the work of team members daily.
- Check whether set goals are still reachable.
- Plan the upcoming working days.
- Communicate impediments before they affect progress.

Prerequisites

- Team members need to be located close enough, preferably in the same room or on the same floor.
- All team members need to be present at the time of the daily meeting.

Necessary commitment

- Fifteen minutes every morning to hold the stand-up meeting.

Tailoring

- Additional standup meetings can be held if it is necessary to discuss progress or if impediments are discovered.
- This practice can be combined with Visualize Workflow (see Appendix B.6) by holding Daily Standup Meetings at the Card Wall. Team members can discuss the Index Cards currently in progress.

B.2.2 Alphas (things to work with)

Day (sub-alpha under *Work*)

Work is discussed every day, giving a form of daily iterations. Note that these iterations do not necessarily result in daily product increments.

Possible states

The day is started with a Daily Standup Meeting, resulting in the Day alpha reaching the Planned State. The Day is Closed when its progress has been discussed in the next Daily Standup Meeting.

Planned The Day is in this state when:

- The Daily Standup Meeting activity is performed

Closed The Day is in this state when:

- The next Day is Planned and the progress made during this Day is discussed.

B.2.3 Activities (things to do)

Daily Standup Meeting (activity under *Coordinate activity*)

The Daily Standup Meeting is held in order to:

- Clarify what has been done the previous day.
- State the plans for this day.
- Indicate blocking issues and impediments.

Discussing this daily, within 15 minutes, helps keep the team synchronized with development and resolve impediments together, before they affect progress.

- **Accountable competency**

Management ensures the Daily Standup Meeting is held within the fixed timeframe and that issues and other lengthy topics are discussed afterwards if necessary.

- **Alpha inputs**

The Daily Standup Meeting activity performs the following operations:

Requirements Progress of achieving the requirements is discussed.

Day The progress of the previous Day is discussed.

- **Completion criteria**

This activity is complete when the Daily Standup Meeting is held and the current Day is Planned. This includes achieving the following:

- The Day sub-alpha is Planned.
- The previous Day sub-alpha is Closed.

B.3 MoSCoW Prioritization

Prioritization of requirements and tasks in levels that specifically tell what happens if they are not met or executed.

Four levels are defined:

- Must Have – crucial for the project and, for high-level requirements, represent the Minimum Usable Subset (MUS).
- Should Have – important but not vital.
- Could Have – wanted or desired but less important.
- Won't Have this Time – these are agreed not to be delivered.

This practice is included in the DSDM Atern methodology [9]. The MoSCoW Prioritization extends the *Requirements* alpha within the Kernel with a Prioritized state, which is described below.

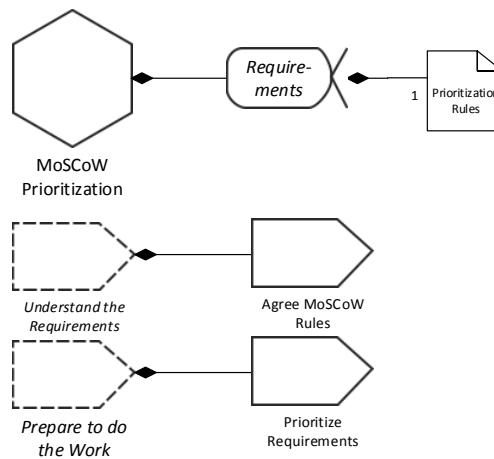


Figure B.3: MoSCoW Prioritization in SEMAT notation.

B.3.1 Usage criteria

Purpose

- Make priority levels of requirements understandable and specific.
- Make it understandable what it means when a requirement or activity has not been met.
- Have a Minimum Usable Subset (MUS) of Must Have requirements that are guaranteed to be delivered by the project.

Prerequisites

- A large enough set of requirements must be clear, i.e. the backlog must be large enough to prioritize.
- High availability of the customer or its representative is necessary for (re-)prioritizing the requirements.
- Development needs to be timeboxed for requirements to be reprioritized after completing an increment.

Necessary commitment

- Setting clear prioritization rules for Must Have, Should Have, Could Have and Won't Have this Time with the customer.
- Reprioritizing requirements after each timebox.

Tailoring

- Reprioritize requirements independent of increments, for instance when delivery is continuous instead of incremental. This does require enough requirements are prioritized as Must and Should Haves at all times to ensure the team focuses on high priority requirements. Prioritization then happens in the context of recent development.
- A team member can be responsible for estimating priority of requirements for customers. The team member must have enough experience with the problem domain to be able to do so.

B.3.2 Alphas (things to work with)

Requirements

The MoSCoW Prioritization practices extends the *Requirements* alpha by adding the Prioritized state to it.

Prioritized Requirements reach this state when:

- Requirements are divided into four sets:
 - Must Have.
 - Should Have.
 - Could Have.
 - Won't Have this Time.
- The Prioritize Requirements activity is performed.

B.3.3 Work products (artifacts to maintain)

Prioritization Rules (work product under *Requirements*)

The Prioritization Rules work product contains the rules with which *Requirements* are prioritized during the Prioritize Requirements activity.

Possible levels of detail

The Prioritization Rules work product can reach the following levels.

None Set The Prioritization Rules work product has reached this level when:

- There are no rules set for dividing requirements into Must Have, Should Have, Could Have and Won't Have this Time.

Rules Set The Prioritization Rules work product has reached this level when:

- The Agree MoSCoW Rules activity is performed.
- Rules are set for dividing requirements into Must Have, Should Have, Could Have and Won't Have this Time.

Sections

Must Have Description of when requirements can be prioritized as Must Have.

Should Have Description of when requirements can be prioritized as Should Have.

Could Have Description of when requirements can be prioritized as Could Have.

Won't Have this Time Description of when requirements can be prioritized as Won't Have this Time.

B.3.4 Activities (things to do)

Agree MoSCoW Rules (activity under *Understand the Requirements*)

Agree on the prioritization rules for Requirements. Below is an example of possible definitions for Must Have, Should Have, Could Have, and Won't Have this Time. The Must Have definition is said to be non-negotiable, as this will have a critical impact on the success of the project.

Must Have Requirements for which the following holds:

- Cannot deliver on target date without this.
- No point in delivering on target date without this.
- Not legal without it.
- Unsafe without it.
- Cannot deliver the business case without it.

Should Have Requirements for which the following holds:

- Important but not vital.
- May be painful to leave out, but the solution is still viable.
- May need some kind of workaround, e.g. management of expectations, some inefficiency, an existing solution, paperwork, etc.

Could Have Requirements for which the following holds:

- Wanted or desirable but less important.
- Less impact if left out (compared to a Should Have).

Won't Have this Time These are requirements the team has agreed it will not deliver.

- **Accountable competency**

Management is accountable for discussing and agreeing prioritization rules with the *Stakeholder Representative*.

- **Alpha inputs**

The Agree MoSCoW Rules activity performs the following operation(s):

Requirements The *Requirements* that will have to be prioritized. The Requirements themselves will be used to discuss when one is a Must Have, a Should Have, etc.

- **Completion criteria**

This activity is complete when rules for assigning priority are agreed upon.

This includes achieving the following:

- The Prioritization Rules work product reaches the Rules Set state.

Prioritize Requirements (activity under *Prepare to do the Work*)

At the end of each increment, as well as when first adopting this practice, prioritize the still unsatisfied requirements (again) to prepare for the next increment. Note that although a requirement can be a Must Have for the entire project, it can assigned a lower priority for an increment if it will be developed later on.

- **Accountable competency**

The *Stakeholder Representative* is responsible for prioritizing requirements, as he is able to determine what has priority and what is the Minimum Usable Set.

Management is responsible for challenging whether a requirement is a Must Have, as requirements in the Minimum Usable Set will have a critical impact on the success of the project.

- **Alpha inputs**

The Prioritize Requirements activity performs the following operations:

Requirements The *Requirements* that are prioritized.

- **Completion criteria**

This activity is complete when Requirements are prioritized according to the Prioritization Rules. This includes achieving the following:

- The *Requirements* alpha reaches the Prioritized state.

B.4 Pair Programming

Pair Programming allows team members to collaborate in pairs on a single task.

This practice allows pairs to:

- Keep each other on the task.
- Brainstorm refinements to the system.
- Clarify ideas.
- Take initiative when the partner is stuck, thus lowering frustration.
- Hold each other accountable to the team's practices.

With Pair Programming, code is written with two team members at one machine. It is important that pairs give each other enough space and that pairs are rotated often. If a pair member needs to explore something on his or her own, this should be allowed. After exploration, bring the resulting idea back to the team and, if accepted, the pair starts working together again.

Pair Programming is described in the Extreme Programming methodology [5]. It has no work products.

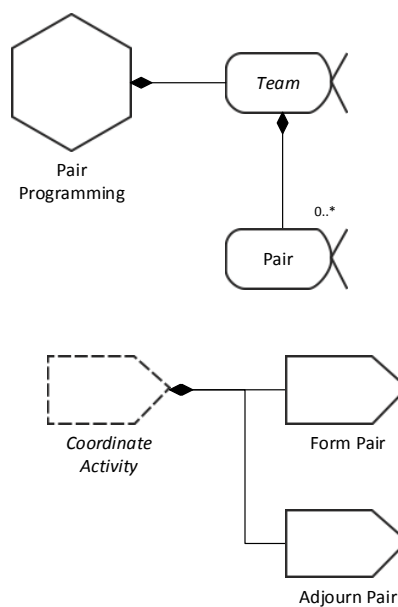


Figure B.4: Pair Programming in SEMAT notation.

B.4.1 Usage criteria

Purpose

- Team members are divided into pairs so that they:
 - Keep each other on the task.
 - Brainstorm refinements to the system.
 - Clarify each other's ideas.
 - Take initiative when the partner is stuck.
 - Hold each other accountable to the team's way of working.

Prerequisites

- Team members must be allowed to work alone on an idea, as long as they return with the idea (not the code) when they are done exploring.
- Team members must be willing to form pairs.
- Team members must be allowed to indicate if they are uncomfortable pairing with a team member.

Necessary commitment

- Assigning tasks to formed pairs of team members.

Tailoring

- Pairs can be rotated every so often, for instance every hour.
- Pair programming can be used only when a team member indicates he needs help on a problem.

B.4.2 Alphas (things to work with)

Pair (sub-alpha under *Team*)

The team is split into Pairs to develop software on one machine.

Possible states

A Pair is first Formed by performing the Form Pair activity. When they have started their assigned tasks, the Pair is in the Working state. When they have completed their assigned tasks, the Pair is Adjourned.

Formed The Pair is in this state when:

- Two team members are assigned to as a Pair.

Working The Pair is in this state when:

- The Pair is working collaboratively to develop software.

Adjourned The Pair is in this state when:

- The assigned tasks of the Pair is completed and the Pair is disbanded.

B.4.3 Activities (things to do)

Form Pair (activity under *Coordinate Activity*)

Form pairs from team members so two programmers develop behind the same machine.

- **Accountable competency**

Management is responsible for forming pairs of developers.

- **Alpha inputs**

The Form Pair activity performs the following operation(s).

Team The team is (partly) grouped into pairs.

- **Completion criteria**

This activity is complete when a pair has been formed. This includes achieving the following:

- The Pair alpha is Formed.

Adjourn Pair (activity under *Coordinate Activity*)

Adjourn pairs when they are finished with their assigned tasks and are not given new tasks. The developers can then form a new pair or continue with tasks on their own.

- **Accountable competency**

Management is responsible for deciding when a pair is adjourned. If pairs have tasks assigned, they can be adjourned by default if these tasks are completed.

- **Alpha inputs**

The Form Pair activity performs the following operation(s).

Pair The pair is adjourned.

- **Completion criteria**

This activity is complete when a pair has been adjourned. This includes achieving the following:

- The Pair alpha is Adjourned.

B.5 Seasons of the Day

This practice allows teams to focus on progress of tasks in the morning, without distractions. The afternoon can be used for meetings, discussions, etc.

Each day is considered as an entire year of seasons:

- The morning is the spring, in which the team can concentrate on tasks.
- The lunch break is the summer, in which the team can rest.
- The afternoon is seen as the autumn, during which focus is lower and distractions are allowed.
- The evening is seen as the winter for sleep.

This practice divides each working day in the morning for focused work, the lunch break for resting, and the afternoon for communication. Adopting this practice will therefore influence other practices of the methodology.

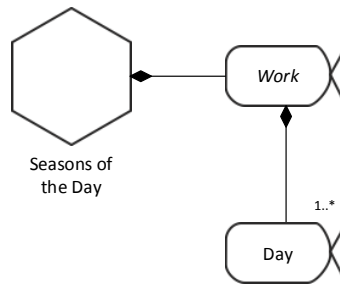


Figure B.5: Seasons of the Day in SEMAT notation.

B.5.1 Usage criteria

Purpose

- Having the morning for focusing on completing assigned tasks without distractions.

Prerequisites

- The possibility to avoid distractions in the morning by deferring them to the afternoon.
- A small team in close proximity, preferably a single room for all development efforts.

Necessary commitment

- Moving any appointments to the afternoon.
- Making arrangements with the rest of the team, for instance the help desk, to avoid distractions in the morning.

Tailoring

- The team can select different time blocks for focused work and for communication. For instance, they can select to hold meetings only on Mondays and Fridays and reply to calls from customers only after 3 p.m.

B.5.2 Alphas (things to work with)

Day (sub-alpha under *Work*)

Each day has a different focus for the morning and afternoon. In the morning, distractions are to be avoided completely, focusing entirely on the assigned tasks. Meetings and discussions should be held in the afternoon.

Possible states

A Day first starts in the morning (or Spring), during which the team can concentrate on their tasks without distractions. The team rests during the lunch break (or Summer) and allows distractions, meetings, discussions, etc. in the afternoon (or Autumn).

Spring The Day is in this state in the morning. The morning is used for focusing on assigned tasks without distractions.

Summer The Day is in this state during the lunch break. The lunch break allows the team to rest from their focused morning.

Autumn The Day is in this state in the afternoon. The afternoon allows for distractions, meetings, and discussions.

Winter The Day is in this state in the evening. The evening is outside of working hours. Team members do not have any responsibility towards the project.

B.6 Visualize Workflow

Visualizing the workflow enables the team to track work items along the project's workflow. Visualize Workflow is described in The Kanban Method, which contains a set of practices to improve the throughput of a project [3].

To apply this practice, tasks are specified on cards, which are put on a card wall. The card wall is divided in columns, representing the workflow of the project. For instance, queued cards first move to development, followed by testing, done, and finally deployed.

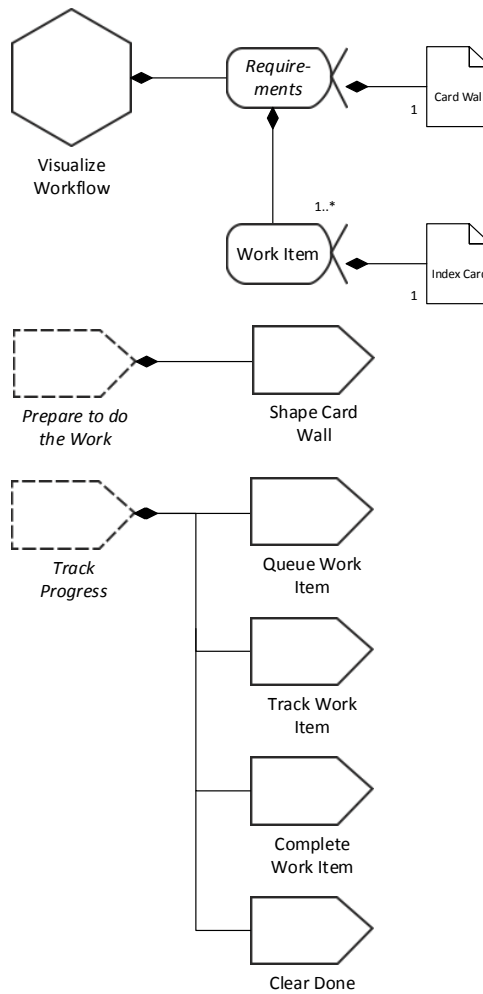


Figure B.6: Visualize Workflow in SEMAT notation.

B.6.1 Usage criteria

Purpose

- Track work items along the workflow of the project.
- Visualize work items on physical index cards which can be put on a card wall representing this workflow.
- Visualize impediments by cards that are stuck in a column.
- Identify possible improvements for the project, such as optimizing bottleneck activities, visualized by an overcrowded column on the card wall.

Prerequisites

- The team should be located close together, with the developers preferably in the same room.
- A card wall in the same room as the developers.

Necessary commitment

- Creating an outline of the workflow within the project.
- Shaping a card wall according to this workflow.
- Queuing and tracking work items along the workflow.
- Clearing index cards whose associated work has been completed and deployed.

Tailoring

- This practice can be combined with the Limit Work in Progress practice, also from the Kanban Method. Progress can be limited by setting a limit to the number of cards in each column of the card wall.
- Use an alternative of a card wall, such as a software system that has the functionality of a card wall.
- Visualize Workflow can be combined with practices that describe work items in a specific form, such as User Stories or Use Cases, which are then described on the index cards.
- Use horizontal swimlanes along the card wall to group certain work items, such as those belonging to a specific feature.

B.6.2 Alphas (things to work with)

Work Item (sub-alpha under Requirements)

A Work Item describes an item of work necessary for completing a requirement.

Possible states

A Work Item is first Created when it becomes known to the team. When it is Queued, an Index Card is Written that references to the Work Item and added to the "To Do" column of the Card Wall.

When progress of the Work Item is started, the In Progress state is reached. The associated Index Card is put into an "In Progress" column of the Card Wall. This indicates the current progress of the Work Item. After all necessary work is done, the Index Card is moved to the "Done" column. Correspondingly, the Work Item reaches the Done state.

Finally, for instance when a release takes place, the Card Wall is cleared and Index Cards that no longer contributing to current development are removed from the Card Wall.

Created The Work Item is in this state when:

- The item of work becomes known to the team.

Queued The Work Item is in this state when:

- An Index Card describing the Work Item is written.
- The Index Card is added to the "To Do" column of the Card Wall.

In Progress The Work Item is in this state when:

- The associated Index Card is moved from the "To Do" column into one of the "In Progress" columns.
- Progress of the Work Item has started.

Done The Work Item is in this state when:

- The associated work is completed.
- The associated Index Card is moved to the "Done" column of the Card Wall.

B.6.3 Work products (artifacts to maintain)

Card Wall (work product under Requirements)

The Card Wall visualizes Index Cards within the outline of the project's workflow.

Possible levels of detail

The Card Wall only has the Shape Decided state, as its shape is decided when it is created. The Card Wall can be created and shaped by performing the Shape Card Wall activity.

Shape Decided The Card Wall has reached this level when:

- The Shape Card Wall activity has been performed.
- Its columns give a detailed outline of the workflow.

Sections

The Card Wall contains three Sections.

To Do The column of the Card Wall where Index Cards are queued.

In Progress The columns of the Card Wall that contain Index Cards whose associated Work Items are in progress.

Done The column where Index Cards of completed Work Items are put.

Index Card (work product under Work Item)

An Index Card describes a Work Item. Index Cards are tangible card to be placed on the Card Wall, where it follows the set workflow of the project.

Possible levels of detail

The Index Card can reach the following levels.

Written The Index Card has reached this level when:

- A card describing the associated Work Item is written.
- The card is added to the Card Wall's "To Do" column.

Cleared The Index Card has reached this level when:

- The associated Work Item is completed.
- The Clear Done activity is performed.

B.6.4 Activities (things to do)

Shape Card Wall (activity under *Prepare to do the Work*)

Shaping the Card Wall can be done so that:

- The Card Wall shows an outline of the workflow.
- Index cards can be put in states on the Card Wall.
- Progress of work items can be visualized on the Card Wall.

To outline the workflow, it may be necessary to perform Value Stream Mapping or a similar technique. Later on, the Card Wall can be fine-tuned to better fit the workflow, as using the Card Wall will make details more clear.

- **Accountable competency**

Management will make an outline of the workflow and visualize it on the Card Wall.

- **Alpha inputs**

The Shape Card Wall activity performs the following operation(s):

Way of Working The outline of the workflow is created from the *Way of Working* within the project.

- **Completion criteria**

This activity is complete when the Card Wall shows the outline of the work process. This includes achieving the following:

- The Card Wall reaches the “Shaped decided” level of detail

Queue Work Item (activity under *Track Progress*)

Work Items are written on an Index Card and queued so that:

- An Index Card is added to the "To Do" column of the Card Wall.
- Items of work become known to the development team.
- Team members can select this item of work to begin its progress.

Queueing a Work Item means physically writing an Index Card and putting it on the Card Wall's "To Do" column.

- **Accountable competency**

Management is responsible for selecting which items of work to queue to ensure the team will focus on those tasks first.

- **Alpha inputs**

The Queue Work Item activity performs the following operations:

Work Item An Index Card for a Work Item is written and put on the board.

- **Completion criteria**

This activity is complete when an Index Card is put on the Card Wall. This includes achieving the following:

- The Work Item sub-alpha becomes Queued.
- The Index Card work product is Written.

Track Work item (activity under *Track Progress*)

Tracking Work Items is achieved by moving the associated Index Cards along the Card Wall's "In Progress" states. This visualizes the Work Item's progress along the workflow of the project.

- **Accountable competency**

Development is responsible for taking on Work Items and moving the associated Index Cards on the Card Wall.

- **Alpha inputs**

The Track Work Item activity performs the following operations:

Work Item The Work Item that is tracked with an Index Card on the Card Wall.

- **Completion criteria**

This activity is completed whenever an Index Card is moved from or between "In Progress" columns on the Card Wall. This includes achieving the following:

- The Work Item sub-alpha is In Progress.

Complete Work Item (activity under *Track Progress*)

When completing work on a Work Item, the associated Index Card is moved to the "Done" column on the Card Wall.

- **Accountable competency**

Development is responsible for finishing items of work and moving the associated Index Card to the "Done" column.

- **Alpha inputs**

The Complete Work Item activity performs the following operations:

Work Item The Work Item of which the work is completed.

- **Completion criteria**

This activity is complete when the associated task is done. This includes achieving the following:

- The Work Item sub-alpha reaches the Done state.

Clear Done (activity under *Track Progress*)

Clearing the "Done" column (often when deploying or releasing a new system) on the Card Wall is done so that only the Work Items whose functionality have yet to be deployed are still visible.

Clearing this column prevents a "Done" column containing old cards that do not visualize the current state of development.

- **Accountable competency**

As *Management* is responsible for deciding when to release or deploy what new functionality, they are also responsible for clearing the "Done" column.

- **Alpha inputs**

The Clear Done activity performs the following operations:

Work Item The Work Items whose Index Cards are cleared from the "Done" column.

- **Completion criteria**

This activity is performed when a release or deployment is done and old cards need to be removed from the "Done" column. This includes achieving the following:

- The *Software System* alpha becomes Operational.
- The Index Card work product becomes Cleared.

Appendix C

Project inventory interview

Interviewee:	
Roles and responsibilities of interviewee:	
Time on the project:	
Years of experience in the field of software development:	
Own evaluation of competence with current roles and responsibilities: Very competent <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Not competent	
Years of experience with using (agile) software development methodologies:	
Own evaluation of knowledge on (agile) methodologies: Very knowledgeable <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> No knowledge	
Perceived team culture: Very participative <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Very autocratic	
Willingness to change the work process and methodology: Very willing <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Not willing	

Goal for using the framework:	
Number, roles and FTE of people involved:	
Distribution of people involved:	<input type="radio"/> Same room <input type="radio"/> Same floor <input type="radio"/> Same building <input type="radio"/> Distributed nationally <input type="radio"/> Distributed globally
Project priorities: <input type="radio"/> Time to market <input type="radio"/> Low costs <input type="radio"/> Correctness <input type="radio"/> Traceability	Other project priorities:
Cost limitations (budget): Very strict <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Very sufficient	
System criticality:	<input type="radio"/> Loss of comfort <input type="radio"/> Loss of discretionary monies <input type="radio"/> Loss of essential monies <input type="radio"/> Loss of life
Problem domain complexity:	
Maintainability and difficulty of adding new (major) features to current solution: Very easy <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Very difficult	

Complexity of current solution (<i>according to measured software metrics?</i>):	
Increment length (number of weeks):	
Length of lowest-level tasks (hours or days):	
Stability of requirements: Very predictable <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Very unpredictable	
Arrival of new requirements in the project:	
Current backlog size (estimated duration):	
Tightness to standards and conventions:	<input type="radio"/> Loose, no standards and conventions set <input type="radio"/> Standards and conventions are set <input type="radio"/> Strict, use is monitored for correspondence
Short project history:	
Planned milestones: <input type="radio"/> Maintenance only <input type="radio"/> Minor upgrades <input type="radio"/> Major upgrades <input type="radio"/> New versions	Other planned milestones:

Customer collaboration (availability and overall communication): Very collaborative <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Not collaborative
Requirements from stakeholders (in terms of deliverables):
Other projects, organizations and external parties involved (the environment):
Stability of environment: Very predictable <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Very unpredictable
Currently used methodology (as a basic workflow):
Things in the project and methodology to keep:
Things in the project and methodology to change or discard:
Anything else:

Appendix D

Social adoption measurement form

Interviewee:							
<i>Daily Standup Meeting</i>							
Given the opportunity to use <i>Daily Standup Meeting</i> , how often do you use it?							
Never	20% or less	20-40%	40-60%	60-80%	80% or more	Always	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
Use of <i>Daily Standup Meeting</i> is encouraged as a common activity within the team							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
Use of <i>Daily Standup Meeting</i> is routine and is used at every opportunity							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
I consistently follow the instructions of <i>Daily Standup Meeting</i> when I use it							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
Instructions of <i>Daily Standup Meeting</i> are precisely described							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree

Kanban board

Given the opportunity to use <i>Kanban board</i> , how often do you use it?							
Never	20% or less	20-40%	40-60%	60-80%	80% or more	Always	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
Use of <i>Kanban board</i> is encouraged as a common activity within the team							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
Use of <i>Kanban board</i> is routine and is used at every opportunity							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
I consistently follow the instructions of <i>Kanban board</i> when I use it							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree
Instructions of <i>Kanban board</i> are precisely described							
Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agree

Can the changed work process have negative impact on the success of the project (and the increment currently under development)?	<input type="radio"/> Yes <input type="radio"/> No
Anything else:	

Appendix E

Questionnaires

E.1 Evaluation by employees

Enquêteformulier

Voor mijn afstuderen doe ik onderzoek naar werkwijzen binnen een project. Om projecten te ondersteunen een passende werkwijze te krijgen is een framework ontwikkeld. Deze enquête wordt afgenomen om te controleren of het ontwerp van dit framework in lijn is met hoe teamleden hun eigen werkwijze zouden willen aanpassen.

Bij voorbaat mijn dank voor het invullen!

Richard Cornelissen
Afstudeerder en part-time developer
Topicus Findesk

	Zeer eens	Eens	Deels eens	Deels oneens	Oneens	Zeer oneens
Ik wil dat management voor mij en mijn team een werkwijze selecteert.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ik wil zelf mijn eigen werkwijze bepalen, onafhankelijk van de rest van mijn team.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ik wil gezamenlijk met mijn team onze werkwijze beslissen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ik vind het de moeite waard om regelmatig met mijn team op onze werkwijze te reflecteren en deze te verbeteren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ik vind het de moeite waard om mijn werkwijze actief aan te passen en nieuwe werkwijzen te verkennen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ik wil de mogelijkheid hebben om nieuwe werkwijzen binnen mijn team voor te stellen en te introduceren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Overige opmerkingen:

E.2 Evaluation at Findesk

Findesk enquêteformulier

Enquête omtrent een framework voor het in samenwerking verkrijgen van een passende methodologie. Onder elke vraag is ruimte voor extra opmerkingen en redenen voor antwoorden.

Bij voorbaat mijn dank voor het invullen!

Richard Cornelissen
Afstudeerder en part-time developer
Topicus Findesk

	Zeereens	Eens	Deelseens	Deelseoneens	Oneens	Zeereoneens
Het framework geeft teams de mogelijkheid een methodologie te ontwikkelen die bij hun werk past.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het framework geeft teamleden de mogelijkheid de methodologie naar hun doelen aan te passen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Om een methodologie passend te maken aan een project is het iteratief verbeteren hiervan essentieel.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Voor het iteratief verbeteren zijn workshops voor gezamenlijke beslissingen met het team bruikbaar.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Een proposal met wijzigingen, met invulling van het team, is een goede voorbereiding voor de workshop.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	Zeer eens	Eens	Deels eens	Deels oneens	Oneens	Zeer oneens
De workshops zorgen voor een betere methodologie dan wanneer management een methodologie kiest.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
De workshops zorgen voor betere adoptie van een methodologie dan wanneer het management kiest.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het houden van workshops is de moeite waard om tot een betere methodologie te komen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het houden van workshops is voldoende om gezamenlijk de methodologie te verbeteren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het meten van adoptie van nieuwe practices is een passende invulling voor verdere workshops.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het framework is bruikbaar om in de praktijk de methodologie van projecten te creëren of verbeteren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

E.3 Evaluation with experienced employees

Expert enquêteformulier

Enquête omtrent een framework voor het in samenwerking verkrijgen van een passende methodologie.
Onder elke vraag is ruimte voor extra opmerkingen en redenen voor antwoorden.

Bij voorbaat mijn dank voor het invullen!

Richard Cornelissen
Afstudeerder en part-time developer
Topicus Findesk

	Zeereens	Eens	Deelseens	Deelseoneens	Oneens	Zeereoneens
Het framework geeft teams de mogelijkheid een methodologie te ontwikkelen die bij hun werk past.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het framework geeft teamleden de mogelijkheid de methodologie naar hun doelen aan te passen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
De inventarisatie verzamelt nuttige eigenschappen van het project ter verbetering van de methodologie.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het uitvoeren van de inventarisatie is de moeite waard om tot een betere methodologie te komen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Om een methodologie passend te maken aan een project is het iteratief verbeteren hiervan essentieel.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Voor het iteratief verbeteren zijn workshops voor gezamenlijke beslissingen met het team bruikbaar.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Een proposal met wijzigingen, met invulling van het team, is een goede voorbereiding voor de workshop.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>






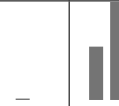
	Zeer eens	Eens	Deels eens	Deels oneens	Oneens	Zeer oneens
De bijgevoegde practice-voorbeelden zijn nuttig voor suggesties ter verbetering van de methodologie.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Doelen, voorwaarden, nodige commitment en variaties van practices zijn een nuttige leidraad voor het maken van een proposal.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
De workshops zorgen voor een betere methodologie dan wanneer management een methodologie kiest.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
De workshops zorgen voor betere adoptie van een methodologie dan wanneer het management kiest.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het houden van workshops is de moeite waard om tot een betere methodologie te komen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het houden van workshops is voldoende om gezamenlijk de methodologie te verbeteren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het meten van adoptie van nieuwe practices is een passende invulling voor verdere workshops.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Het framework is bruikbaar om in de praktijk de methodologie van projecten te creëren of verbeteren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Appendix F

Survey results

This appendix contains the raw results of the surveys discussed in Chapter 7. For each question, the number of answers for each point is displayed. A small graph is added for each question to visualize the spread of the answers.

F.1 Employee survey results

	A	B	C	D	E	F
Strongly agree		1	32	46	24	30
Agree	8	4	55	51	53	55
Partly agree	24	32	16	7	27	17
Partly disagree	25	19	2	1		3
Disagree	37	41			1	
Strongly disagree	11	8				
						

F.2 Findesk team member survey results

	A	B	C	D	E	F
Strongly agree	1		3	2	3	4
Agree	4	5	5	4	6	1
Partly agree	3	2	1	2		3
Partly disagree	1	1				1
Disagree		1		1		
Strongly disagree						

	G	H	I	J	K
Strongly agree	4	1			
Agree	5	7	1	4	5
Partly agree		1	2	3	3
Partly disagree			4	2	1
Disagree			2		
Strongly disagree					

F.3 Experienced employee survey results

	A	B	C	D	E
Strongly agree	2		3	4	2
Agree	4	6	2	2	4
Partly agree			1		
Partly disagree					
Disagree					
Strongly disagree					

	F	G	H	I	J
Strongly agree	2	2	1		
Agree	2	2	5	2	5
Partly agree	1	2		3	1
Partly disagree					
Disagree	1			1	
Strongly disagree					

	K	L	M	N	O
Strongly agree	1	1	1	1	
Agree	4	5	4	5	6
Partly agree	1		1		
Partly disagree					
Disagree					
Strongly disagree					
	