FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Agile Forward: A simple toolkit for process improvement

Augusto Amorim Cravo da Silva

DISSERTATION CONDUCTED IN PARTNERSHIP WITH
ZENDESK, DENMARK

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ademar Manuel Teixeira de Aguiar

Approved on July 6, 2016

Agile Forward:
A simple toolkit for process improvement

Augusto Amorim Cravo da Silva

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Nuno Honório Rodrigues Flores
External Examiner: Doctor Ana Margarida Pisco Almeida
Supervisor: Doctor Ademar Manuel Teixeira de Aguiar

Approved on July 6, 2016

Abstract

As agile methods mature and its successful adoption becomes more and more widespread in the software industry, teams get more open and capable of responding to change. The adoption of agile methods cannot be seen as an end by itself, it must be seen as a mindset pushing teams to keep improving to be more predictable, to deliver faster and with more quality. To achieve this goal, agile methods, e.g. Scrum, include practices, such as retrospectives, that continuously challenge teams to find out how they can improve. However, many teams fail to detect important areas where they can improve.

This work focuses on the activities of agile software process improvement, from analysis to concrete planning and implementation phases, with the goal of understanding the overall phenomena: key obstacles, common practices, and recommendations.

In concrete, we define and propose a practical toolkit to help "Process Masters", e.g. Scrum Masters, and other important players, to detect issues, by collecting relevant data, finding concrete solutions, putting these into practice and evaluating the results.

The work was based on a case study conducted with a team of the software company Zendesk where different methods, tools, practices and techniques were explored and applied in order to improve their process. An action research method was used by following an iterative improvement methodology where the researcher and practitioners worked together to solve the problem.

A validation phase was then conducted for the proposed toolkit. In this phase, other agile teams were interviewed to access the applicability of the toolkit. The results obtained confirm there is an interest in simple tools and ways to conduct process improvement and that the most adequate target for the toolkit are "Process Masters" who are an active part of the development team and thus are not able to neither continuously focus on process improvement activities or have an external vision of the events.

Resumo

À medida que as metodologias ágeis amadurecem e a sua adopção bem sucedida se torna cada vez mais presente na indústria de software, as equipas ficam mais receptivas e capazes de responder à mudança. A adopção de metodologias ágeis não pode ser vista como um fim em si mesma, tem que ser vista como uma mentalidade que faça as equipas continuar a melhorar de forma a serem mais previsíveis, a produzirem mais rapidamente e com maior qualidade. Para atingir este objectivo, as metodologias ágeis, p. ex. Scrum, incluem práticas, tais como as retrospectivas, que continuamente desafiam as equipas a encontrar formas de melhorar. No entanto, muitas equipas falham na detecção de áreas importantes onde podem melhorar.

Este trabalho foca nas actividades de melhoria do processo ágil de desenvolvimento de software, da análise às fases de planeamento concreto e implementação, com o objectivo de perceber o fenómeno geral: obstáculos chave, práticas comuns e recomendações.

Nomeadamente, define-se e propõe-se um kit de ferramentas prático para ajudar "Process Masters", p. ex. Scrum Masters, e outros participantes importantes, a detectar problemas, recolhendo dados relevantes, encontrando soluções concretas e pondo-as em prática.

Este trabalho foi baseado num caso de estudo conduzido na empresa de software Zendesk onde diferentes métodos, ferramentas, práticas e técnicas foram exploradas e aplicadas de forma a melhorar o seu processo. Um método de investigação-acção foi usado seguindo uma metodologia iterativa de melhoria onde o investigador e os praticantes trabalharam juntos para resolver o problema.

Foi depois conduzida uma fase de validação para o kit de ferramentas proposto. Nesta fase, outras equipas ágeis foram entrevistadas para avaliar a aplicabilidade do kit. Os resultados obtidos confirmam que há um interesse em ferramentas e formas simples de conduzir a melhoria de processo e que o público-alvo mais adequado para este kit de ferramentas são "Process Masters" que são uma parte activa da equipa de desenvolvimento e por isso não são capazes de se focarem continuamente em actividades de melhoria do processo or ter uma visão externa dos acontecimentos.

Acknowledgements

The idea of this dissertation started back in April 2015. Until February 2016, this dissertation could have been done in three different companies. One in the UK, another in the Netherlands and another in Denmark. Faith was that it would be done in Denmark. So, before anything else, I would like to acknowledge Zendesk's openness to my internship. After the internship, I can say with confidence I did the right choice. Everyone in the company received me with open arms and made this experience unforgettable.

Then, I would like to thank Professor Ademar for all his support during all these months. More than supervising my work, he was always there for my try-fail cycles. I'd also like to thank Dina Friis, for her preliminary insights of the process followed at Zendesk, Denmark.

A sincere thank you to all my team colleagues at Zendesk that collaborated in this study and listened to my proposals: Libo, Mauro, Christian, Ilkka, Jean-François, Alicia, Cristian, Luís and Mikkel. Also, to the other teams at Zendesk which also contributed to this work and the Scrum Masters at Top Danmark and Farfetch.

Finally, many thanks to Mariana, my parents and sister and my friends for all their motivational words.

Augusto

"Progress cannot be generated
when we are satisfied with existing situations"

Taiichi Ohno

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

TDD	Test Driven Development
XP	Extreme Programming
CMMI	Capability Maturity Model Integration
SPICE	Software Process Improvement and Capability Determination
SBI	Sprint Backlog Item
PBI	Product Backlog Item
QA	Quality Assurance
US	User Story
FDD	Feature Driven Development

Chapter 1

Introduction

As agile methods mature and its successful adoption becomes more and more widespread in the software industry, teams get more open and capable of responding to change. The adoption of agile methods cannot be seen as an end by itself, it must be seen as a mindset pushing teams to keep improving to be more predictable, to deliver faster and with more quality. To achieve this goal, agile methods include practices such as retrospectives that continuously challenge teams to find out how they can improve. However, many teams fail to detect important areas where to improve.

The purpose of this work was to focus on the activities of agile software process improvement, conduct a case study on this matter, and, as a result, build a simple toolkit that is able to help and guide agile teams in process improvement activities, named *Agile Forward - a simple toolkit for process improvement*.

The work focus on process improvement on a team level. It aimed to find simple but relevant methods to improve the software development methodology of an agile team.

In the current chapter, the context, motivation and goals of this dissertation are presented, followed by the work methodology followed, the expected impact as well as an overview of the structure of this document.

1.1 Context

This dissertation addresses the topics of agile methodologies and process improvement. These two fields together lead to process improvements activities in an agile development context, where the process improvement itself must be agile, too. Previous work on these areas tend to rely on external resources and complex methods. Thus, there is a gap in the field when it comes to simple, internal tools, which can be worked on.

This dissertation was conducted in partnership with Zendesk, a cloud-based software company. The practical work was conducted in the company's offices in Copenhagen, Denmark.

Figure 1.1: Zendesk's logo [Zen16].

Zendesk is a software company focusing on customer service platforms. Its main product is also called Zendesk. It is a cloud-based customer service platform. "It enables companies to provide great customer support, scale with self-service options, and differentiate with proactive engagement. The result is customer relationships that are more meaningful, personal, and productive — all at a lower cost." [Zen16]

The author worked at this company as a Software Engineering Intern, and while conducting dissertation related activities, he also joined a Scrum team that develops part of the *Help Center* product, as a web developer.

1.2 Motivation and Goals

The motivation behind this work comes from a preliminary feedback from the company where they came to realize that, although their Scrum teams are stable, they struggle to conduct process improvements, even while having regular Sprint Retrospectives.

With this motivation in mind, this work focuses on the activities of agile software process improvement, from analysis to concrete planning and implementation phase, with the goal of understanding the overall phenomena: key obstacles, common practices, and recommendations. In concrete, we define and propose a practical toolkit to help Scrum Masters, and other important players, to detect issues, by collecting relevant data, finding concrete solutions and putting these into practice.

This work includes a state of the art review on agile methodologies and process improvement, an analysis of the process of the team targeted in the case study, the proposal, enactment and evaluation of process improvement activities in the context of the targeted team, and, based on these, the proposal of a simple toolkit for agile process improvement, followed by validation of the findings.

1.3 Methodology

The research was conducted using *Action Research*. "Action research is an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning." [ALMN99] Using this research method, we were able to try improvement ideas with practitioners in real situations, draw conclusions from it and modify those ideas accordingly. Using action research means that in each iteration we will improve the theory we want to test.

1.4 Expected impact

The lack of software development process improvement may prevent the team to keep delivering faster and with more quality. This not only affects the end product but also the motivation of the team. New ideas can help the team build upon the current spirit and establish better working methods.

The differentiation of this research is centred in the fact that many teams lack resources to bring the change from inside, e.g. depend on external actors like Agile Coaches to conduct process improvement activities.

1.5 Document Structure

Besides this introduction, this dissertation includes six chapters:

- In chapters 2 and 3, the state of the art of related topics is described,
- In chapter 4, we propose a toolkit for agile process improvement,
- In chapter 5, we describe the case study activities that contributed to the toolkit presented, with a focus on Scrum Patterns,
- In chapter 6, the results of the validation of the toolkit with other teams are presented, and,
- In chapter 7, we draw conclusions from the work conducted and suggest points of future work.

Chapter 2

Agile Software Processes

In this chapter we describe the state of the art on agile processes and techniques, going more in depth in two of the most popular: Scrum and Extreme Programming.

2.1 Introduction

The Guide to the Software Engineering Body of Knowledge [Soc14] describes software processes as the following:

> "A set of interrelated activities and tasks that transform input work products into output work products."

This knowledge area includes software processes definition, software life cycles, software processes assessment and improvement, software measurement, and software engineering process tools. In this chapter we focus on a particular family of software processes: agile software processes.

Agile Software Development was formalised in 2001 when sixteen experts gathered to write the *Agile Manifesto* [BBvB+01]. It aimed to provide a large umbrella to the new vision that had started to rise among the software community. This vision is based on four values backed by corresponding principles [BBvB+01].

Individuals and interactions over processes and tools. Agile considers that the software industry is, above all, a people's industry. By this reason, building a good team will be of greater value than to have many processes that guide them through development. Communication should be done face-to-face as much as possible.

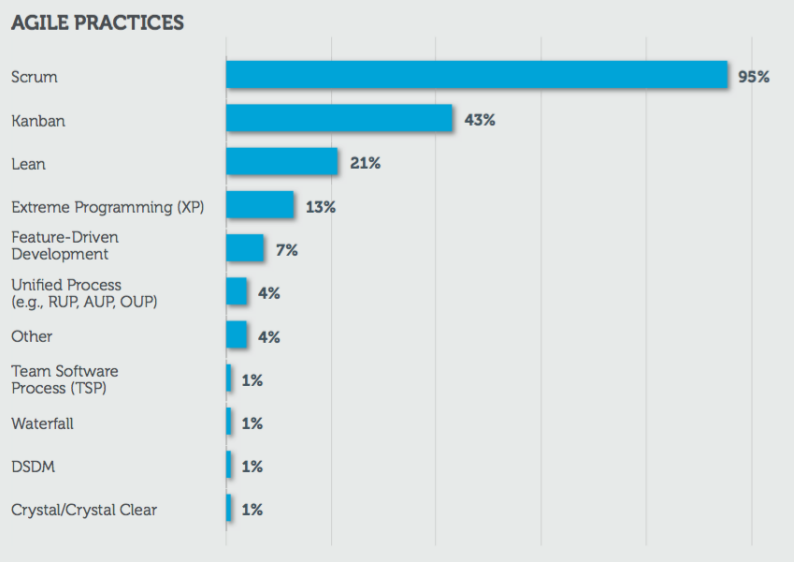

Figure 2.1: Agile practices organisations are using[1] [Sc15].

Working software over comprehensive documentation. Working software is the primary measure of progress and the main goal of a project should be the continuous delivery of valuable software. However, Agile does not consider documentation irrelevant. "It is always a good idea for the team to write and maintain a short rationale and structure document. But that document needs to be short and salient." [MM06] This means that the team should not focus too much on writing documentation but instead on writing quality code.

Customer collaboration over contract negotiation. Customer feedback is considered important since the customer will be the one evaluating the product at the end. Actually, the customer should work daily with the development team. Above all, the product should fit the needs of the customer.

Responding to change over following a plan. This is the value that gives this family of methodologies its name, i.e., *agile software development* must be *agile*. Because customer's needs change, requirements change, therefore developers need to be able to handle it properly. Delivering increments frequently will allow the customer to validate the path that is being followed. In fact, the ability to change will increase the product's competitive advantage.

2.2 Most popular agile methodologies

There are several agile methodologies currently available. The usage of the most popular practices can be seen in figure 2.1. In this section we do an overview of the five most popular ones.

[1] According to a Scrum Alliance study [Sc15]. Multiple answers were allowed.

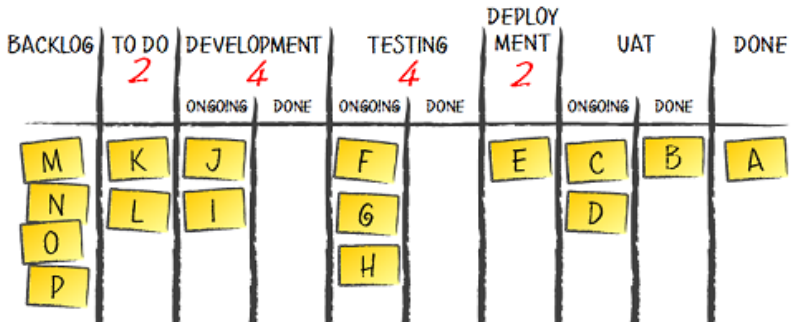

Figure 2.2: Example of a Kanban board [Bro11].

2.2.1 Scrum

Scrum is an iterative methodology that allows teams to adapt in order to deliver the highest possible value. Scrum consists of Scrum Teams and their associated roles, events, artefacts, and rules [SS13]. It does not enforce specific development practices. It focus on fixed time iterations called Sprints where the team plans, builds, delivers and analyses the work done. It aims to provide transparency, inspection and adaptation. Scrum is described in more detail in section 2.4.

2.2.2 Kanban

"Lean and Kanban approaches were introduced in the Japanese manufacturing industry in the 1950s. Kanban is a Japanese word meaning a signboard, and it is used in manufacturing as a scheduling system. It is a flow control mechanism for pull-driven Just-In-Time production, in which the upstream processing activities are triggered by the downstream process demand signals." [AMO13]

The Kanban board - an example is shown in figure 2.2 - provides visibility as it shows assigned work, communicates priorities and highlights bottlenecks. The main goal is to reduce Work in Progress, by limiting the number of items the team can be working on at the same time. "This produces constant flow of released work items to the customers, as the developers focus only on those few items at given time." [AMO13]

Kanban principles are the following:

- Visualise the workflow;
- Limit Work In Progress;
- Measure and manage flow;
- Make process policies explicit;
- Improve collaboratively.

2.2.3 Lean software development

Lean software development is the application of the principles used in the Toyota Production System and other manufacturing companies to software development. Lean Development was first used with the intent of keeping market and customer needs as the primary decision driver. The result of that is the production of vehicles - in the Toyota case, on a very short period of time and always on time. [PC12]

Lean Software Development takes into consideration seven principles:

1. Eliminate waste. In development environment this means that there is no point in building features that do not add value to the product. The most important thing is to deliver what the customer wants and when the customer wants.

2. Amplify learning. Development is seen as an iterative process that can only evolve if there is space for learning.

3. Decide as late as possible. Requirements change frequently so it is important to keep being flexible for as long as possible. "Delaying decisions is valuable because better decisions can be made when they are based on fact, not speculation." [PP03]

4. Deliver as fast as possible. In order to have feedback from the work done, it is important to not delay deliveries. This way, the cycle *design, implement, feedback, improve* will occur more often. "Compressing the value stream as much as possible is a fundamental lean strategy for eliminating waste" [PP03].

5. Empower the team. The development team is the one doing the work so they are the ones taking the technical decisions, these are not coming from above in the hierarchy.

6. Build integrity in. The system built should be coherent, have great usability, and be maintainable, adaptable, extensible. Only this way it is able to evolve in a way the customer needs are met.

7. See the whole. The team should see their contribution to greater picture, and strive for the performance of the overall system and organisation, in contrast to only caring about their specific work.

2.2.4 Extreme Programming

"XP is a style of software development focusing on excellent application of programming techniques, clear communication, and teamwork which allows us to accomplish things we previously could not even imagine." [BA04]

Specifically, Extreme Programming is an iterative methodology that provides a set of well defined practices with the goal of delivering the right product, in time and with quality. It is further presented in section 2.3.

2.2.5 Feature Driven Development

Feature-Driven Development combines the key advantages of agile methodologies with model-driven techniques that scale to the large projects. As the name hints, the processes focus on the features to be delivered. It is an iterative methodology with the following activities [DL16]:

1. Develop an Overall Model
2. Build a Features List
3. Plan By Feature
4. Design By Feature
5. Build By Feature

Each of these activities include a list of tasks that need to be completed to move to the next step. These tasks include many software development best practices such as domain modelling, sequence diagrams, code inspection and unit tests.

In the following two sections we will focus on two of the most popular agile practices: Extreme Programming and Scrum. We opted to do a more in-depth analysis of these two since Scrum is by far the most used one and Extreme Programming offers very practical methods that can complement other methodologies.

2.3 Analysis of Extreme Programming

Extreme Programming (XP) was one of the first agile methodologies to be widely popular. It dates back to 1996, the year the first project using XP was started [Wel13], which is long before the Agile Manifesto was written (2001) [MM06]. It derives from Kent Beck's experience on software development and aims to put together the best practices. In fact, Beck says: "To some folks, XP seems like just good common sense. So why the *extreme* in the name? XP takes common sense principles and practices to extreme levels." [BA04]. The belief is that, while these practices can have problems when standing alone, when together they will support each other.

2.3.1 The problems and variables XP targets

The development of software, like other creation activities, have economic risks that need to be reduced. Those include:

Schedule. There is a risk that the time that it takes to develop software is longer than expected.

Purpose. It may happen that the intended purpose for the software is no longer valid and the project needs to be canceled or completely redefined. As the project evolves the cost of change tends to be higher.

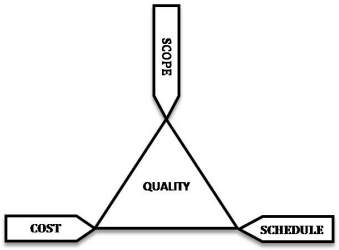

Figure 2.3: The project management triangle [Wik16].

Maintainability. As the software gets more and more features it can become harder to keep it in a organised way, eventually becoming a *big ball of mud* [Mar09].

High defect rate. Without a clear focus on testing, software will become full of *bugs* and any change in legacy code will be reduced to a maximum, because it will possibly break the whole system.

These risks translate in four variables[2] we want to control and predict. These variables are usually considered in project management and represented in a triangle like figure 2.3 shows.

Cost. The amount of investment needed. When starting a project, one should not have all the money available. The project should grow gradually.

Time. There can be constraints like a predefined release date. More time means there can be more quality and a larger scope.

Quality. The standard the customer is expecting. It is possible to reduce quality for some time, while maintaining the system working, but this will become unbearable.

Scope. What the customer wants to include in the software. Less scope means the product is delivered faster and is less expensive, while maintaining the same quality. Scope should be the focus of stakeholders: when a project starts it is difficult to say what is really expected it to become.

When either *cost* and/or *time* are limited we must be aware that *quality* or *scope* will be affected.

XP, as presented in the following subsections, proposes practices that mitigate these risks to a minimum by identifying problems shortly after they arise or by avoiding them completely.

[2]Presented in the first edition of [BA04]

2.3.2 The fundamentals behind XP

XP embraces five *values* to guide development: communication, simplicity, feedback, courage, and respect. From this more abstract values, XP has some fundamental principles that support all its practices. Those are:

Humanity & Diversity & Mutual Benefit & Accepted Responsibility. XP aims at developers, who are above all humans and need a balance between personal and professional lives. Teams are expected to have all kinds of knowledge needed inside of it, through diversity of workers. In XP, people work towards a common goal. "The computer business is really a people business and maintaining working relationships is important" [BA04]. Each team member should say which tasks he wants to do, thus accepting his responsibility in the project.

Failure & Reflection & Opportunity & Improvement & Baby Steps. Failed approaches to a problem are a good thing. With failure, developers will learn how to do it correctly. Teams should think about what they do in order to improve. Each problem should be used as an opportunity to improve. XP aims for continuous improvement, and with its methods tries to find areas where the team can do better. Baby Steps are therefore a metaphor for these practices that take place over the course of time.

Flow & Self-Similarity & Redundancy & Quality. There should be a continuous flow of activities in contrast with large blocks of different activities, e.g. the team should do continuous integration. Moreover, patterns should be valued since these help keeping an organised structure and comprehensive code. Redundancy helps to keep the defect rate very low, e.g. by the use of pair programming. All to achieve quality. Developers should be aware that lowering quality standards often results in later, less predictable delivery.

Economics. In the long run, XP will benefit the company economically by providing better software.

2.3.3 Primary practices

As mentioned above, XP is a set of good practices that gain a new meaning when applied together. To effectively put XP into practice a team should implement *all* of these (as showed in figure 2.4):

1. **Sit Together.** The best way for a team to work, is to be in the same space. This shared space will allow the team members to clarify any doubt they have with any other team member. This will fasten their work;

2. **Whole Team.** Teams should be cross-functional and all team members should focus on only one project at a time;

3. **Informative Workspace.** The workspace should remind the team what they are there to do. The use of a planing board with user stories is one important artefact to have there;

4. **Energised Work.** This practice used to be named *40-hour week*. It translated in having balanced work days. Although over-hours can be done once in a while, there should not be a "Today we will have to stay later *again*" discourse.

5. **Pair Programming.** Pair programming consists in having two individuals sitting side-by-side with only one computer where they code. One individual is the driver, he writes the code. The other is watching and helping the driver. Together they design a complete system. Pairs should not be fixed. A developer should pair, at least, with two other developers per day, allowing the team to interact as a whole;

6. **Stories.** User Stories should have a title and a brief description. They should be estimated early in the development cycle. This way the customer will be more informed when making decisions;

7. **Weekly Cycle.** The start of each week should include: a review of the progress to date, having the customers pick US for that week and having the developers breaking these into tasks, estimating them and taking responsibility over them. After this, acceptance tests should be defined. The purpose of having a one-week cycle is because everyone is focused on Friday.

8. **Quarterly Cycle.** Every three months, a team should stop to review their work and focus on the big picture. Like weeks, seasons can be a natural starting point for each cycle;

9. **Slack.** Every once in a while, there should be time for other activities, usually 20% of the time (e.g. weeks). This will give the project a margin to get back on track if needed, too;

10. **Ten-Minute Build.** Builds should be optimised so they take ten minutes maximum. Long builds mean that programmers will not conduct testing as often as they should;

11. **Continuous Integration.** The code developed by a pair should be integrated every two hours. This will allow problems to arise sooner. It can be done in a synchronised or unsynchronised way. K. Beck recommends to keep it synchronised in order to not lose focus if it fails;

12. **Incremental Design.** Requirements change. Although doing the whole design upfront allows the creation of structured software, the cost of putting it together is high, considering it can change anytime. XP says that teams should design with *Baby Steps.*, i.e., adding new layers as they are needed without loosing sight of the big picture. As mentioned above, because the code is fully tested, coders can efficiently refactor their design without fearing that they will be breaking other part of the system;

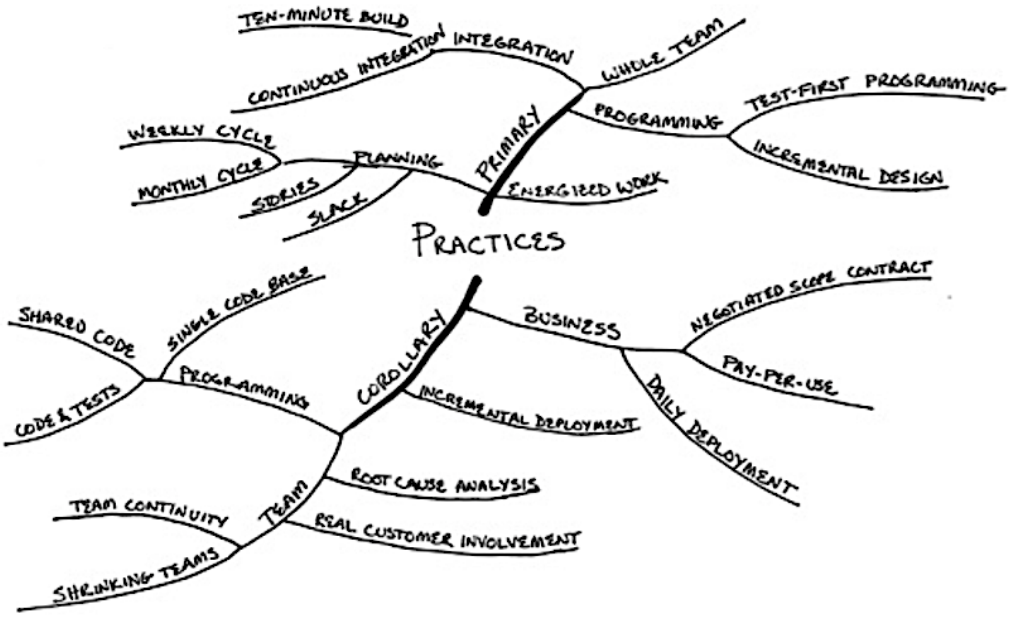

Figure 2.4: An overview of XP practices [BA04].

13. **Test-First Programming.** Also known as TDD, it is considered one of the most important XP practices [Mar09]. Before implementing a task, all tests for it should be written. This will force developers to think before coding and consider different designs that could work. More than that, it will provide a guarantee that tests are not written to fit the implementation. Having a 100% coverage means that one can safely change code being sure it will not break another feature [Mar09].

These practices are complemented with some more advanced ones, called *Corollaries*, that can also be seen in figure 2.4.

2.3.4 Difficulties using XP

The main problem using XP comes from the difficulty to have all practices in place, since it requires an extra effort from developers. Without all the practices in place, XP will not be complete and will not deliver what it promises to. Another difficulty is to provide the customer a real estimate of the cost of the project since the requirements are allowed to change over time. There is also a concern regarding the fact that no documentation is written, because well-written code is considered to be enough. [RS03]

2.4 Analysis of Scrum

Scrum is an agile methodology to manage software projects that was first publicly presented by Ken Schwaber and Jeff Sutherland in 1995 [SS13]. In contrast with other methodologies, Scrum does not cover specific working practices, but rather the way planning is done. It presents a

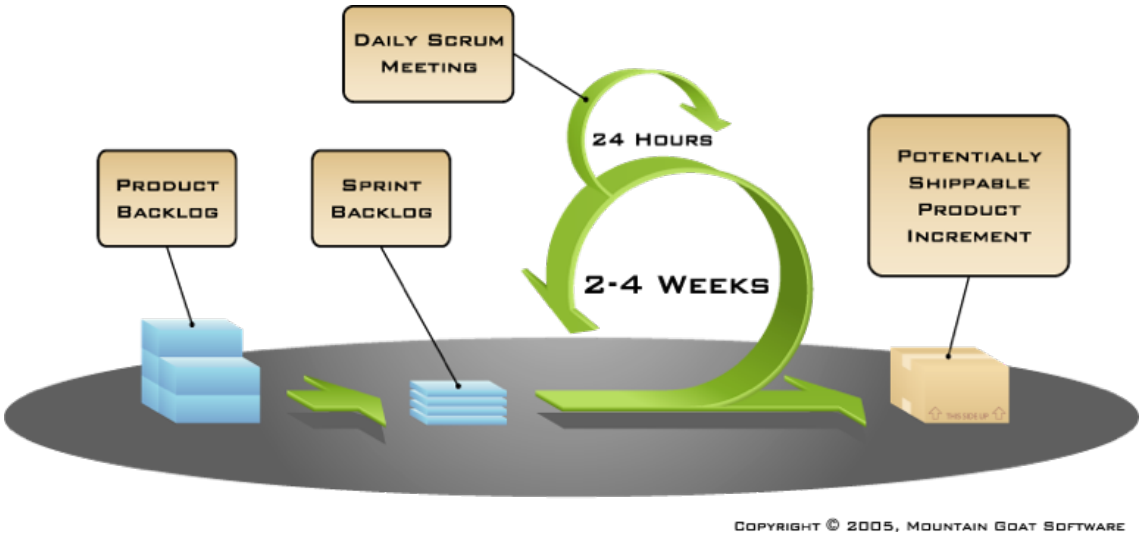

Figure 2.5: An overview of the Scrum methodology [Coh16].

way to manage a software project in an iterative and simplified way. It aims to give teams more productivity while improving *Transparency* between the software team and the client, allowing *Inspection* of the current state of the project and facilitating *Adaptation* if the project goals change. Scrum has made its way to the majority of the software companies worldwide, as showed in figure 2.1.

2.4.1 The Scrum Team

The major players in a Scrum team are [SS13]:

Product Owner. He is the one responsible for ensuring the value to the business and taking business decisions. He manages the features to be included in the product;

Scrum Master. He helps the Development Team and the Product Owner to understand Scrum and improve the way it is implemented. This includes helping the Product Owner to manage the product features and the development team by coaching its members in self-organisation and cross-functionality;

Development Team. As said above, it is a self-organised and cross-functional team. All members are considered to be *developers* and nothing else.

2.4.2 The Scrum process

Scrum defines a process, as shown in figure 2.5, which starts with the creation of the Product Backlog, followed by a Sprint Backlog, leading to a time framed coding activity - a Sprint. Sprint related activities are repeated until there are no more items in the Product Backlog. In the following sections each of the activities included in the process are presented.

Building the Product Backlog. The Product Backlog is where requirements live. These are translated to a collection of items that are ordered considering their priority to the business. Next, these are estimated in relative size to other items, using points. As the product changes, the Product Backlog is updated with new requirements, features, fixes, etc. Highly ranked items have more complete descriptions than lower ranked ones.

Sprint Planning. Before the beginning of each sprint, the Product Owner discusses the Sprint Goal. Then, the team selects the higher ranked items from the Product Backlog they expect to mark Done by the end of the sprint, and thus will be in the next product increment. These go into the Sprint Backlog. The items should be decomposed into working items, which can be marked Done in one day or less.

Daily Scrum. Everyday the team meets for 15 minutes. At this meeting, each team member shares what he did yesterday and what he plans to do today. The team also discusses if there are any major problems that will block the next tasks.

Sprint Review and Retrospective. Although both Sprint Review and Retrospective are held at the end of the Sprint, they serve different purposes. The Sprint Review focus on the work produced during the sprint. It includes presentation of the items marked as Done, discussion of problems during development and analysis of the current backlog and what to do next. It should be conducted in an informal environment, which elicits feedback and fosters collaboration. A Sprint Retrospective, on the other hand, focus on finding ways to improve the development environment of the team including relationships, processes and tools. It is here that teams should discuss the quality of the work and plan ways to improve it.

Releases. By the end of each Sprint, there should be a Product Increment that can be released. The decision to release this new version of the product to the market belongs to Product Owner. The Scrum methodology does not define how many times the product should be released to the market.

2.4.3 Limitations of Scrum

Scrum, introduced as a light weight methodology, can, not only be applied to software projects, but also to other areas of expertise. Compared to XP, it does not set programming practices. This broadness may, sometimes, raise some questions on the right thing to do when a problem arises. Moreover, from a process improvement point of view, there is a moment, the Sprint Retrospective, for the team to think about how it can do better, but no concrete techniques are proposed. In the next section, we review *Scrum Patterns*, which collect and organize best practices of Scrum.

2.5 Scrum Patterns

Following the success of Scrum, it became relevant to identify commonly occurring problems and solutions in the context of Scrum activities, most of them not part of the fundamentals of Scrum mentioned in section 2.4. This approach was first introduced by Beedle et al. in 1998 [BDS98]. Currently, there's a working group on this matter, the *Scrum Pattern Community* [Scr16b, HR14]. The patterns are organised into some main categories: Value Stream, Process Improvement, Product Organisation and Scrum Core.

2.5.1 Value stream

The intent of these patterns is to describe recurrent problems and proven solutions related to activities which lead to product increments. One of its focus is the backlogs including how the predicted velocity affects the team, the visibility of these artefacts and how tasks should be assigned (for example, in order to reduce items dependency). Another focus is the pace the team moves and how it can smooth the flow.

2.5.2 Process improvement

The authors also introduce some process improvement patterns with the purpose of reaching "Hyper-Productivity, more than a 400% increase in velocity over a team's initial velocity" [HR14]. One of these is **Scrumming the Scrum**, a pattern that recommends the use of Scrum itself to improve the team's process by, among others, adding solutions to recognised impediments as backlog items of the sprint following the recognition of these impediments.

2.5.3 Product organisation

Product Organisation patterns cover the roles in Scrum - Product Owner, Scrum Master and Development Team - and how these people can effectively contribute to the success of the product. The proposed solutions aim at improving the communication between the Product Owner and the Development Team and the empowerment of this team with the support of the Scrum Master.

2.5.4 Scrum Core

The Scrum concepts introduced in section 2.4 are also covered as patterns following the common problem-solution perspective.

2.6 Summing up

In this chapter, we started by looking at the main concerns of agile methodologies and then dived into XP and Scrum. We value XP techniques and believe they have the potential to improve a team's process. Scrum is the most used methodology and the one that will be used in the case study. Scrum Patterns refine Scrum and give hints on problem solving.

Chapter 3

Process Improvement

In this chapter, we review relevant software processes assessment and improvement techniques.

3.1 Introduction

Software process improvement activities aim to change organisational activities and culture in order to waste less resources and be more predictable.

"Improvement activities include identifying and prioritising desired improvements (planning); introducing an improvement, including change management and training (doing); evaluating the improvement compared to previous or exemplary process results and costs (checking); and making further modifications (acting)" [Soc14].

In order to improve the process, teams must first assess their current software process. This can be done with capability evaluations - evaluations performed by an external agent - or performance appraisals conducted within an organisation to identify areas of improvement [Soc14].

"A typical method of software process assessment includes planning, fact finding (by collecting evidence through questionnaires, interviews, and observation of work practices), collection and validation of process data, and analysis and reporting" [Soc14].

In section 3.2.1 we explore Capability Maturity Model (CMMI), a broadly used model in the industry. Then, in section 3.2.2, we look into measuring agile teams performance and then how maturity can be assessed based on this (section 3.2.3).

Process improvement should be part of the processes of the organisation. We already reviewed how Scrum introduces this topic with Scrum Retrospectives. CMMI also focus on process improvement. In this chapter we will also analyse two approaches to agile process improvement (section 3.3).

3.2 Process evaluation

As mentioned above, a fundamental part of process improvement is the evaluation of the current status of the process. In this section, we review some approaches to this activity.

3.2.1 Capability Maturity Model Integration

"Capability Maturity Model Integration models are collections of best practices that help organisations to improve their processes. These models are developed by product teams with members from industry, government, and the Software Engineering Institute." [CKS11] CMMI has two types of levels: capability levels and maturity levels. Capability is associated with a continuous representation and it shows the achievements in individual process areas. Maturity is associated with a staged representation across all areas. For a maturity level to be awarded, the organisation needs to achieve a minimum set of goals in every and all process areas.

3.2.1.1 Capability levels

Capability in a certain area can be (definitions from CMMI-DEV guide [CKS11]):

Incomplete. An incomplete process is a process that either is not performed or is partially performed.

Performed. A performed process is a process that accomplishes the needed work to produce work products; the specific goals of the process area are satisfied.

Managed. A managed process is a performed process that is planned and executed in accordance with policy; employs skilled people having adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description.

Defined. A defined process is a managed process that is tailored from the organisation's set of standard processes according to the organisation's tailoring guidelines; has a maintained process description; and contributes process related experiences to the organisational process assets.

3.2.1.2 Maturity levels

Maturity of the whole process includes five stages (definitions from CMMI-DEV guide [CKS11]).

Initial. At maturity level 1, processes are usually ad hoc and chaotic. The organisation usually does not provide a stable environment to support processes. Success in these organisations depends on the competence and heroics of the people in the organisation and not on the use of proven processes. In spite of this chaos, maturity level 1 organisations often produce products and services that work, but they frequently exceed the budget and schedule documented in their plans.

Managed. At maturity level 2, the projects have ensured that processes are planned and executed in accordance with policy; the projects employ skilled people who have adequate resources to produce controlled outputs; involve relevant stakeholders; are monitored, controlled, and reviewed; and are evaluated for adherence to their process descriptions. The process discipline reflected by maturity level 2 helps to ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans.

Defined. At maturity level 3, processes are well characterised and understood, and are described in standards, procedures, tools, and methods. The organisation's set of standard processes, which is the basis for maturity level 3, is established and improved over time. These standard processes are used to establish consistency across the organisation. Projects establish their defined processes by tailoring the organisation's set of standard processes according to tailoring guidelines.

Quantitatively Managed. At maturity level 4, the organisation and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects. Quantitative objectives are based on the needs of the customer, end users, organisation, and process implementers. Quality and process performance is understood in statistical terms and is managed throughout the life of projects.

Optimizing. At maturity level 5, an organisation continually improves its processes based on a quantitative understanding of its business objectives and performance needs. The organisation uses a quantitative approach to understand the variation inherent in the process and the causes of process outcomes.

3.2.1.3 SPICE

The Software Process Improvement and Capability Determination (SPICE) was introduced by the International Organisation for Standardisation and is based on CMMI. The work of this workgroup aims to be the reference model for the maturity models [ISO13].

3.2.2 Agile Development Metrics

As part of our focus on agile methodologies, we looked at different ways to measure process improvement in agile teams. In order to evaluate how a team is performing while practicing XP, Scrum, or other agile methodology, there is a need to define metrics. These metrics will also allow to measure process improvement. According to Hartmann and Dymond [HD06], there should be first a distinguish between the *organizational key metric* and other supporting metrics, which the authors call *diagnostics*. While the first translates the business value of the agile development process, the others help the team improve the processes. To them, a good agile metric should: affirm and reinforce Lean and Agile principles; measure outcome, not output; follow trends, not numbers;

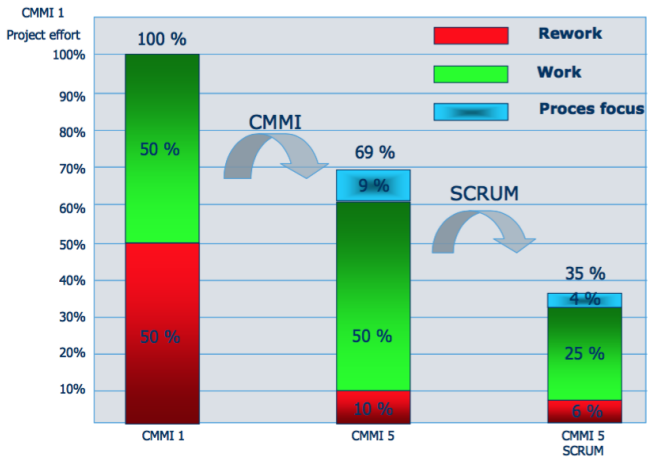

Figure 3.1: Example of project effort evolution when combining CMMI and Scrum [JJ07].

answer a particular question for a real person; belong to a small set of metrics and diagnostics; be easy to collect; reveal, rather than conceal, its context and significant variables; provide fuel for meaningful conversation; provide feedback on a frequent and regular basis; measure value (product) or process; encourage "good-enough" quality. The authors also provide a relevant evaluation checklist to analyze how useful a metric can be for the team.

3.2.3 Agile Practice Maturity

One diagnostic measurement Hartmann and Dymond [HD06] suggest is the *Agile practice maturity*. The maturity of Agile processes can be accessed in different ways from simple checklists and assessment forms, e.g. the ones from Scrum.org [Scr16a], to models that adapt the Capability Maturity Model (CMMI) evaluation to Agile.

One example is the *Scrum Maturity Model* [YdLFdS11, Yin11] that establishes five levels of maturity to Scrum practices, from an unmanaged to an optimised process, introducing concrete evaluation measurements with checklists to correctly place a team in one of the five levels. A similar, more broad work is *Agile Maturity Model (AMM)* [CM09] which also incorporates five maturity levels with specific goals for each of these.

Other authors, like Jeff Sutherland et al. [JJ07, Jak09], explored the combination of CMMI and Scrum in order to balance agility with discipline. The authors analysed the combination of CMMI and Scrum and found out that "Scrum and CMMI together bring a more powerful combination of adaptability and predictability to the marketplace than either one alone." These findings can be seen in figure 3.1, where the project effort has clearly been reduced. This balance between Agility and Discipline was first introduced by Boehm and Turner [BT04] "with a risk-based method for developing balanced strategies that take advantage of the strengths and mitigate the weaknesses of both agile and plan-driven approaches" [BT04].

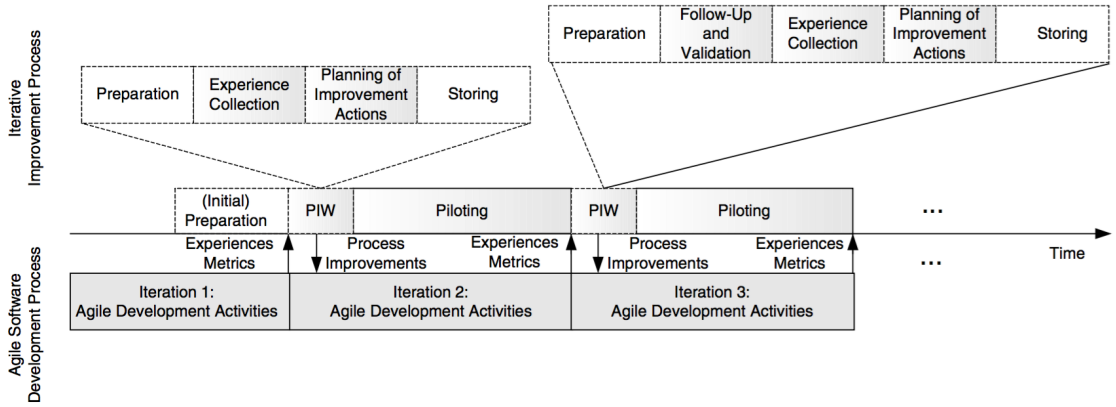

Figure 3.2: The process proposed by O. Salo et al. [SA07].

3.3 Agile process improvement approaches

There are different ways to approach and conduct process improvement. In this section we analyse two different proposals.

O. Salo and P. Abrahamsson propose "An Iterative Improvement Process for Agile Software Development" [SA07], which focus on post-iteration workshops as a way to conduct process improvements. Their approach includes five steps for this workshop:

1. Preparation, where, among other steps, the tools to be used during the workshop are selected;
2. Follow-Up and Validation, where the team discusses what was achieved based on the last workshop;
3. Experience collection, where during the workshop the participants share their negative and positive experiences;
4. Planing of Improvement Actions, "e.g. what the problem is exactly, what the concrete actions are that need to be taken to improve the situation, who is responsible for carrying out the improvement actions and when." [SA07];
5. Storing, where the team registers what it decided on.

An overview of this process can be seen in figure 3.2. This work originated from a case study that was conducted with five Extreme Programming teams.

Abdel-Hamid and Abdel-Kader introduce a different approach named "Process Increments" [AHAK11]. This approach conducts process improvement by applying an agile methodology usually used to manage projects to manage the improvement. The Scrum Pattern *Scrumming the Scrum* follows along the same lines [Scr16b]. The authors recommend using user stories with the following points: summary title, verification points (i.e. conditions of satisfaction), size estimate, process area (management, product development or environment & infrastructure). Then,

iteratively work on them and validate if they met the conditions of satisfaction. In this approach, process improvement is managed by an external group who puts this process in place.

3.4 Summing Up

In this chapter, we reviewed different techniques to evaluate status, and thus progress, and to put process improvement activities in place. We analysed both traditional approaches and agile related ones.

The traditional approaches reviewed demand many metrics to be monitored and, for instance, CMMI places organisations on levels with inflexible criteria. Although, we saw an example of combination of CMMI and Scrum, in a more broad view, these practices many not be compatible with environments that foster agility and, more than that, move in a very fast pace with innovation as a constant.

On the other hand, the agile techniques studied for process improvement focus mostly on retrospectives. As studied in chapter 2, this is indeed an important practice in agile environments. However, agile teams may fail to detect their own problems by not being able to see the bigger picture. Thus, introducing objective metrics, in a manageable amount, can reveal, rather than conceal, less obvious problems.

Considering all this, by this review, we can conclude there is an opportunity to identify an unified and simple approach for agile software improvement than can be conducted by an agile team and that applies different known techniques, which together can deliver better results than when in isolation.

Chapter 4

A simple toolkit for process improvement

During the research conducted, which includes a case study described in chapter 5, we were able to recognise important ways and tools that lead teams to a better development process. In this chapter, we present a proposal of an agile toolkit for process improvement.

4.1 Introduction

We define an agile toolkit for process improvements as a set of procedures and tools that help teams become better in the way they develop new software. It includes diagnostic tools, problem solving techniques and change inducting methods.

In this toolkit, we use the term *Process Master* as a reference to the person that is responsible for ensuring the process is understood and enacted. Specifically, this person is the one responsible for promoting process improvement techniques. In lack of more general names, some Scrum terms referring to concepts also present in other methodologies will be used sporadically.

4.2 Target audience

The main target of this toolkit are agile teams and respective Process Masters that are concerned with process improvement and are looking for a simple, straightforward approach to these concerns. The target team is not closely supported by an Agile Coach and thus is in charge of detecting and solving process problems. The team's Process Master may also be part of the development team while only focusing part of his/her time in this role.

4.3 Approach

Process improvement should follow a process itself. For this toolkit, we define the following steps which we will go into more details in the next sections. Figure 4.1 shows the steps proposed.

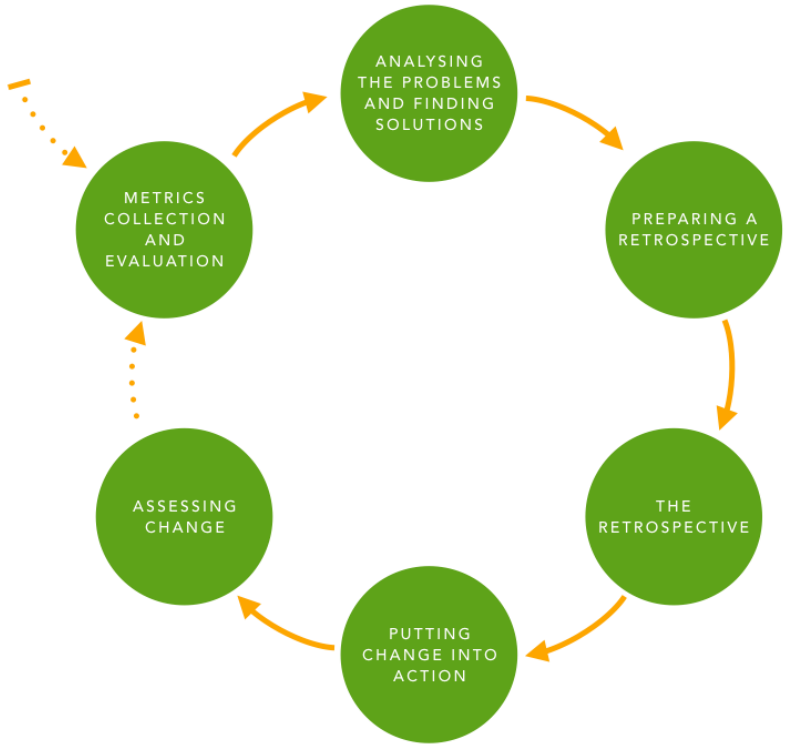

Figure 4.1: The steps of Agile Forward - a simple toolkit for process improvement.

Metrics collection and evaluation. As a Process Master, one must collect relevant metrics during the iterations. These can be of various forms, from concrete numbers to notes of relevant episodes that affected the process.

Analysing the problems and finding solutions. A development team will usually run into similar issues others have had in the past. So, it is useful to match the issues the Process Master was able to identify, to known problems and their solutions.

Preparing a retrospective. Although all problems must be discussed with the team, some may be done during daily meetings while others require a specific moment: retrospectives. Based on the data collected during the iteration, the Process Master needs to select which issues should be discussed during the retrospective, taking into consideration input from the team, too. This step includes setting a strategy on how to introduce each one of the topics in a way that the team recognises the opportunities for improvement.

The retrospective. During the retrospective, the team will reflect on the issues presented by the Process Master as well as the solution he/she may introduce for some of the problems. It is important that the team commits to the solutions they agree on.

Putting change into action. The team may struggle to put into practice the actions they commit to. It belongs to the Process Master the responsibility to make these commitments see the light of the day.

Assessing change. It may seem easy to see if the change introduced is allowing the team to work better, but one may miss out some important issues. In order to better understand the effects, we recommend also conducting measurements.

In the following sections, we further explain each of these steps and introduce the tools and methods to achieve them.

4.4 Metrics collection and evaluation

One of the roles of a Process Master is to, while working with the team, see above the team in order to understand how it can improve [Scr16b, Pattern: Scrum Master]. This implies collecting diverse metrics that he/she can later analyse and introduce during retrospectives. On the other hand, metrics and evaluation will also allow to measure improvement over time, thus having both a *diagnostic* and a *tracking* purpose. These tools should always be viewed as internal to the team and not as reports to the organisation.

We recommend the metrics presented in the following subsections.

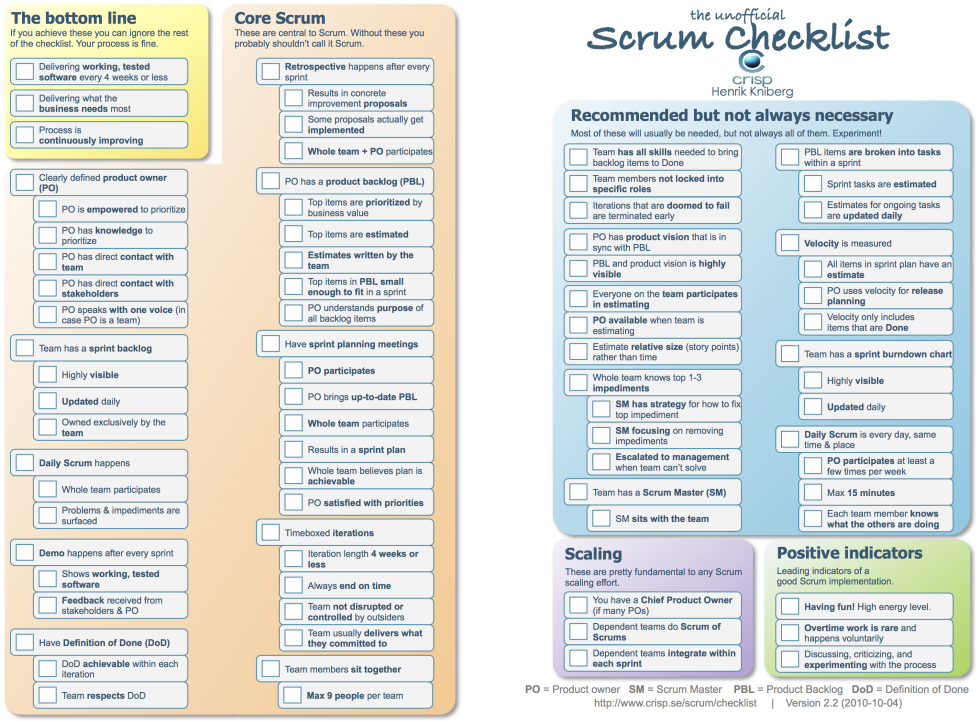

Figure 4.2: Crisp's scrum checklist [Cri16].

4.4.1 The notebook

The first step into collecting relevant metrics is to understand the team and its daily dynamics. During the iteration, the Process Master will usually detect small issues that together contribute to a better understanding of the improvements that are needed. This can happen during planned meetings or during development work. In order to later remember these, we recommend carrying a notebook around to take notes. Organising it by the different kind of problems, will allow a better overview of recurring issues. This technique may seem basic but will reveal very useful when used with other metrics that take into consideration what happens during an iteration.

4.4.2 Process checklist

To understand to what extent the team is putting the methodology into practice, the easiest way is to go through a list where items can be marked as done. This can be done alone or with the team, during a retrospective, for example. One of the most recognised checklists for Scrum is from the consulting firm CRISP [Cri16]. This checklist is presented in figure 4.2. This checklist is important to both identify practices the team has not yet implemented and practices the team has abandoned. In the latter case, the next step is to understand what lead to the abandonment.

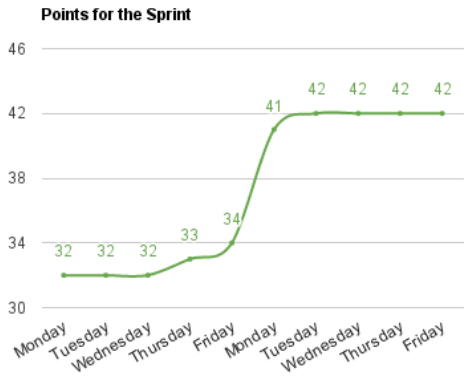

Figure 4.3: Example of scope changes tracking graph.

4.4.3 Performance analysis

Velocity and estimation issues can be identified by analysing the team's commitment versus what it achieves. Moreover, it is also important to analyse how the results are distributed in time. To this end, we recommend the analysis of three important charts.

Iteration burndown. This well-known chart allows Process Masters to diagnose problems that occur during the iteration. As noted in the Scrum Pattern *Track Done* [Scr16b], the burndown chart should only be updated when the item is marked as done, thus preventing false hopes. When analysing it after the end of the iteration, one should see how much work is only concluded in the days preceding the end of the iteration. If this is higher than the rest, it is an indicator that the team has found impediments along the way and has pushed the delivery of the items which can have negative effects in quality. See figure 5.2 for an example of a problematic burndown chart.

Changes in iteration scope. This chart should show the variation of the sum of points of the items in the iteration, along the iteration. An example is shown in figure 4.3. Here, it is key to understand what caused the variation. There are often three main causes: a Product Owner with emergent requirements; too little work for the team to complete; the work is already expected to be done by a specific member. In the first case, there might be a lack of commitment from the Product Owner to thoughtfully order the items in the Product Backlog. The Product Owner should be invited to the next retrospective to find a solution to this problem. In the second case, we identified a planning problem. The team should consider increasing its velocity if this reveals a recurrent issue (it is recommended to consider three iterations [Scr16b, Updated Velocity]). In the third case, there is an overspecialisation problem. The issue should be brought to a retrospective. One possible solution is to conduct *Swarming* [Scr16b], i.e. focus the effort on one item of the iteration backlog until it is done. This will also help team members become more broad skilled.

Items not completed. Another performance problem teams may face is not being able to finish all items before the end of the iteration. This should be tracked by analysing how many times

it happened during the past iterations. Figure 5.5 shows an example of an *Expected vs. real velocity chart* that allows to track how much a team has overcommitted (or undercommited). For each iteration the team should discuss why specific items were not completed and if there were impediments that may have prevented it.

4.4.4 Collecting individual feedback

People are the key for process improvement. Thus, understanding their problems is a simple but effective step towards better performance. By scheduling regular meetings with team members, the Process Master will better grasp the concerns of his/her peers. However, this should be used as a tool to identify the problems. The Process Master role is to encourage the team member to discuss these issues with the team by taking them to the next retrospective. After agreeing on this, during the retrospective the Process Master may introduce the topic and ask the team member to express his/her concerns.

4.4.5 Scrum Patterns

As presented in section 2.5, the Scrum Patterns are a collection of recurrent problems teams face and proven solutions. This way, Scrum Patterns offer an easy way to identify problems a team may be facing. Since the patterns have a clear stated problem, going through them is a task the Process Master can do frequently to surface problems of the team. After identifying potential applicable patterns, the Process Master can go into more detail by reading the first part of the pattern and see if its scope is appropriate. In the website of the project is possible to find a list of the current available patterns and its problem statement.

4.5 Analysing the problems and finding solutions

After identifying the issues the team is facing, the Process Master needs to analyse each of them in order to better understand if it is suitable for discussion in a retrospective or if it should be escalated to management instead, in the case the team's process is restricted by upper level decisions.

The Process Master should also try to understand the impact the problem has in the team's process. In order to identify solutions, the main method we recommend is to look into the Scrum Patterns, specially when the problem was detected with one.

In any case, it is very important to find a well based justification for introducing a change. The team will want to know why they have to change. It can be data-based, e.g. the amount of bugs reported or the iteration burndown chart, or based on an explanation of why doing things differently will improve the process.

We also recommend checking with other teams having similar same background (for example, teams that work on the same office, the same product, etc.) to understand if they faced that problem before and which solution they implemented.

While the Process Master needs to prepare beforehand on all of this, the decision needs to be taken collectively during the retrospective, guided by the Process Master.

4.6 Preparing a retrospective

The first step to prepare a retrospective is to define a purpose for it [Ker13]. The Process Master should go through the identified problems - the ones detected using metrics as the exposed in section 4.4 - and pick the one that seems the most relevant, the one that is blocking the team's performance the most.

According to N. Kerth [Ker13], it is important to sell the retrospective and how it brings value to the teams's work. This can be done during a daily team meeting some days before the retrospective. The Process Master will remind the team of the retrospective and suggest a topic. Upon agreement from the team, the Process Master asks the team to take a moment prior to the retrospective to think about the subject.

The Process Master needs to structure the retrospective and time-box each part of it. The Scrum guide [SS13] recommends a "three-hour time-boxed meeting for one-month Sprints", proportionally adapted to the iteration duration.

Three main moments of a retrospective should be considered when preparing it: *Introduction*, *Analysis* and *Future* [1]. There are many activities possible for each of these moments. For this toolkit we present one example for each:

Introduction. The introduction moment should include the welcoming of the participants and a review of the goal and agenda. A possible exercise is to ask one brief question to be answered by each team member. Considering that the team knows the purpose of the retrospective beforehand the question can be "In a word or two, what are your hopes for the retrospective?".

Analysis. The main portion of the retrospective, where the Process Master will introduce data that reveals the process problem to be addressed and give a brief explanation on why it represents a problem to the team. This is followed by a discussion. One possible activity is to split in pairs and brainstorm on ideas to solve the problem. Each pair can be given sticky notes where they write their ideas. After this moment, each pair presents their findings to the team.

Future. In the last part of the retrospective, the team must decide what to do, followed by a moment to close the retrospective. One way to choose the actions is by asking the team to vote on the value of each proposal using Planing Poker [Mou] points. This exercise is then followed by writing a concrete Definition of Done for the action and adding it to the next iteration's backlog [2].

[1] N. Kerth [Ker13] names the steps "Readying, Past and Future" while E. Derby and D. Larsen [DL06] mention five different kinds of activities that can be grouped in the three mentioned parts: "Activities to set the stage", "Activities to gather data" and "Activities to generate insights", "Activities to decide what to do" and "Activities to close the retrospective".

[2] This proposal is inspired by the pattern *Scrumming the Scrum* [Scr16b].

To end the retrospective, the Process Master can ask the team to quickly evaluate the retrospective by pointing out what they liked in the retrospective and what should be improved [3].

4.7 The retrospective

During the iteration's retrospective, the Process Master will guide the team using the resources he/she prepared while staying outside of the discussion. The Process Master must be a facilitator and not an influencer. However, it must *break the happy bubble* [Scr16b] when necessary, i.e. let the team know the current status and bring attention to the underlaying problems. While facilitating the retrospective, the Process Master must monitor the time to keep the meeting on the predefined time-box and keep the meeting in scope by directing the team to the important issues when they diverge.

The Process Master should document the retrospective for future analysis. The actions the team decided to take are an important part of this document. In the next iteration's retrospective the team must analyse if it was able to accomplish its goals. The team may also set a more distant date to evaluate the improvements if the changes are only expected to be visible in a longer timeframe.

4.8 Putting change into action

While many teams are capable of recognising problems in their process, many fail to continuously improve. In order to promote change, the teams needs to be reminded of its importance and what change actions they decided on. As presented in section 4.6, the team can add the actions to the iteration's backlog, estimate them and thus put it into practice by following the normal iterative process. Another way to give value to the change is to introduce an element that reminds the team of it, e.g. a visual item in the iteration board. As put by L. Rising et al. [RM04]:

> "Keep the new idea visible by placing reminders throughout your organisation. Unless people are reminded, they may forget about the new idea. Post information about the new idea around your organisation – wherever people are likely to see it and discuss it."

4.9 Assessing change

In order to understand the impact of the change introduced, the Process Master must monitor the metrics and see how they evolve. By introducing on change at a time, it will be clear how that change affected the development process. We propose two main methods to assess the success of an introduced improvement:

[3] Activity inspired by the book "Agile Retrospectives" [DL06].

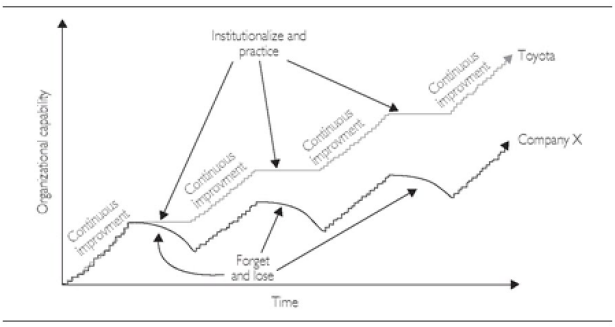

Figure 4.4: The desired evolution of the team's velocity [Scr16b].

4.9.1 Velocity analysis

To monitor the improvement of the team's performance, velocity is a common metric used across agile teams. By registering the amount of work completed in the end of each iteration, the Process Master will be able to tell if the team is succeeding and doing more work in the same time.

It is important to monitor changes for some time in order to make sure the improvements are not forgotten. Scum Patterns [Scr16b] refer to this as a Kaizen Pulse. An illustration of the desired velocity is shown in figure 4.4.

4.9.2 Happiness Metric

Change usually brings discomfort, and so it is important to monitor the status of the team. One way is to apply the pattern *Happiness Metric* [Scr16b] to evaluate if the changes reduced the happiness of the team. The Process Master asks each team member, at the beginning of each retrospective, or via a simple form, to evaluate their happiness with their role and with the company - only two questions. An example of a scale that can be used is shown in figure 4.5.

If the team is able to keep or increase their happiness level, the change positive effects are larger than the discomfort caused by it. An example of an happiness control graph can be seen in figure 5.6.

4.10 Summing up

In this chapter, we introduced a toolkit for the improvement of the agile process for software development. This toolkit focus on simple but effective activities that allow agile teams to deliver faster and better. It acknowledges and fulfils the need for straightforward methods that do not depend on external identities, e.g. external consulting professionals or Agile Coaches, by driving the change from inside the team.

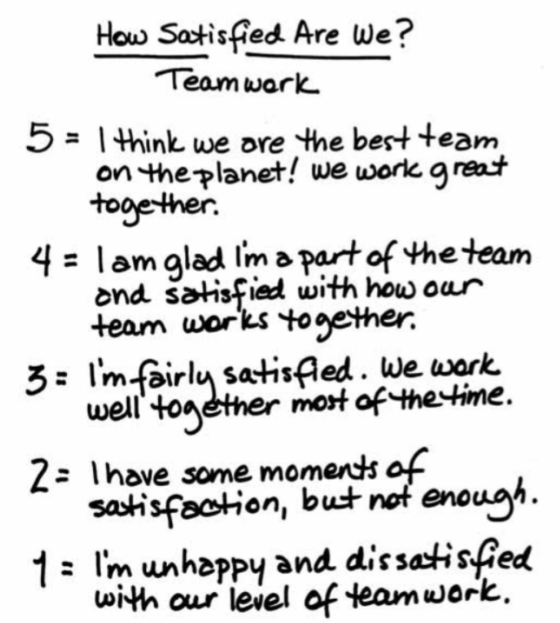

Figure 4.5: An example of a team happiness scale [DL06].

Chapter 5

Case study at Zendesk

As part of the research problem (see chapter 1), a case study was conducted in the software company Zendesk, in the Copenhagen's office. The case study focused on the work of a Scrum team. In this chapter, we will describe how it was conducted and which results were obtained.

5.1 Goals

The primary goal of the case study was to iteratively evaluate the success of different process improvement techniques and build a body of knowledge that resulted in the toolkit presented. The main activities of this case study were the analysis of the process of an agile team followed by discovery of process improvement opportunities and its implementation. By the end of it, we were able to recognise which methods were more valuable to conduct process improvement activities.

5.2 Methodology

For this case study we followed an iterative methodology. This included two key Scrum events: *Sprint Retrospective* and *Sprint Planning*.

During *Sprint Retrospective* the team was invited to find out how it could improve the development process based on the data collected. If the changes would cause an extra effort during the Sprint, the suggestion was that, at *Sprint Planning*, the team would add the most valuable improvement to the Sprint Backlog and agree on the effort it represented.

During the *Sprint* we analysed if the team was actually putting into practice what it committed to and then drew appropriate conclusions. This cycle is shown in figure 5.1.

As exposed in chapter 1, the case study followed an Action Research methodology, as the researcher worked with the team as a team member. The first Sprint was used to get to know the team and the environment. This Sprint was followed by six Sprints where process improvement activities were conducted.

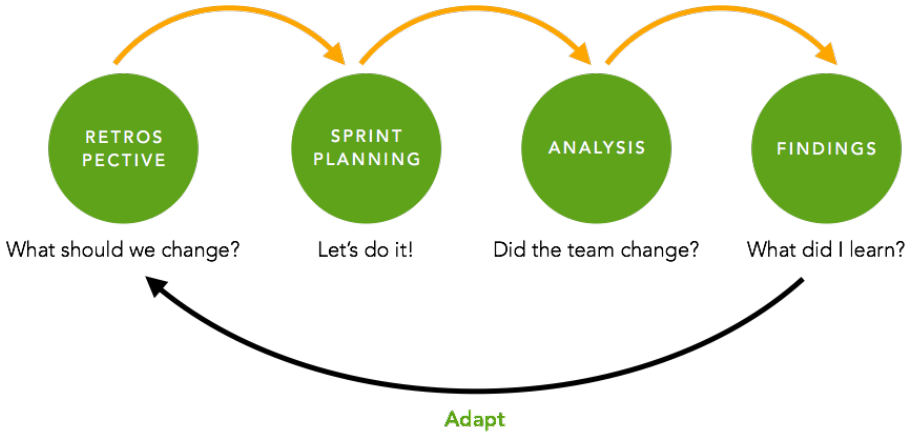

Figure 5.1: The methodology to be followed during the case study.

5.3 Characterisation of the process

Development teams at Zendesk's office in Copenhagen, Denmark follow an agile software process. Teams are free to adapt their process to meet their needs. The team with whom the case study was conducted follows the Scrum methodology putting into practice its roles, cycles and artefacts. The Sprints are of two weeks duration, starting on a Monday and ending on a Friday.

5.4 Characterisation of the team

The Scrum team that participated in the case study is one of three main teams developing a Web App product. The team consists of:

- six developers that, although capable of multi-tasking, focus on one of the areas of web development: front-end or back-end;
- a QA engineer that works mostly remotely;
- a team manager that plays the role of Scrum Master and also does some development work;
- an in-house Product Owner that manages the product of two teams.

The work environment is relaxed, flexible and with balanced work days. The team showed passion and commitment to their work. One day every Sprint is used to explore new technologies and findings (called the "lab day").

The team showed many positive quality indicators from the beginning:

- Code reviews using pull requests;
- Automated tests - both unitary and functional;
- More than 90% code coverage;
- Continuous integration and automated production deploys;

- Code is refactored whenever needed, i.e. they follow incremental design;
- Pairing sessions occur frequently (however, these are not enforced).

5.5 Results

During the case study we were able to identify different problems by using different techniques. In the following sections we present the different ways we identified problems, the actions the team took to solve them, and how they performed.

5.5.1 Detecting and solving day-to-day problems

Some problems are easily detected by watching the day-to-day routine of a team. One of the first identified problems was related to the usage of the process tools provided by the company. The company enforced a backlog management tool but this tool did not allow to have a visible status of the progress, so the team was using a Scrum Board, too, using sticky notes. The usage of a Scrum Board is considered an advantage, since it keeps a visible status of the team progress. [Scr16b, Pattern: Visible Status]. In order to satisfy both requirements and eliminate the time wasted copying the items from the software to the board, the team now uses a printer to transfer the issues to the board - a one-click process.

The team presented another problem related to the backlog. There were some items coming from a secondary backlog, due to a secondary project they had to support. This revealed a problem related to the team's goals we will present in further sections. By following the same automation process to bring the items to the Scrum Board, we were able to give the team a sense of unification.

We identified one more day-to-day problem the team was facing: a reduced presence of the in-house Product Owner. This is due to the fact that the Product Owner is responsible for the backlog of two teams. This problem was discussed during Retrospectives and the Product Owner started participating in all the team meetings. The company is also actively looking for a dedicated Product Owner.

5.5.2 Using a checklist to detect common issues

During the first Sprints of the case study, we focused on detecting common issues. For that, we used a scrum checklist [Cri16] each Sprint to detect missing Scrum practices. We found out the following issues:

- The lack of a burndown chart. As a consequence, the team was not able to visualise how the work done compared to the expectations. We introduced this graph so that the team had a visual indicator of progress.
- The lack of Sprint Review meetings. Although the team had a in-house Product Owner, there was no explicit moment where each team member presented their work. By introducing this meeting, it also gave the team the bigger picture of their work.

- The team size. The recommended size for a team is five, and the team presented a head count of eight. The size of the team exposed another issue, the lack of one well-defined Sprint goal.
- The lack of velocity analysis for release planning. The checklist allowed to detect that the Product Owner does not use the team's velocity for release planning. This prevents the Product Owner of providing a release range based on the available data to the stakeholders. As presented in further sections, this issue was also conditioned by the variability of the team's velocity and other estimation problems.
- The lack of concrete improvement proposals by the end of Sprint Retrospectives. Although the team was conducting regular Retrospectives, most debated issues did not result in tasks for being implemented next Sprints. In section 5.5.6 and following, we present some actions the team was able to introduce during the case study.
- The lack of estimation for all items. The checklist allowed to detect that the team did not split PBIs into SBIs (following an hybrid approach of splitting PBIs when they were too large), some items did not have an estimate when brought into the Sprint, and the SBI estimates were not updated daily. These points were introduced in one Retrospective for team discussion.

5.5.3 Applying Scrum Patterns

Scrum Patterns [Scr16b] were an important tool for process improvement by allowing us to go through a list of common problems accompanied with recognised solutions. Due to its relevance and to easily present sequences of patterns applied, we present these in section 5.6.

5.5.4 Maturity Assessment

Considering the state of art research presented in section 3.2.3, we set out to apply one of the models to the case study: *Scrum Maturity Model* [YdLFdS11]. While we were able to evaluate if the scrum practices were being accomplished or not, the number of metrics required was extensive. Thus, we found that the use of these models are very time-consuming and highlight the need for simple metrics that still reveal process improvement opportunities. Moreover, these models set specific goals per level which may not be fully compatible with the flexibility of agile.

The first maturity assessment conducted with this model - only by briefly verifying if the practices where followed - resulted in level 2, because the team was still failing to accomplish some objectives of level 3.

Considering that this specific model does not enforce the collection of all the metrics (one of level 4 practices says: "All metrics from level 2 to level 4 are monitored and managed for all projects. This is suggested but optional." [YdLFdS11]), after the case study we were able to place the team in level 4, considering that they follow all the relevant Scrum practices and monitor their status. They are near level 5 as they conduct regular improvement activities. However, the team

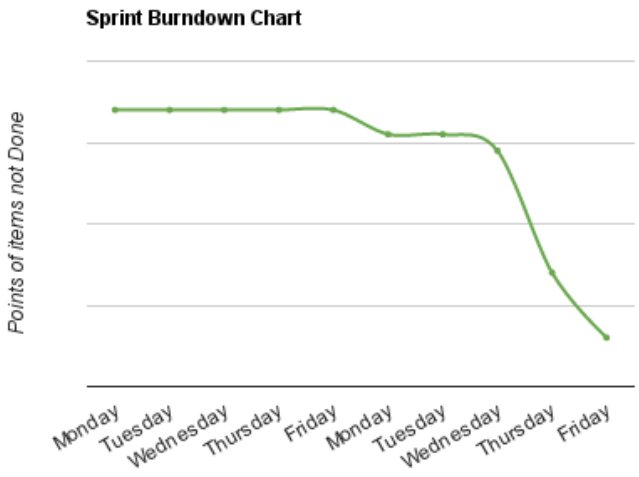

Figure 5.2: Burndown chart of one of the Sprints.

still needs to be able to keep a committed improvement cycle after the case study. It is important to recognise that this model is designed to be applied to an entire organisation, and this case study focus only in one team. Moreover, these must not be confused with CMMI levels.

5.5.5 Conducting velocity analysis

During the case study, we conducted velocity analysis to understand how the team was performing: distribution of items done in a Sprint and velocity across Sprints. Figure 5.2 shows the burndown chart for one of the Sprints. This chart shows that only in the second week of the Sprint, items were marked as done. By revisiting what happened in that Sprint to look for causes, we noticed two issues that were potentially related: the team was interrupted by company activities and the size of the SBIs were large. This lead to bottlenecks in quality assurance and a rush to deploy changes in the last days. While the company activities were decided by management - one could still advice management to consider the influence of the events in the team's rhythm - the size of the items was something that could be easily addressed by the team. In the following Sprint Planning meetings the team was asked to split large items into smaller ones.

Retrospectives play a very important role in agile process improvement and, as a consequence, in our case study, as presented in figure 5.1. In the following sections we describe the most important Retrospectives the team had during the case study and to which actions they led.

5.5.6 Preparing varied Retrospectives

One of the first points the team made in the beginning of the case study was that their Retrospectives where mostly based on the activity *Start, Continue, Stop doing* - an activity where each team member writes in sticky notes a proposal for each of these topics - and they would prefer different

Figure 5.3: The captain magnet being used in an item of the Scrum Board.

dynamics for the Retrospectives. Using resources like the books “Agile Retrospectives” [DL06] and “Project Retrospectives” [Ker13], we were able to provide different experiences. The Retrospectives activities included: analysis of the team’s Sprint burndown chart; free discussions of one topic selected based on the issues detected during the Sprint; sharing of moments they valued during the Sprint; moments for introspection followed by opinion sharing and discussion; etc.

5.5.7 Introducing swarming

To introduce the concept of Swarming [Scr16b, Pattern: Swarming: One-Piece Continuous Flow], a small presentation on the topic was done followed by a discussion. Being a large team, the team considered that it would be valuable for some of its members to swarm on items that needed help to be completed. With swarming we introduced the concept of Captain, the one that would take the important item and whom the others should assist. To remind the team to consider swarming and who was the Captain, a new magnet was added to the Scrum Board with a captain hat (see figure 5.3).

5.5.8 Bug triage

Being a customer service software maker, Zendesk uses its system to assist customers and escalate problems to developers. Since the target team is the core team for the product they help develop, it was getting many problem tickets. The existing flow was that the problems were addressed immediately or sometimes kept in the ticket queue without updates. However, this was causing too many interruptions to the team’s work since many reports were not critical and some could even be seen as feature requests. During a Retrospective, this topic was introduced and the team discussed a way to do bug triage. They came up with a flow that allowed them to still address important issues right away while postponing other matters. The latter were directed to the Product Owner for prioritisation. The team now assigns a person to be responsible for bug triage each week. The flow created by the team is illustrated in figure 5.4.

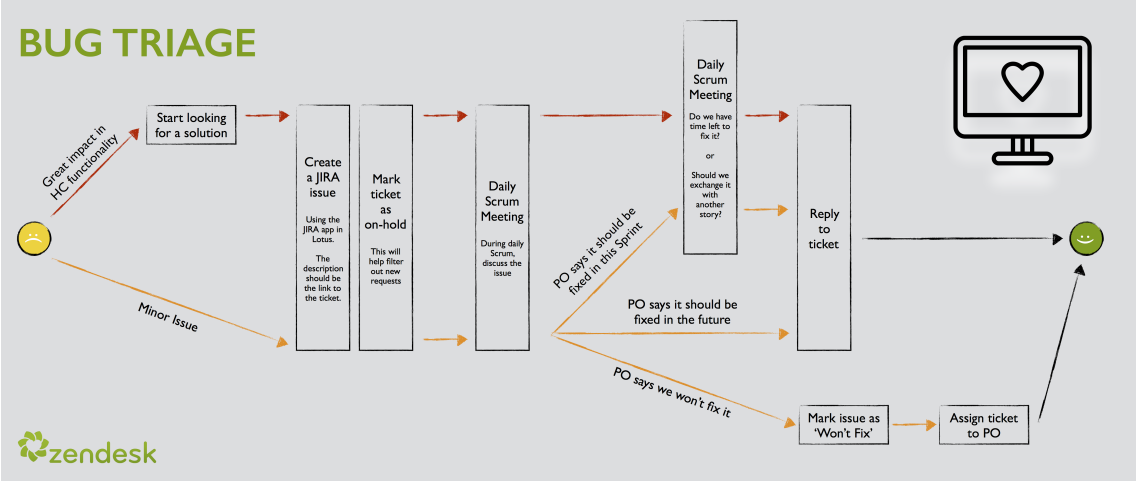

Figure 5.4: The bug triage flow created by the team.

5.5.9 Analysing velocity and estimation

During Retrospectives, the team was invited to analyse graphs like the one in figure 5.2, in order for them to understand how they were performing and how they could improve. These discussions led to decisions on what to do to remove impediments and how to handle items that where not completed by the end of the Sprint.

Along the way, some team members raised questions about the usefulness of estimating. The reason was that for many months they were able to deliver without caring much about estimations. The members then realised that the team has changed since then, because they had acquired new members, thus creating new team dynamics and challenges. Estimation was recognised as an important method to foster team performance.

5.5.10 Actions to team happiness

In one of the Retrospectives, following advise from the Scrum Pattern *Happiness Metric* [Scr16b], the team was presented with two questions they had to answer individually and them discuss with the group. The questions were "Based on your past experience, do you feel the team is working well? Why, why not? Point out biggest issues. Think about development process, communication, human resources, amount of work completed, product proudness, product quality, etc." and "What would increase your happiness with your work and role the most? What about with the company?".

For the second question, the team was then asked to give points - using the same cards as in poker planing - to rank the value of each proposal [1]. These allowed everyone in the team, including the team manager, to understand what was the most important concern to be addressed. In the specific case, the action was to escalate the issue to management.

[1] This is an exercise proposed by Jeff Sutherland et al [Sut14].

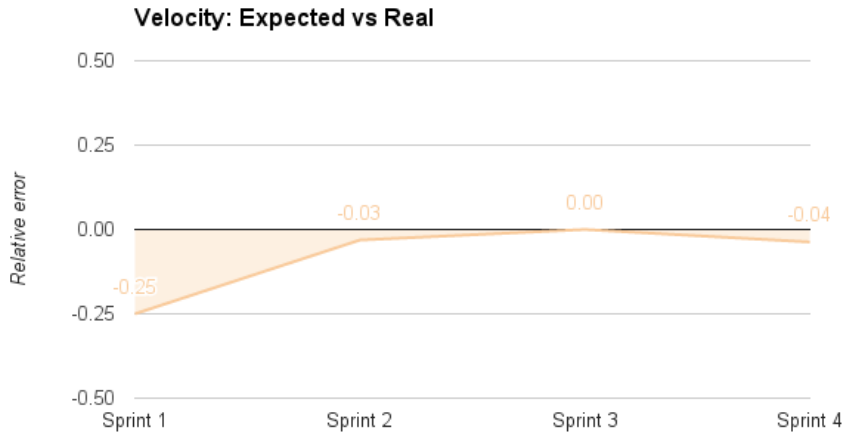

Figure 5.5: Relative error between expected and real velocity of the team.

This method revealed to be effective, easily time-boxed, and with concrete results. It triggered important discussions, in contrast with other methods, which leave the team in the more superficial issues.

5.5.11 Evaluating improvement

In order to evaluate the successfulness of the process improvement activities, we conducted different analysis. In this section, we present the most relevant ones.

Overheads reduced. One of the results we obtained was the reduction of the overheads in some of the tasks, for example, with a better usage of planning tools as detailed in section 5.5.1, and with a bug triage process that keeps one person in charge (presented in section 5.5.8). The reduction of these overheads and interruptions, lead to more time of focused work.

A more stable velocity. As presented in this chapter, one of the main challenges the team faced was related to estimates. In order to track the evolution of the team on the predictably of their work, we monitored, among other metrics, the expected velocity in comparison to the points of the items marked as done by the end of the Sprint. Figure 5.5 shows the relative error between these two measurements. As shown, the team was able to reduce it from 25% to less that 5%. As the team is now more predictable, it is now capable of improving its velocity. "Not until the system has normative behaviour does history predict the future" [Scr16b, Pattern: "Updated Velocity"].

Happiness Metric. In order to assure that the level of happiness of the team members was not affected by the process improvement activities, each Sprint the team was asked if they were happy with the company and with their role. Figure 5.6 shows the results obtained. We were able to keep a stable happiness metric while introducing changes that naturally cause discomfort.

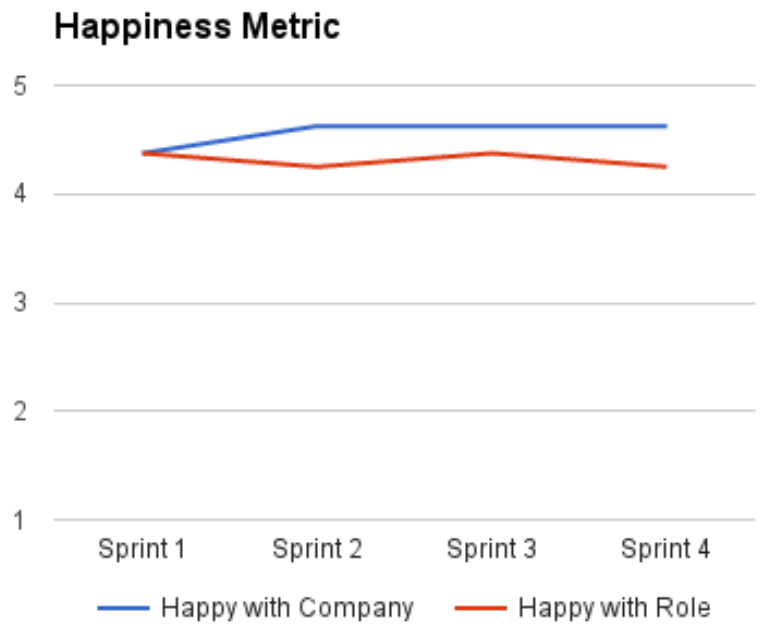

Figure 5.6: Example of chart to track team happiness.

5.6 Scrum Patterns: applied

One of the most effective and complete ways to identify advanced process problems was by analysing the Scrum Patterns. Reading the summary problem of each pattern helped to identify the relevant ones and by reading it we were able to evaluate the severity of the problems.

In this section we present which patterns were applied to each specific most relevant problem that the team had in an effort to show their applicability and importance.

5.6.1 Team work

As the team of the case study was a well established team, with Scrum in place, one would assume that patterns associated with new teams would not apply. That was not what we found out.

For example, the pattern *Small Teams* [Scr16b] says:

> " If you have a lot of work ahead of you there is a temptation to throw the world at it. But work that requires a high degree of shared knowledge and coordination, such as product design and development, defies such approaches.
>
> *Therefore:* Use small teams of people working on serialised work rather the striving for false parallelism. "

This was a problem that appeared in the team along time. In the beginning, the team was small and there was a very well defined *Circle of Trust* [Scr16b], but as the company grew, new

members were integrated. Right now, the team has eight elements. Coordination issues are often frequent, even with daily scrum meetings. Although there is a clear solution - split the team in two, that requires approval from the company and a commitment of the team members to their new roles. It also requires more effort from the Product Owner since he would need to be managing two backlogs and attend related meetings.

This solution would also allow the team to solve another problem they have, related to the *Vision* pattern [Scr16b],

> " How can we create an environment where everyone on the team is heading in the same direction without locking in our final result?
>
> The Product Owner creates and articulates a vision for the product, which will be elaborated into a Product Backlog. The vision is a description of the desired future state the Scrum Team will create. "

In the specific case, the problem with the *Vision* is that the team is currently working on two distinct sub-products thus lacking a vision of what the team builds.

During a Retrospective (as detailed in section 5.5.10), each team member was asked what would increase their happiness the most and then all team members gave value points to each proposal. The one with more points was "All the team works on the same goal", with everyone giving it the highest score. Splitting the team in a way that each new team would work on one of the two sub-products, would greatly contribute to its performance.

The size of the team, complemented by the specialisation of its members, introduced another issue: every Sprint the team needs to bring in stories to keep all occupied, thus having to negotiate the backlog ordering with the Product Owner.

Nevertheless, in order to reduce the amount of items in progress and to make the team work together when needed, the pattern *Swarming: One-Piece Continuous Flow* [Scr16b] was valuable:

> " Working on too many things at once leads to radical reduction in the effectiveness of an individual, in the velocity of a team, or a company. It can cut velocity by 50% and sometimes reduce it to zero.
>
> *Therefore*: Focus maximum team effort on one item in the Sprint Backlog and get it done as soon as possible. Whoever takes this item is Captain of the team. Everyone must help the Captain if they can and no one can interrupt the Captain. As soon as the Captain is Done, whoever takes responsibility for the next priority backlog item is Captain. "

In section 5.5.7, we detailed how this pattern was implemented.

5.6.2 Item estimation, velocity and scope

As the team got more mature, it started not feeling the need to estimate. But, with the introduction of new members, there was more unpredictability of how much they could deliver. *Yesterday's weather* pattern says [Scr16b]:

> " It's human nature that individuals and teams with self-esteem set increasingly higher goals for themselves. (...) It is more common that these higher levels are unsustainable in the long term.
>
> *Therefore*: In most cases, the number of Estimation Points completed in the last Sprint is the most reliable predictor of how many Estimation Points will be completed in the next Sprint. "

So the team restarted estimating all items and not bringing to the Sprint more than they did the previous Sprint. Because *Teams that finish early accelerate faster* [Scr16b]:

> " Teams often take too much work into a Sprint and cannot finish it. Failure prevents the team from improving.
>
> *Therefore:* Take less work into a Sprint. "

As the team continued, velocity was updated according to the pattern *Updated Velocity* [Scr16b].

The team also improved their commitment accuracy by directing more attention to *Definition of Ready* [Scr16b]:

> " If work items are not precisely understood, development effort (and time) tend to balloon, which in turn cause the Sprint to fail.
>
> *Therefore:* Each item must meet at least the following criteria before it can go into Sprint Planning as candidates to be put in the Sprint Backlog:
>
> - The work to be done has value
> - It has been estimated
> - It is testable, and the tests for it have been defined
> - The pieces are sized appropriately "

In the effort of trying to reduce interruptions, the team introduced a new bug triage process as described in section 5.5.8. According to this new process, the Product Owner is able to ask for new issues to be taken to the Sprint. This, however will add more work to a full Sprint. The solution for this problem was found using the pattern *Change for free* [Scr16b] which states that the Product Owner is allowed to exchange items as long as the effort to materialize the new one is not superior than the current one.

5.6.3 Patterns for process improvement

During the case study, patterns that advise on process improvement techniques were also applied. In order to foster discussion on underlaying problems, it was necessary to apply *Beyond the Happy Bubble* [Scr16b]:

> " Teams can get to a state where they are happy and things are going well. However, this allows dysfunctions to persist. Whether or not they recognise it, the team is in

> a rut, problems are not fixed, and performance begins to decline. It is like they are surrounded by a bubble that insulates them from unpleasant but true information.
>
> *Therefore:* Jolt the team into awareness of their situation (pop the happy bubble): force the team to confront their happy-bubble-ness by showing them important deficiencies. Then together with the team, make actions to improve. Create a culture in the team of relentless, continuous self-examination and improvement. ”

Then, following a *Kaizen Pulse* [Scr16b] (see example in figure 4.4):

> “ Because it takes time to establish a statistically sound baseline, it’s difficult to show improvement from minute to minute, hour to hour, or even from Sprint to Sprint. *Therefore:* Alternate periods of controlled velocity with spikes of process improvement. ”

During Sprint Retrospectives, there were many ideas for improvement. In order to be able to pick one to be implemented, the pattern *Happiness Metric* was applied: “Find out what one improvement will increase the happiness of the team the most, and implement that improvement in the next Sprint” [Scr16b].

This pattern is then followed by *Scrumming the Scrum*: “Identify the single most important impediment at the Sprint Retrospective and remove it before the end of the next Sprint” [Scr16b] by adding it as an item to the Sprint Backlog with a Definition of Done, thus also applying *Testable Improvements* [Scr16b] by defining concrete actions, to make sure that their goals where accomplished. The team has yet to achieve a *Scrumming the Scrum* status for all the issues they found.

5.7 Summing up

During the case study, the team conducted different process improvement activities that lead to a better workflow and more capacity. From a research point of view, we were able to identify different techniques that can help teams improve their process without depending on an external person - in the case study the researcher was part of the team. This research contributed to the toolkit presented in chapter 4. We also conducted one of the first trials of the project “Scrum Patterns”, reporting a good fit of the patterns to the scenario we were presented with during the case study.

Chapter 6

Validation

In this chapter, we present the validation actions conducted to understand the applicability of the toolkit presented in chapter 4.

6.1 Methodology

In order to evaluate the usefulness and appropriateness of the toolkit presented in this dissertation, we formulated a questionnaire to be used in interviews with Process Masters (e.g. Scrum Masters), then conducted these interviews and analysed the results.

We conducted eight interviews with an approximate duration of thirty minutes each. The interviewees where given freedom to answer beyond the specific questions, thus not being limit to the points addressed in the questions.

Due to the openness of the questions, in the analysis of the results, we group the answers as possible based on their contents.

6.2 Questionnaire

The questionnaire included six main sections:

1. Questions for characterisation of the interviewee, supported teams and methodology used;
2. General questions to understand how the interviewee views process improvement and how useful a toolkit for process improvement would be for the interviewee;
3. Questions on the metrics used by the team and which ones the Process Master values;
4. Questions on ways to identify and solve problems;
5. Questions to understand how the Process Master usually conducts retrospectives and what he/she values in it;

6. Questions on assessment of change following the introduction of improvements.

The complete questionnaire can be found in appendix A.

6.3 Analysis of results

In this section, we analyse the results and take appropriate conclusions to the toolkit. The collected answers can be found in appendix A.

6.3.1 Characterisation of the interviewees

The interviews where conducted in three different companies, with different approaches when it comes to the process they follow. In the first company (interviews from A to C), the interviewees also have other responsibilities, so their focus in the methodology is more reduced. By contrast, in the two other companies, the interviewed Process Masters focus only on this role and support more than one team (they could also be seen as agile coaches of teams that do not have a Process Master). Except from team B, all the teams follow Scrum. The iterations range from one week to three. All teams have in-house Product Owners.

6.3.2 Overview of process improvement practices followed

All participants said they were concerned in process improvement and took the time to think about ways to improve the way their team works. Thus, **all participants are interested in conducting process improvements.** However, only one of the three people who also work as developers revealed a high concern with these.

When asked how they could evaluate improvements, most affirmed they would gather feedback from the team. Therefore, highlighting **the importance of the satisfaction of the team** as a positive indicator of the change.

6.3.3 Expectations for a process improvement toolkit

The majority of the respondents (6 in 8) would **potentially use a small toolkit for process improvement**. Many said that they would like to analyse its contents to better understand if it would be useful or not. When asked what the guide should contain, most expect it to **contain different approaches or possibilities** to conduct process improvement. Many implicitly assumed the toolkit would be focusing on retrospectives.

6.3.4 Value of different metrics

In order to evaluate the value the interviewees give to different metrics proposed in the toolkit, we indirectly asked about them. Converting back to the metrics presented, the results were as follows.

The notebook. When asked if it would be useful to note down issues to discuss with the team later, 5 in 8 consider it relevant while the others said the issues should be handled immediately. We now understand that this technique can potentially be scoped down to be a notebook of interesting behaviours in the team that do not require an immediate action, but do indeed need improvement.

Process checklist. A process checklist was considered relevant by 4 out of the 8 respondents. **There is an important relation between the appraised usefulness of a checklist and the years of experience of the Process Master.** More mature Process Masters say that this is something they have in their mind. When asked if they would consider using the checklist in a retrospective, among other similar opinions, interviewee F said "I don't think they would pay any attention to it". Overall, this tool still seems useful for Process Masters that do not completely dominate the methodology - a possible target of the toolkit presented.

Performance analysis. When asked about the usefulness of tracking velocity, the opinions diverged: 5 in 8 said it was important. The ones who considered this metric important, correlate to the more experienced Process Masters. Thus, revealing **the importance of this metric in the toolkit as a way to foster its use among the less experienced ones**. All people interviewed revealed to be concerned when the team fails to be delivering regularly. One of the interviewees said, however, that the evolution of a story burndown chart can be a flat line in the beginning without that representing a problem, since the team works with large stories and small tasks. Interestingly, 4 in 8 showed no concern with iterations where the team is not able to deliver what it should. This can be, once again, correlated to less experienced Process Masters and more relaxed environments.

Collecting individual feedback. All participants in this study said they did or wanted to collect individual feedback from team members. Some did it formally while others more informally. The majority added this is a very useful way to understand personal problems that may affect the team.

Scrum Patterns. Most respondents (7 in 8) said they would possibly **use a list with problems teams usually face** to look there for recommended solutions. One Process Master showed concerns that it could be too general while another said there should be practical evidence of the usefulness of the solutions.

6.3.5 Steps followed to find solutions

Before introducing, in the conversation, the possibility to use a list of recurrent problems with associated solutions (Scrum Patterns), we asked what they would do if they were looking for solutions. All respondents mentioned that **one of the main possibilities would be to discuss the problem with their peers**. This is a point that is mentioned in the toolkit.

6.3.6 Conduction of retrospectives

The questions on retrospectives aimed to understand how the interviewees conducted retrospectives. The answers where very coherent among respondents. All said it was **very important to set actions by the end of the retrospective**. When asked about the activities conducted, most mentioned activities were varied and that they took time to prepare them beforehand. Thus, showing the **importance of good retrospectives**.

6.3.7 Value of different methods for change evaluation

All the participants said that it was relevant to verify if the changes lead to an improvement. Most say they do this by discussing with the team in the retrospective that follows the change implementation. We concluded that, although this way of assessing change is subjective, it can be a good first indicator. Thus, **a possible addition to the toolkit**. In what velocity analysis is concerned, only 3 in 8 valued it. **Some pointed out that is it difficult to link velocity variation to changes in the process.** One of the participants mentioned the difficulty of using it when the velocity is unstable. This is a point covered in the toolkit.

While all the respondents affirmed the **extreme importance of making sure the team is happy**, only 3 in 8 captured this information in a more objective manner.

6.4 Summing up

All-in-all, after conduction the validation phase we were able to better understand both the expectations of Process Masters and the relevance of the points covered in the toolkit.

From the analysis we did in section 6.3, we can confirm the most adequate target group of the toolkit presented are Process Masters that are an active part of the team and thus are not able to neither continuously focus on process improvement activities or have an external vision of the events. Interviewing “multi-team”, “more advanced” Process Masters showed how some of the practices the toolkit puts forward, by not being so elementary, can foster the evolution of the Process Master as one. On the other hand, practices considered not so useful by more experienced ones, were said to be useful by the remaining Process Masters, thus also pointing out their value.

While analysing the results in section 6.3, we identified very specific areas were the toolkit could be improved. In a more broad approach, we believe that to better meet the expectations of Process Masters the toolkit could increase the number of examples and give some more focus to retrospectives.

Chapter 7

Conclusions and future work

Agile methodologies introduced a new level of freedom and adaptability. However, this cannot be seen as an excuse for a processless environment. Thus, it is of the most importance that teams are able to improve their process.

In this work we reviewed agile methodologies and process improvement techniques, we conducted a case study where we tried different improvement methods and contributed to a raise in the team's performance, and proposed a simple but effective way to conduct agile process improvement activities. Agile Forward, the toolkit introduced, focus on the different steps needed for process improvement: Metrics collection and evaluation; Analysing the problems and finding solutions; Preparing a retrospective; The retrospective; Putting change into action; Assessing change.

In the validation phase we learned that there is a clear interest from the target audience in improving their process. The different items of the toolkit described in this dissertation appear to match the needs of the interviewees and encapsulate ways to detect and solve problems in areas they value.

We set out with the goal of understanding the overall phenomena of process improvement, from analysis to concrete planning and implementation phase. We believe we accomplished this goal by identifying key obstacles, common practices, and recommendations.

Considering not only the input from the validation phase but also our perception of the work done, we believe future work could focus on the following aspects:

1. Based on the feedback received, test new ways of process improvement to be potentially included in the toolkit;

2. Based on the feedback received, a next version of the toolkit could include more examples on each item so that the users of the toolkit could better understand the purpose and applicability of each item;

3. Conduct more validation interviews to collect feedback from even more different environments thus increasing the possibility of generalisation of the toolkit;
4. Focus more on other agile methodologies to increase the applicability of the toolkit;
5. Carry out other case studies with the current version of the toolkit to evaluate its applicability in practice;
6. Follow-up for a longer period with the team targeted in the case study to see if they were able to keep improving.

To end with, this dissertation, as a learning experience, made the author realise the importance and complexity of process improvement. A complexity that is subjective and ranges from phycology to core engineering, and must meet in a balanced point where a Software Engineer is able to make improvements happen. Software engineering is still a new industry and a different one: one that depends on each single person that help build a product. If we are able to empower them to their best, the end-product will be by far superior.

References

[Sc15] Scrum Alliance. State of Scrum Report - 2015 [online]. July 2015. URL: https://www.scrumalliance.org/why-scrum/state-of-scrum-report/2015-state-of-scrum.

[AHAK11] Amr Noaman Abdel-Hamid and Mohamed Amr Abdel-Kader. Process Increments: An Agile Approach to Software Process Improvement. In *2011 AGILE Conference*, pages 195–200. IEEE, July 2011.

[ALMN99] David E Avison, Francis Lau, Michael D Myers, and Peter Axel Nielsen. Action research. *Communications of the ACM*, 42(1):94–97, January 1999.

[AMO13] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. Kanban in software development: A systematic literature review. *39th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 9–16, 2013.

[BA04] Kent Beck and Cynthia Andres. *Extreme Programming Explained*. Embrace Change. Addison-Wesley Professional, November 2004.

[BBvB+01] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, and Ken Schwaber. Manifesto for Agile Software Development [online]. 2001. URL: http://www.agilemanifesto.org.

[BDS98] Mike Beedle, Martine Devos, and Ken Schwaber. SCRUM: An extension pattern language for hyperproductive software development. December 1998.

[Bro11] Pawel Brodzinski. Kanban: Mapping Process to Kanban Board [online]. August 2011. URL: http://brodzinski.com/2011/08/map-process-to-kanban-board.html.

[BT04] Barry W Boehm and Richard Turner. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. *ICSE*, pages 718–719, 2004.

[CKS11] Mary Beth Chrissis, Mike Konrad, and Sandra Shrum. *CMMI for Development*. Guidelines for Process Integration and Product Improvement. Pearson Education, March 2011.

[CM09] Patel Chetankumar and Ramachandran Muthu. Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices. 2(1), January 2009.

[Coh16] Mike Cohn. Scrum Methodology and Project Management [online]. 2016. URL: https://www.mountaingoatsoftware.com/agile/scrum.

[Cri16] Crisp. Scrum Checklist [online]. 2016. URL: https://www.crisp.se/gratis-material-och-guider/scrum-checklist.

[DL06] Esther Derby and Diana Larsen. *Agile Retrospectives*. Making Good Teams Great. 2006.

[DL16] Jeff De Luca. The Latest FDD Processes [online]. 2016. URL: http://www.nebulon.com/articles/fdd/latestfdd.html.

[HD06] Deborah Hartmann and Robin Dymond. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In *AGILE 2006*, pages 6 pp.–134. IEEE, 2006.

[HR14] Neil Harrison and Joel Riddle. Teams That Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams. *HICSS*, pages 4722–4728, 2014.

[ISO13] ISO. ISO/IEC 15504-6:2013 , 2013.

[Jak09] Carsten Ruseng Jakobsen. Scrum and CMMI Going from Good to Great. In *2009 Agile Conference (AGILE)*, pages 333–337. IEEE, 2009.

[JJ07] Carsten Ruseng Jakobsen and Kent Johnson. Scrum and CMMI Level 5: The Magic Potion for Code Warriors. In *AGILE 2007*, pages 272–278. IEEE, 2007.

[Ker13] Norman Kerth. *Project Retrospectives*. A Handbook for Team Reviews. Addison-Wesley, July 2013.

[Mar09] Robert C Martin. Clean Code - a Handbook of Agile Software Craftsmanship. *Prentice Hall 2009*, 2009.

[MM06] Micah Martin and Robert C Martin. *Agile Principles, Patterns, and Practices in C#*. Pearson Education, July 2006.

[Mou] Mountain Goat Software. Planning Poker: An Agile Estimating and Planning Technique.

[PC12] Mary Poppendieck and Michael A Cusumano. Lean Software Development: A Tutorial. *Software, IEEE*, 29(5):26–32, 2012.

[PP03] Mary Poppendieck and Tom Poppendieck. *Lean Software Development*. An Agile Toolkit. Addison-Wesley, May 2003.

[RM04] Linda Rising and Mary Lynn Manns. *Fearless Change*. Patterns for Introducing New Ideas. Pearson Education, October 2004.

[RS03] Doug Rosenberg and Matt Stephens. *Extreme Programming Refactored*. The Case Against XP. Apress, August 2003.

[SA07] Outi Salo and Pekka Abrahamsson. An Iterative Improvement Process for Agile Software Development. *Software Process Improvement and Practice*, 12(1):81–100, 2007.

[Scr16a] Scrum.org. Open Assessments [online]. 2016. URL: https://www.scrum.org/Assessments/Open-Assessments.

[Scr16b] ScrumPLoP. Scrum Patterns [online]. 2016. URL: https://sites.google.com/a/scrumplop.org/published-patterns/home.

[Soc14] Ieee Computer Society. *Swebok*. Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Press, January 2014.

[SS13] Ken Schwaber and Jeff Sutherland. The Scrum Guide [online]. July 2013. URL: http://www.scrumguides.org/.

[Sut14] J J Sutherland. *Scrum*. The Art of Doing Twice the Work in Half the Time. Crown Business, September 2014.

[Wel13] Don Wells. Extreme Programming: A gentle introduction [online]. 2013. URL: http://www.extremeprogramming.org.

[Wik16] Wikipedia. Project management triangle - Wikipedia, the free encyclopedia [online]. 2016. URL: https://en.wikipedia.org/wiki/Project_management_triangle.

[YdLFdS11] Alexandre Yin, Soraia de Lemos Figueiredo, and Miguel Mira da Silva. Scrum Maturity Model. January 2011.

[Yin11] Alexandre Yin. Scrum Maturity Model. Master's thesis, Instituto Superior Técnico, Lisboa, September 2011.

[Zen16] Zendesk. Press Releases & Announcements | Zendesk [online]. 2016. URL: https://www.zendesk.com/company/press/.

REFERENCES

Appendix A

Validation interviews

In this appendix, it is possible to find the complete questionnaire used in the validation phase presented in chapter 6 as well as the individual answers to it.

A.1 Complete questionnaire

A.1.1 Characterisation

1. Years of experience in agile
2. Current team details:
   (a) Size
   (b) Methodology
   (c) Iteration duration
   (d) End product
3. Process Master is also team manager?
4. Is current team supported by a coach?
5. Does the team have an in-house PO, how many teams does the PO support?

A.1.2 General questions

1. *Openness to conduct process improvement:* Do you usually think of different ways you could improve the way the team works?
2. *Verification of improvement:* If so, how do you know if the change had a positive effect?
3. *Availability to learn:* Do you try to bring new ideas to the team you read somewhere else?

4. *Availability to conduct process improvement:* Do you dedicate time to think about process improvement activities?

5. *Availability to use the toolkit:* If you were given a mini-guide to help with process improvement, what would be the chances of using it?

6. *Expectations for the toolkit:* What do you think this guide should contain?

A.1.3 Metrics collection and evaluation

1. Would it be useful to note down issues you see to discuss them with the team later?

2. If you add a checklist of practices you should be doing in methodology would you use it? Alone or with the team?

3. Do you think it is useful to track the velocity of the team?

4. Would you be worried if the team seems to not perform as it should during an iteration, and only in the last days everything is completed?

5. Would you care if your team didn't finish what it commits to in the beginning of the iteration?

6. Do you have private meetings with other team members regularly? Do you discuss problems with the team members, trying to collect feedback on the process or is it more personal?

A.1.4 Analysing the problems and finding solutions

1. If you found a problem and were wondering of a solution for it, what would you do?

2. If there was a list with problems teams usually face and corresponding recommended solutions, would you look there for solutions?

A.1.5 Preparing and conducting retrospectives

1. Duration, frequency and participants of the retrospective.

2. Do you take time to prepare the retrospective beforehand?

3. Do you vary the activities?

4. How important it is to set concrete actions by the end of the retrospective? If you set actions in the previous retrospectives, did you personally made sure they happened, i.e. remind the team?

5. During a retrospective do you try to lead it, or do you let the discussion go?

A.1.6 Putting change into action and assessing change

1. Imagine you put an improvement in place, is it important to see if there were actual improvements?
2. How do you know if what you decided to change, actually improved the process?
3. Would it be important to see if the velocity changed?
4. Is it important to make sure the team is not unhappy due to a change in the way they work?

A.2 Answers

The anonymised answers to the questionnaire are presented in table A.1.

Table A.1: Answers to questionnaire

	Characterization of the process and background information							
ID	**Market**	**Years working with Agile**	**Team(s) size**	**Methodology**	**Iteration duration**	**End product**	**Scrum master and developer?**	**Is current team supported by a coach?**
A	Customer Service Software	6	1 team - 3 elements	Scrum	2 weeks	Web App	Yes	No
B	Customer Service Software	10	1 team - 8 elements	Kanban - not completely	1 week	Web App	Yes	No
C	Customer Service Software	3	1 team - 7 elements	Scrum	2 weeks	Web App	Yes	No
D	Insurance	1	2 teams - 3 and 7 elements	Scrum	2 and 3 weeks	mainframe	100% SM	N/A
E	Insurance	5	2 teams - 9 and 4 elements	Scrum	2 and 3 weeks	mainframe	100% SM	N/A
F	Insurance	8	3 teams - 4, 5 and 5 elements	Scrum	2 and 3 weeks	frontend and mainframe	100% SM	N/A
G	E-commerce	4	4 teams - around 7 each	Scrum	2 weeks	mobile	100% SM	N/A
H	E-commerce	4	4 teams - around 5 each	Scrum	2 weeks	financial applications	100% SM	N/A

	Characterization …	General questions			General questions			Metrics & …
ID	**Does the team have an in-house PO, how many teams does the PO support?**	**Do you usually think of different ways you could improve the way the team works?**	**If so, how do you know if the change had a positive effect?**	**Do you try to bring new ideas to the team you read somewhere else?**	**Do you dedicate time to think about process improvement activities?**	**If you were given a mini-guide to help with process improvement, what would be the chances of using it?**	**What do you think this guide should contain?**	**Would it be useful to note down issues you see to discuss them with the team later?**
A	In-house: 2 teams	more or less	asks the team in the next retrospective	no	yes	very probable	recipe on how to do things or a guide to help reflect on the current process, suggestions how to engage the team	Yes
B	In-house: 2 teams	yes	sets a date to evaluate, changes 1 thing at a time	yes, and sends articles to team, inspire	yes - spends 5 to 6 hours a week reading about ways to improve the process	very probable	techniques for process improvement but with explanations why it works / reasons behind	yes, the team keeps a shared board for this porpose
C	In-house: 2 teams	more or less, takes time every week to team for team managing/SM tasks	feeling	yes	yes	more or less, all the teams are different	ideas to try	yes
D	In-house: 2 teams	yes	by creating a culture of feedback	yes	N/A (full-time)	very probable	practical examples	more or less, many issues need to be discussed the moment they happen
E	In-house: 2 teams	yes	from feedback from the team	yes	N/A (full-time)	believes there is more value in team dynamics	purpose of each proposal, what the outcome should be, expected reactions from the team	yes, if it is not handled in small discussions during the sprint
F	In-house: 2 teams	yes	feeling	yes	N/A (full-time)	maybe. Would take a freely approach	exercises, tips and actions on different areas	usually will not wait, will talk individually to team members. Sometimes puts it on the Scrum Board to be discussed the next day.
G	In-house: 1 team	yes	the team decides in the next retrospective if it was beneficial	yes	N/A (full-time)	no, because the Agile values are enough to know how to improve	N/A	Does not wait
H	In-house: 1 or 2 teams	yes	based on the feedback provided by the team in a retrospective	yes	N/A (full-time)	maybe, after analysing its contents and discussing the possible usage with the team	First, it shouldn't be a recipe. Help to indentify areas for improvement and then which possibilities to solve a problem. It should mostly try to place people in a process improvement mindset.	yes, if it was difficult to deal with and needed to be further discussed or was of low priority and could be introduced in a retrospective

	Metrics & detection of problems							Retrospectives
ID	**If you add a checklist of practises you should be doing in methodology would you use it? Alone or with the team?**	**Do you think it is useful to track the velocity of the team?**	**Would you be worried if the team seems to not perform as it should during an iteration, and only in the last days everything is completed?**	**Would you care if your team didn't finish what it commits to in the beginning of the iteration?**	**Do you have private meetings with other team members regularly? Do you discuss problems with the team members, trying to collect feedback on the process or is it more personal?**	**If you found a problem and were wondering of a solution for it, what would you do?**	**If there was a list with problems teams usually face and corresponding recommended solutions, would you look there for solutions?**	**Duration, frequency and participants of the retrospective.**
A	Yes, but not trying to implement all. Alone and with the team.	yes, for big teams	yes	No	Yes, but don't discuss process	The SM will usually ask peers and the team.	Yes	1 hour once a month with team and PO
B	just a quick look, will not follow.	N/A	yes, may suggest that the team is using the tool as reporting, stories are not gradual enough, very risky.	N/A	Yes, process discussions are not regular. The SM tries to ask people to bring the issues to the retrospective but sometimes people are shy.	Will search on-line, ask peers in the office, describe the problem to the team and ask if they also feel it is an issue.	Yes	Retrospectives each week with a duration of half-an-hour with team and PO
C	yes, but skeptic, first alone.	the team does not estimate items	yes, not ideal. If systematic, the stories should be splited. Can also be impacted by external factors like company events.	The team usually takes less than what they can do. Otherwise, wouldn't care much.	Yes, not usually about process; Will bring the issue to retrospectives if it makes sense.	The SM will talk with the manager and with other team leads.	Yes, if there are practical examples	Once a month, 1 hour with team and PO
D	yes, alone.	no	yes, that is a current problem.	Depends on why the team doesn't finish.	No, but would like to do it. Seems that in retrospectives there is no sincere feedback.	The SM will ask other SMs in the company and his leader.	Yes	Between 1 hour and 1h30 per sprint with team and PO
E	not for SM, but maybe for the team to have a look.	yes	yes, suggests a problem with the break down of stories to tasks.	Yes.	Not formally, casual discussions.	First, search in the web and then discuss with colleagues.	Reduced chance, believes it depends very much on the team	1 hour each sprint with team and PO
F	no	yes	yes. The SM will ask the team for reasons behind it if she sees it happening.	Yes.	No, has some reservations because it may look like she is the leader of the team.	The SM will discuss it with colleagues and leader.	Yes, would like to try it	Between 1 hour and 2 hours per sprint with team and product owner
G	no, knows what matters.	yes, very important	yes, but that is something the team discusses every day.	Yes, the team has a carry over chart. The SM tries to understand why it is happening during the sprint.	Very informally during breaks.	Will discuss it with some members of the team, manager and peers.	Yes, SM is familiar with Scrum Patterns. Believes it is very relevant.	1 hour per sprint with team and PO
H	Sometimes to be reminded of each point, and understand why they are not doing some. Would analyse alone and then bring the most important point to the team.	yes, to know how much they can deliver	no, if doing a story burndown chart, it is normal to take some time to finish the first ones since stories are large and then splitted into tasks.	yes, considers it a problem and tries to make sure the teams do not carry over stories from sprint to sprint	Informally but mostly about personal problems. Prefers to discuss isues with the all team.	Would ask his peers about their opinion and also for some articles he could read on the issue	yes, definitely	1 hour per sprint with team and PO

	Retrospectives				Measuring Improvements			
ID	**Do you take time to prepare the retrospective beforehand?**	**Do you vary the activities?**	**How important it is to set concrete actions by the end of the retrospective? If you set actions in the previous retrospectives, did you personally made sure they happened, i.e. remind the team?**	**During a retrospective do you try to lead it, or do you let the discussion go?**	**Imagine you put an improvement in place, is it important to see if there were actual improvements?**	**How do you know if what you decided to change, actually improved the process?**	**Would it be important to see if the velocity changed?**	**Is it important to make sure the team is not unhappy due to a change in the way they work?**
A	No	No	Very Important	Leads the discussion	Yes	The team usually revisits the actions from last retrospective, tries to understand how the teams feels about it	Does not believe there is a correlation between those	Yes, but not objectively.
B	Yes	Yes, some different activities and some offsite	Very important, usually needs to remind the team on the actions	In the beginning uses 5 minutes to tell his opnion, leading seat is rotating	Yes, based on hapiness	Doesn't measure it objectively. Tries to perceive if the ROV is improved.	NA	Yes.
C	No	No, but would like to do that	Important, usually actions are assigned to team members	Tries not to lead, hopes someone else takes that responsibility	Yes	Usually checks in the next retrospective.	NA	Yes, very important.
D	Yes	Yes	Very important, it is necessary to remind the team	Lets the discussion flow	Yes	Asks individually.	Does not value it	Yes, important.
E	Yes	Yes	Important but the discussion is also very important, needed to remind the team.	Both	Yes	Asks the team in the next retrospective.	Yes, but currently the teams have problems keeping the velocity stable	Yes, it is the most imporant indicator. The SM says she doesn't do it objectivily
F	Yes	Yes	Very important to have at least 1 action point. An element of the team is assigned to it.	Mostly leads it	Yes	Sets a date with the team to evaluate change.	Yes	Yes, the most important. Usually asks to how happy each team element is from 1 to 5 in retrospectives.
G	Yes	Yes, is building a Retrospectives Toolkit	Essential, actions are added to the Scrum Board and treated and other regular items.	Depending on the retrospective, both are possible	Yes	Subjective analysis in the next retrospective.	Yes	Very important. Prepares activities in retrospectives to track the level of hapiness.
H	Yes	Usually uses one of three activities he considers to be the best	Very important in most cases. In some cases, it is acceptable to use the retrospective as a deep sharing moment. Most of the times tries to assume the responsability of the change, otherwise remind the team.	Makes sure the time-boxes are respected but lets the discussion flow	Yes	Usually an improvement comes from a team member feedback. If that feedback changes, the action worked.	No, it is very difficult to mach actions to velocity changes.	One of the most important points. Once a quarter, measures it with 3 hapiness levels.