Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Laurea Magistrale in Computer Science LM-18

Double Degree with

Universidad Nacional de Catamarca (UNCa)

Analysis and Definitions of Effective Metrics for Scrum Teams

Candidate:
Federico Ramayo

Supervisor (UNICAM):
Prof. Andrea Polini

Supervisor (UNCa):
Prof. Juan Pablo Moreno

Academic Year 2016/2017

Analysis and Definitions of Effective Metrics for Scrum Teams

Federico Ramayo

July 2017

Abstract

Nowadays, organizations that are not ready to adapt to changes that are happening every day, are most likely to lose any competitive advantage they have achieved, leaving them lagging behind those who did. In software industry, this can be seen in companies that deliver software once every several months, or do not have a good communication with their customers and do not deliver the functionalities they need/want. In order to counter these problems Agile methodologies were born. Although these methodologies make promises regarding teams' productivity, this aspect does not depend only on the use of a framework but also the practices, processes, and the culture within a team and an organization.

The current thesis sets the necessary concepts to establish useful metrics that any software development team practicing Scrum should take into account. To achieve this, in first place Agile, along with its principles and values, have been defined and explained, as well as the concepts that rule Agile's most used framework, Scrum. In second place, different metrics that are useful for Scrum teams were classified and explained. To support the selection of really helpful metrics a survey was carried out. The results obtained, along with an analysis of the available tools for software project management, settled the basis for the establishment of a working environment, which consists of open source applications. These tools cover almost all the spectrum in a software project management, from the specification of features to develop, till the deployment of the product, passing through of the coding tasks that developers have to perform along with documentation, testing, among others, as well as the analysis of different aspects involved in the team's performance.

To my parents,
for giving me wings.

Acknowledgments

First of all, I would like to thank my supervisors Professor Andrea Polini and Professor Juan Pablo Moreno for providing me their support and the guidance over these months. Without them, this thesis would not be possible.

Thank you to the National University of Catamarca not only for giving me this opportunity but also for giving me the tools to develop as a professional anywhere in the world. I am also grateful to the University of Camerino for teaching me that "Il futuro non crolla", and no matter how great a setback may be, with effort and work you can get ahead.

Special thanks to two excellent people that Camerino and the destiny made me meet, PhD Cesar Nieto Coria for being there, giving his help since I put my first step in Castelraimondo, and PhD Garima Tiwari for her infinite help.

Last but not the least, I want to thank my parents for giving me all the support to fulfill this dream, even in the most difficult moments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The software history is populated with examples about developments that failed because they were delivered out of time or because their cost was higher than the planned one. Because of these reasons, in 2001, a group of developers had a meeting in Utah to establish the basis for a new and better way to develop software. As a result of this meeting, the Agile Manifesto [5] was born, in which the developers enunciated four core values and twelve principles. The main ideas proposed by these values and principles encourage the people developing software to embrace the change and get really involved with the different stakeholders. The ideas in the manifesto were the cornerstone for the methodologies that would be born during coming years such as Scrum, Kanban, eXtreme Programming (XP), among others. But it is needed to say that Agile is not only a set of methods that conform to a methodology, its concept goes beyond that and tries to change the way in which a developer, team or company thinks and behaves, so it is also possible to call it a different mindset [62].

1.1 Measurement and Metrics

For years now, humans have been measuring things such as distance, time, velocity, temperature, etc. and the software industry is not an exception. For Information Technology (IT) managers, it is important to measure how much or how long a project will take, as well as different aspects of the development life-cycle or team's performance. On the other hand, for software developers it is important to have predictable and repeatable development processes that allow teams to shift the software to production in a controlled manner at any time [14]. That is why they began to measure things like cost, effort, defect rates, lines of code (LOC), pages of documentation, among others [63]. But despite the efforts, the delivery of software is generally not reliable, this produces projects that are often delivered late, with a reduced scope of features.

Despite the fact that measuring different variables during the life-cycle of a development seems like an easy task, it is difficult to do it right because there are so many variables implied during the process that it can be hard to select the right ones to measure. If to time and budget are added other variables such as the methodology (e.g. Waterfall, Scrum) used to develop and/or the type of delivery to the customer (i.e. annually, quarterly, monthly, weekly or even continuous delivery), the selection of the variables to be measured becomes a really complex work. Although it can be a hard task to determine what to measure, it is necessary because the measurements provide a full picture of what is happening in a project, or within a team. Further, measurements can provide insight on how changes in aspects such as team's composition, used tools, methods, or methodologies, affect the team's performance. This information is useful not only for developers in a team, but also for managers who want to know what to change to keep improving his or her teams.

While *measurement* is a quantitative observation of a specific attribute, a *metric* is the result of taking a certain measurement recurrently [45]. It somehow needs to have informational, diagnostic, motivational or predictive power, so that it can help in understanding how far from the expected results a project is or what is the impact of a change. In software development, metrics have the main purpose of directing the work towards a goal by showing how the actual performance differs from the expected one, and guiding process improvements by showing the variability in the performance of a process after changes in practices or methods used. As it was settled, it is simple to collect data, even though there are related costs and efforts while tracking metrics. This is the reason why each metric should be justifiable and have a purpose. In other words, a metric should be pragmatic and provide useful information to stakeholders that have to take decisions.

According to Nicolette [45], the effects produced by a metric allows to classify them in three categories:

- Informational: the metric provides plain information.

- Diagnostic: the metric calls attention to a problem.
- Motivational: the metric influences people's behavior.

There is a big number of metrics that can be grouped by the methodology applied during development, as well as some metrics that are not attached to a methodology in particular. Because of this wide range of possibilities, it can be difficult for a team or a manager to select a reduced set of metrics at the beginning of a project, among all the metrics available. Despite this situation, it is highly recommended to revise and adapt periodically the set of metrics chosen.

According to Rod Stephens [63], the characteristics that any metric must have are:

- **Simple**: the metric must be easy to understand.
- **Measurable**: it must be possible to measure the attributes that conform a metric.
- **Relevant**: the metric must provide useful information that leads to actions.
- **Objective**: the results must come from objective data rather than subjective opinions.
- **Easily obtainable**: it must not take too much time to team members to gather data.

1.1.1 Measurement of Scrum

Scrum is an Agile framework that allows companies of all stripes (e.g. Information Technology (IT), Product Development, Operations, Finance) to innovate and remain competitive [1]. This framework has a team-based approach to delivering value to business frequently in short periods of time that goes from one week to four weeks. This approach provides a faster feedback to the teams, giving them the chance to adapt to changes more quickly.

But not all the teams that start trying this framework have successful experiences with it. One possible solution is to obtain metrics that can help in having a perspective on how well a Scrum Team is performing, or what difficulties a team has in order to increase its productivity.

Because of these reasons, Agile and Scrum are two core concepts studied in the current work. This will be explained in depth on Chapter 2.

1.2 Research Problem

In light of the above discussion on measurements and metrics, the research problem that this thesis aims to address is how to measure the performance of Scrum Teams more effectively. Usually, a metric program is expensive and time-consuming, but without it, it is difficult to have evidences that support decisions taken in a project or an organization. Metrics give quantitative insight about performance and provide measurable goals. But to be successful with metrics, it is necessary to measure things that can be measured and can lead to different actions. It is futile to measure things just because they are easy to measure, or to have a metric that does not give insight about the impact that a decision or an action had.

Measuring is a good action to be carried out in software development, but if it is not done in right way, it can lead to different problems. One of them is that sometimes managers try to measure a lot of variables, generating a lot of information. This massive amount of information causes the teams and managers to lose focus and they end working on weaker aspects. Another problem is that when the wrong variables are measured, a manager will not notice that something is going wrong in the development till it is really late. Also there is the possibility that the developers adapt their behavior according to what is measured by the managers to give them the impression that they are performing well. Another aspect to consider is that the collection of data should not cause interruptions or slow down the team's work.

In order to succeed with a measurement program, it is important to establish the goal of the metric program at its very beginning. This will allow to have a clear view of the purpose of it and avoid information overload.

1.3 Objectives

Following are the main objectives that this thesis aims to achieve:

- Investigate about useful metrics for working environments where the Scrum framework is applied.

- Provide a software solution that facilitates the measurement procedure, through the integration and implementation of different tools.

1.4 Outline

The remainder of this document is divided into the following chapters.

Chapter 2 and Chapter 3 while providing a detailed explanation on Agile teams and Scrum and Metrics for Scrum teams uses the existing literature and material on the topic to provide the basis for the current thesis as a progress over the already available material. These two chapters therefore, establish the theoretical framework for the thesis.

In addition, Chapter 4 discusses existing surveys with an additional survey conducted for the thesis to corroborate and add to the previous studies.

Chapter 5 provides for the analysis of available tools for software project management and their measurement. This chapter will clearly detail the tool chosen as result of the analysis, and describe the implementation of the solution for generation of useful metrics for better environments for Scrum Teams. Thus, the main aim of the chapter is to elaborate the architecture of the application developed, metrics obtained and results of the implementation.

Finally, Chapter 6 concludes the work by reviewing its main contributions, and possible future work that can be done.

Chapter 2

Agile Teams and Scrum

Even though the Agile movement started at the beginning of the XXI century, it is only today that the companies have realized that they need to adapt quickly to the changes to remain competitive in the volatile market. While a mere quick adaptation will not guarantee success, it is a good way to achieve the main goal of any company, that is to have happy customers. And one of the best ways to go along this path is incorporating the Agile values and principles through at least one of its mature set of methodologies.

According to a 2016 survey known as the 11th annual State of Agile Report, sponsored by the company VersionOne [1], respondents say that the top five reasons for adopting agile are: Accelerate product delivery, Ability to manage changing priorities, Increased team productivity, Improved project visibility, and Enhancement of software quality. And despite the fact that 98% of the respondents' organizations have realized success from agile projects, more than 50% of their teams are not practicing agile [54]. This is due to some challenges that organizations experience while adopting agile such as company philosophy (or culture) does not match with agile values, lack of experience on agile methods, lack of management support, among others.

Another interesting fact that can be pointed out from the report is that since at least three years Scrum is the most used methodology by respondents' organizations, with more than 50%. And the second most used is a hybrid between Scrum and XP, with 10%.

The present chapter will introduce the necessary concepts to understand

[1]https://blog.versionone.com/insights-from-11th-annual-state-of-agile-report

agile and its philosophy (Section 2.1), and Scrum, the most used framework to adopt agile (Section 2.2). These concepts will set the basis for the Section 2.3 and the Chapter 3.

2.1 Agile Concepts

Agile can be defined as a *mindset* and a set of *methods* and *methodologies* which provide a way to think more effectively, work more efficiently and make better decisions. And it also makes the following promises [62]:

- Projects delivered on time.
- High-quality software.
- Code well constructed and highly maintainable.
- Happy users.
- Developers working normal hours.

Even though the adoption of Agile can seem a safe bet, it is important to remark that it is not only the adoption of practices or techniques, but a full change of mindset that can lead to a complete overhaul. If there is a fractured perspective in the team, meaning that if each team member cares only about his or her job, the results may be better but they will be far from the promised ones.

The mindset shift is about sharing the same information and a feeling of ownership among all the members of a team. This can be achieved by opening up planning, design, and process improvement. Also it implies that each person's opinion matters, because if one of the members does not agree with the way that the team will work, it can drop the performance of the whole team.

2.1.1 Values and Principles

The ideas implied in the agile culture differ completely from the ones in waterfall process, in which at the very beginning of a project all that is needed at the end of it, is known. In a waterfall process, the requirements must be well planned and defined upfront in a detailed document called specification.

This document is given to the developers who will develop what they interpret about what is written. After development comes a testing phase, and, finally, the product is delivered to the customers. But it is very likely that there were misinterpretations or misunderstandings about the requirements, or that the needs of the customer have changed since they settled the requirements. In this situation the team has to start the entire process again.

The agile mindset is the result of the ideas, principles and values established in the Manifesto for Agile Software Development (also known as Agile Manifesto) [5], created in 2001 by a group of people who wanted to change the way in which software was being developed. In this document, they wrote four core values:

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

The authors of this document do agree that there is value in the items on the right, but they value more the ones on the left.

With these concepts in mind, it is possible to interpret the ideas present on the values as follow:

- the ideas, concerns, motivations, interrelations and communication among the people who are working in a team are more important than the tools or processes that will be used;
- a team should concentrate on delivering software that adds value to the organization instead of writing complete and comprehensive documentation, but this does not mean that a team should not document;
- a customer should be involved with the team in the software development process;
- and a team needs to make sure that it responds appropriately any time there is a change in the needs of the customer, or the way the software needs to be built.

The signatories of the Agile Manifesto created 12 principles for agile software development [2], to support the above-mentioned values and to motivate

[2]https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

teams to build software that the users actually need. The principles are the following:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The main purposes of these principles are: to enhance the software delivery, the communication both among team members and with customers, as well as the execution of the software development during its whole life cycle, and to be constantly improving both the project and the team.

2.1.2 Methodologies

Agile methodologies are a collection of practices, ideas, advice and a body of knowledge [62], constructed on the basis of the agile values and principles, to help to adopt them. They also help to see the practices in context, that allows novel agile practitioners to get in touch with agile concepts more easily and quickly.

Some of the most known methodologies are: Scrum, it is an iterative and incremental way to produce software that will be described deeper in the Section 2.2; eXtreme Programming (XP), in which teams produce deployable software every week using specific development practices that help to keep the software simple and maintainable [4]; Lean, is a mindset with its basis on seven principles drawn from the principles of Lean Thinking, the main idea behind these principles is to let customers take decisions about what they want as later as possible, and when they want something give it to them as soon as possible [51]; and Kanban, this is a simple but powerful agile method with the main idea on limiting the Work In Progress (WIP) done on each stage of the software development process, this provides transparency [27].

2.2 Scrum concepts

Scrum was born in 1993 at the Easel Corporation as a structure for iterative, incremental software development framework, and nowadays it is the most used methodology by small and large companies trying to be agile, all over the world.

According to Jeff Sutherland, one of the fathers of Scrum together with Ken Schwaber, Scrum is an agile method designed to add energy, focus, clarity, and transparency to project planning and implementation [64]. In The Scrum Guide of 2016, Ken Schwaber and Jeff Sutherland define Scrum as a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [65]. This framework consists of Scrum Teams with associated roles, events, artifacts, and rules. Each of these components has a specific purpose that makes impossible to succeed with Scrum if they are not well employed and considered essentials.

The three main pillars of Scrum are transparency, inspection and adaptation. They come to life when the values of commitment, courage, focus, openness and respect are shared and lived by the Scrum Team [32].

2.2.1 Scrum Teams

The Scrum Guide [65] establishes three main roles for a Scrum Team as follows:

- *Product Owner:* is the person responsible for maximizing the value of the product and the work of the Development Team throughout the management of the Product Backlog. Some of the management tasks include: clearly expressing Product Backlog items, ordering them to best achieve goals, ensuring that is visible and the Development Team understands its items.

- *Development Team:* is the group of professionals who do the work of delivering a potentially releasable Increment of the product at the end of each Sprint. The people in the team must have cross-functional skills in order to be able to create a product Increment. And only they can set how to turn the Product Backlog items into Increments. An important aspect about a Development Team is its size, a rule of thumb is to have a team of 7 ± 2 members, without including Product Owner and Scrum Master.

- *Scrum Master:* is responsible for ensuring that Scrum is understood and all adheres to its theory, practices, and rules. She is also a servant-leader who helps the Product Owner (e.g. finding techniques for effective Product Backlog management) as well as the Scrum Team (e.g. removing impediments to the team's progress) and also the Organization (e.g. planning Scrum implementations within the organization).

The above-mentioned roles are designed to optimize flexibility, creativity and productivity. There are no Product or Project Manager roles. The self-organization allows team to choose how best to accomplish their work, while with the cross-functionality a team does not depend on people that are not part of it to accomplish the work.

2.2.2 Scrum Events

Events in Scrum provide opportunities for inspection, adaptation, and transparency. They all have a maximum duration, that is why they are called time-boxed events. Next there is a list of the events and a brief description of them [65]:

- *The Sprint:* is a time-box of four weeks or less, in which the Development Team creates a usable and potentially releasable product Increment. This helps limiting the risk to a maximum of one calendar month of risk. It has to be considered that once a Sprint starts, it cannot change its duration. Immediately after a Sprint finishes, the next one starts, this is the reason why Scrum is iterative and incremental. The Sprint is the heart of Scrum because it contains all the other events.

  It is recommended for productivity reasons to keep the Development Team focused on one Product for one Sprint, and avoid multitasking across multiple applications.

- *Sprint Planning:* this events occurs at the beginning of each Sprint, where the entire Scrum Team plans the work to be performed in that Sprint. Its time-box is of a maximum of eight hours for a one-month Sprint, with a rule of thumb of two hours per week that the Sprint lasts. Usually it is divided in two parts. In the first part of the meeting, the Development Team and the Product Owner review the Product Backlog, the first forecast the functionality that will be developed during the Sprint, while the second gives context to the items. Finally, both of them set a Sprint Goal, which is an objective to be met within the Sprint through the implementation of Product Backlog, and provides guidance to the Development Team.

  In the second part of the Sprint Planning, the Development Team designs the system and breaks the Product Backlog items down to smaller pieces called Tasks, which have a duration of a day or less. The list of tasks is recorded in a document called the Sprint Backlog.

- *Daily Scrum:* also known as the Daily Stand-Up Meeting, is a time-boxed event of fifteen minutes held at the same place and time each day. It helps to synchronize activities and create a plan for the next 24 hours, presenting the information needed to inspect progress. Holding this meeting each work day has the following benefits: improvement in communications and the Development Team's level of knowledge, elimination of other meetings, identification of impediments to development for removal, and promotion of quick decision-making.

- *Sprint Review:* this event is held at the end of a Sprint to inspect the Increment and adapt the Product Backlog if needed. The main intention of this meeting is not to give status to the stakeholders, but elicit feedback from them and promote their collaboration. The result of the Sprint Review is a revised Product Backlog for the next Sprint. This is a four-hour time-boxed event for one-month Sprint, so it is possible to set a rule of thumb of one hour for each week that the Sprint lasts.

  The Sprint Review involves inspect and adapt regarding the *product*.

- *Sprint Retrospective:* in this meeting, the Scrum Team inspects itself regarding to people, relationships, process and tools. As a result, they agree on what is working and create a plan to improve in the next Sprint the way the Scrum Team does its work. This event is a three-hour time-boxed meeting for one-month Sprints, and it is usually shorter for shorter Sprints.

  The Sprint Retrospective involves inspect and adapt regarding the *process and environment*.

The events can be summarized as follow in the table 2.1:

Event	**Max Duration**	**Participants**
Sprint	4 weeks	SM, PO, DT
Sprint Planning	2 hours per week of Sprint	SM, PO, DT
Daily Scrum	15 minutes each day	SM, DT
Sprint Review	1 hour per week of Sprint	SM, PO, DT, SH
Sprint Retrospective	45 minutes per week of Sprint	SM, PO, DT

Table 2.1: Scrum events summary. SM = Scrum Master, PO = Product Owner, DT = Development Team, SH = Stakeholders.

2.2.3 Scrum Artifacts

The Scrum's artifacts are a mechanism to provide transparent information to the Scrum Team as well as any stakeholder or even the organization [69]. The information provided by the artifacts creates opportunities for inspection and adaptation. The artifacts are defined as follow [65]:

- *Product Backlog:* this is an ordered list of everything that might be needed by the product (i.e. functions, features, fixes, etc.), and which is constantly evolving with the product Sprint after Sprint. The Product Owner is the responsible for its content, availability, and ordering. The Development Team is responsible for the estimates of the items on the Product Backlog. The Scrum Team as a whole is responsible for refine the Product Backlog adding details, estimating and ordering its items.

  Those items that are on the top of the list, have enough detail and can be "Done" by the Development Team within a Sprint, are deemed "Ready" for selection in a Sprint Planning.

  The definition of "Done" is an agreement that conforms to an organization's standards, conventions and guidelines. This document defines all the activities that are needed for creating a Potentially Shippable Product Increment.

- *Sprint Backlog:* this is the set of Product Backlog items selected for a Sprint, plus a plan with the work needed for delivering the Increment and meeting the Sprint Goal. The Development Team is the only one who can add or remove items from a Sprint Backlog during a Sprint. This artifact provides a highly visible, real-time picture of the work that the Development Team plans to accomplish during a Sprint.

- *Increment:* this is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. It must be usable and meet the Scrum Team's definition of "Done", this is the standard that a team has to accept an item as a complete one, this will expand as the Scrum Team matures. Each increment is additive to all prior Increments, so they must be able to work together.

The roles, events and artifacts of Scrum are summarized in the Figure 2.1:

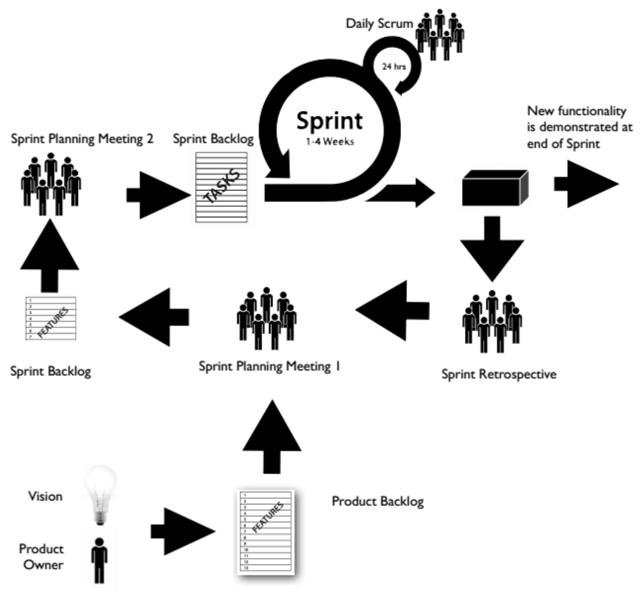


Figure 2.1: Scrum overview. Source [64]

2.3 Conclusion: Agile teams practicing Scrum

A team in which there exist only the roles listed in Section 2.2.1, drives and respects the time-boxing of each event in Section 2.2.2 and builds all the artifacts presents in Section 2.2.3, is called a Scrum Team.

These teams are characterized by self-management, a very visual way to see how a project is going throughout different graphics (e.g. task board), and a quick adaptation to changes, despite the seniority of its members.

Nevertheless, all of this does not guarantee the success of the team if there is no real commitment with the project, trust among its members, or they do not agree with the methodology or the agile values.

Chapter 3

Metrics for Scrum Teams

As it has been depicted in Chapter 2, there are different promises that the Agile software development approach makes, but this approach (and its methodologies) is not exempted from the inconveniences present in any software development process, even if the philosophical barrier related to the mindset change is overcome. Since it is pertinent to know how well the agile process is applied in a team, or how happy the customers are with the product, it is necessary to measure different aspects during the software life-cycle.

According to the 11th annual State of Agile survey [54], the top five measurements where agile practitioners focus to evaluate the success of their agile initiatives are the followings:

- On time delivery (53%).
- Business Value (46%).
- Customer/user satisfaction (43%).
- Product Quality (42%).
- Product Scope (40%).

On the other hand, it is different how the success of an Agile Project is measured, as it is possible to observe in the following top five:

- Velocity (67%).
- Iteration burn-down (51%).

- Release burn-down (38%).
- Planned vs. actual stories per iteration (37%).
- Burn-up chart (34%).

In the above-mentioned ranking, the business value delivered is in the 11th position with a 23%.

These differences can be due to many reasons such as a weak metric plan, management and teams not sharing the same objective, or that the management asked the team to adopt agile values but it did not yet. In order to avoid these kind of situations, the management needs to set a concrete plan to collect the metrics in which the importance, purpose, cost, target, etc. of each metric is well established. This is needed to have a clear vision of the goal pursued by the organization and avoid information overload.

In the following sections the key concepts to select meaningful metrics will be established (Section 3.1), classify them according to different aspects (Section 3.2), define useful metrics for Scrum teams (Section 3.3), and, finally, a set of metrics that have to be avoided will be presented (Section 3.4) .

3.1 When is a metric useful?

According to Nicolette [45], a metric that helps a stakeholder to make a decision is a pragmatic one. This idea is independent of the approach applied to develop software, because for a plan-driven approach metrics will be used to track team's performance compared with a plan which has a fixed budget, scope and schedule. For an Agile approach, metrics will be used to assess whether the scope, schedule, or budget has to be adjusted to keep work on track. In addition to the general approach, the author proposes that the process (i.e. Linear, Iterative, Time-boxed, Continuous flow) and the delivery mode (i.e. discrete project or ongoing development and support) have to be considered.

Hartman and Dymond [23] distinguish between metrics and diagnostics, former being defined by the organizations and the latter, determined by the teams. To support their selection, they give a list of ten items that a good Agile metric (or diagnostic) needs to have:

1. **Affirms and reinforces Lean and Agile principles**. The metric should be consistent with the values and principles seen in Section 2.1.1.

2. **Follows trends, not numbers**. Numbers by themselves do not have any meaning, but when they are observed throughout the historical data, and giving them a context, they become a valuable tool.

3. **Belongs to a small set of metrics or diagnostics**. If a metric is general or has multi purposes, it is likely that it does not meet expectations of any of the objectives for which it was created.

4. **Measures outcome, not output**. Output can be defined as any physical or virtual product, while an outcome is the benefit that customers receive from these products [1]. Because the aim of Agile methodologies is to give customer values, a metric that provides information about outcome will generate information that is useful not only for Scrum Teams, but also for the customers.

5. **Is easy to collect**. The collection of metrics should not interfere with the work of the developers, they do not have to feel that they are wasting time while providing information.

6. **Reveals, rather than conceals, its context and significant variables**. Agile methodologies promote openness and trust among team or organization members, this only can be achieved through sharing information with people that may or may not be involved directly in a project.

7. **Provides fuel for meaningful conversation**. A metric that does not generate any discussion between organizations members is not working well. When there is trust among team members, it is possible for them to have discussions regardless their position nor their experience. This generally leads to improvement regarding not only the people involved in the discussion, but also their organizations or teams.

8. **Provides feedback on a frequent and regular basis**. It is probable that when a person asks for a metric, at that moment it is necessary to act of urgency. The metrics should be available every time a person asks for it.

9. **May measure Value (Product) or Process**. According to Pressman, measurements can be applied to the software process to improve

[1] https://hbr.org/2012/11/its-not-just-semantics-managing-outcomes

it continuously, or can be applied to a project (or product) to adapt the workflow of a team. The metrics that measure the process have strategical purposes (e.g. release frequency), while the product metrics have tactical purposes (e.g. estimation of time or costs, reduction of risks) [53].

10. **Encourages "good-enough" quality**. In every software development process, quality is a key value that allows an easy maintainability of a system.

The above-mentioned elements provide the management and teams with a unified criteria to establish good metrics on their metrics plan.

3.2 Classifying Metrics

In any agile program it is important to track both business metrics and agile metrics. While business metrics focus on whether the solution is meeting the market need, the agile metrics measure aspects of the development process. It is also valuable for Scrum Team's adaptation and improvement to have a success criteria which guides the team on its future steps.

The bibliography about metrics classification offers a wide range of different perspectives in which different aspects such as target audience, information provided, area of usage, etc. are taken into account to categorize the metrics. The current section will discuss some points of view that will help in the following sections to better understand the metrics and their implications.

3.2.1 What type of variable do they measure?

The first criteria that can be used to make a distinction between metrics is also used to distinguish variables. In this, they are divided in two groups: qualitative (e.g. value delivered last sprint, team's progress in adopting agile techniques and processes) and quantitative (e.g. Lead Time, Cycle Time, number of defects escaping to production) metrics.

While the first ones provide better insight for teams, the last ones do so for the organizations. Despite of the type of metrics collected, teams and organizations should pay more attention to the trend than the absolute number showed by them because the trend will tell the effect of the changes applied to improve a practice, technique, or process.

3.2.2 What information do they provide?

Eric Ries defines two types of metrics: vanity metrics and actionable metrics [55]. Actionable metrics are those in which the causes of their increment or decrease are clear, and with this knowledge, the teams can replicate results. Besides, metrics must be accessible, this means that reports must be as simple as possible to make them understandable and reachable by everyone who need them. Furthermore, it is essential for good metrics to be "auditable", which means the data employed to generate the metrics must be credible and the results obtained can be tested in the real world without the tool that provided the metrics.

All the other metrics are vanity metrics (e.g. gross number of costumers), because they do not give insight about why something is happening and this can lead to misconceptions about what actions are needed for improvement.

3.2.3 Do they look at the present or the past?

Mike Cohn introduced the concepts of leading and lagging indicators [10]. A lagging (or trailing) indicator is something you can measure after a series of actions have been executed, and can be used to determine if a goal was achieved, e.g. the number of defects reported in the first 30 days after a release.

On the other hand, a leading indicator is available in advance and can tell a team or a project manager if a goal is likely to achieve. An example of a leading indicator for a team that wants to improve its quality could be the number of nightly test that pass.

A similar way to classify metrics was used by Nicolette [45] who called traditional support metrics as backward-facing metrics because it is needed to face the past in order to see the target. Similarly, the forward-facing metrics to those that support adaptive development, because in order to see the target, it is needed to face the future.

3.2.4 What are they used for?

Kupiainen et al. [30] classified the metrics according to the reason they were used in the following categories:

- Iteration Planning: here the metrics were focused to help in prioritization of tasks or features, forecast effort estimation, or Development Team's velocity.
- Iteration Tracking: here metrics can be found to monitor, identify problems, or predict results, as well as progress metrics that are useful to focus work on tasks that are really important.
- Motivating and Improving: the metrics in this section are used to motivate people and support team level improvement of practices/processes.
- Identifying Process Problems: in this section the metrics are often used to avoid problems in processes and workflows.
- Pre-release Quality: metrics in this category are used to prevent defects reaching customers and to understand the current quality of the product.
- Post-release Quality: it consists of those metrics that allow to evaluate the quality of the product after it has been released.

They also analyzed how the metrics in each category support the twelve principles of Agile Manifesto seen in the previous chapter (2.1.1).

3.3 Scrum Metrics

The Scrum Guide [65] establishes the rules that govern the Scrum framework as seen in the previous Chapter (2.2). And also it sets two things to monitor, but it does not define a specific way to do so, giving to Scrum Teams the freedom to choose their own methods.

The first thing to monitor is the total work remaining to reach a goal (progress towards a goal) which should be controlled at least every Sprint Review, and be available to any Stakeholder.

The second one is the total work remaining in the Sprint Backlog, which should be tracked at least at every Daily Scrum, so that the Development Team can manage its progress.

The current section will show the common metrics used by Scrum Teams as well as the metrics which different authors have proposed in order to track different aspects that could help teams and management to improve products or processes.

3.3.1 Burn-down

A way to show progress in a Project for either a Sprint or a Release is through the Burn-down chart (see Figure 3.1). It is a graph that consists of two axis in which the horizontal axis represents time, and the vertical axis represents the amount of work remaining (usually measured in Story Points or ideal engineering hours). Then a line is drawn to connect the dot in the vertical axis that represents the amount of work addressed by the team in the current Sprint, and the dot in the horizontal axis corresponding to the last day of the Sprint. This line is useful to check if the team is going to have problems to reach the goals of a Sprint or a Release. The difference between a Sprint and a Release Burn-down is that while in the first chart the unit of time is days, in the latter chart the time is measured in Sprints.

As the Sprint advances, each day the Scrum Team adds a dot corresponding to the amount of Story Points remaining in the Sprint Backlog. The chart will be modified only when any Sprint Backlog item have reached the team's definition of Done.

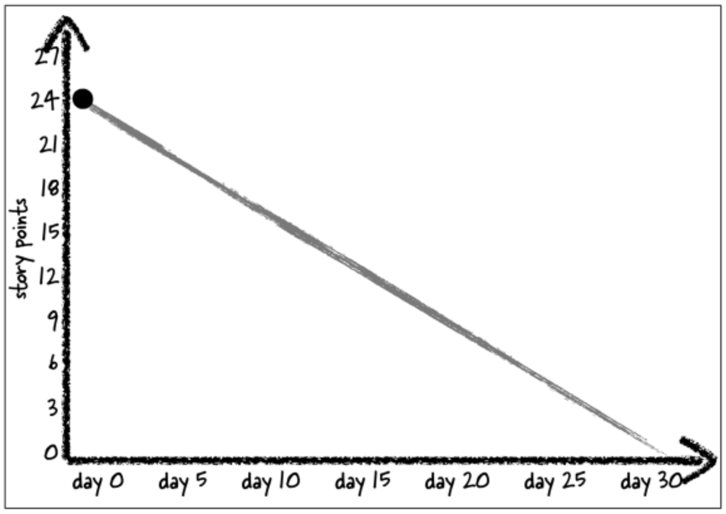


Figure 3.1: Example of Burn-down chart. Source [62]

This graph allows the whole team to realize when they have committed to more/less work than they could deliver, or what are the impediments to making progress. And it should be used as a debating tool to make trade-offs or improve the communication within the team or with other sectors of the organization, instead of being used as a command and control tool.

A variant of this chart is known as the burn-up chart, which shows the accumulated work Done by day.

3.3.2 Velocity

The velocity is used as a forecast measurement. This variable is calculated as the average of the Product Backlog items really Done in previous Sprints, and it represents the amount of work that a team is likely to address in the next Sprint. The units used to measure this variable are the same that are used to estimate the items in the Product Backlog. Usually teams measure those items in Story Points, they can either represent how many work days or hours a User Story will take to develop, or be used as an abstract comparative measurement among User Stories. This method of measurement is inherent to each Scrum Team, which means that it varies from a team to another. As a consequence of it, this variable cannot be used to compare teams.

There are some factors that make a team's velocity volatile such as incorporation of new team members, team members leaving, seniority levels, holiday or sick leave, working unknown territory, among others. Even then, it is important for a team to know its velocity because they can realize how a change in a process or a practice impacts on its velocity. This variable also helps to predict how quickly a team can work through the Product Backlog, giving an idea of how many iterations will be needed to complete the required work, but it does not give any information about the progress or success of a Project. It should neither be used as a driving nor an agile adoption metric, because it may either cause the team over-estimates the User Stories or a decrease in the quality of the software produced with the consequent slow down in the value delivered to the customers. On the other hand, a team with excellent velocity could spend time delivering software that the customer does not want or has been waiting for.

Managers must consider that it is expected for a stable team on the same project with the required resources to gain velocity during the course of the Project until it stabilizes.

3.3.3 Productivity

The productivity of a Scrum Team can be measured in different ways. The use of Story Points or ideal days to do this task is not recommended because it can lead the team to inflate its estimates if they are under pressure to improve or deliver more points per Sprint. A better way to do this is through the number of Product Backlog items delivered or the percentage of Product Backlog items completed versus planned at the end of a Sprint.

From a management perspective this can be measured as the division between the amount of Story Points completed in a Sprint and the budget that it costs.

3.3.4 Lead Time and Cycle Time

Two interesting metrics that are used by Kanban teams but can be beneficial for Scrum Teams as well are Lead Time and Cycle Time. The Lead Time is a measure of how long it takes for an item to be delivered to the customers from its creation [52]. A Scrum Team can measure its responsiveness by tracking the time from when an item enters in the Product Backlog until it either comes out of an iteration or is delivered to the customer. Using Mike Cohn's classification (see 3.2.3), it can be said that as a Trailing Indicator historical trends of Lead Time give a way to understand how long it is likely to take to deliver similar work items. While as a Leading Indicator it can be used to indicate the likely duration of the end-to-end flow of work through the system, it is necessary to consider that Lead Time by itself does not help to visualize the causes and impacts of variability within a team or an organization.

The Cycle Time measures the total time from "In progress" to "Done" of all the items in a Sprint (i.e. tasks, User Stories), and also the time spent by the items on each workflow state. As a Trailing Indicator a Cycle Time analysis of historical trend data shows where work items are getting stuck. While as a Leading Indicator, it can be used to predict likely duration of similar work items within a range of possibilities.

For any team, its goal should be to have a short and consistent cycle time, which will translate into higher throughput and more predictability. This measurement is an efficient tool to see almost immediately how a change in team's process impacts on it. With these definitions in place it can be inferred that the Lead Time is what is relevant from a business perspective, because it gives insight into the customer's experience; while Cycle Time is a measure of team's process capability, helping to identify impediments and improvements opportunities within an organization.

3.3.5 Value

Metrics should help validate business investments and the value delivered by teams. Some key metrics of interest to the business are the amount of capital required and the return on investment (ROI), which can be captured with Value. Value is defined as software put into production that can return an investment over time [23].

There are many ways to measure business value delivered such as Net Present Value (NPV), Internal Rate of Return (IRR), or Return on Investment (ROI). In an agile project, the ROI begins since the software is released into production for first time. It is necessary to define if Value will be accounted at the close of each Iteration, or at the time of release to production, the last option being the most recommended one.

3.3.6 Quality

Monitoring software quality is a non-trivial task because it cannot be performed in the same way across different projects. It is a process that requires the selection of relevant quality attributes which are project specific, and the selection of appropriate metrics to quantify these attributes. Software quality metrics aim to measure how "good" a software is, from the point of view of being error-free, and easy to modify and maintain.

In their study, Arvanitou et al. suggest that the same metric can be used for assessing more than one attribute. The following list shows some of the most used quality attributes and the metrics associated to them [3]:

- Maintainability:
  - Depth of Inheritance Tree (DIT).
  - Lines of Code (LOC).
  - Weighted Methods per Class (WMC).
  - Cyclomatic Complexity (CC).
- Re-usability:

  - Lack of Cohesion of Methods-1 (LCOM1).
  - Lines of Code (LOC).
  - Coupling Between Objects (CBO).

- Change proneness:
  - Depth of Inheritance Tree (DIT) .
  - Number of Children (NOCC).
  - Coupling Between Objects (CBO).
- Understand-ability:
  - Lines of Code (LOC).
  - Depth of Inheritance Tree (DIT).
  - External Class Complexity (ECC).
- Test-ability:
  - Response for Class (RFC).
  - Coupling Between Objects (CBO).
  - Lack of Cohesion of Methods-1 (LCOM1).
- Modify-ability:
  - Depth of Inheritance Tree (DIT).
- Stability:
  - Weighted Methods per Class (WMC).
  - System Design Stability (SDI).

Concas et al. based their study on some of the above-mentioned metrics (present in the Chidamber and Kemerer suite) applied on a single software development [12]. They found that there is a clear correlation between good Agile practices and an improvement of the quality metrics. But Destefanis et al. tried to verify this and they investigated if there was a relationship between software metrics obtained from open source software developed using Agile methodologies and plan-driven methodologies. They conclude that the use of Agile methodologies does not influence the distribution of the measured metrics in the classes, this means that metrics distributions are similar for software developed using Agile methodologies and software developed using plan-driven methodologies [15].

These totally opposite results open the door to other questions such as Does the software that has been developed use Agile methodologies from its very beginning? Do teams know and believe in the Agile values? Are teams skilled in Agile practices?

There is a large number of quality metrics that can be applied to Agile Software Development such as number of defects found during development or after a release, number of defects deferred to a future release, percentage of automated test coverage, release frequency, or delivery speed, among others. Alperowitz et al. propose the usage of some quality metrics to increase the manageability of a software engineering project course. The set of metrics proposed by the authors were based on good Agile practices present in the XP methodology, and they enabled the instructors of the course to have a high-level overview over the projects to recognize problems as early as possible [2]. They organized the metrics as follow:

- Merge Management:
  - Lifetime: the measure of the average lifetime of a merge request between the creation of the request and the point at which it was merged.
  - Workflow: this measures the percentage of merge requests with feedback (comments or tasks) in a week.
- Continuous Integration:
  - Time to fix: is the average period of time between a failed build and the first successful build on the team's main development branch.
  - Builds: the absolute numbers of successful and failed builds.
- Continuous Delivery:
  - Delivery to customer: the number of successful downloads of a release by customers.

  - Downloads: the number of times that a successful biuld was downloaded to a team member's device.

Another aspect related with the quality of the software is known as Technical Debt, which is a metaphor about the consequences of poor software development practices. The emphasis on quick deliver in Agile Software Development makes this way to develop software more prone to incur on Technical Debt, and if it is not strategically controlled, Scrum Teams will have to make a great effort into fixing defects or addressing stability issues [6]. As a consequence of this the team is likely to reduce its working speed and productivity, which will not only have a technical impact on the project but also an economical one. In order to measure the Technical Debt, it is possible to use code analysis tools which can measure things like percentage of repeated code, Cyclomatic Complexity, or number of lines of code in the largest class, among others. But the code analysis tools are not enough because most of the time the Technical Debt is not only related with the code, but also with architectural decisions or technological gaps.

3.3.7 Metrics for Hyper-productive Teams

Downey and Sutherland [16] say that Story Points, Burn-down chart and Team Velocity are lightweight tools that provide insufficient information. In order to develop and sustain Hyper-productive teams, the authors propose ten metrics that can be collected without slow a team down, and used to compare Scrum Teams with different reference scales. According to the authors, the following metrics clarify the impact of any modification (i.e. tools, technology, process, team composition), help the Scrum Team measure their own performance and make changes based on them, and are meaningful outside of their team of origin:

1. Velocity: is the sum of all the original estimates of the work that a team addressed in a Sprint.

$$\text{Velocity} = \sum \text{of original estimates of all accepted work}$$

2. Work Capacity: is the sum of all work reported during the Sprint, including the work done on non finished Sprint Backlog Items (SBI). This leads to a team to have a higher Work Capacity than its Velocity.

To track the work done over each SBI, the following questions are used in the Daily Scrum:

- What did WE achieve on the highest priority SBI that is not yet completed (Priority 1)?
- What was OUR contribution on Priority 1 worth in Story Points?
- What is OUR plan for completing Priority 1 today?
- What, if anything, is blocking US or has the potential to slow US down today?

This shift the focus of the Daily Scrum from the individuals to the Sprint Backlog, and improves the quality and the speed of estimation.

3. Focus Factor: is the relation between Velocity and Work Capacity, and it should remain in the neighborhood of 80% on average.

$$\frac{\text{Velocity}}{\text{Work Capacity}}$$

4. Percentage of Adopted Work: Adopted Work is work that is brought forward from the Product Backlog at any point during the Sprint because the team has completed the original Forecast early.

$$\frac{\sum \text{Original Estimates of Adopted Work}}{\text{Original Forecast for the Sprint}}$$

5. Percentage of Found Work: the Found Work is work associated with a piece of Forecast Work which is above and beyond what was initially expected but which must be completed to deliver the original work item.

$$\frac{\sum \text{Original Estimates of Found Work}}{\text{Original Forecast for the Sprint}}$$

6. Accuracy of Estimation: this metric refers to the team's ability to correctly estimate the body of work during Sprint Planning. This number should remain around 80% in healthy Scrum Teams. When this metric goes above 88%, on average, it is likely that the team is being overly conservative. And when it goes below 72%, on average, it can be due to many reasons such as poor understanding of the SBI, the Product

Owner is not available to the team during the Sprint, the team does not understand the technology or the product that they are building, among others.

$$1 - \frac{\sum \text{Estimate Deltas}}{\text{Total Forecast}}$$

7. Accuracy of Forecast: this metric refers to the team's ability to come together in the Sprint Planning, select and devote to themselves to a body of work that represents what they can achieve during the coming Sprint. This number also should remain around 80%. When it goes above 90%, the Scrum Master should evaluate if team is feeling safe forecasting work, because they can be under-forecasting to avoid reprisals. And when it goes below 75%, it is generally because the Scrum Master does not protect the team well during the Sprint or the team is heavily randomized.

$$\frac{\sum \text{Original Estimates}}{\sum \text{Original Estimates} + \sum \text{Adopted Work} + \sum \text{Found Work}}$$

8. Targeted Value Increase (TVI+): this metric measures how the team's velocity varies along a Sprint.

$$\frac{\text{Current Sprint's Velocity}}{\text{Original Velocity}}$$

9. Success at Scale: this metric gives teams confidence during Sprint Plannings, and it also helps them select the right granularity for User Stories.

   It groups the User Stories by their size according to the Fibonacci Scale (1, ,2, 3, 5, 8, 13, ...), and based on historical data it gives an idea how likely is a team to succeed with a User Story of a given size. For each point on the Fibonacci Scale (F_p), the formula is:

$$\frac{\sum \text{Number of Accepted Attempts of scale } \text{F}_\text{p}}{\text{Number of All Attempts of scale } \text{F}_\text{p}}$$

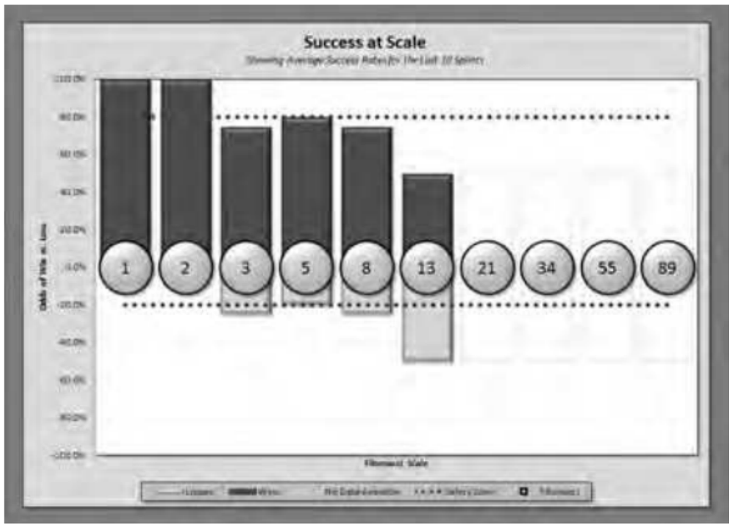


Figure 3.2: Example of Sucess at Scale. Source [16]

10. Win/Loss Record: each Sprint is a Win only if:
    (a) A minimum of 80% of the Original Forecast is Accepted by the Product Owner.
    (b) Found + Adopted Work During the Sprint remains at 20% or less of the Original Forecast.

3.3.8 Actionable Metrics

Vacanti and Vallet carried out a study at Siemens Health Services [67], in which they detailed how to shift from traditional Agile metrics (i.e. Velocity, Story Points) to actionable flow metrics (i.e. Work In Progress, Cycle Time and Throughput). The main reason to shift was that although Scrum and Agile practices improved different aspects such as collaboration across roles, speed, or code quality, a change was needed because the traditional Agile metrics did not provide to the company the required level of transparency to manage software product development at big scale. They established three metrics to understand the Company's flow of work:

- Work In Progress (WIP): any work item (e.g. User Story, Bug, etc.) that is between "Sprint Backlog" and "Done" steps.

- Cycle Time (see 3.3.4).
- Throughput: the number of work items that entered the "Done" status for an arbitrary unit of time (e.g. User Stories per week). This metric differs from velocity, which measures Story points per Iteration.

The above-mentioned metrics are inextricably linked through a simple yet powerful relationship known as *Little's Law*:

$$\text{Average Cycle Time} = \frac{\text{Average Work In Progress}}{\text{Average Throughput}}$$

These metrics are more transparent, because they provide a high degree of visibility into the teams' progress, and they are more actionable as well, because they suggest the needed team interventions to improve the overall performance of the process. And they provide improvements such as reduction of the Cycle Time, increase in the productivity and the overall predictability, along with a better process performance [68].

3.3.9 Metrics for Understanding Flow

In his study, Ken Power presents the following set of metrics that reveal how work flows in an organization [52]:

- Cumulative Flow Diagram (CFD): it is a chart that allows to visualize the amount of work in each of the defined workflow states, and trends over time. This type of chart is useful to understand the behavior in queues, and for diagnosing problems such as delays or other impediments. As a Trailing Indicator, CFD shows a history of how well a team's process is working for them. While as a Leading Indicator, CFD can be used to forecast the probability of either delivering a desired set of content by a given date, or the date by which a subset of the content might be delivered.
- Throughput Analysis with Demand Analysis: a Throughput Analysis (work items "Done") reveals the rate of the flow of work through the system over time. And in combination with a Demand Analysis, it shows how much work is Value Demand versus Failure Demand. As a Trailing Indicator, Throughput Analysis shows how much work is moving through teams or within an organization, and when it is combined with

Demand Analysis, it shows what type of demand is being placed on different teams. On the other hand, as a Leading Indicator, Throughput Analysis trends provide an indicator of where the demand will be in the near future, but it does not give the reasons why particular demands have impact in a system.

- Cycle Time and Lead Time (see 3.3.4).

The metrics provide an increased visibility throughout the organization about how they are working and the progress they are making.

3.4 Conclusion: Things to avoid

Some times there is confusion between measuring to deliver value (the objective of the Agile world) and data-gathering to document or justify the Agile approach in the wider world of methodologies. Different authors across the bibliography agree on tracking the Scrum Team as a whole and its context, rather than focusing on the individuals. A couple of examples of metrics that teams or management should avoid are Story Points per developer per time interval, or number of certified team members.

Also it is important to set that the metrics should never be used to blame individuals, firstly because in Scrum the entire team is responsible for the results of a project, and secondly this behavior may have negative consequences such as undermining the team's confidence to innovate, or diminish its transparency to the managers.

Chapter 4

Metrics Selection

In Chapter 3, a large list of metrics used by Scrum teams was shown along with a variety of points of view (i.e. business, performance, quality), was established. In this chapter, information related to metrics in Scrum environments will be shown, that have been gathered from different surveys, one carried out by the company VersionOne and the other carried out as part of the current thesis. The information will help in the reduction of the scope of the metrics that can be measured by Scrum teams and provide them better actionable information.

4.1 Supporting metrics selection

In order to support the election of metrics to be measured by the software tool which will be developed, the following data from the 11th annual State of Agile survey [54] were taken into account.

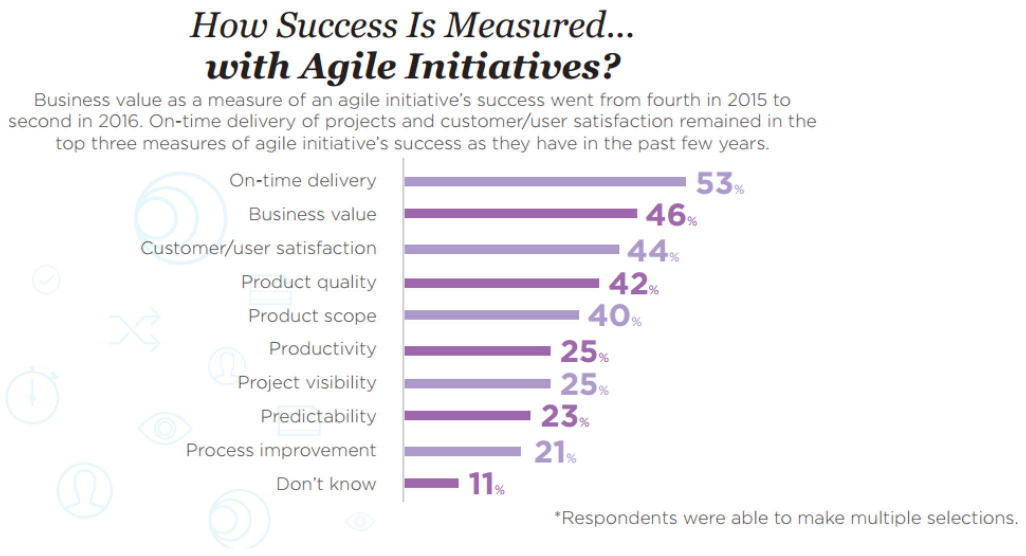


Figure 4.1: How success is measured with Agile Initiatives. Source: [54]

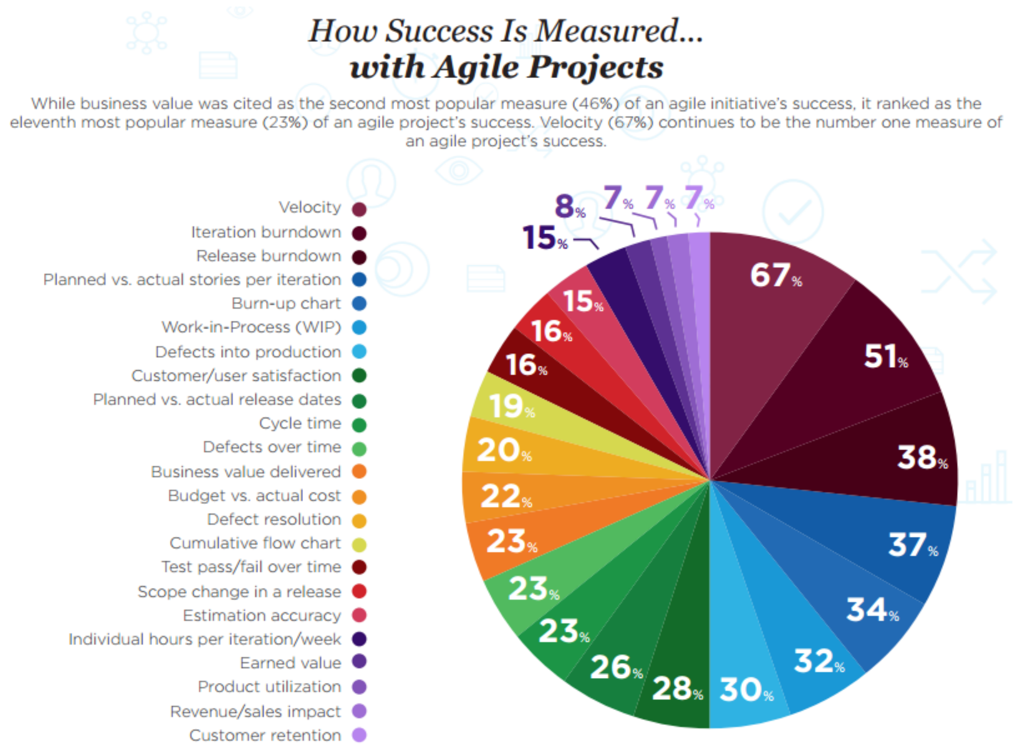


Figure 4.2: How success is measured with Agile Projects. Source: [54]

From the graphs in figures 4.1 and 4.2 it is visible that there exist two different levels of measurement of success on Agile, and each of them has different goals. One is at an *Initiative* level, which is more related to managerial positions in the hierarchy of an organization. In this level, different managers (i.e. business, portfolio, project) want to deliver the product within a defined budget and time, and also have happy customers who are more likely to promote the product. For these reasons, the top five metrics used by the management to measure how well an Agile Initiative is performing are on-time delivery, business value, customer/user satisfaction, product quality and product scope.

On the other hand, the main purpose of a team practicing Scrum is to deliver what their customers consider the most valuable functionalities, and they want to do this frequently in order to adapt quickly to any possible change. In this case, teams' top five metrics are velocity, iteration burn-down, release burn-down, planned vs actual stories per iteration, burn-up chart, which allow them to see how they are performing during a project.

4.2 Survey

In order to corroborate and expand the information exposed in Section 4.1, a survey was carried out. In the following subsections the survey will be detailed starting from the questions asked, continuing with where was it published, and finally depicting the results obtained.

4.2.1 Questions

Following is a list with the questions asked in the survey and their purpose:

- **Question 1:** *Do the teams in your organization use Scrum?*

  The respondents were able to choose one out of the following three options:

  1. Yes, all of them.
  2. Some of them.
  3. No, none of them.

  The purpose of this question was to corroborate whether or not the respondents were familiar with Scrum and its practices.

- **Question 2:** *What metrics do you use in your organization?*

  The respondents were allowed to choose more than one of the following options:

  - Velocity
  - Iteration burn-down
  - Release burn-down
  - Lead Time
  - Cycle Time
  - Cumulative Flow Diagram (CFD)
  - Net Promoter Score (NPS)
  - Internal Rate of Return (IRR)
  - Return of Investment (ROI)
  - Throughput
  - Quality metrics
  - Other/s
  - None

  The above-mentioned options have basis on the analysis done on Chapter 3, and which comprise different aspects such as work-capacity, work-flow, business, and quality.

  The purpose of this question was to have insight about which metrics are usually used within Scrum teams, regardless the level in which they are measured.

- **Question 3:** *What metrics do you think are useful?*

  This question had the same options than **Q2** (except for the last one, None), and it also allowed the respondents answer to it by choosing more than one option.

  This question has a purpose to check if the metrics that are being measured in **Q2** are really useful, or which ones the respondents think should be measured in a Scrum environment.

- **Question 4:** *Which metrics do you think are left in the previous questions?*

  In this open question the respondents were allowed to give their point of view about metrics that were not taken into account in the current survey.

It is important to remark that none of the questions were mandatory, and even then only one (1) out of the seventy-four (74) answers obtained was completely empty.

4.2.2 Making it public

This survey had as target audience people with a profile related with Scrum, its practices and philosophy. In order to obtain answers from people with certain knowledge about Scrum topic, a link [1] to the survey was posted on the following Scrum forums:

- The LinkedIn group "Agile and Lean Software Development": https://www.linkedin.com/groups/37631/37631-6278613219368009728

- The LinkedIn group "Agilidad y Calidad en el Software y los Sistemas de Información (CSSI)": https://www.linkedin.com/groups/2977088/2977088-6278615618174357505

- The Tuleap developers forum and its Slack channel: https://tuleap.net/plugins/forumml/message.php?group_id=101&topic=38994&list=1, https://tuleap.slack.com/archives/C4A086HEY/p1496996487659469

- The Slack channel of the LinkedIn group "Agile Clinic – Best Practices For Agile Change": https://hands-onagile.slack.com/archives/C2PUGTWRJ/p1496938300659933

In some of the above-mentioned forums, the survey served as a debate initiator where different techniques for measurement, types of metrics, among other topics were discussed.

[1] https://goo.gl/forms/bnGKOD6sepHKQj4A2

4.2.3 Results

In this section the results for the above-mentioned questions are presented. Because of the freedom that the respondents had to choose to answer any of the four questions asked, the number of answers can vary from one question to another.

The charts (figures 4.3, 4.4, and 4.5) that represent the results obtained in the survey, and which were automatically generated by the Google Forms platform (with exception of **Q4** which was an open question), are shown below.

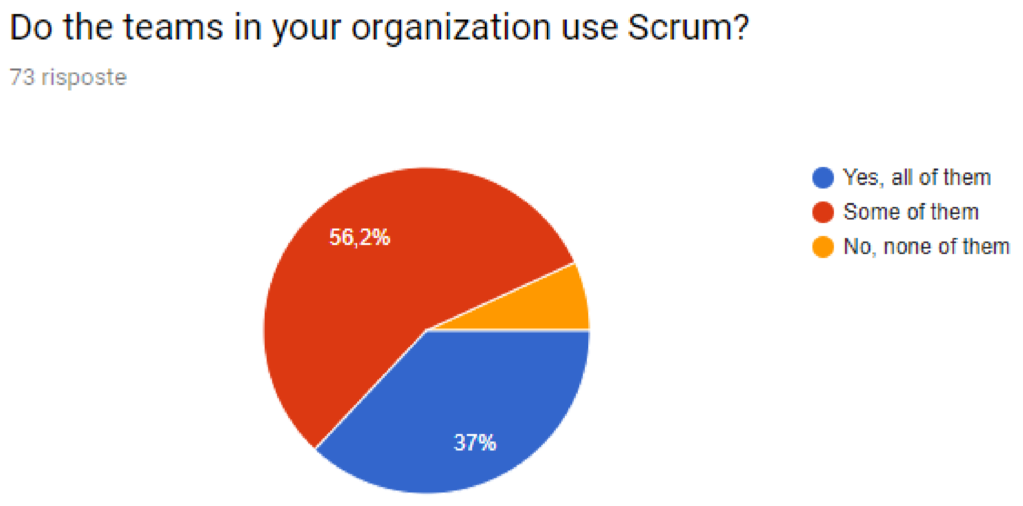


Figure 4.3: Results obtained for Question 1.

Here it is possible to see that the big majority (93.2%) of the respondents were members of organizations where Scrum is practiced by at least one team within them. And only 6.8% of the respondents' organizations did not have any team practicing Scrum, but this does not mean that they were not following other type of Agile practices. This helps to confirm that the targeted respondents' profile was reached.

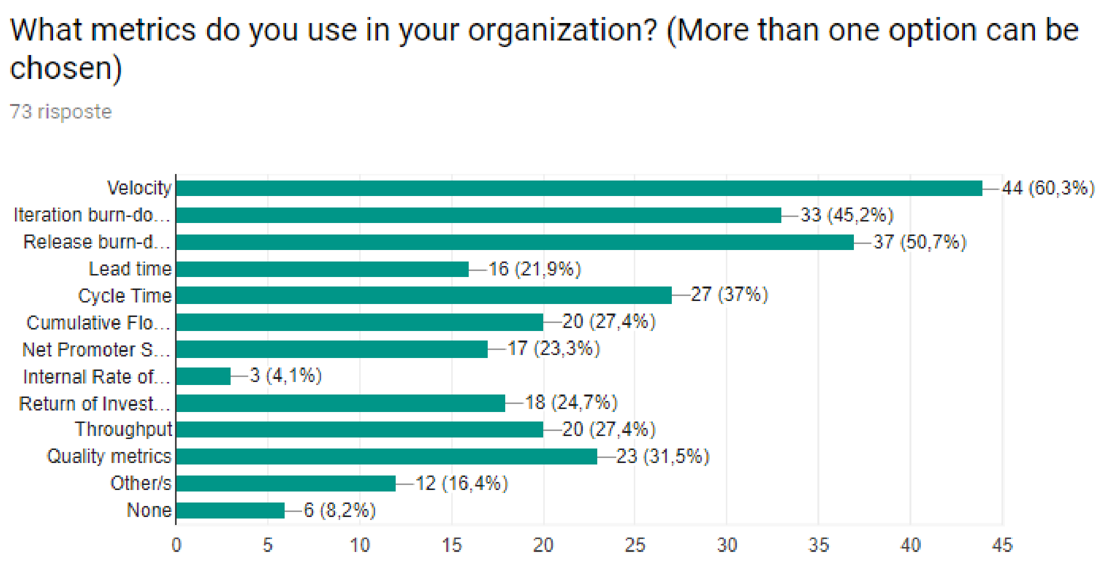


Figure 4.4: Results obtained for Question 2.

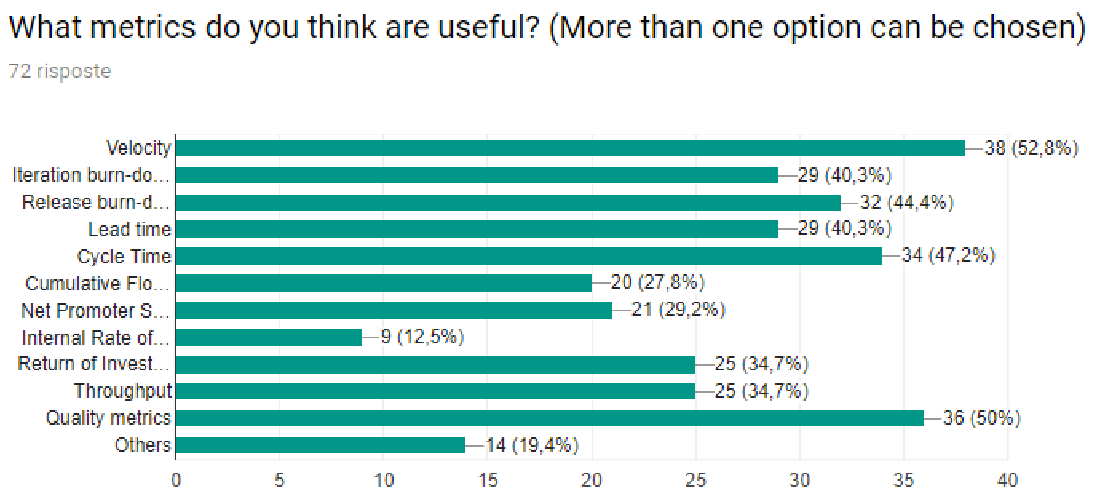


Figure 4.5: Results obtained for Question 3.

The results obtained in **Q2** (figure 4.4) and **Q3** (figure 4.5) are shown together in the following table to better understand them:

Option	**Q2 %**	**Q3 %**	**% Difference**
Velocity	60.3	52.8	↓ 7.5
Iteration burn-down	45.2	40.3	↓ 4.9
Release burn-down	50.7	44.4	↓ 6.3
Lead time	21.9	40.3	↑ 18.4
Cycle Time	37	47.2	↑ 10.2
Cumulative Flow Diagram (CFD)	27.4	27.8	↑ 0.4
Net Promoter Score (NPS)	23.3	29.2	↑ 5.9
Internal Rate of Return (IRR)	4.1	12.5	↑ 8.4
Return of Investment (ROI)	24.7	34.7	↑ 10
Throughput	27.4	34.7	↑ 7.3
Quality metrics	31.5	50	↑ 18.5
Other/s	16.4	19.4	↑ 3
None	8.2	-	-

Table 4.1: Percentages of answers in questions Q2 and Q3 of the survey.

The table 4.1 shown above helps to visualize that there is a decrease in percentage regarding the three most used metrics (velocity, iteration burn-down, and release burn-down) and what the respondents think about their usefulness, despite these metrics being very relevant. Also it is possible to see that there are metrics which the respondents think are useful, such as lead time, cycle time, and quality metrics, but their implementation is not too much spread across Scrum Teams. This situation can be due to different situations such as metrics are imposed by the management, teams provide metrics that make management “happy”, among others.

In the following table 4.2 the answers obtained in **Q4** are shown, where either new or different metrics to the ones given in the options of **Q2** and **Q3** were proposed. The full list of answers can be found in the Appendix A.

Answers
Business Value Delivered
Scope change
Deployment rate, business value delivered, employee satisfaction
Developers satisfaction
Happiness score
Sprint speed metric
E-NPS (Employee NPS), win/loss rate of iterations. acceleration, percentage of forecast done, escaped defects, financial metrics, sustainability, value stream analysis
Relative business value estimated, relative business value delivered
User value received, development org value received
On scope, automation coverage, code coverage
Team Happiness index, Customer satisfaction
Team Satisfaction, how many bugs. Division of type of work
Frequency of deploying new functionality to the customer
Value outcome
Customer satisfaction, number of users, employee satisfaction
Velocity deviation

Table 4.2: Answers obtained in question Q4 of the survey.

From the table 4.2 it is possible to distinguish that the most repeated metric is related to team's satisfaction/happiness, this is an important metric because happier people are 12% more productive according to Oswald et al. [47]. Some Scrum practitioners recommend to ask team members during the Sprint Retrospectives [2] how happy they feel on a scale from one (1) to five (5). Another technique is the niko-niko calendar [3] [57] where in a calendar that is visible for all the team members, they draw day by day a face which represents how a person felt that day. Nevertheless, while this require a high level of openness and trust among team members, this can be difficult to achieve even in teams that have a good performance practicing Scrum.

Moreover, other metrics proposed by the respondents were related to the value delivered to the business and customers, and quality metrics such as

[2] https://web.archive.org/web/20140721134516/http://scrum.jeffsutherland.com:80/2010/11/happiness-metric-wave-of-future.html

[3] https://www.agilealliance.org/glossary/nikoniko/

number of bugs that escape into production, automation or code coverage, or frequency of deployment.

The answers obtained in **Q4** were insightful to expand the spectrum of metrics, regardless if they can be implemented or not through a software tool.

4.3 Conclusion: Selected metrics

According to Davies [14], it is important that developers, product owners, and project managers understand why a metric is measured, what it is measuring, and how it is collected. This is important because they have different points of view of the same product/project, but the metrics tell how the team is performing as a whole, not individuals.

With the information collected from the survey carried out and the information obtained from the annual State of Agile survey, we may conclude that the following important metrics should be measured in a Scrum Team:

- **Velocity**: this implies the velocity that is forecast for a Sprint, the team's committed work and the real velocity, this is the work completed ("Done Done") and accepted by the Product Owner. These values will provide information regarding the team's pace, which can be used to forecast how many Sprints will be needed to deliver a certain amount of functionality to the customers. Also, the variability of team's velocity can be an alert that something is happening within a team.

- **Iteration burn-down** and **Release burn-down**: these metrics were depicted on Secction 3.3.1. They help Scrum Teams to visualize how far they are from the goal that they established for that Sprint or Release.

- **Lead Time** and **Cycle Time**: a description of this metrics can be seen on Section 3.3.4. They indicate whether or not there is a bottleneck in the workflow implemented by a team.

- **Throughput**: this is how many functionality was delivered in a period of time (e.g. User Stories per week). This metric helps to understand the team's behavior, for example, if a team delivers regularly functionality or it does one big delivering near the end of the Sprint/Release.

- **Type of work committed**: this metric helps teams to visualize how much effort a team dedicates to deliver value to the customer, and how much is dedicated to fix problems. If a team dedicates most of the effort of Sprints on fixing bugs rather than producing new features, this may mean that the team is generating too much technical debt, or that the testing is deficient, among others.

These metrics have as purpose to help Scrum Teams to detect quickly when processes or practices are having unwanted behaviors.

Chapter 5

Implementation

So far, the activities Scrum Teams have to carry out according to the Scrum Guide (Section 2.2), and the metrics to measure different aspects of the software development where Scrum practices are adopted (Chapter 3) have been studied.

In the following sections of the current chapter, an analysis of available tools not only for management of software projects but also for their measurement will be provided. Further, a description of the tool chosen as result of the analysis, as well as a description of the solution implemented to produce useful metrics will be presented. The aim of the analysis and the solution implemented is to establish and provide Scrum Teams with a suitable working environment where most of the execution and tracking of the activities can be carried out.

5.1 Analysis of Tools

In the current section, tools for managing software throughout its life cycle, as well as tools for measuring a variety of variables and parameters related to Agile or Scrum in a project, will be shown.

5.1.1 Application Life-cycle Management Tools

Before obtaining any metric, it is important to establish the software tool that will be the source of information for the application provider of metrics. In order to select the appropriate tool, an analysis focused on the available software for Agile project management was carried out.

The first aspect analyzed was the type of license on the source code of the software. It is possible to differentiate between *open source* and *proprietary* software. The software belonging to the first group gives to users the possibility to adapt the software to their needs by adding functionalities or modifying the behavior of the software. While the second provides software to which the user must adapt.

The table 5.1 shows the different software separated in the before-mentioned groups.

Open Source	**Proprietary**
• Agilo for Trac, • MY Collab, • OpenProject, • OrangeScrum, • Phabricator, • Taiga, • Tuleap	• Asana, • Axosoft, • CA Agile Central, • FogBugz, • IceScrum, • Jira, • Kanbantool, • LeanKit, • Mingle, • Planbox, • PivotalTracker, • SwiftALM, • VersionOne

Table 5.1: Open source and proprietary tools for Agile project management.

From this first stage of the analysis, it has been decided that the tool must be an open source one. In this way, the organizations that want to start using a tool for their teams using the Scrum framework, do not have to spend a lot of money buying licenses or paying a subscription fee for using software. Following open source tools for project management are briefly depicted:

- **OpenProject** [1]: a collaborative, web-based project management software written in Ruby.
- **]project-open[** [2]: a web-based project and service management tool with focus on finance and collaboration. Its last stable version is the 4.0 from 2013, even though the software has a version 5.0 that is in development currently.
- **Trac** [3]: an enhanced wiki and issue tracking system for software development projects. There exist a tool based on Trac that is called Agilo for trac [4], which supports the agile Scrum software development process. Both tools, Trac and Agilo for trac, are written in Python.
- **MY Collab** [5]: a collaboration platform management that provides a comprehensive set features of Project Management, Customer Relationship Management (CRM), and Document Management. The main programming language of this tool is Java.
- **OrangeScrum** [6]: a project management and collaboration suite that provides a centralized management for projects and tasks, collaborative resources, tracking of time and generation of invoices, visualization of real-time analytics. The tool has its basis on the PHP programming language.
- **Taiga** [7]: a project management platform for agile developers, designers, and project managers.

[1] https://www.openproject.org/
[2] http://www.project-open.com/
[3] https://trac.edgewall.org/
[4] http://www.agilofortrac.com/
[5] https://www.mycollab.com/
[6] https://www.orangescrum.org/
[7] https://taiga.io/

- **Tuleap** [8]: a platform for Agile management and software development.

The second aspect taken into account to make a decision about the tool was the frequency of actualization of the tools, but almost all of them had their last versions released the current year, having a pace of releasing a new version every one or two months. Even the pace of actualization of the tools was not very helpful to make a decision, it is a good and healthy sign that the tools have the support of the companies behind them, but also of the communities supporting the software.

Because the purpose of the tool to be selected is not only to provide the data to the software to be developed but also to be integrated with other tools, the third aspect taken into account was the integration that the tools had with Mylyn. Mylyn [9] is a framework for Eclipse that realigns the Integrated development environment (IDE) around tasks through a task-focused interface. It provides a task management tool for developers which can be easily integrated with a wide range of Application Life-cycle Management (ALM) software. The tool gives to developers the information they need to accomplish their work and update the status of their tasks within the IDE, reducing the changes of context produced while switching between applications and which have as consequence a decrease in productivity.

From the above-listed tools, the only two that have a plugin for Mylyn are Tuleap and Trac. Tuleap (see Section 5.2) is the tool chosen between these two options because it integrates more functionalities since the very beginning and the possibility to extend them via plugins, instead Trac provides a basic functionality which can be extended through a series of plugins.

5.1.2 Measurement Tools

Most of the tools mentioned above (Section 5.1.1) provide a dashboard where teams can visualize the status of the different tasks they are executing, but only few of them offer a set of metrics, which is often not too deep. Following software developed with the goal of measure aspects such as team performance, Agile adoption, or the adherence of a team to Scrum standards, will be depicted.

ActionableAgile^{TM} Analytics [10] is a tool that reveals the efficiency of

[8] https://www.tuleap.org/

[9] http://www.eclipse.org/mylyn/

[10] https://www.actionableagile.com/

a Lean or Agile process and provides insights for determining what process tweaks can be made to improve these processes.

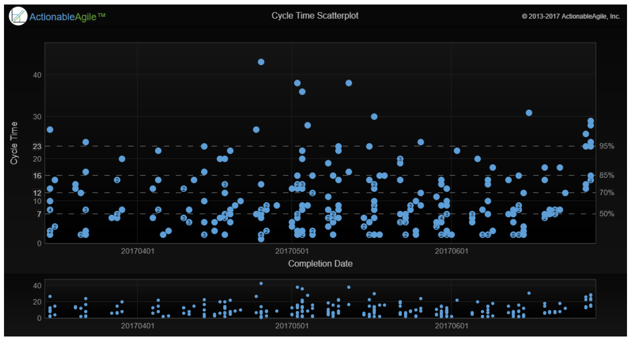


Figure 5.1: Example of a chart provided by the ActionableAgile Analytics tool. Source: https://www.actionableagile.com/analytics-demo/

AgilityHealth [11] is a platform that allows teams and companies to visualize, through different charts called radars, the performance and health of the Agile implementation across multiple teams and organization.

[11] https://agilityhealthradar.com/

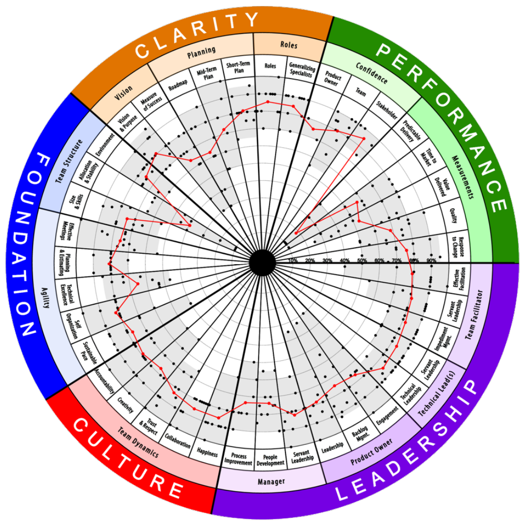
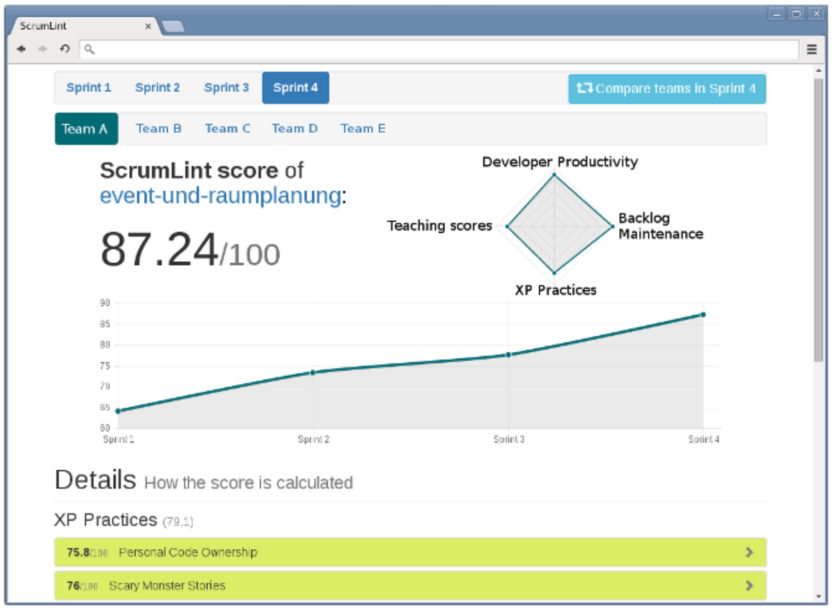


Figure 5.2: Example of an AgilityHealth radar. Source: https://agilityhealthradar.com/team-health-radar-assessment/

ScrumLint [12] [39] is a tool that analyzes development artifacts in order to provide process metrics. These metrics contain the core ideas of agile methods and report deviations, which gives development teams immediate feedback on their executed development practices. This information can be used to improve their workflows.

[12]https://github.com/chrisma/ScrumLint



ScrumLint score of event-und-raumplanung:

87.24/100

Details How the score is calculated

XP Practices (79.1)

75.8/100 Personal Code Ownership

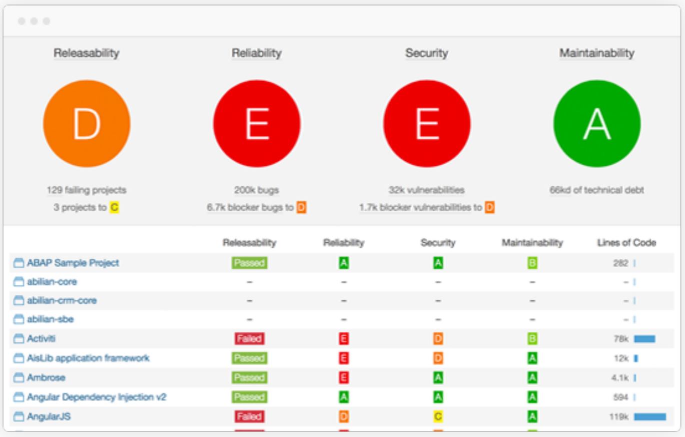
76/100 Scary Monster Stories

Figure 5.3: Screenshot of ScrumLint. Source: [39]

As can be observed, the offer of open source software for metrics obtainment is far from being well developed as the offer of ALM tools.

Another aspect the Scrum Teams should care about, and which was pointed out in the results of the survey carried out (see Section 4.2.3), is the quality of the software they are developing. For this task there exist a widely used open source tool denominated SonarQube [13]. It is a platform that provides continuous inspection of code quality that shows the health of an application regarding characteristics of the code such as quality, security, bad smells, and consistency with the language.

[13]https://www.sonarqube.org/



Releasability	Reliability	Security	Maintainability
D	E	E	A
129 failing projects 3 projects to C	200k bugs 6.7k blocker bugs to D	32k vulnerabilities 1.7k blocker vulnerabilities to D	66kd of technical debt

	Releasability	Reliability	Security	Maintainability	Lines of Code
ABAP Sample Project	Passed	A	A	B	282
abilian-core	–	–	–	–	–
abilian-crm-core	–	–	–	–	–
abilian-sbe	–	–	–	–	–
Activiti	Failed	E	D	B	78k
AisLib application framework	Passed	E	D	A	12k
Ambrose	Passed	E	A	A	4.1k
Angular Dependency Injection v2	Passed	A	A	A	594
AngularJS	Failed	D	C	A	119k

Figure 5.4: Example of a SonarQube dashboard. Source:https://www.sonarqube.org/

5.2 Tuleap

Tuleap is a 100% *libre* and open source tool, and it is sponsored and developed by Enalean, a French company. The company behind the tool has what they call Enalean's Open Roadmap [14] in which every month they deploy a new version of the software based on user's feedback and customer's needs, this keeps Tuleap continuously evolving. The system can be used by organizations of all sizes from small to large, and from a wide range of industries, some examples are Eclipse Foundation, OW2 Consortium, Airbus Group, Orange^{TM}, JTEKT Corporation, Portuguese Public Institute, among others.

The software provides an environment where most of the needs for a software project are covered, and where all the people involved (i.e. Scrum Masters, Product Owners, Team Developers, Stakeholders) can meet. With Tuleap, it is possible to create customized dashboards (see Figure 5.5) for any type of team, no matter the life-cycle they use (i.e. Waterfall, Scrum, or Kanban) nor the process they follow. One of the main characteristics of Tuleap is its high degree of adaptability. With this software, teams can track any type of item (see Figure 5.6) such as requirements, tasks, bugs,

[14] https://blog.enalean.com/enalean-open-roadmap-how-it-works/

documents, among others, as well as customize any of these trackers or create new ones.

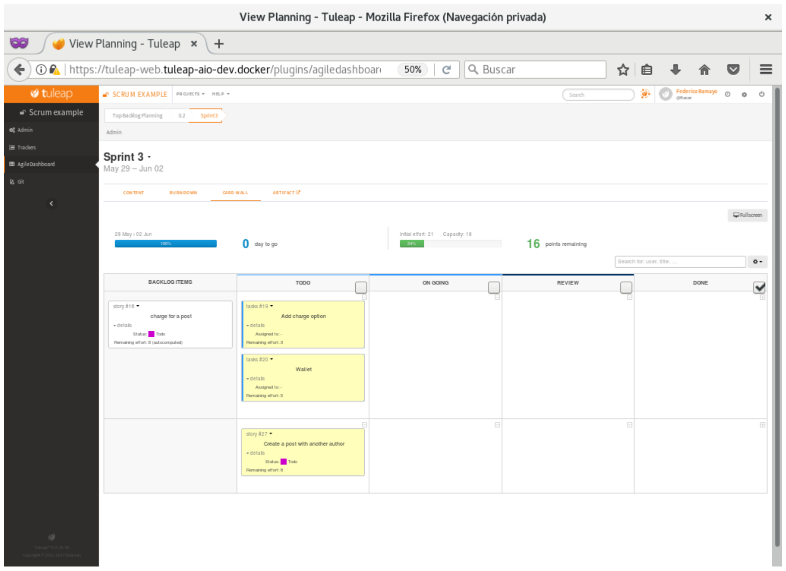
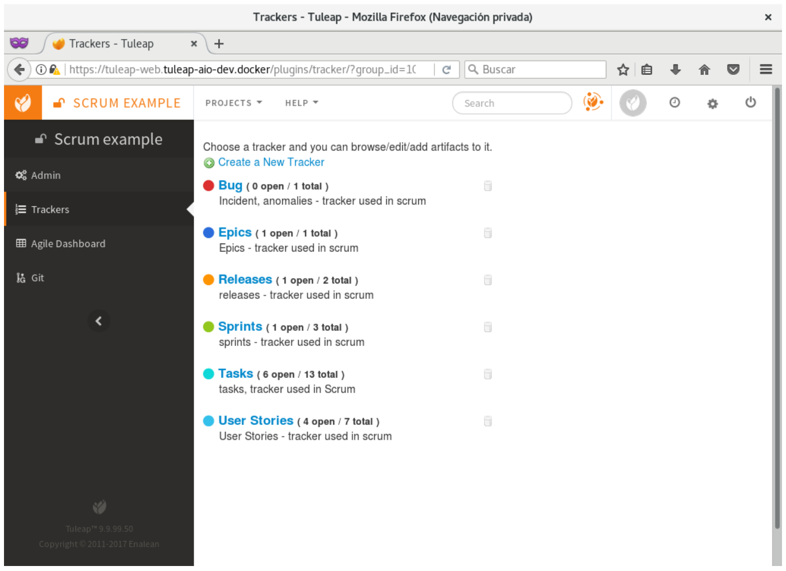


Figure 5.5: Scrum dashboard in Tuleap.



Choose a tracker and you can browse/edit/add artifacts to it.
Create a New Tracker

● **Bug** (0 open / 1 total)
Incident, anomalies - tracker used in scrum

● **Epics** (1 open / 1 total)
Epics - tracker used in scrum

● **Releases** (1 open / 2 total)
releases - tracker used in scrum

● **Sprints** (1 open / 3 total)
sprints - tracker used in scrum

● **Tasks** (6 open / 13 total)
tasks, tracker used in Scrum

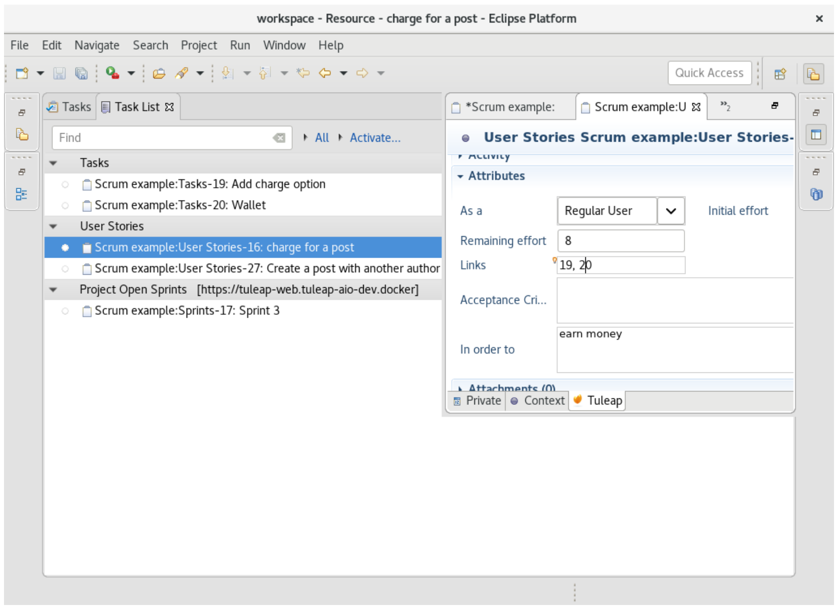
● **User Stories** (4 open / 7 total)
User Stories - tracker used in scrum

Trackers - Tuleap - Mozilla Firefox (Navegación privada)
Trackers - Tuleap
https://tuleap-web.tuleap-aio-dev.docker/plugins/tracker/?group_id=10
Buscar
SCRUM EXAMPLE
PROJECTS
HELP
Search
Scrum example
Admin
Trackers
Agile Dashboard
Git

Figure 5.6: Trackers for a Scrum project in Tuleap.

Another interesting characteristic of Tuleap is the integration with different tools that it provides through either a Simple Object Access Protocol (SOAP) Application Programming Interface (API) or a Representational State Transfer (REST) API, this openness helps to cover all the aspects on a software development life-cycle. For version control system Git or Apache Subversion (SVN) can be selected, with an unlimited number of repositories per project. Before a series of changes are pushed to the code repository they should be tested on a continuous integration (CI) tool, in this case Tuleap can connect with Jenkins. Through another tool called Gerrit, Tuleap also gives teams the possibility to review the code that is going to be added or modified in the code repository, allowing them to discuss about changes and share knowledge about the project.

Moreover, Tuleap can interact with Mylyn (see Figure 5.7) through the Tuleap Mylyn and Agile planner connector [15], which enables Development Team members to create and edit artifacts (i.e. tasks, bugs, support requests), and synchronize them with the Tuleap server.



Tasks
- Scrum example:Tasks-19: Add charge option
- Scrum example:Tasks-20: Wallet

User Stories
- Scrum example:User Stories-16: charge for a post
- Scrum example:User Stories-27: Create a post with another author

Project Open Sprints [https://tuleap-web.tuleap-aio-dev.docker]
- Scrum example:Sprints-17: Sprint 3

As a: Regular User
Initial effort
Remaining effort: 8
Links: 19, 20
Acceptance Cri...
In order to: earn money

Figure 5.7: Mylyn showing tasks retrieved from a Tuleap server.

[15] https://marketplace.eclipse.org/content/tuleap-mylyn-and-agile-planner-connector

5.3 Solution

Tuleap offers release and iteration burn-down metrics, which is a good way to track the progress of a team against release or sprint deadline but is not enough to have a clear view of how well a team is performing, where the impediments are, or why they appear.

The aim of the solution developed is to expand the functionalities of Tuleap with an application that provides a set of useful metrics (see Chapter 4) for Scrum Teams. The source code of the application has been released under the GNU General Public License v3.0, and it is available online on the GitHub platform at `https://github.com/El-Flaco/ScrumMetrics`.

In the current section, the architecture (i.e. technology and structure) of the application developed will be depicted, followed by an explanation of how the metrics are obtained, and finally the results of the implementation will be shown.

5.3.1 Architecture

The technological stack of the application is compounded by :

- **Client-side**:
  - **HyperText Markup Language (HTML) & Cascading Style Sheets (CSS)**: the first term refers to the markup language that allows the creation of web pages rich in text, images, videos, etc; while CSS is the language that describes the style of an HTML documents and how the elements of the web page will be displayed. The use of both technologies permits to provide to the users a responsive interface through any web browser, this is important to keep a consistent user experience across different devices (e.g. desktop, laptop, mobile).
  - **JavaScript (JS)**: it is an interpreted programming language that allows to create dynamic web pages. Using this technology it is possible to interact with the web page through the Document Object Model (DOM), which gives developers the possibility to do things like catching the user's interactions within the application and respond according them, modify any part of the web page when a condition is reached, among others.

  - **Asynchronous JavaScript And XML (AJAX)**: it is a web development technique to build Rich Internet Applications (RIA). This the technology makes possible keep asynchronous communications with the server while the users interact with the web application. AJAX accesses to the data retrieved by the server through an object available on all the current web-browsers called XMLHttpRequest. This object can be easily manipulated using JavaScript code.

- **Server-side**:

  - **PHP**: this is the technology in which the application relies on, and which allows the interaction with the Tuleap server in order to obtain data, process it and generate the metrics.

In the following section, the structure used to separate the code will be explained:

- **Classes**: here it is possible to find PHP classes created in order to cover different needs. The classes contained in this folder are the TuleapUser class which manages the user's connection with the Tuleap server, the CurlManager class that manages the interaction with the REST API provided by the Tuleap server, and a iFileManager interface with a set of methods that any class implementing it must have, as well as a JsonFileManager class which implements this interface to manage the operations over JSON files.

- **Drawers**: this folder contains the PHP files where the data retrieved by the Tuleap server is processed and utilized to generate charts representing the metrics obtained (e.g. drawArtifactsByType.php, drawVelocity.php). Moreover, the Topnew SVG Chart [16] (i.e. cms_chart.php), the file which allows to generate Scalable Vector Graphics (SVG) charts that can be embedded to a web page, is placed here.

- **Functions**: the files present in this folder accomplish specific functions such as interact with the REST API of the Tuleap server (e.g. getProjects.php, getArtifactInformation.php), or perform calculations using the data provided by the interactors (e.g. calculateVelocity.php, LeadTimeCalculator.php).

[16] http://topnew.net/chart/

- **Site**: this is where the client-side part of the application relies. It is possible to find the three (3) main user interfaces (i.e. home.html, login.html, and project.html), as well as the following folders:
  - **css**: this folder contains the CSS files that give the Look and Feel to the application.
  - **js**: here it is possible to find the JS files that manage the user's interaction with the web page, and where the AJAX calls to the server are carried out.
  - **php**: in this folder are present PHP files that manage the user's session (e.g. login.php, logout.php), as well as the ones that manage the AJAX calls from the clients.

5.3.2 Interaction with the Tuleap REST API

The application needs a source from where to collect data that will be used in the calculus of different metrics. Tuleap provides a REST API that makes it possible to interact and this allows developers to gather data that after being processed can provide useful information. This API [17] can be accessed anonymously or with authentication (this will depend on the configuration of the projects) through the following Uniform Resource Locator (URL, colloquially called web address) `https://<tuleap-host-name>/api/`. The Hypertext Transfer Protocol (HTTP) request methods available to interact with the API are:

- **POST**: this method submits data to be processed to a specified resource. There is no restriction neither in the type nor the length of the data that can be send to the server. The data sent is not displayed in the URL, nor stored either the web browser history or the web server log, which gives the method a layer of security.

- **GET**: method used to request data from a specified resource. Here the data is sent as part of the URL, which makes the method restricted on the lenght of the URL to 2048 characters, and also to be less secure than POST because the data is visible to everyone in the URL.

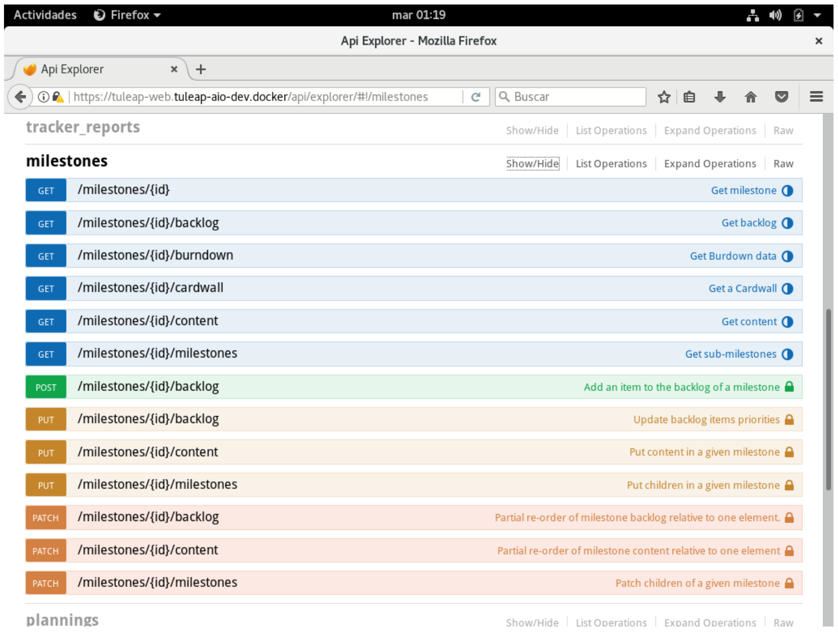
[17] `http://tuleap-documentation.readthedocs.io/en/latest/user-guide/rest.html`

- **DELETE**: when this method is called, it deletes an specified resource.
- **PUT**: with this method it is possible to create and store an entity on the server. When the entity already exists on the server, it does not create a new entity, but update its information.
- **PATCH**: this method is used to update partial resources. While with PUT method is necessary to send to the server the whole representation of a resource to update its information, with PATCH method it is possible to update a specific field of the resource, saving bandwith in consequence.

The API gives access to: projects, tokens, user_groups, users, phpwiki, jwt, system_event, git, trackers, artifacts, artifact_files, trackers_reports, milestones, plannings, backlog_items, kanban, cards, gerrit, svn. The application interacts with the following elements of the API:

- **tokens**: the TuleapUser.class.php makes a POST request to the `https://<tuleap-host-name>/api/tokens` URL when the user logs in into the application, in order to obtain a token that allows him/her to interact with the API. Also a DELETE call to `https://<tuleap-host-name>/api/tokens/{tokenID}` is made whit the identification (ID) of the token, when the user logs out the application, in order to delete the token generated during the login.
- **projects**: the getProjects function calls with the GET method to `https://<tuleap-host-name>/api/projects` in order to obtain the projects which the authenticated user has granted access. Having the ID of a project, it is possible to obtain its plannings through the getPlannings function, which makes a GET call to `https://<tuleap-host-name>/api/projects/{projectID}/plannings` in order to get the ID of the planned entities (i.e. Releases or Sprints plannings).
- **plannings**: with the ID of a planning, information about its milestones (i.e. Releases, Sprints) can be gathered through the getMilestones function This function makes a GET request to `https://<tuleap-host-name>/api/plannings/{planningID}/milestones` and as result it gives a list with the IDs of the milestones related to that planning entity.
- **milestones**: the function getMilestoneInformation retrieves information about a specific milestone, which is identified by its ID, such as

its capacity (in Story Points) and status (e.g. Open, Closed). To make this possible, the function makes a GET request to `https://<tuleap-host-name>/api/milestones/{milestoneID}`.
Also it is possible to obtain data of the artifacts (i.e. User Stories, Bugs) that are part of a milestone, this can be done through the getMilestoneContent function which makes a GET request to `https://<tuleap-host-name>/api/milestones/{milestoneID}/content`. The information regarding to the artifact includes its ID, its type and status, and the effort (in Story Points) required to accomplish that artifact. The function provides all the before-mentioned information of all the artifacts belonging to a milestone.


Actividades
Firefox
mar 01:19
Api Explorer - Mozilla Firefox
Api Explorer
https://tuleap-web.tuleap-aio-dev.docker/api/explorer/#!/milestones
Buscar

tracker_reports — Show/Hide | List Operations | Expand Operations | Raw

milestones — Show/Hide | List Operations | Expand Operations | Raw

GET	/milestones/{id}	Get milestone
GET	/milestones/{id}/backlog	Get backlog
GET	/milestones/{id}/burndown	Get Burdown data
GET	/milestones/{id}/cardwall	Get a Cardwall
GET	/milestones/{id}/content	Get content
GET	/milestones/{id}/milestones	Get sub-milestones
POST	/milestones/{id}/backlog	Add an item to the backlog of a milestone
PUT	/milestones/{id}/backlog	Update backlog items priorities
PUT	/milestones/{id}/content	Put content in a given milestone
PUT	/milestones/{id}/milestones	Put children in a given milestone
PATCH	/milestones/{id}/backlog	Partial re-order of milestone backlog relative to one element.
PATCH	/milestones/{id}/content	Partial re-order of milestone content relative to one element
PATCH	/milestones/{id}/milestones	Patch children of a given milestone

plannings — Show/Hide | List Operations | Expand Operations | Raw

Figure 5.8: Tuleap API explorer screenshot.

- **artifacts**: the getArtifactInformation function, which makes a GET call to `https://<tuleap-host-name>/api/artifacts/{artifactID}`, retrieves the status, date of creation, and the date of last modification of an artifact.
  Moreover, it is possible to obtain the historical status changes (e.g. from Todo to On Going) ocurred to user stories and bugs through the getUserStoryStatusChangesets and getBugStatusChangesets functions.

Both functions make a GET call to `https://<tuleap-host-name>/api/artifacts/{artifactID}/changesets`, which retrieves information about the status of the artifact and the date in which this changes happened.

All the above-mentioned interactions with the Tuleap API are part of a chain that makes it possible to calculate helpful metrics.

The main purpose of the application is to provide Scrum Teams with the metrics they need, in the moment they ask for them. For this reason the application does not store data in a data base (DB), but it processes the users request and calculates the metrics each time they are requested.

5.3.3 Result

Scrum Teams must have a way to share and radiate information among the members of the team, as well as people outside of it. For this reason, the application has implemented a web interface in which users can request for a set of metrics. Different interfaces of the application are shown as follows.

The first interface that a user sees is shown in the Figure 5.9, this is necessary in order to obtain the user's token that will allow his/her interaction with the Tuleap API.

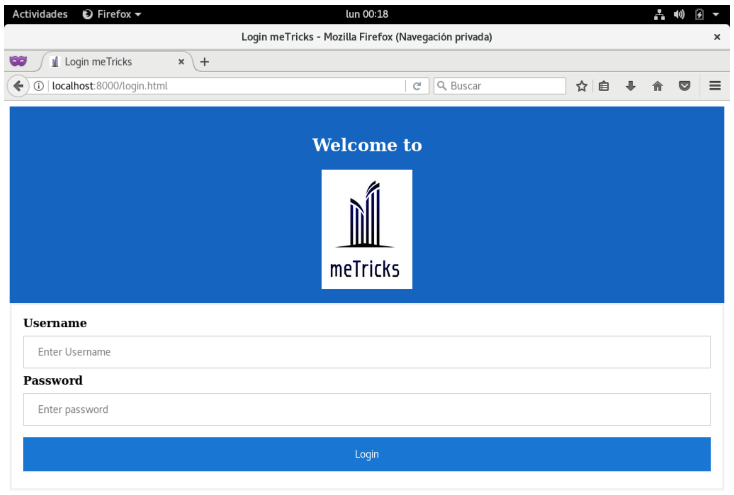


Figure 5.9: Login to meTricks plugin.

Once the user is logged in, as it is possible to observe in Figure 5.10, a list of projects (either public or where he/she is member) is shown. This gives to the user the possibility to choose from which project he/she wants to retrieve the metrics.

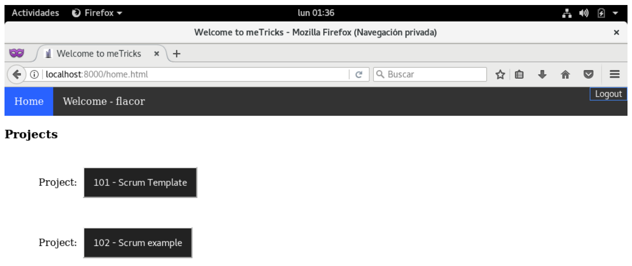


Figure 5.10: Project selection screen.

After selecting the project, the user can request for any metric from the list, i.e. Velocity, Artifacts by Type, Sprints Lead Time, Artifacts Lead Time, Artifacts Cycle Time (see Section 4.3 for more details), and after a calculation which takes few milliseconds the chart corresponding to the metric will be shown. Below, examples of the metrics that an user can obtain are presented:

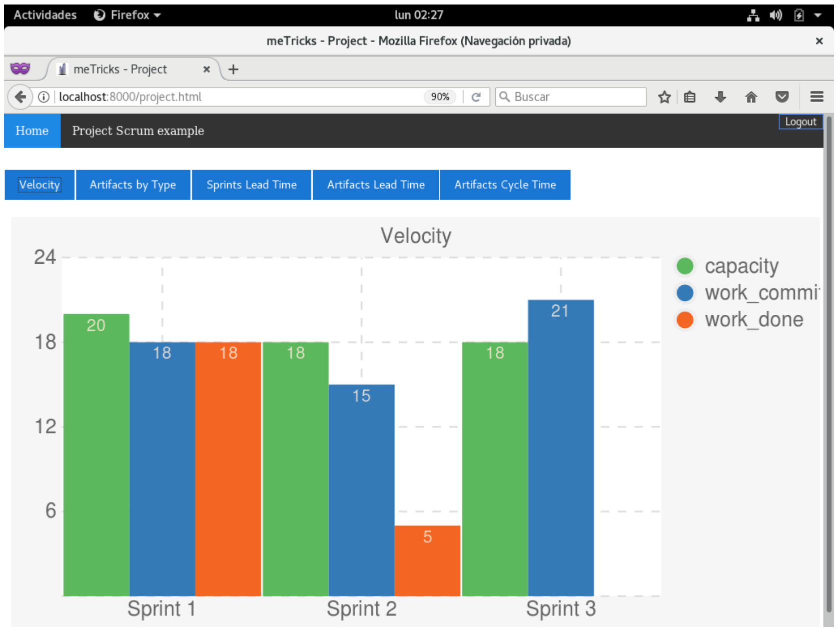


Figure 5.11: Example of Velocity metrics.

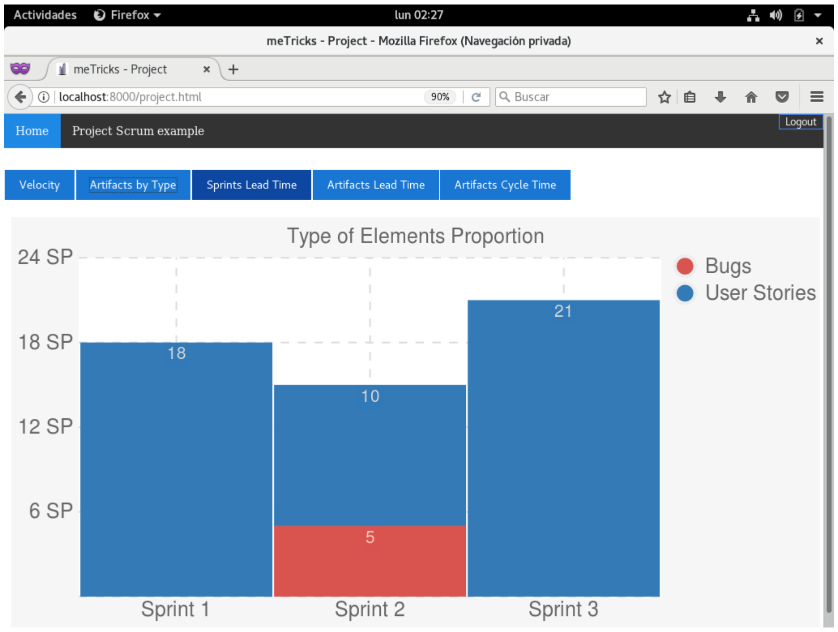


Figure 5.12: Example of Artifacts by Type.

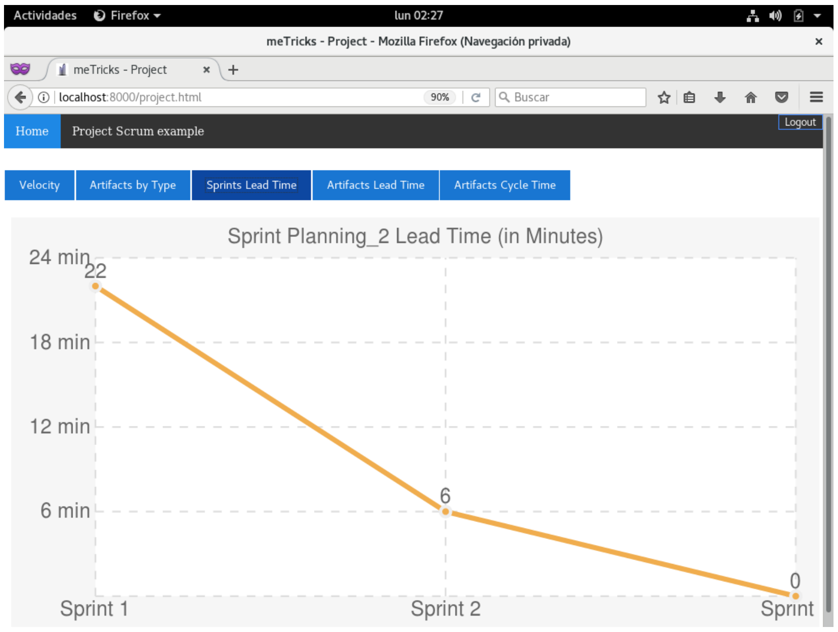


Figure 5.13: Example of Sprints Lead Time.

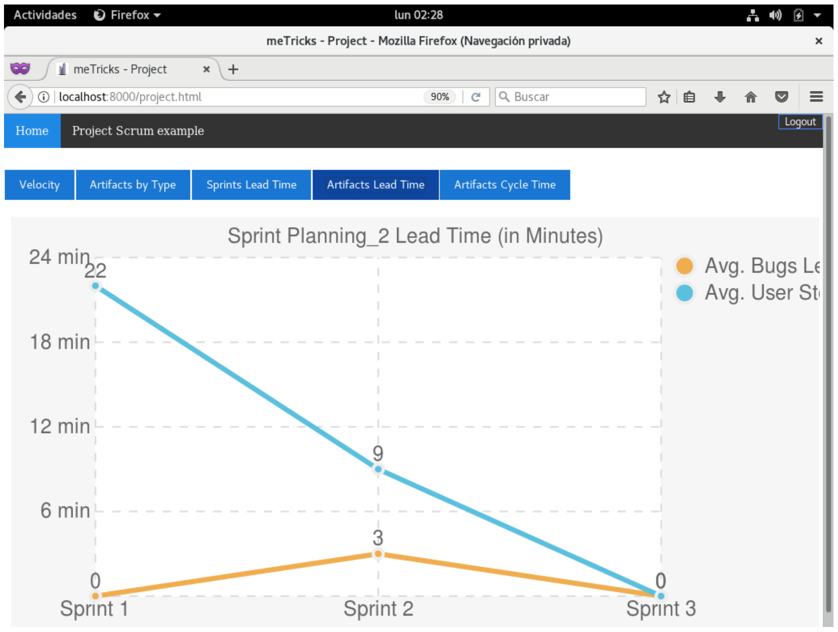


Figure 5.14: Example of Artifacts Lead Time.

5.4 Conclusion: Working Environment

After analyzing the available ALM tools, a working environment was proposed. It consists of Tuleap, a tool that allows software teams (whether o not using Agile methodologies) to have control over all the aspects involving software life cycle (e.g. features to be developed, hosting and control of the source code), as well as meTricks, an application to provide teams with metrics that will help them to improve their performance. It is important to remark that since both the tools are free and open source, these characteristics make it possible that the tools adapt to the teams' processes and practices and not the other way round, and also prevent an organization from having to invest a significant sum of money.

A schema summarizing the resulting system is shown in Figure 5.15.

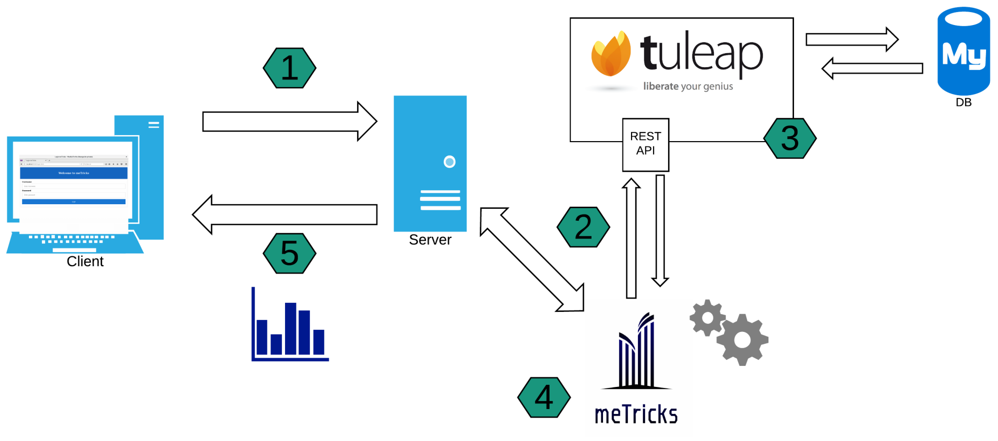


Figure 5.15: Architecture of the working environment.
(1) The user requests a metric via web browser.
(2) On the server, meTricks interacts with Tuleap through its REST API.
(3) Tuleap provides the information stored in its data base.
(4) meTricks process the data gathered and sends the results to the user.
(5) A chart representing the metrics obtained is shown.

Chapter 6

Conclusions

This thesis presents the resources needed to set a working environment for Scrum Teams where it is easy for them to track and measure its progress over a project life cycle. To achieve this the core concepts of Agile and Scrum were explained, the metrics used by teams that have adopted Agile practices were analyzed and described, as well as an analysis of the available tools for life cycle project management was carried out. These analysis were the basis to develop an application that gathers the metrics from a server and shows the results to the users. In order to support the metrics that are shown by the application a survey was carried out.

The current chapter presents the conclusions of the work developed. First, the main points of the work done are reviewed and thereafter the possibilities for future work are addressed.

6.1 Contribution

In this work the Agile concepts were presented along with Scrum which is one of the frameworks that implements these concepts. Although Agile does not ensure success to organizations, it provides them with the necessary ideas to transform them into organizations well prepared to face changes that may occur immediately or in the near future. This gives organizations a big competitive advantage, because they are neither attached to a business model, nor a technology, or a specific group of clients.

Scrum is the most used framework to adopt the Agile concepts due to its maturity (it was born in 1993). The creators of this framework update a guide regularly (the last version is from June, 2016) where the roles, events, artifacts, and rules of Scrum are established and depicted in depth. This provides a great background for teams that want to start the adoption of Agile values and principles.

Also, the current work has shown that any software development team that wants to succeed must have a defined measurement plan establishing metrics that are useful for the team. These metrics should be pragmatic, have a clear purpose, and provide useful information (i.e. information that lead to actions). In this way the team can identify where it is failing, or which points have margin for improvement.

A series of points that are pertinent for identifying the useful metrics were studied, to provide the basis on which teams can rely and start measuring their performance in different aspects of the software development process. Also, different classifications of metrics were provided according to aspects such as the type of variable measured to obtain the metric, the information provided by the metric, the point in the time (i.e. past or future) that they observe, and the purpose of their use. Furthermore, metrics strongly related with Scrum were depicted, as well as metrics that were thought useful for other Agile methodologies, but are still useful despite the framework adopted. Among these metrics it is possible to find Velocity, Burn-down and Burn-up charts, Lead and Cycle Time, Throughput, Quality metrics (e.g. number of defects during development, percentage of automated test coverage), Value metrics (e.g. ROI, NPV, IRR), and a set of ten metrics for increasing the productivity of a team.

It is clear that the number of metrics can be overwhelming (even for experienced teams), for this reason it is recommended for teams that are initiating with Scrum to measure only few, but insightful, attributes at the beginning, and after some time practicing this framework to adapt the metrics to their own process. The scope of metrics for teams that want to attempt to start measuring themselves with the aim of improving, either practices or processes, was reduced through the information gathered from the 11th annual State of Agile, which inspired this work to carry out a specific survey about the metrics that Scrum practitioners use and the ones that they think are really useful (even if they use them or not). The later survey also provided insight about metrics that were not considered during the analysis, but those that Scrum practitioners think are needed to measure, even if they are hard to measure or not very objective (e.g. team members' happiness).

With all the information collected, and after an analysis of the available tools (open source and proprietary) for software project management, a working environment for Scrum Teams was settled. This environment is conformed by Tuleap, an open source ALM tool that can be integrated with other applications which gives it the ability to cover all the team's needs in a project (i.e. tracking artifacts and bugs, storage of documents, code repositories, continuous integration, etc.), and an open source application called meTricks was developed by this work.

The purpose of meTricks is to provide Scrum Teams with useful metrics. To achieve this, it gathers data by making use of the REST API provided by the Tuleap server running in the working environment, and, after processing the collected data and making the calculations, it provides the results to the users through a web-based interface. Among the metrics provided by the application it is possible to find team's Velocity along with the commited work and the work that have accomplished the team's definition of Done, the type of work that a team is addressing (i.e. creating new features or fixing problems), as well as the time needed to fix a bug or deliver a feature to the customer (Lead and Cycle time). Due to the open source nature of meTricks, it can be easily extended or modified by any person interested in doing so, giving them the chance to adapt the tool to what they need.

In summary, this work establishes the necessary concepts to allow any person attempting to approach the ideas such as Agile or Scrum, which go beyond mere software development and provide a friendly perspective. Also, to those software teams or organizations that having adopted Scrum want to start improving their performance, a free and open source working environment was proposed. This environment has its basis in the analysis of available metrics as well as tools.

6.2 Future Work

The following list summarizes the possible directions that can be taken in order to give the current work a wider spectrum of application.

- **Integrate the application with Tuleap**. For now the application developed can be executed in a server that supports PHP, but the aim is that the application can be added to Tuleap as a plugin. This will allow users to have most of the tools together in one place, reducing the effect of switching between applications. To achieve this it is necessary to adapt the architecture of the app to the architecture that Tuleap establishes for its plugins. Also it is necessary that the application should have written a series of test cases (which must be successfully passed), and pass a series of specific test cases written by Tuleap developers. The purpose of this is that the plugin written will not produce any critical error nor any misbehavior in the software as a whole.

- **Making the application available for any other tool**. The current application is developed having in mind the interaction with a Tuleap server. But due to the lack of free and open source tools for software metrics, there exist space in this area to develop an application that can interact with any ALM tool, providing them with useful metrics.

- **Development of a Chatbot**. From the analysis of the answers to the question four (Q4) of the survey carried out (see Section 4.2.3), it is possible to observe that there is a need to measure the happiness of the team members, as well as the happiness of the customers. In order to gather this information without disturbing neither the team members nor the customers, a chatbot that asks either direct or indirect questions can be implemented. This chatbot can be another source of information for the application.

Appendix A

Survey Model and Answers

In this appendix, the model of the survey carried out by using the platform provided by Google Forms, as well as all the answers obtained will be shown.

A.1 Survey Model

Following is the survey that the respondents had to answer and which has been depicted in Chapter 4:

Metrics for Scrum Teams

I would like to ask you four (4) questions as a part of a Master's Thesis at University of Camerino.

I thank you in advance for your contribution, and feel free to contact me.

Federico Ramayo.
feederico.ramayo@studenti.unicam.it

1. **Do the teams in your organization use Scrum?**
   Contrassegna solo un ovale.
   - ◯ Yes, all of them
   - ◯ Some of them
   - ◯ No, none of them

2. **What metrics do you use in your organization? (More than one option can be chosen)**
   Seleziona tutte le voci applicabili.
   - ☐ Quality metrics
   - ☐ Velocity
   - ☐ Iteration burn-down
   - ☐ Net Promoter Score (NPS)
   - ☐ Return of Investment (ROI)
   - ☐ Cumulative Flow Diagram (CFD)
   - ☐ Release burn-down
   - ☐ Internal Rate of Return (IRR)
   - ☐ Cycle Time
   - ☐ None
   - ☐ Throughput
   - ☐ Lead time
   - ☐ Other/s

3. **What metrics do you think are useful? (More than one option can be chosen)**
Seleziona tutte le voci applicabili.

- ☐ Velocity
- ☐ Iteration burn-down
- ☐ Release burn-down
- ☐ Lead time
- ☐ Cycle Time
- ☐ Cumulative Flow Diagram (CFD)
- ☐ Net Promoter Score (NPS)
- ☐ Internal Rate of Return (IRR)
- ☐ Return of Investment (ROI)
- ☐ Throughput
- ☐ Quality metrics
- ☐ Others

4. **Which metrics do you think are left in the previous questions? (Write as many as you want)**

A.2 Survey Answers

The results obtained (see Section 4.2.3) have basis on seventy-four (74) answers obtained, which are shown in the following subsections. It is necessary to remark that given the anonymous nature of the survey, the answers have a time-stamp as identifier.

A.2.1 Answers to Q1

The following table shows the answers for question one (**Q1**):

Time-stamp	**Do the teams in your organization use Scrum?**
08/06/2017 18.16.41	Yes, all of them
08/06/2017 19.26.33	Yes, all of them
08/06/2017 19.56.53	Some of them
08/06/2017 20.42.09	No, none of them
08/06/2017 20.47.50	Some of them
08/06/2017 21.34.50	Some of them
08/06/2017 22.20.22	No, none of them
09/06/2017 0.39.09	Some of them
09/06/2017 1.37.25	Some of them
09/06/2017 2.52.04	Yes, all of them
09/06/2017 8.06.45	Some of them
09/06/2017 8.21.32	No, none of them
09/06/2017 8.49.30	Some of them
09/06/2017 8.59.37	Some of them
09/06/2017 9.51.59	Yes, all of them
09/06/2017 10.02.04	Yes, all of them
09/06/2017 10.13.11	Yes, all of them
09/06/2017 23.54.29	Yes, all of them
10/06/2017 3.16.25	Some of them
10/06/2017 4.13.06	Yes, all of them
10/06/2017 12.24.04	Some of them
10/06/2017 13.30.18	Yes, all of them
10/06/2017 13.32.39	Some of them
10/06/2017 13.53.39	Some of them
10/06/2017 13.53.48	Some of them

Continued on next page

Table A.1 – *Continued from previous page*

Time-stamp	**Do the teams in your organization use Scrum?**
10/06/2017 14.29.04	Some of them
10/06/2017 15.07.49	Yes, all of them
10/06/2017 15.15.51	Yes, all of them
210/06/2017 15.46.29	Yes, all of them
10/06/2017 16.23.49	Yes, all of them
10/06/2017 16.38.21	Yes, all of them
10/06/2017 17.29.06	Yes, all of them
10/06/2017 17.57.01	Some of them
10/06/2017 18.15.14	Yes, all of them
10/06/2017 18.37.40	Yes, all of them
10/06/2017 18.47.24	Some of them
10/06/2017 19.36.21	Some of them
10/06/2017 20.02.20	Some of them
10/06/2017 20.03.24	Some of them
10/06/2017 20.07.33	Yes, all of them
10/06/2017 20.23.10	Some of them
10/06/2017 20.36.03	Yes, all of them
10/06/2017 20.40.31	Some of them
10/06/2017 21.19.03	No, none of them
10/06/2017 21.23.27	Some of them
10/06/2017 21.32.29	Some of them
10/06/2017 23.48.16	Yes, all of them
11/06/2017 0.35.29	Yes, all of them
11/06/2017 1.27.07	Some of them
11/06/2017 3.26.54	Yes, all of them
11/06/2017 6.26.53	Some of them
11/06/2017 8.14.38	Some of them
11/06/2017 8.54.24	Some of them
11/06/2017 10.26.23	Some of them
11/06/2017 10.50.29	Some of them
11/06/2017 13.09.47	No, none of them
11/06/2017 13.38.33	Some of them
11/06/2017 13.55.06	
11/06/2017 14.56.49	Some of them

Continued on next page

Table A.1 – *Continued from previous page*

Time-stamp	Do the teams in your organization use Scrum?
11/06/2017 17.12.20	Yes, all of them
11/06/2017 20.30.08	Yes, all of them
11/06/2017 21.26.02	Some of them
12/06/2017 1.03.09	Yes, all of them
12/06/2017 8.52.34	Some of them
12/06/2017 9.12.41	Yes, all of them
12/06/2017 13.41.29	Some of them
12/06/2017 15.55.55	Some of them
12/06/2017 22.06.25	Yes, all of them
13/06/2017 7.07.54	Some of them
13/06/2017 12.00.31	Some of them
13/06/2017 17.04.41	Some of them
13/06/2017 22.19.40	Some of them
14/06/2017 14.14.53	Some of them
14/06/2017 22.02.18	Some of them

Table A.1: Answers to question one (Q1).

A.2.2 Answers to Q2

The following table shows the answers for question one (**Q2**):

Time-stamp	What metrics do you use in your organization?
08/06/2017 18.16.41	None
08/06/2017 19.26.33	Velocity, Other/s
08/06/2017 19.56.53	None
08/06/2017 20.42.09	Cumulative Flow Diagram (CFD), Net Promoter Score (NPS)
08/06/2017 20.47.50	Velocity, Iteration burn-down, Release burn-down
08/06/2017 21.34.50	Velocity, Release burn-down, Cycle Time
08/06/2017 22.20.22	Cycle Time, Net Promoter Score (NPS), Throughput
09/06/2017 0.39.09	Release burn-down, Cycle Time, Net Promoter Score (NPS), Other/s
09/06/2017 1.37.25	None

Continued on next page

Table A.2 – *Continued from previous page*

Time-stamp	**What metrics do you use in your organization?**
09/06/2017 2.52.04	Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Throughput
09/06/2017 8.06.45	Velocity, Cycle Time
09/06/2017 8.21.32	Other/s
09/06/2017 8.49.30	None
09/06/2017 8.59.37	Velocity, Release burn-down
09/06/2017 9.51.59	Velocity, Iteration burn-down, Release burn-down
09/06/2017 10.02.04	Velocity, Iteration burn-down, Release burn-down, Cycle Time
09/06/2017 10.13.11	Cycle Time, Net Promoter Score (NPS), Quality metrics, Other/s
09/06/2017 23.54.29	Velocity, Release burn-down, Cycle Time, Quality metrics
10/06/2017 3.16.25	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 4.13.06	Velocity, Cycle Time, Net Promoter Score (NPS), Throughput
10/06/2017 12.24.04	Velocity, Lead time, Cycle Time, Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 13.30.18	None
10/06/2017 13.32.39	Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), None
10/06/2017 13.53.39	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics, Other/s
10/06/2017 13.53.48	Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS)
10/06/2017 14.29.04	Lead time, Cycle Time, Other/s
10/06/2017 15.07.49	Quality metrics, Other/s

Continued on next page

Table A.2 – *Continued from previous page*

Time-stamp	**What metrics do you use in your organization?**
10/06/2017 15.15.51	Velocity, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 15.46.29	Velocity, Iteration burn-down, Release burn-down, Net Promoter Score (NPS), Return of Investment (ROI)
10/06/2017 16.23.49	Return of Investment (ROI), Throughput
10/06/2017 16.38.21	Cycle Time, Cumulative Flow Diagram (CFD), Throughput, Quality metrics
10/06/2017 17.29.06	Velocity, Release burn-down
10/06/2017 17.57.01	Velocity, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD)
10/06/2017 18.15.14	Velocity, Iteration burn-down, Release burn-down
10/06/2017 18.37.40	Velocity, Iteration burn-down, Release burn-down
10/06/2017 18.47.24	Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Return of Investment (ROI)
10/06/2017 19.36.21	Velocity, Lead time, Quality metrics
10/06/2017 20.02.20	Velocity, Iteration burn-down, Release burn-down, Quality metrics, Other/s
10/06/2017 20.03.24	Velocity, Return of Investment (ROI), Throughput
10/06/2017 20.07.33	Iteration burn-down
10/06/2017 20.23.10	Velocity, Release burn-down
10/06/2017 20.36.03	Velocity, Iteration burn-down, Release burn-down, Quality metrics
10/06/2017 20.40.31	Velocity, Iteration burn-down, Release burn-down, Cycle Time, Cumulative Flow Diagram (CFD), Throughput
10/06/2017 21.19.03	Lead time, Throughput, Quality metrics, Other/s
10/06/2017 21.23.27	Iteration burn-down, Release burn-down, Lead time, Return of Investment (ROI)
10/06/2017 21.32.29	Iteration burn-down, Net Promoter Score (NPS), Return of Investment (ROI), Quality metrics
10/06/2017 23.48.16	Velocity, Iteration burn-down, Throughput
11/06/2017 0.35.29	Velocity, Iteration burn-down
11/06/2017 1.27.07	Velocity, Release burn-down, Cumulative Flow Diagram (CFD), Quality metrics

Continued on next page

Table A.2 – *Continued from previous page*

Time-stamp	**What metrics do you use in your organization?**
11/06/2017 3.26.54	Release burn-down, Quality metrics
11/06/2017 6.26.53	Velocity, Release burn-down, Cycle Time, Net Promoter Score (NPS)
11/06/2017 8.14.38	Velocity, Iteration burn-down, Quality metrics
11/06/2017 8.54.24	Velocity, Release burn-down, Cycle Time, Quality metrics
11/06/2017 10.26.23	Iteration burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Throughput, Quality metrics
11/06/2017 10.50.29	Velocity, Iteration burn-down, Return of Investment (ROI), Quality metrics
11/06/2017 13.09.47	Cycle Time, Cumulative Flow Diagram (CFD), Throughput, Other/s
11/06/2017 13.38.33	Other/s
11/06/2017 13.55.06	
11/06/2017 14.56.49	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Throughput, Quality metrics
11/06/2017 17.12.20	Velocity, Iteration burn-down
11/06/2017 20.30.08	Velocity, Iteration burn-down, Release burn-down
11/06/2017 21.26.02	Velocity, Iteration burn-down
12/06/2017 1.03.09	Velocity, Iteration burn-down, Cumulative Flow Diagram (CFD), Throughput
12/06/2017 8.52.34	Cumulative Flow Diagram (CFD), Throughput
12/06/2017 9.12.41	Release burn-down, Cycle Time, Return of Investment (ROI)
12/06/2017 13.41.29	Throughput, Other/s
12/06/2017 15.55.55	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Return of Investment (ROI), Quality metrics
12/06/2017 22.06.25	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Quality metrics
13/06/2017 7.07.54	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Return of Investment (ROI)
13/06/2017 12.00.31	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Return of Investment (ROI)

Continued on next page

Table A.2 – *Continued from previous page*

Time-stamp	**What metrics do you use in your organization?**
13/06/2017 17.04.41	Velocity, Iteration burn-down, Release burn-down, Net Promoter Score (NPS)
13/06/2017 22.19.40	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics
14/06/2017 14.14.53	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time
14/06/2017 22.02.18	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Return of Investment (ROI)

Table A.2: Answers to question two (Q2).

A.2.3 Answers to Q3

The following table shows the answers for question three (**Q3**):

Time-stamp	**What metrics do you think are useful?**
08/06/2017 18.16.41	Quality metrics
08/06/2017 19.26.33	Velocity, Others
08/06/2017 19.56.53	Lead time, Cycle Time, Net Promoter Score (NPS), Others
08/06/2017 20.42.09	Cumulative Flow Diagram (CFD), Net Promoter Score (NPS)
08/06/2017 20.47.50	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Throughput
08/06/2017 21.34.50	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time
08/06/2017 22.20.22	Lead time, Cycle Time, Net Promoter Score (NPS), Throughput, Others
09/06/2017 0.39.09	Release burn-down, Lead time, Cycle Time, Net Promoter Score (NPS), Others
09/06/2017 1.37.25	Lead time, Return of Investment (ROI), Throughput
09/06/2017 2.52.04	Lead time, Cycle Time, Net Promoter Score (NPS), Throughput
09/06/2017 8.06.45	Velocity

Continued on next page

Table A.3 – *Continued from previous page*

Time-stamp	**What metrics do you think are useful?**
09/06/2017 8.21.32	Velocity, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics, Others
09/06/2017 8.49.30	Others
09/06/2017 8.59.37	Release burn-down, Cycle Time, Throughput
09/06/2017 9.51.59	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics
09/06/2017 10.02.04	Velocity, Iteration burn-down, Lead time, Cycle Time, Quality metrics
09/06/2017 10.13.11	Cycle Time, Net Promoter Score (NPS), Quality metrics
09/06/2017 23.54.29	Velocity
10/06/2017 3.16.25	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics, Others
10/06/2017 4.13.06	Net Promoter Score (NPS), Quality metrics
10/06/2017 12.24.04	Velocity, Lead time, Cycle Time, Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 13.30.18	Iteration burn-down, Cycle Time, Cumulative Flow Diagram (CFD)
10/06/2017 13.32.39	Velocity, Iteration burn-down, Release burn-down
10/06/2017 13.53.39	Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics, Others
10/06/2017 13.53.48	Velocity, Internal Rate of Return (IRR), Return of Investment (ROI)
10/06/2017 14.29.04	Lead time, Cycle Time, Others
10/06/2017 15.07.49	Cycle Time, Return of Investment (ROI), Quality metrics

Continued on next page

Table A.3 – *Continued from previous page*

Time-stamp	**What metrics do you think are useful?**
10/06/2017 15.15.51	Velocity, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 15.46.29	Velocity, Iteration burn-down, Release burn-down, Net Promoter Score (NPS)
10/06/2017 16.23.49	Velocity, Iteration burn-down, Release burn-down, Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 16.38.21	
10/06/2017 17.29.06	Iteration burn-down, Release burn-down
10/06/2017 17.57.01	Velocity, Release burn-down, Quality metrics
10/06/2017 18.15.14	Velocity, Iteration burn-down, Release burn-down
10/06/2017 18.37.40	Iteration burn-down, Release burn-down, Cycle Time, Throughput, Quality metrics
10/06/2017 18.47.24	Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Quality metrics
10/06/2017 19.36.21	Lead time, Cycle Time, Return of Investment (ROI), Quality metrics
10/06/2017 20.02.20	Velocity, Release burn-down, Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 20.03.24	Velocity, Lead time, Quality metrics
10/06/2017 20.07.33	Iteration burn-down
10/06/2017 20.23.10	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Return of Investment (ROI), Throughput, Quality metrics
10/06/2017 20.36.03	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics, Others
10/06/2017 20.40.31	Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Throughput, Quality metrics
10/06/2017 21.19.03	Iteration burn-down, Release burn-down, Lead time, Cycle Time, Return of Investment (ROI), Throughput, Quality metrics, Others

Continued on next page

Table A.3 – *Continued from previous page*

Time-stamp	**What metrics do you think are useful?**
10/06/2017 21.23.27	Velocity, Iteration burn-down, Release burn-down
10/06/2017 21.32.29	Net Promoter Score (NPS), Return of Investment (ROI)
10/06/2017 23.48.16	Velocity, Throughput
11/06/2017 0.35.29	Velocity, Iteration burn-down, Internal Rate of Return (IRR), Quality metrics
11/06/2017 1.27.07	Velocity, Release burn-down, Cumulative Flow Diagram (CFD), Quality metrics
11/06/2017 3.26.54	Quality metrics
11/06/2017 6.26.53	Velocity, Net Promoter Score (NPS), Quality metrics
11/06/2017 8.14.38	Velocity, Release burn-down, Quality metrics
11/06/2017 8.54.24	Velocity, Cycle Time
11/06/2017 10.26.23	Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Throughput
11/06/2017 10.50.29	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics, Others
11/06/2017 13.09.47	Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Return of Investment (ROI), Throughput, Quality metrics
11/06/2017 13.38.33	Others
11/06/2017 13.55.06	
11/06/2017 14.56.49	Velocity, Lead time, Cycle Time, Return of Investment (ROI), Throughput, Quality metrics
11/06/2017 17.12.20	Lead time, Cycle Time
11/06/2017 20.30.08	Others
11/06/2017 21.26.02	Velocity, Iteration burn-down, Release burn-down
12/06/2017 1.03.09	Velocity, Iteration burn-down, Cumulative Flow Diagram (CFD), Quality metrics
12/06/2017 8.52.34	Cumulative Flow Diagram (CFD), Throughput
12/06/2017 9.12.41	Release burn-down, Return of Investment (ROI), Quality metrics
12/06/2017 13.41.29	Return of Investment (ROI)

Continued on next page

Table A.3 – *Continued from previous page*

Time-stamp	**What metrics do you think are useful?**
12/06/2017 15.55.55	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Cumulative Flow Diagram (CFD), Net Promoter Score (NPS), Internal Rate of Return (IRR), Return of Investment (ROI), Throughput, Quality metrics
12/06/2017 22.06.25	Internal Rate of Return (IRR), Return of Investment (ROI), Quality metrics
13/06/2017 7.07.54	Velocity, Iteration burn-down, Release burn-down
13/06/2017 12.00.31	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Quality metrics
13/06/2017 17.04.41	Velocity, Iteration burn-down, Release burn-down, Net Promoter Score (NPS)
13/06/2017 22.19.40	Iteration burn-down, Cycle Time, Net Promoter Score (NPS), Return of Investment (ROI)
14/06/2017 14.14.53	Velocity, Iteration burn-down, Release burn-down, Lead time, Cycle Time, Quality metrics
14/06/2017 22.02.18	Velocity, Iteration burn-down, Release burn-down, Cumulative Flow Diagram (CFD), Return of Investment (ROI)

Table A.3: Answers to question three (Q3).

A.2.4 Answers to Q4

The following list is an exact reproduction of the nineteen (19) answers obtained in question 4 (**Q4**) of the survey:

- Business Value Delivered
- Scope change
- Deployment rate, business value delivered, employee satisfaction
- The problem is not whether these things are useful (or indeed whether they are metrics) - it is whether and under what circumstances they give significant value to offset their down-sides and their cost. Tools to give visibility are useful, but start using them as metrics and they become damaging.
- Developers satisfaction

- Happiness score
- We use the sprint speed metric to see if we are on target: *https://pm.stackexchange.com/questions/21527/alternatives-to-sprint-burndown-is-it-deprecated/21530#21530*
  It is simple and it leads to actions, burns-downs suck :)
- I call it E-NPS (Employee NPS), win/loss rate of iterations, acceleration (based on velocity trend), % of forecast done, escaped defects (into production), financial metrics, sustainability, value stream analysis,....
- Relative business value estimated, relative business value delivered
- 1) User value received. Development org value received.
- On scope, automation coverage, code coverage
- Team Happiness index, Customer satisfaction
- Team Satisfaction, how many bugs. Division of type of work
- Frequency of deploying new functionality to the customer
- Please see my paper presentation in my profile where I had explained about Agile Metrics
- The metrics you have focused on using here are quantitative metrics. When you truly work in an agile way you "metrics" are of a more qualitative nature. So if I was to give you an example. On the projects I work on a measure success by the value it is giving to our customers. Is what I have built being used? do they get use out of it?what can I do to improve it? what don't they like? I also use metrics such as the secondary impact of the release into production onto the business itself... i.e has it reduced calls to a call centre if so by how much and what is the knock on effect to that. But the value is in reducing calls to call centre and allowing customers to self serve giving a happier customer. What I try and get clients to care about less is the metrics used to help get code to production and delivery... Which is most of the options you have in your questions. The reason for this is because what I could be building maybe of no use at all... So why would I then say it is a success just because we built what you asked for, rather than what was needed. I hope this feedback is at least a bit more constructive to

you than some of the feedback you have received on LinkedIn. From a delivery manager based in the UK.

- Value outcome
- Customer satisfaction, number of users, employee satisfaction
- Velocity deviation which helps establish how predictable a team is..this is why I have ticked velocity, velocity as a number is not a metric

Bibliography

[1] 2016 state of scrum report. Technical report, Scrum Alliance, 2017.

[2] Lukas Alperowitz, Dora Dzvonyar, and Bernd Bruegge. Metrics in agile project courses. In *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*. ACM Press, 2016.

[3] Elvira Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, Matthias Galster, and Paris Avgeriou. A mapping study on design-time quality attributes and metrics. *Journal of Systems and Software*, 127:52–77, may 2017.

[4] Kent Beck and Cynthia Andres. *Extreme Programming Explained*. Addison Wesley, 2004.

[5] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. *Software Development*, 9(8):28–35, 2001.

[6] Woubshet Nema Behutiye, Pilar Rodríguez, Markku Oivo, and Ayşe Tosun. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology*, 82:139–158, feb 2017.

[7] Martin P. Boerman, Zeeger Lubsen, Damian A. Tamburri, and Joost Visser. Measuring and monitoring agile development status. In *Proceedings of the Sixth International Workshop on Emerging Trends in Software Metrics*, WETSoM '15, pages 54–62, Piscataway, NJ, USA, 2015. IEEE Press.

[8] Mike Cohn. *User Stories Applied*. Addison Wesley, 2004.

[9] Mike Cohn. *Agile Estimating and Planning.* Prentice Hall, 2005.

[10] Mike Cohn. *Succeeding with Agile.* Addison Wesley, 2009.

[11] Ken W. Collier. *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing (Agile Software Development Series).* Addison-Wesley Professional, 2011.

[12] Giulio Concas, Michele Marchesi, Giuseppe Destefanis, and Roberto Tonelli. An Empirical Study of Software Metrics For Assessing the Phases of an Agile Project. *International Journal of Software Engineering and Knowledge Engineering*, 22(04):525–548, jun 2012.

[13] Subhajit Datta. Agility measurement index. In *Proceedings of the 44th annual southeast regional conference on - ACM-SE 44.* ACM Press, 2006.

[14] Christopher W. H. Davies. *Agile Metrics in Action: How to Measure and Improve Team Performance.* Manning, 2015.

[15] Giuseppe Destefanis, Steve Counsell, Giulio Concas, and Roberto Tonelli. Agile processes in software engineering and extreme programming. In Giovanni Cantone and Michele Marchesi, editors, *Lecture Notes in Business Information Processing*, chapter Software Metrics in Agile Software: An Empirical Study, pages 157–170. Springer-Verlag, Berlin, Heidelberg, 2014.

[16] Scott Downey and Jeff Sutherland. Scrum metrics for hyperproductive teams: How they fly like fighter aircraft. In *2013 46th Hawaii International Conference on System Sciences.* IEEE, jan 2013.

[17] Yael Dubinsky and Orit Hazzan. Using a roles scheme to derive software project metrics. In *Proceedings of the 2004 Workshop on Quantitative Techniques for Software Agile Process*, QUTE-SWAP '04, pages 34–39, New York, NY, USA, 2004. ACM.

[18] Robert Galen. *Agile Reflections: Musings Toward Becoming "Seriously Agile" in Software Development.* RGCG, LLC, 2012.

[19] Robert Galen. *Scrum Product Ownership: Balancing Value from the Inside Out.* RGCG, LLC, 2013.

[20] Ilan Goldstein. *Scrum Shortcuts without Cutting Corners.* Addison Wesley, 2013.

[21] Yiwei Gong and Marijn Janssen. Measuring process flexibility and agility. In *Proceedings of the 4th International Conference on Theory and Practice of Electronic Governance - ICEGOV '10.* ACM Press, 2010.

[22] Warren Harrison. A flexible method for maintaining software metrics data: a universal metrics repository. *Journal of Systems and Software*, 72(2):225–234, jul 2004.

[23] D. Hartmann and R. Dymond. Appropriate agile measurement: Using metrics and diagnostics to deliver business value. In *AGILE 2006 (AGILE'06).* IEEE, 2006.

[24] David I Heimann, Peter Hennessey, and A Tripathi. A bipartite empirically-oriented metrics process for agile software development. *ASQ Software Quality Professional*, 9(2):36–48, 2007.

[25] André Janus, Andreas Schmietendorf, Reiner Dumke, and Jens Jäger. The 3c approach for agile quality assurance. In *Proceedings of the 3rd International Workshop on Emerging Trends in Software Metrics*, WETSoM '12, pages 9–13, Piscataway, NJ, USA, 2012. IEEE Press.

[26] Henrik Kniberg. *Scrum and XP from the Trenches - 2nd Edition.* LULU PR, 2015.

[27] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum - Making the Most of Both.* AL LAVALLIS ENTERPRISES LLC, 2010.

[28] Oualid Ktata and Ghislain Lévesque. Designing and implementing a measurement program for scrum teams: What do agile developers really need and want? In *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, C3S2E '10, pages 101–107, New York, NY, USA, 2010. ACM.

[29] Martin Kunz, Reiner R. Dumke, and Niko Zenker. Software metrics for agile software development. In *Proceedings of the 19th Australian Conference on Software Engineering*, ASWEC '08, pages 673–678, Washington, DC, USA, 2008. IEEE Computer Society.

[30] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. Why are industrial agile teams using metrics and how do they use them? In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*, WETSoM 2014, pages 23–29, New York, NY, USA, 2014. ACM.

[31] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. Using metrics in agile and lean software development – a systematic literature review of industrial studies. *Information and Software Technology*, 62:143–163, jun 2015.

[32] Mitch Lacey. *The Scrum Field Guide.* Addison Wesley, 2015.

[33] Valentina Lenarduzzi, Ilaria Lunesu, Martina Matta, and Davide Taibi. Functional size measures and effort estimation in agile development: A replicated study. In *Lecture Notes in Business Information Processing*, pages 105–116. Springer International Publishing, 2015.

[34] V. Mahnic. A case study on agile estimating and planning using scrum. *Electronics and Electrical Engineering*, 111(5), jun 2011.

[35] V Mahnic and Ivan Vrana. Using stakeholder driven process performance measurement for monitoring the performance of a scrum-based software development process. *Elektrotehniski vestnik*, 74(5):241–247, 2007.

[36] V. Mahnic and N. Zabkar. Measuring progress of scrum-based software projects. *Electronics and Electrical Engineering*, 18(8), oct 2012.

[37] Viljan Mahnic and Natasa Zabkar. Measurement repository for scrum-based software development process. In *Proceedings of the 2Nd WSEAS International Conference on Computer Engineering and Applications*, CEA'08, pages 23–28, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[38] Viljan Mahnic and Natasa Zabkar. Using cobit indicators for measuring scrum-based software development. *W. Trans. on Comp.*, 7(10):1605–1617, October 2008.

[39] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker, and Hasso Plattner. Scrumlint: Identifying violations of agile practices using development artifacts. In *Proceedings of the 9th International*

Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '16, pages 40–43, New York, NY, USA, 2016. ACM.

[40] Deepti Mishra, Eda Balcioglu, and Alok Mishra. Measuring project and quality aspects in agile software development. *Technics Technologies Education Management*, 7(1):122–127, 2012.

[41] Sanjay Misra and Martha Omorodion. Survey on agile metrics and their inter-relationship with other traditional development metrics. *SIGSOFT Softw. Eng. Notes*, 36(6):1–3, November 2011.

[42] Alessandro Murgia, Giulio Concas, Sandro Pinna, Roberto Tonelli, and Ivana Turnu. Empirical study of software quality evolution in open source projects using agile practices. -, 2009.

[43] Nancy Y. Nee. Metrics for agile projects: finding the right tools for the job. *PMI® Global Congress 2010*, September 2010.

[44] Marta Olszewska (née Pląska), Jeanette Heidenberg, Max Weijola, Kirsi Mikkonen, and Ivan Porres. Quantitatively measuring a large-scale agile transformation. *Journal of Systems and Software*, 117:258–273, jul 2016.

[45] David Nicolette. *Software Development Metrics.* Manning, 2015.

[46] Frederico Oliveira, Alfredo Goldman, and Viviane Santos. Managing technical debt in software projects using scrum: An action research. In *2015 Agile Conference*. IEEE, aug 2015.

[47] Andrew J Oswald, Eugenio Proto, and Daniel Sgroi. Happiness and productivity. *Journal of Labor Economics*, 33(4):789–822, 2015.

[48] K. Petersen and C. Wohlin. Measuring the flow in lean software development. *Software: Practice and Experience*, 41(9):975–996, apr 2010.

[49] Andrew Pham and Phuong-Van Pham. *Scrum in Action: Agile Software Project Management and Development.* COURSE TECHNOLOGY, 2011.

[50] Roman Pichler. *Agile Product Management with Scrum.* Addison Wesley, 2010.

[51] Mary Poppendieck and Tom Poppendieck. *Lean Development Software*. Addison Wesley, 2003.

[52] Ken Power. Metrics for understanding flow. In *Agile Software Development Conference(Agile 2014), Orlando, FL, USA*, 2014.

[53] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education - Europe, 2014.

[54] Analysis.Net Research. The 11th annual state of agile report. Survey report, VersionOne Inc., 2016.

[55] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011.

[56] Kenneth S. Rubin. *Essential Scrum*. Addison Wesley, 2012.

[57] Akinori Sakata. Niko-niko Calendar. `http://www.geocities.jp/nikonikocalendar/index_en.html`. [Online; Accessed: 10-July-2017].

[58] Marco Scotto, Alberto Sillitti, Giancarlo Succi, and Tullio Vernazza. Non-invasive product metrics collection. In *Proceedings of the 2004 workshop on Quantitative techniques for software agile process - QUTE-SWAP '04*. ACM Press, 2004.

[59] Pedro Serrador and Jeffrey K. Pinto. Does agile work? — a quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040–1051, jul 2015.

[60] Alberto Sillitti, Barbara Russo, Paolo Zuliani, and Giancarlo Succi. Deploying, updating, and managing tools for collecting software metrics. In *Proceedings of the 2004 workshop on Quantitative techniques for software agile process - QUTE-SWAP '04*. ACM Press, 2004.

[61] Miroslaw Staron and Wilhelm Meding. Monitoring bottlenecks in agile and lean software development projects – a method and its industrial use. In *Product-Focused Software Process Improvement*, pages 3–16. Springer Berlin Heidelberg, 2011.

[62] Andrew Stellman and Jennifer Greene. *Learning Agile*. O'Reilly UK Ltd., 2015.

[63] Rod Stephens. *Beginning Software Engineering.* John Wiley & Sons Inc, 2015.

[64] Jeff Sutherland. Jeff sutherland's scrum handbook. *Scrum Training Institute*, page 66, 2010.

[65] Jeff Sutherland and Ken Schwaber. The scrum guide. *The Definitive Guide to Scrum: The Rules of the Game. Scrum. org*, page 17, July 2016.

[66] C.J. Torrecilla-Salinas, J. Sedeño, M.J. Escalona, and M. Mejías. Estimating, planning and managing agile web development projects under a value-based perspective. *Information and Software Technology*, 61:124–144, may 2015.

[67] Daniel Vacanti and Bennet Vallet. Actionable metrics at siemens health services. *Agile Conference (AGILE), 2014*, 2014.

[68] Daniel S. Vacanti. *Actionable Agile Metrics for Predictability: An Introduction.* Daniel S. Vacanti, Inc., 2015.

[69] Stacia Viscardi. *The Professional Scrummaster's Handbook.* Packt Publishing, 2013.

[70] Harikesh Bahadur Yadav and Dilip Kumar Yadav. Construction of membership function for software metrics. *Procedia Computer Science*, 46:933–940, 2015.