FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Metrics-based evaluation and improvement of source code design

João Pedro Bandeira Fidalgo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Filipe Alexandre Pais de Figueiredo Correia

Second Supervisor: Eduardo Guerra

July 23, 2021

Metrics-based evaluation and improvement of source code design

João Pedro Bandeira Fidalgo

Mestrado Integrado em Engenharia Informática e Computação

July 23, 2021

Abstract

Analytics has been emerging, and there are various tools to gather data from the source code, which helps to have a better insight into software quality issues. Software Analytics is an analytics process that aims to improve the software development process and software quality. Large companies currently maintain and develop many software projects with different characteristics, which can be a problem to maintain and improve if the project is complex or extensive. The motivation for this work is the necessity of having a more informed way to improve software quality during the product's development.

To inform the teams about the current quality of the software, they can implement a Software Quality model that will measure the quality of software according to a set of quality elements. Several Software Quality models were developed, such as McCall's model, Boehm's model, ISO/IEC 9126, Quamoco, and Q-Rapids. Although these quality models can give a good measurement of the software quality to the teams, they do not provide a way to assess and improve it. That is why it is important to have assessment methods to assess the code, get insights into what is wrong, and fix those problems. Our review found EMISQ, SCQAM, CQMM, GQM, and SA Canvas, where the EMISQ and SCQAM are already used in a large company, Siemens, but they require an expert team to do the assessment job. CQMM, GQM, and SA Canvas appear to monitor and improve the code quality continuously. In this dissertation, we use SA Canvas as the base of our work.

Based on our state-of-the-art revision, we formulated a hypothesis and formulated four research questions to prove our hypothesis. With this work, we aim to gather more profs of the usefulness and ease-of-use of the SA Canvas and analyse its impact on the decision-making process and code quality. To do this, we conducted an industrial study with a software development team formed by 3 members. In order to obtain the participants' feedback while being able to explain and clarify the usage of the SA Canvas, we decided to use Cooperative Method Development as our research methodology, where we work together with the team.

This study lasted for 12 weeks, which were equivalent to 3 full sprints of the team and the start of the next one. To obtain data to our study we conducted a several surveys. At the end of this study, we analysed the surveys' results, the SA Canvas itself, the difficulties felt during the meetings with the teams, and the evolution of the code metrics to give answers to the research questions. During this study, we have identified a possible new pattern, the BROAD-SPECTRUM DIAGNOSTIC that aims to help the team to be more aware of the kind of information it can generate through the use of metrics and SA, and the issues that can be addressed with its analysis. In terms of answers to the research questions, the study results showed that the SA Canvas was hard to understand at the beginning of its usage, but with the adoption of the new pattern, the team started to understand its objectives and usage. Their answers to the sprints surveys showed that they were neutral in terms of ease-of-use and considered the approach useful for the project. In terms of impact in their decision-making and impact in the code quality, the team perceived benefits in the practices behind the SA Canvas, but not so much in the SA Canvas itself. It did not impact the code quality during the time of the study, but the decisions derived from its usage will, in the future of the project. In the end, we conducted a refinement session which led to some suggestions for the SA Canvas from the participants of the study that could be considered if verified in more future studies.

Keywords: Software analytics, software analytics canvas, internal software quality, code quality, large company environment, code metrics, software analytics patterns

Resumo

Analytics tem vindo a emergir, e existem várias ferramentas para recolher dados a partir do código-fonte, o que ajuda a ter uma melhor perceção dos problemas de qualidade do software. Software Analytics é um processo analítico que visa melhorar o processo de desenvolvimento de software e a qualidade do software. Atualmente, grandes empresas mantêm e desenvolvem um grande número de projectos de software, com características diferentes, o que pode ser um problema para manter e melhorar caso o projecto seja complexo ou extenso. A motivação para este trabalho é a necessidade de ter uma forma mais informada para melhorar a qualidade do software durante o desenvolvimento do produto.

Para informar as equipas sobre a qualidade atual do software, pode ser implementado um modelo de Qualidade de Software, que irá medir a qualidade do software de acordo com um conjunto de elementos de qualidade. Atualmente, já foram desenvolvidos vários modelos de Qualidade de Software, tais como o modelo McCall, o modelo Boehm, ISO/IEC 9126, Quamoco, e Q-Rapids. Embora estes modelos de qualidade possam dar uma boa medição da qualidade do software às equipas, não proporcionam uma forma de a avaliar e de a melhorar. É por isso que é importante dispor de métodos de avaliação para avaliar o código, obter conhecimentos sobre o que está errado, e corrigir esses problemas. Na nossa análise encontramos os métodos: EMISQ, SCQAM, CQMM, GQM, e SA Canvas, onde o EMISQ e SCQAM já são utilizados numa grande empresa, a Siemens, mas requerem uma equipa de peritos para fazer o trabalho de avaliação. O CQMM, GQM, e SA Canvas são métodos para monitorizar e melhorar continuamente a qualidade do código. Nesta dissertação, utilizamos o SA Canvas como base do nosso trabalho.

Com base na nossa revisão do estado da arte, formulámos uma hipótese e quatro perguntas de investigação para provar a nossa hipótese. Com este trabalho, pretendemos reunir mais provas da utilidade e facilidade de utilização do SA Canvas e analisar o seu impacto no processo de tomada de decisão e na qualidade do código. Para o efeito, realizámos um estudo industrial com uma equipa de desenvolvimento de software, formada por 3 membros. A fim de obter o feedback dos participantes ao mesmo tempo que explicavamos e utilizavamos em conjunto com a equipa o SA Canvas, decidimos utilizar o Cooperative Method Development como metodologia de investigação.

Este estudo durou 12 semanas, o equivalente a 3 sprints completos da equipa e ao início do sprint seguinte. De forma a obter dados para o nosso estudo realizamos vários questionários. No final deste estudo, analisámos os resultados dos inquéritos, o próprio SA Canvas, as dificuldades sentidas durante as reuniões com as equipas, e a evolução das métricas do código para dar respostas às perguntas da investigação. Durante este estudo identificámos um possível novo padrão, o BROAD-SPECTRUM DIAGNOSTIC que visa ajudar a equipa a estar mais consciente do tipo de informação que pode gerar através da utilização de métricas e de SA, e das questões que podem ser abordadas com a sua análise. Em termos de respostas às perguntas da investigação, os resultados do estudo mostraram que o SA Canvas foi difícil de compreender no início da sua utilização, mas, com a adoção do novo padrão, a equipa começou a compreendê-lo melhor e as suas respostas aos inquéritos de sprints mostraram foram neutras em termos de facilidade de utilização e consideraram a abordagem útil para o projeto. Em termos de impacto no seu processo de tomada de decisões e impacto na qualidade do código, a equipa viu alguns benefícios nas práticas por detrás do SA Canvas, mas não tanto no próprio SA Canvas, e acabou por não alterar a qualidade do código, mas é algo que com as decisões tomadas e implementação das alterações definidas irá ser alterado, melhorando assim a qualidade do código. No final, realizámos uma sessão de melhoramento que levou a algumas sugestões para o SA Canvas por parte dos participantes do estudo, que podem ser tomadas em conta caso se verifiquem em mais estudo no futuro.

Keywords: Software analytics, software analytics canvas, large companies, qualidade interna de software , qualidade de código, ambiente de empresa grande, métricas do código, padrões de software analytics

Acknowledgements

First, I must express my profound gratitude to my parents for being there when I needed, supporting and providing me with the means to complete my studies. Thanks to all my family for helping me and encourage me during this phase.

Thanks to all my friends and colleagues who helped me during these times and to all my close friends for advising and providing me with happy moments when I was down and unmotivated.

I'd also like to share my profound gratitude to Joelma Choma and my supervisors Eduardo Guerra, and Filipe Figueiredo Correia, for sharing their knowledge and guiding me during this dissertation period. They were always available to answer my doubts and my problems regarding this work.

Last but not least, thanks to the faculty of FEUP that helped me learn during these years and provided me with wonderful moments and people.

João Fidalgo

“Remember to celebrate milestones as you prepare for the road ahead”

Nelson Mandela

Contents

List of Figures

List of Tables

Abbreviations

SA	Software Analytics
EMISQ	Evaluation Method for Internal Software Quality
SCQAM	Structured Code Quality Assessment Method
CQMM	Code Quality Monitoring Method
GQM	The Goal, Question, Metric
SQL	Structured Query Language
CMD	Cooperative Method Development
TDD	Test Driven Development
RQ	Research Question
Q	Question

Chapter 1

Introduction

1.1 Context

An increasing number of companies are already using analytics as a part of their decision-making process by making decisions based on data collected [12]. The analytics' goal is to help decision-makers insight important and valuable information about a product that would be hidden without analytics [7]. An example is Web Analytics that has been used to inform the developers and managers about many aspects of their business, improving characteristics such as advertisement and user experience [22]. Web Analytics's difference from Software Analytics is that Web Analytics looks to improve business activities, and Software Analytics focuses on improving the software development process and software quality.

Large companies usually maintain and develop a large number of software projects with different characteristics. Maintaining and improving all of these projects can be an expensive and challenging task. With lack of time and pressure from stakeholders to deliver new features and the final product makes agile teams abandon the usage of analytics tools in their code because of the time these tools can take to use and understand their results [8]. So, agile teams tend to focus more on delivering the product and less on improving product quality by implementing improvements to their source code, such as refactoring and test coverage, which cause an increase in the product's technical debt. That is why it is essential to have a way to decrease that technical debt and increase software quality during its development without consuming much time from the product development activities.

1.2 Motivation

There are various analytics tools to gather data from the source code, which helps to better insight into what can be wrong with the code. However, the increase of software's complexity, extension, and lifetime makes the analytics process harder. After all, when dealing with large and complex code bases, teams can get lost analysing the results from those tools, making them abandon those analytics tools' usage because they do not find benefits. Also, if an undetected error by analytics tools appears, it can not be easy to find a practical and insightful solution rapidly. So it is essential to have an approach that helps teams organise analytics activities during software development. Nevertheless, this approach needs to be scalable to many different projects, as companies usually have several projects ongoing, with different complexities and criticalities. Usually, teams use their knowledge and past experiences to solve software issues instead of using decisions based on reasoning and analysis of the software project. The application of software analytics approaches helps teams solve this problem, promoting logic reasoning based on analytics tools' results.

1.3 Objectives

This dissertation's purpose is to conduct a study of the adoption of data-driven approaches for code quality improvement, like Software Analytics Canvas [8]. With this dissertation, we aim to:

- Introduce important concepts and present currently available methods related to Software Analytics, code quality improvement and Software Quality.
- Propose Software Analytics as part of the development process.
- Encourage teams to use Software Analytics approaches like SA Canvas.
- Conduct an industrial study of SA Canvas usage in different teams and projects in a large company context.
- Generate new refinements for the SA Canvas based on the participants' perceptions.

1.4 Dissertation Structure

The rest of this dissertation contains a background, state of the art review, and problem statement. Chapter 2 introduces important concepts for readers to understand better the rest of the dissertation. Chapter 3 presents approaches related to Software Analytics and Code Quality, briefly describing and comparing them. Chapter 4 describes the case study in detail, with its goals, design, execution, analysis of results, answers to the defined research questions, and also a new pattern discovered during the study. Finally, Chapter 5 presents the conclusions of our work, with a brief summary of what was done and achieved, the main contributions of our work and the possible future work that can be done.

Chapter 2

Background

In this chapter, we introduce essential concepts for readers to better understand the rest of the dissertation. We start by explaining what software quality is according to different sources and the impact on software. Then we introduce and explain the concept of technical debt that is important for this context, following a definition of software analytics and its use.

2.1 Software Quality

[29] defines software quality as "the degree to which a software product satisfies stated and implied needs", also [21] refers to it as "lack of bugs in the product". Software quality can impact the overall product. If the product does not meet the requirements, it will lack quality, and it will probably not function as it was supposed to. Different models appeared and defined software quality as a set of attributes, such as reliability, functionality and maintainability. These models are explained later in this dissertation in the State of the Art section. However, software quality is something abstract, it has no physical existence, and it is not constant because what can be a quality aspect for a project does not mean that it will be for another. So it is important to focus on the quality aspects that meet the project's demands [17].

Software quality divides into external and internal Software Quality. **External Software quality** refers to quality aspects that users identify, like usability and functionality [33]. **Internal Software Quality** covers aspects visible to developers, more focused on source code and documentation, and this is the one **addressed in this dissertation**. Internal Software Quality affects external aspects, even though users cannot see it [14].

Monitoring software quality can have a crucial impact on the overall development of software products and lead to better decisions when deciding the product's changes and improvements [16].

2.2 Technical Debt

Technical debt is defined as the consequences of limited software development [37]. This is a concept introduced by Ward Cunningham to explain how important is refactoring to the stakeholders and defined it as "not quite right code which we postpone making it right" [10]. However, it had a low degree of specification of what it is, and several authors expanded this concept, such as [27, 15]. [36] refers that technical debt has been used to describe "anything that stands in the way of deploying, selling, or evolving a software system".

Figure 2.1 shows how technical debt elements can organise, representing two types, the visible elements and the invisible elements. The visible represent elements visible to the software users, and the invisible represent elements mostly visible to the software developers and not to the users. Also, these elements can be organised as oriented to evolution or maintainability [24].

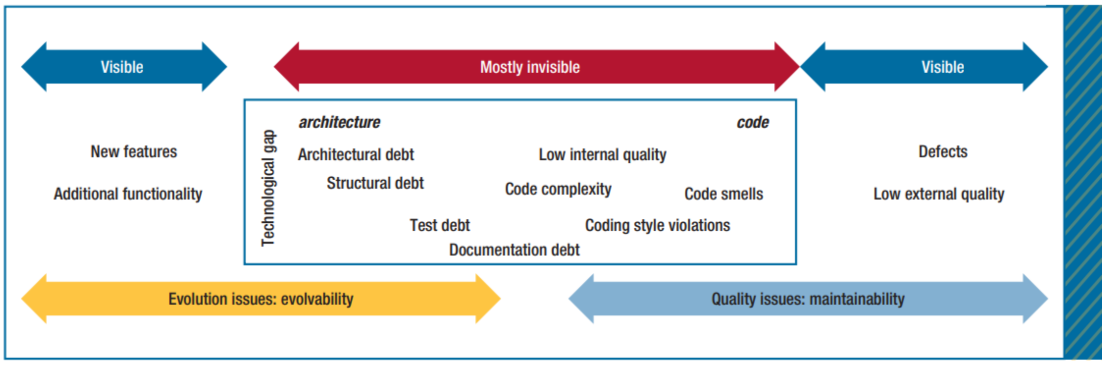


Figure 2.1: Technical debt elements organization [24]

Technical debt usually increases because of faster development approaches, where developers prioritise the delivery over the product's quality, which means that many agile teams are affected by accumulating technical debt. This can be caused by the lack of systematic testing and systematic assessment of quality issues, which can escalate the amount of technical debt in the software, making it harder to decrease the feature. So it is essential to be aware of the technical debt of software in the early stages of the development and continuously address these issues during the software's development [5].

2.3 Software Analytics

Software Analytics consists of analysing data from the software to obtain insightful and actionable information. [40] states that "Software analytics aims to obtain insightful and actionable information from software artefacts that help practitioners accomplish tasks related to software development, systems, and users.". The goal of software analytics passes by describing, monitor and improve software development and maintenance by analysing software data, such as **code, repositories, development environments, user reports and data** [6].

Several analytics tools have been developed to help analyse software data and produce automatic insights, such as [38, 11]. These tools help developers develop and get constant insights from software and see possible improvements to the overall software's quality.

Software Analytics can significantly impact decision-making processes in the present and future, as these processes are becoming harder with the tendency to grow on the size and complexities of projects[40]. So it is essential to find tools and approaches to use analytics activities during the all development of a product, and those tools need to be easy to use, implement, and with fast and consistent results [40]. However, there is a lack of approaches to monitoring, controlling, and introducing the analytics activities to the teams' development process, as we will see in Chapter 3.

2.4 Conclusions

This chapter introduced essential concepts for this dissertation. We can see that the first two are connected because the increase of software quality means the decrease of technical debt, as many of the debts pointed are connected with quality elements. This chapter also shows the importance of adopting analytics applied to software during the development process to help monitor and address the technical debt in the early stages of development and increase the overall software quality.

The next chapter addresses some approaches and methods to measure the software quality and systematically assess the software product using analytics tools to improve internal software quality.

Chapter 3

State of The Art

There are already multiple tools and methods to improve software quality. This chapter describes the state of the art of SA. Section 3.1 focuses on Software quality models present and used in many SA methods. Section 3.2 presents the SA methods, what they try to solve, a brief introduction of each method, similarities, differences, main problems found, and some key points worth mentioning.

3.1 Software Quality Models

Quality models define a set of characteristics, with relations among them to evaluate software quality. Quality models have been evolving throughout time. The firsts quality methods were created more than 40 years ago with basic quality definitions. **McCall's and Boehm's quality model** represent the firsts quality models developed. After this, one of the first standards of software quality appeared, ISO 9126 standard [4].

McCall's model [26] classifies quality considering: Product operation, Product revision, and Product transition [29]. Then the quality attributes are defined considering each product perspective.

Boehm's model presents a hierarchy of quality characteristics based on three dimensions: utility, maintainability, and portability.

3.1.1 ISO/IEC 9126 standard

ISO/IEC 9126 [3] is a group of standards to specify quality's characteristics and subcharacteristics of a software product. It also defines metrics associated with each of its subcharacteristics, that are used to measure those subcharacteristics. This standard divided the software quality into three types: internal quality, external quality, and quality in use. The defined characteristics and subcharacteristics are:

- **Functionality**: how capable the software is to meet needs and goals when used in certain circumstances. Suitability, accuracy, interoperability, security, functionality, and compliance represent functionality's subcharacteristics [20].

- **Reliability**: how capable the software is to maintain the same performance level when used in certain circumstances. Maturity, fault tolerance, recoverability, and reliability compliance represent reliability's subcharacteristics [20].

- **Usability**: how capable the software is to be learned and used by users when used in certain circumstances. Understandability, learnability, operability, attractiveness, and usability compliance represent usability's subcharacteristics [20].

- **Efficiency**: how capable the software is to provide the required performance, considering the number of resources used. Time behaviour, resource utilization, and efficiency compliance represent efficiency's subcharacteristics [20].

- **Maintainability**: how capable the software is to be modified. These changes can be improvements, environment changes and requirement changes. Analyzability, changeability, stability, testability, and maintainability compliance represent maintainability's subcharacteristics [20].

- **Portability**: how capable the software is to be transferred between environments. Adaptability, installability, replaceability, coexistence, and portability compliance represent portability's subcharacteristics [20].

Figure 3.1 provides an example of how the characteristics, subcharacteristics and metrics can relate with each other, in this case for Maintainability. The measurement of subcharacteristics uses multiple metrics, and these metrics can be used in various subcharacteristics' measurements.

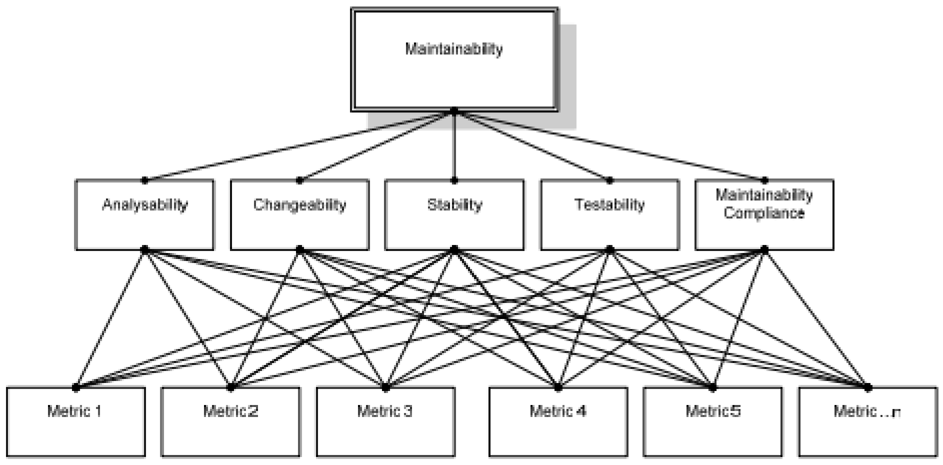


Figure 3.1: ISO/IEC 9126 Maintainability Hierarchy [1]

3.1.2 Quamoco Quality Model

Quamoco [39] is a model designed to close the gap between models that either provide abstract quality attributes or concrete quality assessments. The objective was to build a model that describes software quality that allows quality assessment. A meta-model was designed to specify the concepts that influence software quality.

Figure 3.2 shows a UML diagram that represents those concepts and their relations. The **Factor** resides in the centre, and it can be a **Quality Aspect** or a **Product Factor**. **Product Factors** have an impact on **Quality Aspects**. Every **Factor** has an evaluation associated and represents a characteristic of an **Entity**. **Entities** are aspects important for quality. The base model of Quamoco presents 112 entities and 286 factors, where 221 of those factors have evaluations. Studies were conducted to prove the usefulness of Quamoco in detecting quality differences on different systems.

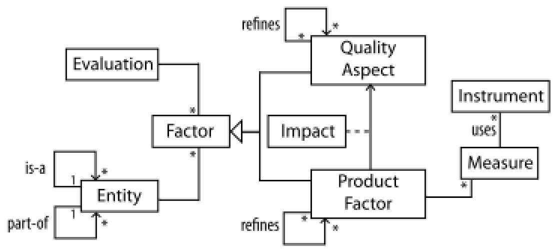


Figure 3.2: Quamoco meta quality Model [39]

3.1.3 Q-Rapids quality model

Q-Rapids quality model [25] was developed to help teams discover software problems during development and support the decision-making process. This model was created based on four use cases: Nokia, Bittium, Softeam, and ITTI. [19] defines a use case as a "Software system developed in an Rapid Software Development process" that uses Q-rapids as a quality model to provide help in decision making and quality management. This model's main elements are quality aspects, product and process factors, assessed metrics, and raw data. The process factors and product factors relate to different quality aspects like **maintainability, reliability, functional suitability, and productivity**. The product factors present in Q-Rapids are:

- **Code quality**: Usage of metrics like complexity, comment density, and duplication density, to monitor maintainability aspects of the code.

- **Blocking code**: Usage of metrics to determine the status of quality rules and technical debt. The identification of regularly changed files has higher chances of bringing problems to the project.

- **Testing Status**: Usage of metrics to determine test coverage and percentage of tests failed to verify the quality and stability of testing, which will help tests be more effective and significant.
- **Software stability**: Usage of metrics to detect crashes and errors during runtime, which will help developers have better reports of causes to the crashes and correct errors.
- **Software Usage**: Usage of metrics to detect user's usage of the software such as most used features and most used screens, which will help teams remove features that are not used and focus more on the most used ones.

The only process factor present in Q-Rapids is **Issues' velocity** that represents the velocity in which teams resolve a software project's issues. This factor helps stakeholders track the development team's productivity and improve the planning and estimation of future cycles.

3.1.4 Conclusion

Quality models provide a good measurement of software quality. They can be essential to verify the strong aspects and the quality weaknesses of a project, which can help teams organize better to improve those weaknesses. But they lack on how to implement the metrics needed and use the findings to improve. That's why Quamoco and Q-Rapids created assessment methods on top of their quality models, in order to assess and use the quality issues to suggest possible solutions and measures to take.

Although quality models can have an essential role in measuring software quality, teams need methods and tools to assess, insight and implement actionable changes to improve that software quality. The next section presents some of these methods and tools.

3.2 Systematic assessment methods

Systematic assessment methods include the usage of code inspection tools and manual inspection of code. They consist of a group of phases and steps to monitor and improve code quality, where the teams have an active role in identifying the problems and coming up with solutions to those problems. Some large companies already use these types of methods.

For example, Siemens, Corporate Technology, Development Center, Asia Australia, a company that develops software to different industry sectors, and with many large projects, already uses different proactive analytics methods, [33, 18, 35, 28], referred as quality assessment methods, that require an assessment team to analyse, get results and present them to the Project development team.

3.2.1 EMISQ: Evaluation Method for Internal Software Quality

EMISQ (Evaluation Method for Internal Software Quality) is used in several quality assessment projects, from mid-size to large projects, consisting of eight main activities, divided into five to ten subactivities [33]. This method appears as a base for two other methods, SCQAM and CQMM, explained latter in this section. The activities of EMISQ are:

- **Establish the purpose of evaluation** - identification and definition of goals for the evaluation project.
- **Identify types of projects** - Focuses on source code quality, and this activity proposes identifying what components will be assessed.
- **Specify quality model** - Definition of which quality model will be used, to evaluate the assessed project's overall quality.
- **Select metrics** - Metrics and tools that will be used to help identify possible problems and monitor future alterations.
- **Produce an evaluation plan** - Produces a detailed evaluation project plan.
- **Take measurements** - Implement code analysis tools and perform the manual inspection by the assessors.
- **Assess and document results** - Produce documentation about quality issues, measurement results. These documents contain an analysis of strengths, weaknesses, opportunities, and threads and recommendations to improve the quality issues.
- **Assure internal reuse of results** - The results of an evaluation project should be available to other projects instead of being considered only for one project.

3.2.2 SCQAM: Structured Code Quality Assessment Method

SCQAM (Structured Code Quality Assessment Method) was created to speed up the EMISQ method. To do that, SCQAM simplifies and automates most of the steps of EMISQ [18]. SCQAM consists of three phases:

- **The Input phase**, where the assessors ask the teams for code access, guidelines about the dependencies, and build environment.
- **The Process phase** that divides into 3 steps:
  - **Preparation step** - the team provides information about the project; the assessment team selects the most critical components in terms of weight in the business context; implements the automatic tools that analyse source code; defines the project team's dates and expectations.
  - **Measurement step** - implement, run, and save results from automatic tools.

- **Assessment step** - use an automatic tool, like COSMOS, to group all results from code analysis tools, in order to provide findings after analysing those results; perform a manual inspection of code, mainly on selected components, defined in the Preparation step; group all findings, and document them.

- **The Output phase** - present documented findings and critical problems to the Project team. After this, both teams work together to validate results and create actionable items.

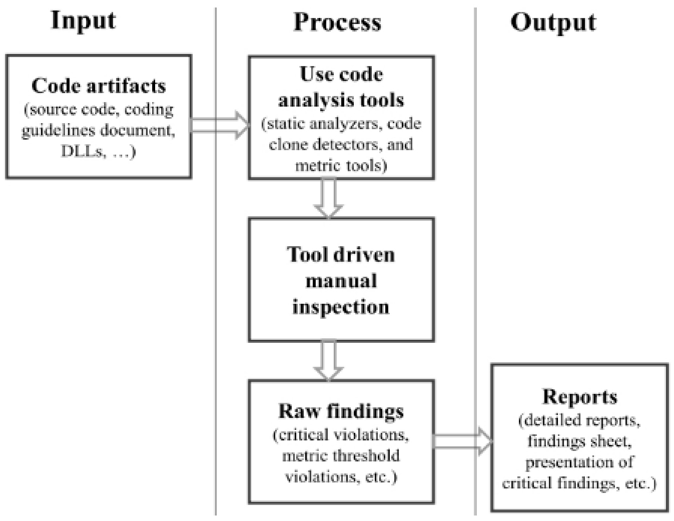

Input
Process
Output
Code artifacts (source code, coding guidelines document, DLLs, ...)
Use code analysis tools (static analyzers, code clone detectors, and metric tools)
Tool driven manual inspection
Raw findings (critical violations, metric threshold violations, etc.)
Reports (detailed reports, findings sheet, presentation of critical findings, etc.)

Figure 3.3: SCQAM Method's phases and process [18]

3.2.3 CQMM: Code Quality Monitoring Method

Code Quality Monitoring Method (CQMM) is meant to improve code quality by continuously monitor code quality during development [32]. Even though CQMM is built on top of EMISQ essentials, it differs from EMISQ because it is meant to the development team, and not for an expert assessment team. CQMM divides into 11 steps, grouped into three phases, as we can see in Figure 3.4. Those three phases consist of:

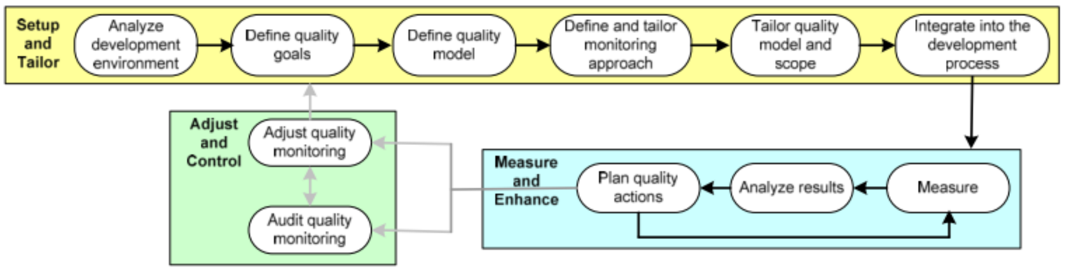

Setup and Tailor
Analyze development environment
Define quality goals
Define quality model
Define and tailor monitoring approach
Tailor quality model and scope
Integrate into the development process
Adjust and Control
Adjust quality monitoring
Audit quality monitoring
Measure and Enhance
Plan quality actions
Analyze results
Measure

Figure 3.4: CQMM's phases and steps [32]

- **Setup and Tailor phase** - divides into **six steps**, that basically are the selection and implementation of the analysis techniques and tools, definition of the quality goals and models that meet the project requirements and integration of the CQMM tasks into the development process.

- **Measure and Enhance phase** - the idea of continuous monitoring quality consists of repeating this phase periodically to ensure good code quality throughout the development process. This phase divides into **three steps** where the team measures and obtains the results from the static analysis tools, analyses those results, and plans the actions to take to improve and meet quality thresholds.

- **Adjust and Control phase** - divides into **two steps**, where the team adjusts the new requirements of a project, removing and adding metrics or rules and changing the monitoring tools. To ensure the changes to the quality monitoring project, an external quality assessor evaluates the actions taken, but this part can be optional but essential in critical projects.

Firstly, this method was tested on two pilot studies with different characteristics and sizes of 100k and 30k lines of code. After this, a more deep study was conducted with open source projects large scale projects, and these studies showed that the effort to use it and implement the tasks was too high on large projects [23].

3.2.4 GQM: Goal, Question, Metric

The Goal, Question, Metric (GQM) is meant to help teams define, trace and reach goals [2]. GQM consists of three levels:

- **Conceptual Level** - Goals are defined concerning different quality models, with several points of view. The objects that can be measured are the products, processes and resources.

- **Operation Level** - Use a set of questions to define how to achieve a specific goal. These questions relate to quality issues.

- **Quantitative level** - Use a group of metrics to answer the raised questions quantitatively. That data can be objective or subjective. Objective data represents data that only depends on the object measured. Subjective data relates to data that depends on the object measured, but also on the environment and viewpoint of the object.

Figure 3.5 shows that GQM has a hierarchical structure that begins in the goal divided into several questions, leading to metrics. These metrics can be common between different questions and goals. GQM specifies that the metrics should be defined after defining the questions related to some goals that we want to reach, which means that the metrics should answer the questions in order to reach the defined goals.

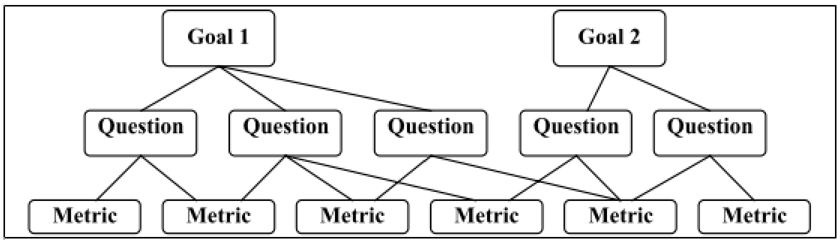


Figure 3.5: GQM hierarchical structure [2]

This method has expanded to more areas than software quality. [34] refers that it can also be used in case studies that involve the usage of metrics, where the researchers should define the goals of the study, then define the questions that will answer those goals, and then decide which metrics will provide them data that helps them answer the questions and reach the goals.

3.2.5 Software Analytics Canvas

Software Analytics Canvas (SA Canvas) is also a method oriented for project development teams, just as CQMM [8]. SA Canvas is a board tool that promotes understanding, discussion, creativity and analysis on a given subject [30]. SA Canvas is considered a **goal-focused approach**, and it is really similar to the Goal Question Metric (GQM) as it consists of firstly defining the goals and questions that we want to answer with the usage of metrics. SA Canvas supports the SA activities, where the teams can guide and understand their decisions related to software quality.

3.2.5.1 Software Analytics Patterns

SA Canvas was built based on a group of patterns described to help teams adopt SA practices in their development process. Some of these patterns described in [9] were defined because of the lack of patterns on how to adopt SA into an industrial environment. After that, when SA Canvas was built, more three patterns were found helpful to adopt in SA [8]. Figure 3.6 shows an overview of those patterns, followed by a brief description of each pattern.

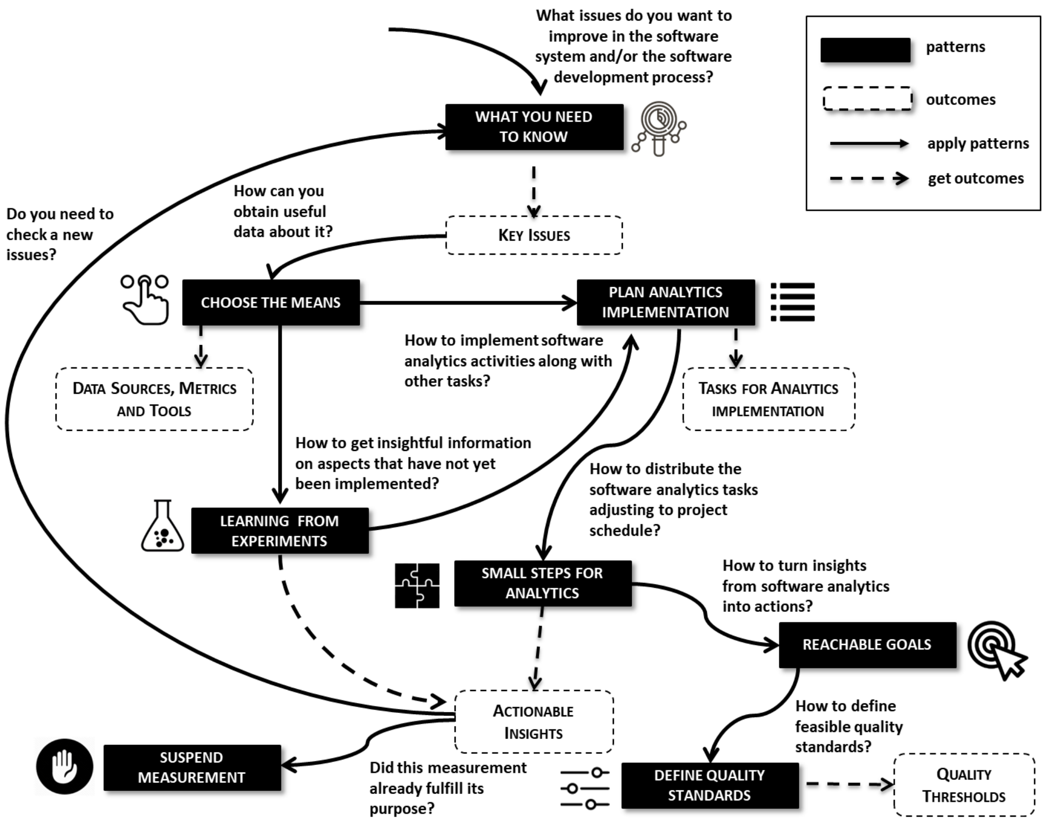

What issues do you want to improve in the software system and/or the software development process?
WHAT YOU NEED TO KNOW
patterns
outcomes
apply patterns
get outcomes
Do you need to check a new issues?
How can you obtain useful data about it?
KEY ISSUES
CHOOSE THE MEANS
PLAN ANALYTICS IMPLEMENTATION
How to implement software analytics activities along with other tasks?
DATA SOURCES, METRICS AND TOOLS
TASKS FOR ANALYTICS IMPLEMENTATION
How to get insightful information on aspects that have not yet been implemented?
How to distribute the software analytics tasks adjusting to project schedule?
LEARNING FROM EXPERIMENTS
SMALL STEPS FOR ANALYTICS
How to turn insights from software analytics into actions?
REACHABLE GOALS
ACTIONABLE INSIGHTS
How to define feasible quality standards?
SUSPEND MEASUREMENT
Did this measurement already fulfill its purpose?
DEFINE QUALITY STANDARDS
QUALITY THRESHOLDS

Figure 3.6: Overview of the patterns and their relationships [9]

1. **WHAT YOU WANT TO KNOW**: where the team defines the issues that need attention, and these will be a kind of guide in identifying which data needs to be extracted to help find solutions to those issues. These can be seen as questions that the team will try to answer and verify with metrics and analytics.

2. **CHOOSE THE MEANS**: based on the previous pattern, the team defines the tools and techniques to select and collect data useful for the specified issues.

3. **PLAN ANALYTICS IMPLEMENTATION**: is where the tasks related to SA are defined and prioritised according to the team's urgency for each task.

4. **SMALL STEPS FOR ANALYTICS**: the SA tasks are organised throughout the project planning, where each task can be completed in small steps. This pattern helps the team integrate the analytics tasks in their development process without impeding other activities. The teams in large companies have stricter deadlines and rules.

5. **REACHABLE GOALS**: where the team defines goals according to the SA findings. Those goals break into small actions in order to fit in the team's development activities and other tasks.

6. **Learning from Experiments**: where the team defines different solutions using different data to answer the questions raised. This can happen when the defined solution does not answer the questions raised.

7. **Define Quality Standards**: the team defines the maximum, and minimum values for the metrics collected and aims to meet those values in their project. This pattern helps the team to continually ensure that the project implementation meets the defined quality standards.

8. **Suspend Measurement**: when a measurement takes too much effort, costs too much time or other constraints, the team should decide if they keep using those measurements or suspend them. This pattern makes that all the metrics are necessary and valuable for the project.

3.2.5.2 Software Analytics Canvas Design

Figure 3.7 represents the first version of SA Canvas, and a brief description of each step is presented forward.

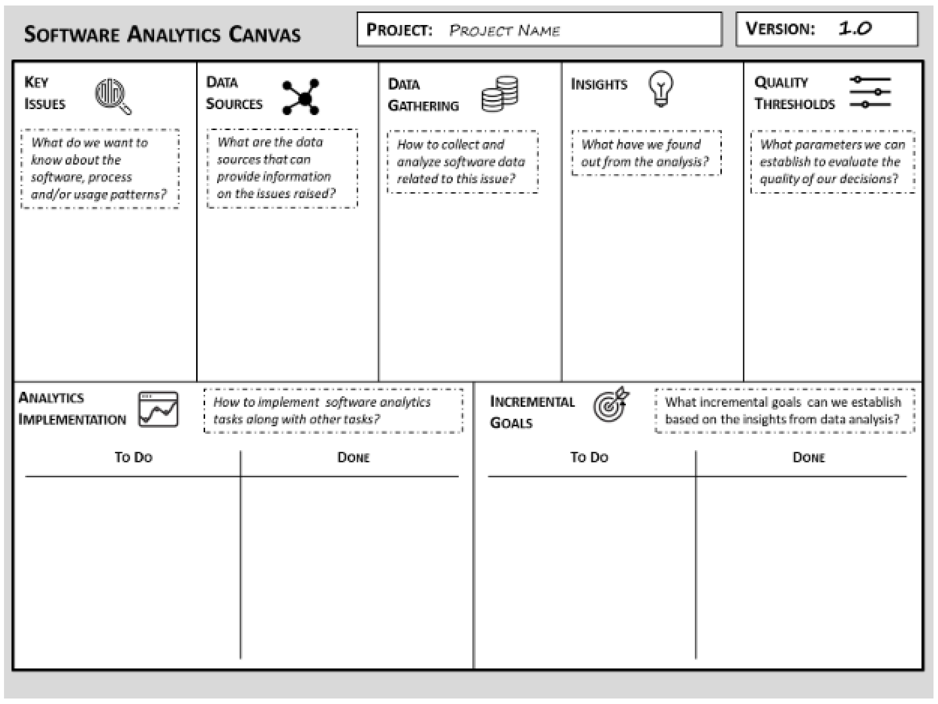

Software Analytics Canvas
Project: Project Name
Version: 1.0
Key Issues
What do we want to know about the software, process and/or usage patterns?
Data Sources
What are the data sources that can provide information on the issues raised?
Data Gathering
How to collect and analyze software data related to this issue?
Insights
What have we found out from the analysis?
Quality Thresholds
What parameters we can establish to evaluate the quality of our decisions?
Analytics Implementation
How to implement software analytics tasks along with other tasks?
To Do
Done
Incremental Goals
What incremental goals can we establish based on the insights from data analysis?
To Do
Done

Figure 3.7: Software Analytics Canvas Version 1.0 [8]

- **Key issues** - where the team raises issues that need to be verified, analysed and improved. The guiding question to help in this block is: "What is do we want to know about the software, process and/or usage patterns?". This step is related to the pattern What you need to know.

- **Data Sources** - where the team identifies the sources and data needed for understanding the issue raised. The guiding question to help in this block is: "What are the data sources that can provide information on the issues raised?". This element is related with two patterns: CHOOSE THE MEANS and LEARNING FROM EXPERIMENTS.

- **Data Gathering** - where the team decides which tools and metrics will be used to obtain the data, the methods and tools to analyse the data collected, and the tools to support their analysis. The guiding question to help in this block is: "How to collect and analyse software data related to this issue?". This step is related with the same patterns as the data sources.

- **Analytics Implementation** - where the team plans the analysis activities of the collected data in order to obtain valuable information. The analysis activities can be included in a to-do list and integrated into the development tasks. The guiding question to help in this block is: "How to implement software analytics tasks along with other tasks?". The patterns related with this block are: PLAN ANALYTICS IMPLEMENTATION and SMALL STEPS FOR ANALYTICS.

- **Insights** - where the team raises possible solutions based on the discussion and analysis of the data collected results. Suppose the analysis of those results does not reveal important information about the issue. In that case, the team decides if they continue to investigate using new tools and/or data or discard the issue. The guiding question to help in this block is: "What have we found out from the analysis?". This block is the element that leads to the REACHABLE GOALS pattern.

- **Incremental Goals** - where the team defines the goals in order to implement the solutions defined. The goals can be reached incrementally by integrating those goals into the product development goals. The guiding question to help in this block is: "What incremental goals can we define based on the insights from data analysis?". The pattern related with this step is the REACHABLE GOALS if the goals are not fulfilled, otherwise is SUSPEND MEASUREMENT.

- **Quality Thresholds** - where the team evaluates the impact of the changes, if those changes affect other features, and if the issue is solved. The stakeholders can participate in this step to help check if the issue is really solved. After this evaluation, if the issue is still there, the team needs to decide if they will try to re-analyse it using different tools and data or discard it. However, if the issue is solved, the team should decide if they want to monitor and implement measures to prevent that issue in the future. The guiding question to help in this block is: "What parameters we can establish to evaluate the quality of our decisions?". DEFINE QUALITY STANDARDS is the pattern that relates with this element.

After the development of SA Canvas, two studies were conducted to evaluate the method [8]. After these studies, the participants were asked for possible improvements to the method, where they suggested to change the blocks at the top of the canvas to *Key Issues*, *Data Sources*, *Incremental Goals*, and *Quality Threshold*, where this last one divides into two parts, the minimum acceptable value and goal to achieve. At the bottom of the canvas, *Analytics Implementation* changed to *Analytics Tasks* divided into four parts to organise the analytics tasks. After some more refinements of these suggestions, a new version of the SA Canvas was generated. Figure 3.8 shows the second version of the SA Canvas, after some refinements to it.

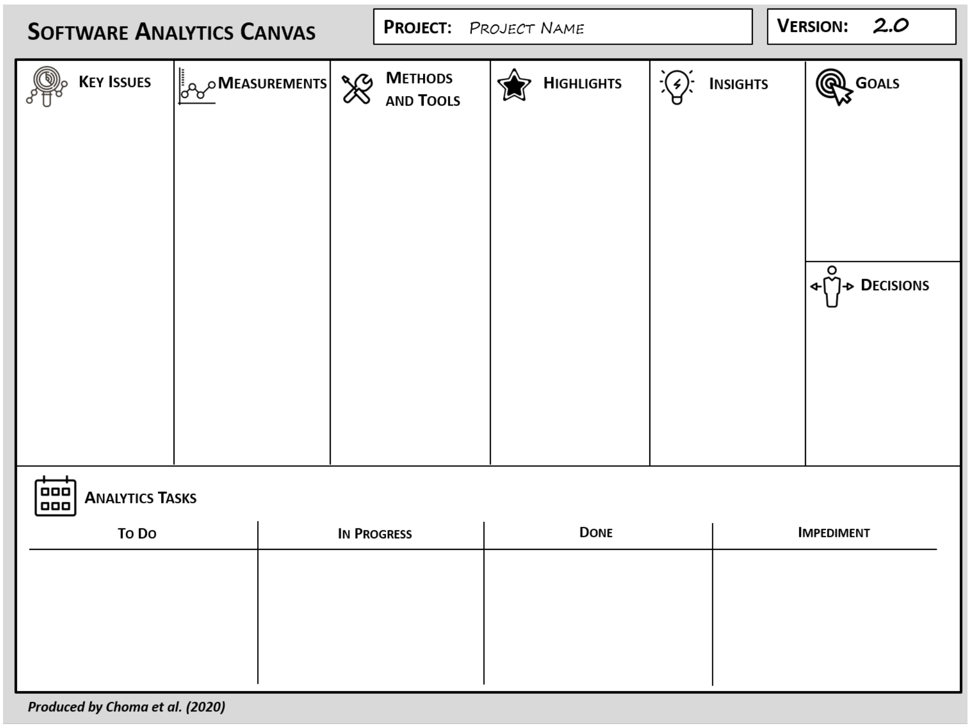


Figure 3.8: Software Analytics Canvas Version 2.0 [8]

This version of the SA Canvas has some differences from the first version, and the description of each step is:

- **Key Issues** - where the team needs to raise the issues that need to be verified, analysed and improved. This is the same as in the first version of the SA Canvas.
- **Measurements** - where the team identifies the sources and data needed for understanding the issue raised and will help them find answers or solutions to them. This element is related to the *Data Sources* element of the first version.
- **Methods and Tools** - where the teams decide how they will obtain the measurements defined, using tools or methods to get those metrics. This can be using a static analysis tool that provides some metrics or even using scripts to obtain specific information about the code. This step is the same as *Data Gathering* in the first version of the SA Canvas.

- **Highlights** - based on a first look at the metrics' values, the team will highlight values or information that are out of what they consider as normal. This is a new element in the second version of the SA Canvas.

- **Insights** - where the team will analyse more deeply the reasons behind the highlights. This is where the team will find causes to the highlights and possibly to the issues defined. This has some differences from the *Insights* element from the first version because this is more related to the analysis of the highlighted values.

- **Goals** - where the team defines the goals in order to implement the solutions defined. These goals can be reached incrementally in order to reach them in the future, as they can be goals hard to reach in a short period of time. This step is the same as in the first version.

- **Decisions** - this element refers to decisions taken to solve the issue defined. These decisions need to be taken as a team after the analysis of the *Insights* and *Highlights*.

- **Analytics Tasks** - where the team can organise the analytics tasks. The team needs to define the tasks needed to implement any of the elements in the top part of the SA Canvas. This element has some differences from the first version, where in the first version, the tasks would go from *To Do* to *Done* directly, and in this second version, we have 4 sections for the tasks. These tasks will be changed from *To Do* to *In Progress* and to *Done* if everything went normal. However, if during the performance of the task there is any problem that prevents it from being performed, then it will go to the *Impediment* section.

3.2.6 Conclusion

SCQAM and **EMISQ** are considered expert assessment methods because they require an expert assessment team to implement the method, allowing development teams to focus on developing the product. However, this means that instead of only having one team pointed to the product, they also require another one to use the assessment method to inform the project development team about quality issues. So there is a necessity to train teams to know how to use these methods to provide satisfactory results and increase quality.

CQMM, GQM, and **SA Canvas** were created for development teams usage and are considered goal-oriented methods. These three methods help teams to better organise towards code quality improvement. **CQMM** has the advantages of being flexible because the team decides all the tools to be used and is integrated into the development process. It guarantees that the analytics process is focused on the project goals and that all the tools are updated to new requirements. However, after a more deep study with open source projects and large scale projects, the method showed that the effort to use it and implement the tasks was too high on large projects.
SA Canvas shows favourable aspects for agile teams because it can be integrated as small tasks

into the agile development cycles, and it ensures continuous improvement of code quality. However, it still has a small number of studies with teams, requiring more practical studies with different projects characteristics and sizes to prove its usefulness and possibly new refinements resulting from those studies.

Chapter 4

Case Study

This chapter describes the case study in detail, following the guidelines in [34]. This chapter starts by describing the scope of our work, followed by the hypothesis that will be verified with the study. Then the goals of the study are presented along with the research questions. After this, the case study design is described in detail, followed by the analysis of the results and the answers to the research questions. Also, we present a possible new pattern in Software Analytics that we detected while conducting the study with the team. In the end, we show some threats to the validity of the study and a discussion of the case study.

4.1 Scope

The lack of time and the pressure from stakeholders to deliver new features causes teams to poorly assess their code quality, or not asses at all, making the product more prune to bugs, harder to maintain and develop new features. Many teams already use multiple static analysis tools, but often they get lost with the number of metrics those tools provide, because they can have troubles selecting which ones can be important for their projects, if there are any available on those tools, which can make the teams abandon or ignore its usage.

There are already many different models to measure software quality, helping teams to check their software quality. Still, these methods are not enough to assess and lead to improvements in their code quality, because they don't really guide the teams into the causes of their poor or good quality, neither help them solve their quality issues.

Several methods and approaches were developed to assess and improve internal software quality, more specifically 'code quality'. Even though EMISQ and SCQAM are used in a large company context, these methods require an expert team to assess the code, which can be a problem for many companies to afford because they might not have the means in terms of people or investment.

That's why approaches that guide the teams to continuously assess their code quality can play an important role in assessing the technical debt while developing the product. These approaches can lead the teams into creating a product with more quality while meeting the deadlines and have the advantage of not needing an expert team to do this job.

4.2 Goals

With this work, we aim to encourage teams to adopt data-driven approaches during their product development, and study the benefits of these approaches, such as the SA Canvas, in a large company environment. To do that, we performed a literature review to find current methods and approaches in use, where we compared the different approaches, pointing out the main problems and benefits to each one. With this literature review, we understood the main impediments and advantages of SA and have different perspectives on introducing SA into the teams' development process.

It is important for many software development teams to have an approach that guides them into continuously assessing their code quality, using metrics to get valuable insights into their code quality problems and possibly get some solutions. To address this, we use Software Analytics Canvas as a support of the SA activities. So we conduct an industrial study to validate the SA Canvas as support for the SA activities and be aware of possible changes in SA Canvas and possibly new software analytics' implementation patterns in the team development process.

4.3 Hypothesis

This dissertation is based on the following hypothesis:

"The adoption of Software Analytics Canvas results in a better organisation of the software analytics activities, more informed decisions, better insight into source code issues and an increase in code quality.

To clarify some points of this hypothesis:

- Better organisation means everyone in the team understand why the metrics were used, for what propose and what insights came from the use of them.
- The more informed decisions derives from the fact that decisions are based on insights from the source code derived from the usage of useful methods and metrics for the project.
- Code quality is based on a couple of measures like maintainability, reliability, readability and portability, that are measured with the help of static analysis tools.

To verify the validity of this hypothesis, we formulated four research questions presented in the next section.

4.4 Research Questions

This dissertation addresses the following research questions:

- **RQ1** *"What are the participants' perceptions about the usefulness and ease-of-use of the SA Canvas?"* - As it is a tool meant for the teams' organisation, this is an important point to evaluate.

- **RQ2** *"What are the participants' perceptions about the approach's benefits in their organisation and decision-making process into finding solutions and solving issues?"* - As this is a tool to support the SA activities, it is essential to know if it impacts the decision-making process of finding solutions to issues and solving them in a more informed way. So with this question, we aim to evaluate if the teams with the canvas feel that the SA Canvas impacts their organisation and their decision-making process when looking for solutions to issues and solving them. To do this, we also use the **participants' answers to the surveys** concerning this point.

- **RQ3** *"What is the impact of the adoption of SA Canvas in the code quality?"* - By answering to this question we can verify part of the hypothesis, by analysing the impact of the SA Canvas in the team's code quality, and if it really leads to an increase in the code quality.

- **RQ4** *"Which elements can be improved in SA Canvas?"* - As this tool is meant to be helpful to the teams, it is essential to know how the team feels about the tool at the end of its usage, where they can propose changes and improvements to it. This can be a very important question because it can lead to alterations that increase the tool's usefulness or remove some difficulties that the team felt during its usage.

4.5 Case Study Design

This section describes the study design in detail, describing the **participants**, **data sources**, **procedure**, **data collection**, and how the **data analysis** was performed. Figure 4.1 shows a diagram that gives an overview of how this study was designed.

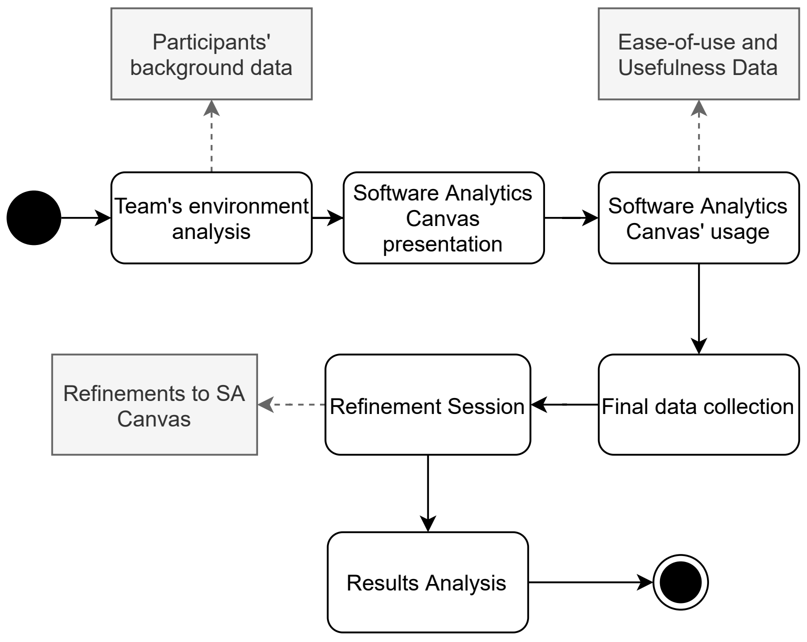

Participants' background data
Ease-of-use and Usefulness Data
Team's environment analysis
Software Analytics Canvas presentation
Software Analytics Canvas' usage
Refinements to SA Canvas
Refinement Session
Final data collection
Results Analysis

Figure 4.1: Study design diagram

4.5.1 Participants and Environment

The **participants** of this case study were from a small team of a large company **environment**. The team we worked with was formed by three members, the team leader and two more developers, where one had recently joined the team. During the study, one of the members changed, making a total of four participants. These are the participants directly involved in the SA Canvas' usage. The three members of the team were in charge of both backend and frontend development.Also, a business analyst was involved in the team's development process, that was in charge of passing the product owners' feedback and demands, building the user stories, defining priorities and validating the deliveries in each sprint. The study needed to be designed in order to fit their time available from the backend and frontend development. Table 4.1 presents the initial information gathered of that project.

Table 4.1: Project characteristics

Organisation	**Development**	**Description**
2 Developers 1 Team Leader 1 Business Analyst	Non-standard SCRUM, with sprint planning, sprint retrospective and daily meetings, but sprints with variable duration	Related with the financial sector

4.5.2 Data Collection

For this case study, we collected data from four different **data sources**: 1) **surveys** built using the Microsoft Forms [1]; 2) **SA Canvas** itself; 3) observations during the meetings where the SA Canvas was used; 4)**static analysis tools** like SonarQube [2].

For this case study, we created 4 different surveys:

- **Background survey** that was based on questions regarding their experience as developers, experience in the company, their definition of what Software Analytics is, the metrics used in the project, the importance that code quality has in their project and their decision-making process. It was composed of small open answer questions, some multiple answer questions and some Likert items with the format: 1) very rarely, 2) rarely 3) occasionally, 4) often, 5) very often.

- **Sprint surveys** that had questions concerning the participants' perceptions about the usefulness and ease-of-use of the SA Canvas during that sprint. This survey has an open answer question concerning their difficulties felt with the SA Canvas during that sprint, and mainly composed of Likert items with the format: 1) strongly disagree, 2) disagree, 3) neutral, 4) agree, 5) strongly agree.

[1]https://forms.office.com/
[2]https://www.sonarqube.org/

- **Final survey** that had the objective of gathering the final perceptions of the participants during all duration of the study. It is mainly related to the tool's usefulness and the team's intentions to maintain: the usage of the metrics defined in the SA Canvas, the SA as part of their development process, and the SA Canvas as support for the SA activities. This survey was only composed of Likert items with the same items as the Likert items in the sprint surveys.

- **Refinement Survey** with questions concerning the participants' perceptions of each element of the SA Canvas and suggestions to it. In this survey, the questions related with their perceptions were built using Likert items with the same items as in the sprint surveys, and the suggestions' questions were open answer questions for the participant to write down the details of their suggestions. It has one open answer for each element of the SA Canvas, where the participant will answer if they felt that element was not important or useful, and a final open answer question regarding their suggestions of changes to the SA Canvas.

The observations collected during the meetings were **the difficulties they felt during the SA Canvas' usage** and the **team's questions during these meetings**. Also, after analysing the surveys' answers, we would talk with the participants to understand better their answers if we felt that it was necessary.

To create the SA Canvas online and share it with the team, we used Lucid[3]. In the **SA Canvas**, we analysed the **number of tasks generated**, **sections of the SA Canvas with more usage** and **the overall changes from sprint to sprint**.

The **static analysis tool** used was SonarQube, and we looked for the **evolution in the metrics the team defined as important**. These were: 1) test coverage, 2) number of issues detected by SonarQube.

4.5.3 Procedure

In order to obtain the participants' feedback while being able to explain and clarify the usage of the SA Canvas, we decided to have a more active role in this process. So we decided to use **Cooperative Method Development** (CMD) as the research methodology [13]. CMD is an adaptation of action research oriented for software development, where the researcher cooperates with the participants to implement the innovations in their software. It consists of three phases: understanding, deliberating change, and implementation and evaluation of improvement.

In the first phase, **we analysed the team's environment**, where we collected information about their organisation, technologies used, analytics tools implemented, and main problems already known. To do this, we had **meetings with the team and with their coordinator**, and we ask them to answer a **survey concerning their background on SA**, some practices and other information that can impact the usage of the SA Canvas. This survey was conducted before explaining any concept related to SA and before any presentation of the SA Canvas.

[3]https://lucid.app/

In the second phase, we **introduce SA Canvas** to the team by firstly introducing some **concepts of SA** and then **presenting the SA Canvas itself along with practical examples** to help them understand better its objectives and usage. During this presentation, we answered the **participants' doubts** and try to understand the difficulties felt in the first impact with the SA Canvas.

In the third phase, the team **use SA Canvas as part of their development process for 12 weeks**, where we have the role of guiding and providing some suggestions during the meetings where the tool is used. In every sprint retrospective, we meet with the team to use the SA Canvas. During these meetings, we have an active role, where we participate in the SA Canvas usage, helping the team define the metrics for their issues, generate tasks, analyse the metrics' results and, if needed, give suggestions of decisions to make based on this usage. At the end of these meetings, we ask the team to answer the sprints' surveys to monitor their opinion about the SA Canvas usage and usefulness across the different sprints. During the rest of the time, we work together with the team to address the tasks defined in these meetings and update SA Canvas's status. **In total the SA Canvas was used in 4 sprint retrospectives, and lasted for 3 full sprints, because the study ends after the 4th usage**.

In the end, a **final data collection** is performed, where we use the evolution of the metrics defined as important by the team to evaluate the impact of the SA Canvas' usage in the code quality. Also, we ask participants to answer the final survey to obtain the participants' final perceptions of the SA Canvas. This final survey has the objectives of gathering the final perceptions of the participants concerning the usefulness of the tool and their intentions of continuing to use SA and the SA Canvas as part of their development process. Finally, we perform a **refinement session** meeting with the team. During this meeting, where we ask them to suggest changes to the SA Canvas and to explain the reasons for those changes. At the end of this discussion, they answer a survey related to these refinements. This survey will reflect this discussion's results and gives them more freedom to answer with their personal opinions. The objective is to analyse the results of the survey and possibly get a new version of the SA Canvas.

4.5.4 Data Analysis and Replication

The data analysis was performed with the help of Microsoft Excel [4]. The participants' answers to the surveys were all aggregated in a single excel file where we generated some plots of the evolution of the answers and of their distribution of answers. A replication package with all the surveys used, all the answers to each survey and an excel file to replicate the plots used in our analysis can be found in [5].

4.6 Execution

In this section we present an overview of the execution of the study.

[4]https://www.microsoft.com/microsoft-365/excel
[5]https://github.com/wortz/Metrics_Evaluation_DISS

The first important point to notice is that this study was conducted during a pandemic situation, where every participant of the study was working remotely. This difficulted our work, as we had no real contact with the team and every meeting was via Microsoft Teams[6].

Starting the study we asked the team to answer the **background survey** and then we explained some concepts of SA and how the SA Canvas was meant to be used and its objectives. In this first contact, the team had a hard time understanding the SA Canvas and its objectives, which caused a freeze in the usage of the approach. That led to an adoption of a different approach, described latter in this dissertation as part of a possible new pattern observed. With this we could help them understand better some of the SA Canvas' objectives and the team could start its usage. This event occupied a full sprint, where we couldn't measure the usage of the SA Canvas and understand its usefulness and ease-of-use. During the rest of the dissertation, this was the sprint that we name as Sprint 0.

Starting the next sprint, the one we name as Sprint 1 in our study, the team had already used the SA Canvas in the previous sprint retrospective, which led to some tasks defined meant to be done during this sprint. This sprint lasted for 4 weeks and the development process went normal. Ending this sprint the team assessed the SA Canvas in the retrospective.

In the beginning of the next sprint, Sprint 2, **the team had some members in vacations, which delayed the sprint planning, starting only one and a half week after the sprint 1 retrospective**. Also, **during this sprint a change was made in the team composition, where one the members left the team and a new one joined**. This caused this sprint to last much longer, lasting for six weeks, because of presentations of the project to the new member and a personal problem of the Team Leader. **This delay led to a reduce in the number of entire sprints of the study, from 3 full sprints to 2**. At the end of sprint 2 the team also assessed the SA Canvas, but the study ended before the start of the next sprint.

4.7 Data Analysis

In this Section, we analyse the data collected throughout the study to obtain possible answers to the research questions. We analyse the team's background, their usage of the SA Canvas, and their perceptions regarding ease-of-use and usefulness. Also, we try to evaluate the code quality impact of the approach, and we present the possible refinements suggested by the participants.

4.7.1 Team's background in Software Analytics

Starting the first phase described in Section 4.5.3, we had a meeting with the project coordinator, and we questioned him about which metrics were already being used in the project. His answer was that **they had no metrics, tests, and analytics tools implemented in the project**. After this, we had our first experiences with the team, and we tried to understand their background in SA. To

[6]https://www.microsoft.com/microsoft-teams

do this, we asked the **Background survey** described in Section 4.5.2. Table 4.2 and Figure 4.2 show the questions and answers to the background survey.

Table 4.2: Answers to the background survey where Q1 and Q2 were open answers' questions with the scale of years, Q3 and Q6 were open answers' meant for the participants to write small answers and the rest of the questions were Likert items

	Project Leader	Developer	Junior Developer	$\bar{x}$
Q1	7	2	0.5	3.17
Q2	2	0.5	0.5	1
Q3	A way to improve and understand better your code and eventual flaws	Is a way to analyze a software components' performance in order to improve processes	I think a software analytics have a job more related with computer architecture	
Q4	2	2	2	2
Q5	2	2	2	2
Q6	-	User Experience	-	
Q7	4	2	3	3
Q8	2	2	3	2.33
Q9	4	5	4	4.33

Q1 How many years of experience you have?
Q2 How many years you have in this company?
Q3 What is Software Analytics to you? Briefly describe it.
Q4 How often does your team use Software Analytics?
Q5 How often your team use metrics to inform their decisions?
Q6 If you answered 3, 4 or 5 in the previous question, which metrics? Do you think that those metrics give you important information for the problems you try to solve? Why?
Q7 How often your team defines quality thresholds concerning code quality?
Q8 How often do you use refactoring in your code?
Q9 What is the importance of code quality in your project?

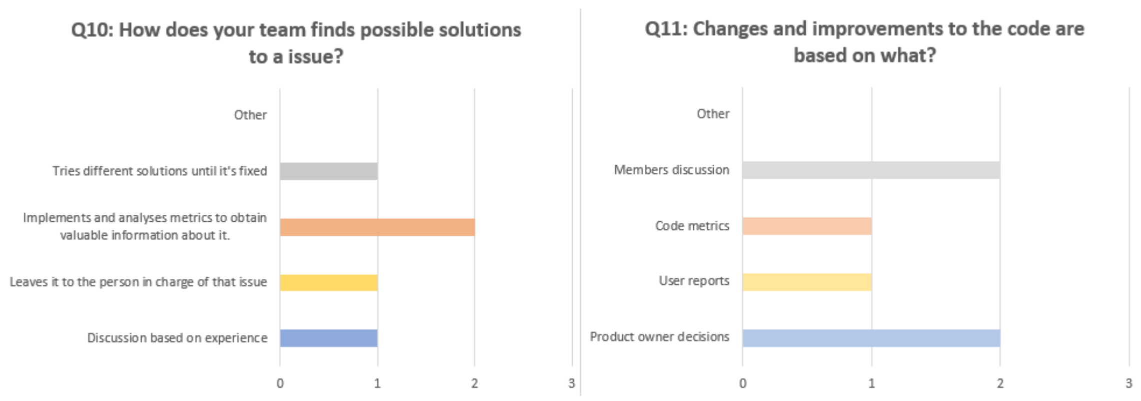


Figure 4.2: Answers to the background's multiple choice questions

Looking at Q1 and Q2, we can see that **one of the members has low experience as a professional developer** and that **two of them have low experience working in the company**. Analysing the answers to Q3, we can verify that the team had a **definition of SA based on basic and wrong concepts**, where the Team Leader knows it is meant to improve and understand the code better and find possible issues. In the answers to Q4 and Q5, we can also notice that **they did not have the habit of using SA and metrics** in their project. Also, during our contact with the team, we verified that the most crucial point for the project was the user feedback about it and that this was how they know which changes and improvements needed to be done.

In terms of **code quality**, Q9 shows us that the team members perceive it as a **crucial point in their project**. However, their decision-making is only **based on product owner feedback**. After getting the product owner feedback, they discuss the changes needed based on that feedback. In that discussion, it was evident that the **Team Leader was the person that mainly guides the team** on what needs to be changed or implemented in order to correspond to the product owners feedback. However, when answering Q10 and Q11, some members referred that they use metrics to find solutions to issues and implement changes in their project. Hence, we tried to understand this contradiction better by asking those members directly. They explained that those metrics are more related to users and product owners' feedback, not with code quality metrics, which **strengthens our observation that they have low experience with SA**.

4.7.2 Software Analytics Canvas' usage and evolution

The team that participated in the study assessed the SA Canvas in 3 different retrospectives, and its usage lasted 2 full sprints. In each sprint, the SA Canvas was used during the sprint retrospective, and the tasks originated by its use were included in the next sprint planning.

In the first usage, instead of starting with the issues known by the team, we started by giving the team some contact and visualisation of metrics by showing the SonarQube's report and discussed some problems identified by it. After this, the team started to organise the most crucial problems identified by the SonarQube in the SA Canvas with the metrics of the SonarQube associated. During this meeting, the team also realised that the usage of tests in their code could be beneficial to avoid possible future bugs. Also, the team already got to some highlights and insights because the metrics were already available for analysis. As we can see in Figure 4.3, most of the occupied lanes were the "Measurements" and the "Methods and Tools" derived by the early implementation of the SonarQube, but with this, the team could identify some bugs and code smells in the code that were important for the them to fix. In this first usage, they already got two tasks derived from the SA Canvas usage:

- bugs and code smells to be fixed in the backend;
- an implementation of a method defined in "Methods and Tools";

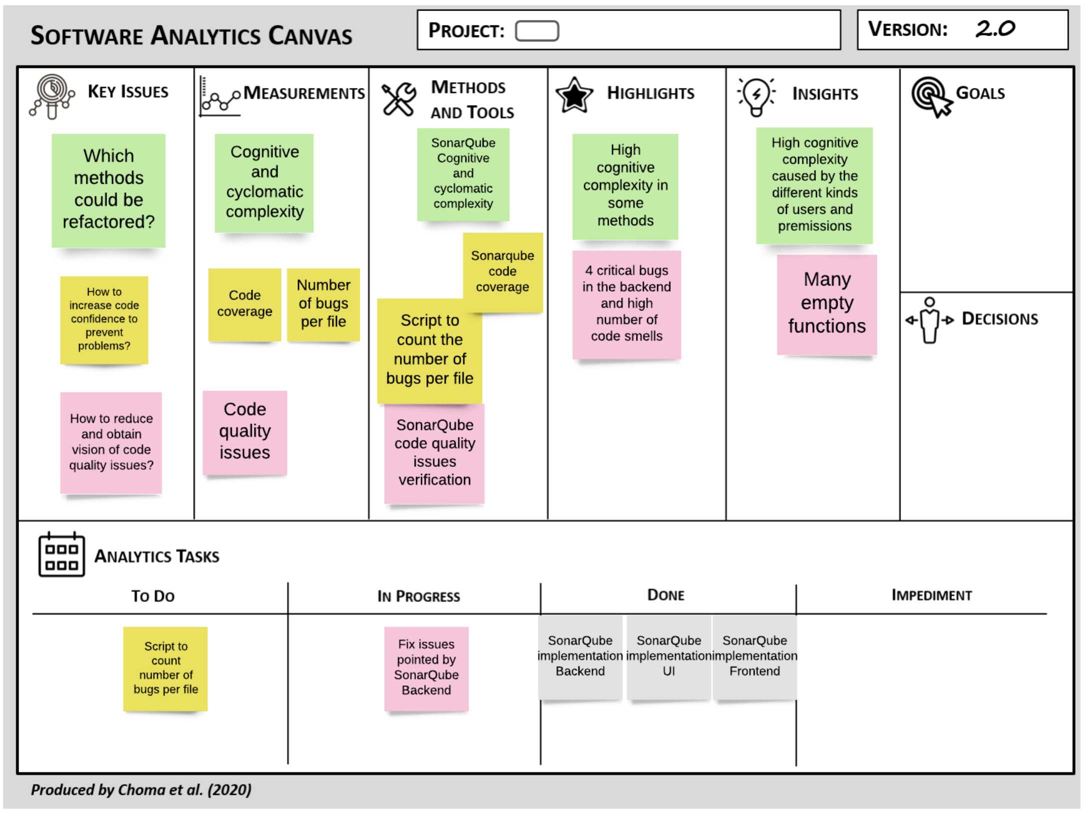

SOFTWARE ANALYTICS CANVAS
PROJECT:
VERSION: 2.0
KEY ISSUES
Which methods could be refactored?
How to increase code confidence to prevent problems?
How to reduce and obtain vision of code quality issues?
MEASUREMENTS
Cognitive and cyclomatic complexity
Code coverage
Number of bugs per file
Code quality issues
METHODS AND TOOLS
SonarQube Cognitive and cyclomatic complexity
Sonarqube code coverage
Script to count the number of bugs per file
SonarQube code quality issues verification
HIGHLIGHTS
High cognitive complexity in some methods
4 critical bugs in the backend and high number of code smells
INSIGHTS
High cognitive complexity caused by the different kinds of users and premissions
Many empty functions
GOALS
DECISIONS
ANALYTICS TASKS
TO DO
Script to count number of bugs per file
IN PROGRESS
Fix issues pointed by SonarQube Backend
DONE
SonarQube implementation Backend
SonarQube implementation UI
SonarQube implementation Frontend
IMPEDIMENT
Produced by Choma et al. (2020)

Figure 4.3: First SA Canvas' usage - during sprint 0 retrospective.

In the second usage, represented in Figure 4.4, the team analysed SonarQube's bugs and code smells in the frontend with more detail and looked into a script implemented to get the number of bugs per file. In the analysis of the bugs and code smells, they saw that there were many possible problems, and they realised that they could have time troubles in fixing those problems in one sprint. In order to obtain a list of the most indicated files to start the implementation of tests, the team analysed the number of bug fixes per file. However, they decided to change the script to the number of bug fixes per file in the last three months, as this could be different from all-time results. In terms of tasks generated from the SA Canvas usage, we defined two:

- Change in the script of the number of bug fixes per file to look only for the last three months

- Fix the important bugs and code smells pointed by the SonarQube

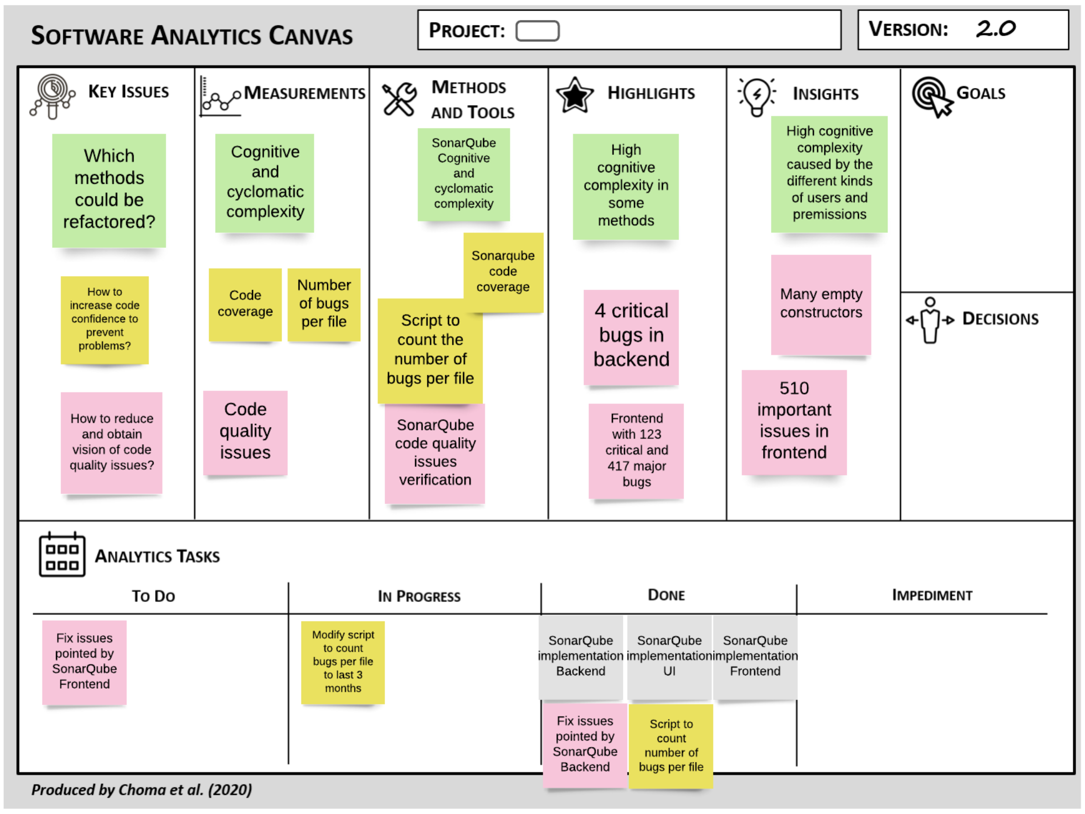

Software Analytics Canvas
Project:
Version: 2.0
Key Issues
Which methods could be refactored?
How to increase code confidence to prevent problems?
How to reduce and obtain vision of code quality issues?
Measurements
Cognitive and cyclomatic complexity
Code coverage
Number of bugs per file
Code quality issues
Methods and Tools
SonarQube Cognitive and cyclomatic complexity
Sonarqube code coverage
Script to count number of bugs per file
SonarQube code quality issues verification
Highlights
High cognitive complexity in some methods
4 critical bugs in backend
Frontend with 123 critical and 417 major bugs
Insights
High cognitive complexity caused by the different kinds of users and permissions
Many empty constructors
510 important issues in frontend
Goals
Decisions
Analytics Tasks
To Do
In Progress
Done
Impediment
Fix issues pointed by SonarQube Frontend
Modify script to count bugs per file to last 3 months
SonarQube implementation Backend
SonarQube implementation UI
SonarQube implementation Frontend
Fix issues pointed by SonarQube Backend
Script to count number of bugs per file
Produced by Choma et al. (2020)

Figure 4.4: Second SA Canvas' usage - during sprint 1 retrospective.

The third and final usage, represented in Figure 4.5, was based on an analysis of the script of the number of bug fixes per file, where the team realised that there were two files in the backend with a high number of bug fixes in the last three months compared with the rest of the files. Also, one of those files had one of the highest cognitive complexities, which led to the conclusion that this should be the file where the implementation of tests should begin. Another thing analysed, common to all the other sprints, was the issues pointed by the SonarQube. In this case, the backend issues were analysed in more detail than in sprint 1, and 510 issues were considered important to be checked during changes to the files involved in them. This high number of issues found led to the creation of a *Decision* in the SA Canvas, which was the integration of SonarQube into the team's development process. In more detail, when they are changing a file, they should check if there is any issue associated with it and fix it if it makes sense, in their opinion. This will also help prevent some issues during the development of new features or alter the existing code.

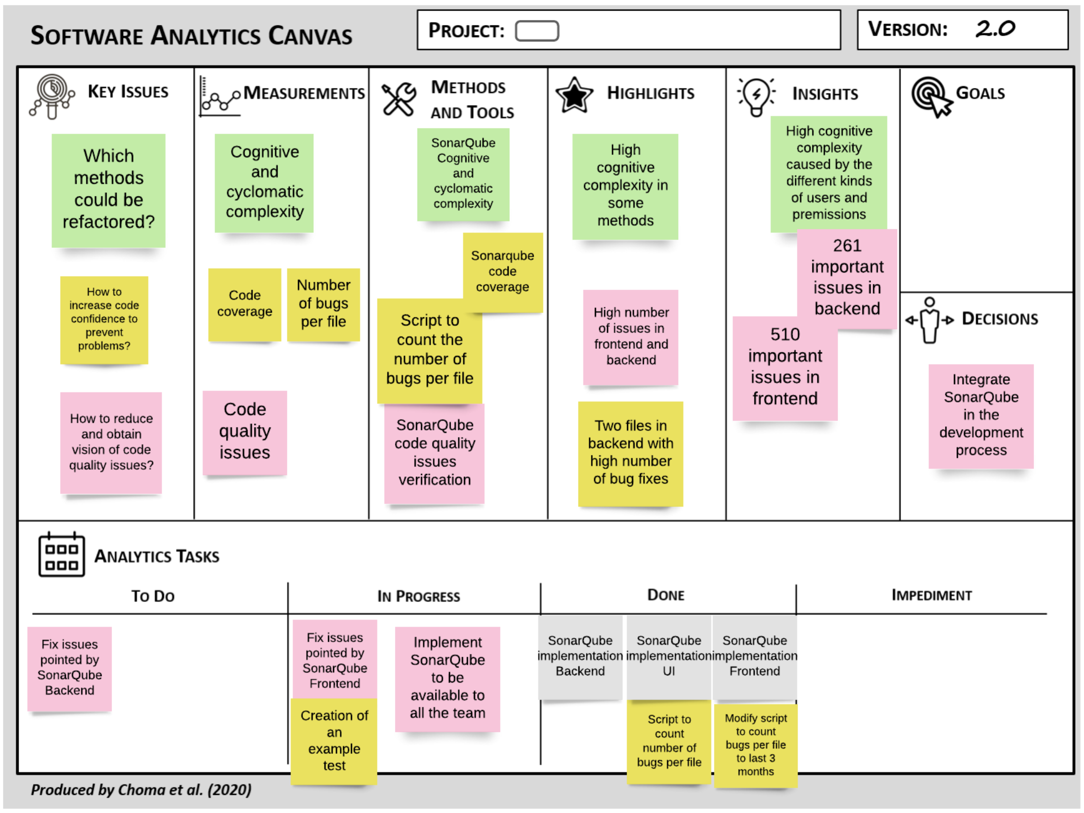

SOFTWARE ANALYTICS CANVAS
PROJECT:
VERSION: 2.0
KEY ISSUES
Which methods could be refactored?
How to increase code confidence to prevent problems?
How to reduce and obtain vision of code quality issues?
MEASUREMENTS
Cognitive and cyclomatic complexity
Code coverage
Number of bugs per file
Code quality issues
METHODS AND TOOLS
SonarQube Cognitive and cyclomatic complexity
Sonarqube code coverage
Script to count the number of bugs per file
SonarQube code quality issues verification
HIGHLIGHTS
High cognitive complexity in some methods
High number of issues in frontend and backend
Two files in backend with high number of bug fixes
INSIGHTS
High cognitive complexity caused by the different kinds of users and permissions
261 important issues in backend
510 important issues in frontend
GOALS
DECISIONS
Integrate SonarQube in the development process
ANALYTICS TASKS
TO DO
IN PROGRESS
DONE
IMPEDIMENT
Fix issues pointed by SonarQube Backend
Fix issues pointed by SonarQube Frontend
Creation of an example test
Implement SonarQube to be available to all the team
SonarQube implementation Backend
SonarQube implementation UI
SonarQube implementation Frontend
Script to count number of bugs per file
Modify script to count bugs per file to last 3 months
Produced by Choma et al. (2020)

Figure 4.5: Third SA Canvas' usage - during sprint 2 retrospective.

4.7.3 Ease-of-use and Usefulness

To evaluate the usefulness and ease-of-use, we used the **sprints surveys** and the **final survey** to get the team's feedback during the different usages of the SA Canvas. By doing this, we could see the evolution of their perceptions about the SA Canvas. The questions and answers of this survey can be seen in Tables 4.3 and 4.4. In the end, as described in 4.5.3, we conducted the final survey, which can also help us understand the usefulness of the SA Canvas. When analysing the evolution of their perceptions **we gave more importance to the Team Leader and Junior Developer's answers** because they were present throughout the study.

As it was said in Section 4.6, the team had a hard time understanding the objectives, the usage and perciving usefulness in the SA Canvas in the first experience with it. This could have been caused by their lack of experience with SA and their understanding of the objectives and usefulness of metrics in evaluating their code and informing their decisions, as was verified in Section 4.7.1.

To assess the ease-of-use, we questioned the overall understandability of the objectives of the SA Canvas and the ease-of-use of every step in the Canvas. These questions were conducted in the end of sprint 1 and sprint 2.

Table 4.3: Responses to the sprints' surveys regarding ease-of-use of the SA Canvas.

	Team Leader		Developer 1	Developer 2	Junior Developer		$\bar{x}$	
	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Sprint 1	Sprint 2	$\overline{x_1}$	$\overline{x_2}$
Q1	4	4	4	5	4	3	4	4.3
Q2	4	3	2	5	4	3	3.3	3.7
Q3	3	3	2	5	3	3	2.7	3.7
Q4	3	3	3	5	4	4	3.3	4
Q5	3	3	2	5	4	4	3	4
Q6	3	3	2	5	3	4	2.7	4
Q7	-	3	-	5	-	4	-	4

Q1 It was easy to understand what were the objectives of Software Analytics Canvas.
Q2 I found it easy to define Key Issues in the SA Canvas.
Q3 I found it easy to define the Measurements for the defined Key Issues.
Q4 I found it easy to define the Methods and Tools that obtain the Measurements defined
Q5 I found it easy to define Highlights after implementing the Methods and Tools.
Q6 I found it easy to define Insights after analysing the Methods and Tools.
Q7 I found it easy to define Decisions after analysing the Insights and Highlights

Analysing the responses to the Q1, we can see that the team could then understand the objectives of the SA Canvas after its first usage and that **these responses were more or less constant** during both sprints. Regarding the ease-of-use of every step of the SA Canvas, we can see that the answers of Q2 to Q7, shows that the team leaned towards a **neutral opinion about the ease-of-use of each step** and that it was **maintained during the time of the study**, with some increases and decreases in some answers. Also, in both surveys, there was a question regarding difficulties felt in the SA Canvas' usage, and despite that only one answer was given, it can be an important point, as it concerned the identification of *Key Issues*. Developer 1 answered this question by saying that it was hard to identify the key issues as the project was quite recent and, in his opinion, well built. By looking at Figure 4.6, we can verify that the Team Leader and Junior Developer's opinion regarding the ease-of-use of the tool has **slightly decreased** from sprint 1 to 2. However, it does not represent a significant change to be considered as a decrease in their perceptions. Looking at Developer 1's answers, we can verify that he had some difficulties understanding the SA Canvas, as his average was 2,5. Regarding the new member's perception we can see that his answers were 5 to every question, which means that he thought that every step and that the SA Canvas itself is easy to understand, but this answers may not reflect the truth about the tool.

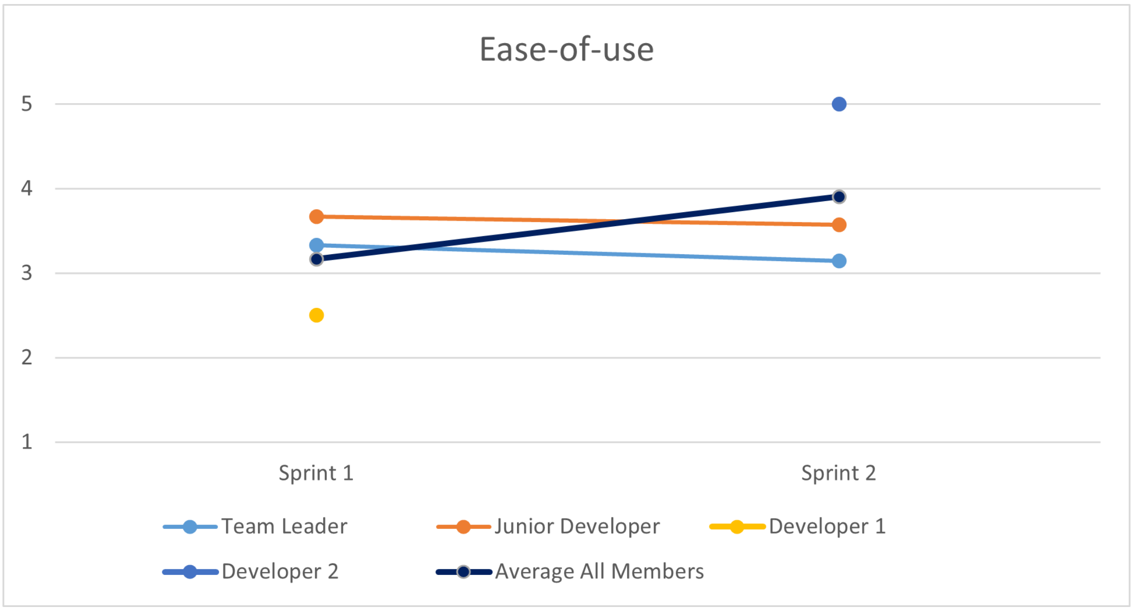


Figure 4.6: Ease-of-use surveys' chart considering the averages of each member's answers and the average of all answers

Table 4.4: Responses to the sprints' surveys regarding the usefulness of the SA Canvas.

	Team Leader		Developer 1	Developer 2	Junior Developer		$\overline{x}$	
	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Sprint 1	Sprint 2	$\overline{x_1}$	$\overline{x_2}$
Q1	3	4	3	5	4	4	3.3	4.3
Q2	3	4	4	5	4	3	3.7	4
Q3	4	3	2	5	3	3	3	3.7
Q4	3	4	4	5	4	4	3.7	4.3
Q5	2	3	3	5	3	4	2.7	4
Q6	3	3	3	5	3	4	3	4

Q1 The usage of Software Analytics Canvas was useful.
Q2 The SA Canvas helped the team to think about the quality problems in the project.
Q3 The SA Canvas helped the team to think about the best metrics for the identified problems.
Q4 The implemented metrics in the usage of SA Canvas led to improvements in the code quality.
Q5 The SA Canvas brings greater transparency to the decision-making process.
Q6 The time that SA Canvas took from other development activities was beneficial to the project.

To evaluate the usefulness of the SA Canvas, we asked the team about their perceptions of some important points during each sprint's SA Canvas' usage. Figure 4.7 shows the evolution of each member's average along with the average of all members together.

Analysing the answers from each sprint's survey, available in Table 4.4, we can see by Q1 that the team was somewhat **neutral in terms of perceived usefulness in the first sprint, but there**

was an increase in the second sprint in the team leader's perception. If we look at the average of the answers, we can see that all the averages increased from sprint 1 to sprint 2, mainly caused by the new developer's perceptions of the tool. After further analysis of the new developer's opinion, Developer 2, we found that he was used to have metrics implemented in its projects pipelines and thinks that will increase the overall code quality and that the team should be using metrics to improve their decisions. Also, if we look at the Team Leader and Junior Developer's answers, we can see that there was an **increase in some answers**, which can be caused by the better understanding of the SA Canvas and more results from the metrics analysis, in terms of possible changes and improvements to the code quality. Looking at Figure 4.7 we can verify a considerable increase in both Team Leader and Junior Developer's perception, from 3 to 3,5 and 3,8 in the Team Leader's, and 3,5 to 3,7 and 4,2 in the Junior Developer's answers. Also, this Figure has a dark line that represents the average of all participants' answers, and it shows that their **overall perceptions of usefulness increased** from 3,2 to 4,1 from sprint 1 to 2, and 4,3 in the final survey.

Additionally, by looking at the answers of the final survey, available in Table 4.5, we can also evaluate the team's perceptions in terms of usefulness. In these answers, we can see that most of them are favourable. The answers to questions 1 to 4 show that **the SA Canvas and its usage have proven useful**. Also, the answers to Q6 show that the team may **keep the usage of SA practices** in their development process. However, despite having a favourable average in questions 5 and 8, we can see that the **Team Leader has a neutral position in terms of SA Canvas' usefulness in supporting analytics tasks and his intentions of continuing to use it**. This could mean a possibility of **dropping the SA Canvas usage in the future**, as the Team Leader is the one that most affects the decisions in this team.

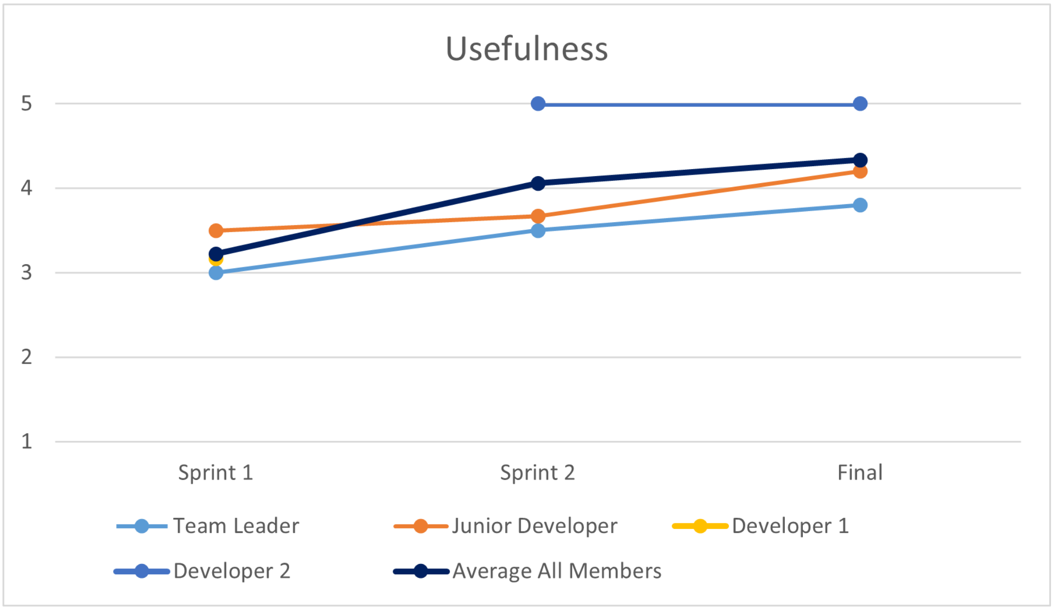


Figure 4.7: Usefulness surveys' chart considering the averages of each member's answers and the average of all answers

Table 4.5: Responses to the final survey.

	Team Leader	Developer 2	Junior Developer	$\bar{x}$
Q1	4	5	4	4.3
Q2	4	5	5	4.7
Q3	4	5	4	4.3
Q4	4	5	4	4.3
Q5	3	5	4	4
Q6	4	5	3	4
Q7	4	5	4	4.3
Q8	3	5	4	4

Q1 The Canvas' usage was useful
Q2 The issues raised were important to our project
Q3 The metrics implemented leaded to useful insights of our code quality.
Q4 The Canvas' usage has leaded to improvements in the code quality
Q5 The Canvas represents a good support for the analytics tasks
Q6 The metrics and tools implemented will continue to be used in the future
Q7 Software Analytics represents a valuable method to use in the future of our project
Q8 The Canvas will continue to help us organize and implement analytics in the future

4.7.4 Impact on code quality

In order to evaluate the impact of the SA Canvas' usage on code quality, we analysed the team's code through the use of metrics and monitored their evolution throughout the study. Firstly, as we implemented the SonarQube to help the team better understand the SA Canvas objectives, we also used its metrics to evaluate their code quality.

One of the metrics was **the number of issues** pointed out by the SonarQube. This metric was the one that the team mainly focused its attention on. In all sprints, the SA Canvas' usage had some time invested in looking at those issues and evaluate which ones have a real impact on the code quality in the team's opinion. During this evaluation, different kinds of issues were identified and considered important to fix, such as:

- Duplicated imports - frontend
- Duplicated and unused styles - frontend
- Empty constructors - frontend
- Useless assigns in variables that were not used since the assign - frontend and backend
- Identical methods - backend
- Some methods' parameters that were not used - backend
- Unused temporary variables - backend

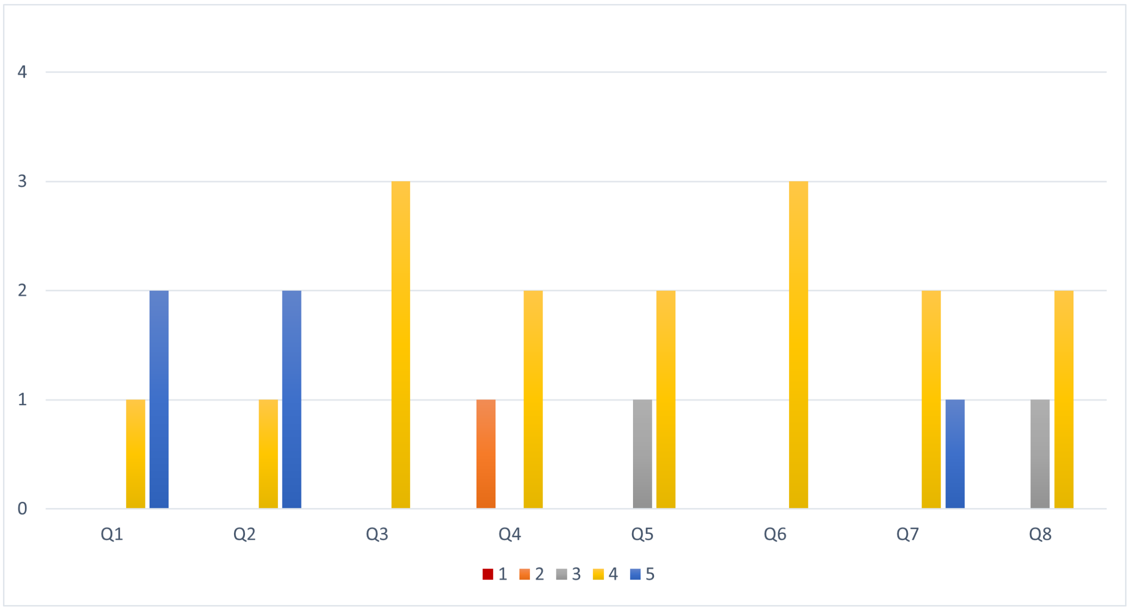
The total number of important issues for the team found was 510 in the frontend and 261 in the backend. This large number of issues caused the team to decide that they would keep the list of issues and **fix them when they make changes in the files related to those issues**. Although the **fix of some of those issues was already done** during this study, **the total number of issues did not decrease**. In reality, **this number increased** because the rest of the changes and implementations in the code kept on creating new issues.

The second metric used was **code coverage**. This metric had an **initial value of 0%**, but this was a concern for the team because they wanted to start to implement tests in their code. During the usage of the canvas, the team tried to understand how they would start using tests in a large project that had no implementation of any test. So this process took some time, due to some drawbacks in the team's members and organisation. During this time, we analysed where to start implementing tests, which files would be most indicated, that had a higher frequency of bug fixes. This has led to **no alteration in this metric**, but with the objective that in the near future, the team **will gradually start to increase the test coverage** on files with the highest frequency of bug fixes, and this metric will have its value increased.

4.7.5 Refinement Session

As described in Section 4.5.3, we met with the team to perform the refinement session, where they shared some perceptions of the SA Canvas, some difficulties they felt and proposed some changes to it. Then, based on this meeting, they answered the **refinement survey**. Figure 4.8 shows the answers to the Likert Items of this survey.

Figure 4.8: Answers to the refinement survey's Likert items



Q1 I think that *Key Issues* plays an important role in the SA Canvas.
Q2 I think that *Measurements* plays an important role in the SA Canvas.
Q3 I think that *Methods and Tools* plays an important role in the SA Canvas.
Q4 I think that *Highlights* plays an important role in the SA Canvas.
Q5 I think that *Insights* plays an important role in the SA Canvas.
Q6 I think that *Decisions* plays an important role in the SA Canvas.
Q7 I think that *Goals* plays an important role in the SA Canvas.
Q8 The bottom row of the SA Canvas helps the team to organize the analytics tasks during the development process.

During the meeting, the main topics brought to the table were:

- The similarities they felt between the *Highlights* and the *Insights*, were they felt that they were really close, and couldn't see any benefit from having them separately.
- The integration of the SA Canvas with their tasks' definition tool, where the SA Canvas should be inserted into that tool, so they do not need to look for two different places to follow their development process.

Analysing the **answers to the refinement survey** we can see:

- By looking to Figure 4.8, they did not felt that the *Highlights* was an important element in the SA Canvas, as in they saw them really similar to the *Insights*. This could have been caused by their lack of experience in using SA practices. This caused them to have difficulties differentiating the highlights that they get from looking at the metrics results for the first time and highlighting strange values, from analysing those highlights to get insights of why the metric has that value.

- Also, the *Insights*, Q5, and *Analytics tasks* row, Q8, got lower answers when comparing to the others. In the case of the *Insights*, it was also caused by the similarities they felt to the *Highlights* section. For the *Analytics tasks* it can be related to the suggestion they made of integrating the SA Canvas with their tasks' organisation tool, in this case the tools was Jira [7].

- Finally, there was a suggestion in the open questions' answers of this survey, saying that the *Methods and Tools* could be moved to the first column, which was derived from the adoption of the different approach in the Sprint 1. Because of this, during the 3 usages of the SA Canvas, the team gave enormous importance to the SonarQube analysis and started to use it as the main base of their analytics process as it was described in Section 4.7.2.

4.8 A new pattern to implement Software Analytics in development teams: Broad-spectrum Diagnostic

In this section, we describe a new pattern observed during this study. The description of this pattern was written separately from the rest of the dissertation, so there could be some repetition.

Introducing Software Analytics to software teams can be challenging, especially when those teams have low experience with analytics and have no usage of them in their current project. They can have trouble understanding some key points to implement Software Analytics in their development process. This can make them abandon the analytics usage at the start or not correctly use Software Analytics.

4.8.1 Problem

The team does not have an understanding of the type of questions that can be answered through software analytics.

Even though Software Analytics practices help solve code quality issues and improve software quality based on code metrics and insights of those metrics, teams might **have no experience** with these practices, leading to difficulties **understanding what issues can be solved, verified or improved** with the help of metrics and their analysis. This can get the teams stuck at the start of the adoption of the Software Analytics practices. However, the existence of some analytics tools that are **easy and fast to implement** and that give a **good overview of the system quality, issues, and metrics**, can help them to get a more practical example of the objectives and process of Software Analytics and make the process of implementing analytics practices in their development process more accessible.

[7]https://www.atlassian.com/software/jira

4.8.2 Solution

Use tools that make a general diagnostic of the system and show a couple of metrics, to help detect a set of initial issues that can be used as a starting point to the software analytics practices' usage.

Starting with the usage of software analytics practices can be challenging when introducing them to teams with low or no experience with metrics and analytics practices. This is when the BROAD-SPECTRUM DIAGNOSTIC pattern can be used. To implement it, the team chooses a static analytics tool that gives a good view of the overall quality of the system, like SonarQube, Grafana; these tools will help them look at some metrics and analysis of their system. By looking at those metrics, the team will have a more practical understanding of what metrics can be implemented, some examples of some issues that can be solved or improved with the help of those metrics.

After implementing the tool, the team should **analyse the tool's results** and analyse the importance each issue has in their project. This analysis should be taken carefully because the tool shows the same metrics to every kind of project, so the team should look to the metrics they find helpful for them and to the important issues to be solved in their project. Many of the tools offer severity levels to the issues found, and this can be a good way to start by looking to the most critical issues identified by the tools.

Finally, **select some issues** from the tool report and then use the WHAT YOU NEED TO KNOW with these issues in mind and follow the rest of the analytics process, so they can better understand how to use Software Analytics practices and how the patterns described in section 3.2.5.1 can guide them to solve different kinds of problems.

4.8.3 Consequences

As a consequence, the team members can **learn more about Software Analytics** and better understand **which metrics they can integrate** into their system, **the issues that can be verified and solved** with the usage of Software Analytics practices, but also help them to **understand how to use the analytics** as part of their development and improvement process. In the end, the team will have already a **tool integrated with the environment** which can be useful in the future because some metrics are already available to analyse.

A negative consequence is that the tool's report can guide the team in the **wrong direction** because without a good analysis of the tool's results, they can be looking to metrics and issues that are not relevant for their project or cause the team to **lose interest** by showing irrelevant issues.

4.8.4 Example

We observed the pattern in the context of a large company, where data security and privacy were critical points to the project. It was a recent project with 3 members involved in the development team and various clients governing the project and giving feedback.

After analysing the team's background in Software Analytics, we figured that they had low experience in analytics and metrics usage. Most of their decisions were based on experience from the members and mostly on user and product owners feedback.

When we introduced the Software Analytics practices to them, they experienced difficulties in understanding the objectives and issues that could be solved using analytics, possibly caused by the lack of experience or knowledge of software analytics. In the first try of usage, we had no progress with the team, where they had difficulties identifying the issues to solve or verify with analytics, more specifically in the pattern WHAT YOU NEED TO KNOW.

With this difficulty, we decided to use another approach. We implemented SonarQube and showed the team its report, with all the issues it pointed out and some interesting metrics to promote discussion. After analysing and discussing the importance of each issue and metric, both the team and we started to fit the issues, or the group of issues in the patterns described in Section 3.2.5.1, in order to understand better to have a more practical example for them to work with. With this, we were able to move forward, and the team started to bring more issues to the table and understand better the objectives of Software Analytics.

4.9 Research Answers

This section presents answers to the defined research questions defined in section 4.4.

RQ1 *"What are the participants' perceptions about the usefulness and ease-of-use of the SA Canvas?"*

The participants did not felt the SA Canvas either hard or easy to use, but felt that its usage was useful to the project and to the development process.

Using the analysis of **sprints' surveys and the final survey** made in section 4.7.3, we can conclude that the participants felt that the SA Canvas was **hard to understand in their first impressions**, where they had difficulties understanding its objectives and its usage, but also understanding the concepts of SA. After explaining better and showing some examples of metrics and tools of their code, they could better understand the objectives of the SA Canvas, but **their perceptions were neutral in terms of ease-of-use and were constant during the study**.

In terms of usefulness, the **participants also started by being neutral**, but **their perceptions increased** as they used more the SA Canvas and started to have more insights into their code quality issues. By their responses to the continuity of the SA Canvas, metrics, and SA in their development process shows that **they perceived more usefulness in the process of SA than in the SA Canvas itself**. They said that the overall usage of the SA Canvas and the practices behind it **brought value to their development and decision-making process**. These perceptions of usefulness increased over time. These perceptions would increase even more if the study time were extended because they would see more improvements and changes being implemented in their code.

RQ2 *"What are the participants' perceptions about the approach's benefits in their organisation and decision-making process into finding solutions and solving issues?"*

The participants saw benefits in the SA Canvas practices but not so much in the SA Canvas itself. Also, their perceptions of the benefits increased throughout the study.

In terms of their perceptions about the benefits in their organisation and decision-making process, they **presented neutral responses** to the questions related to this RQ, with a **slight increase from one sprint to another**. At the start of the study, **they felt that the SA Canvas was confusing and that it did not show many benefits for their development process**. However, after understanding better the SA' concepts and the SA Canvas itself, **they started to see benefits in the practices behind the SA Canvas**. They affirmed that the usage of metrics and team discussion of those metrics could improve their decisions because they were based on information of their code. We can affirm that for this case, **the team perceived benefits in the practices behind the SA Canvas, but not so much in the SA Canvas itself**.

RQ3 *"What is the impact of the adoption of SA Canvas in the code quality?"*

SA Canvas did not impact the code quality during the time of the study, but we believe that the decisions derived from its usage will, in the future of the project. **We can say that the SA Canvas takes some time to impact the overall code quality.** To measure the impact of the SA Canvas on the code quality, we used metrics that try to give an overview of this code quality. For this case study, **we could not see much of an impact of the approach in those metrics' values** because, during the time of the study, the team did not really implement many of the insights and decisions taken from the SA Canvas' usage. **However, we believe that those metrics' values will be affected as they will implement the changes defined**. These changes will affect directly the metrics we used to estimate their code quality, as the team decided to implement tests gradually in the future, decided to use the SonarQube as part of their code quality issue detection in the future of the project, and fix all the issues already known and important for their project.

RQ4 *"Which elements can be improved in SA Canvas?"*

As it was analysed in Section 4.7.5, the main refinements suggested by the participants of this study were: 1) the integration of the SA Canvas in the tool the team uses to organise their development tasks, 2) merge of the *Highlights* column with the *Insights*. The second suggestion was much influenced by their inexperience with SA practices and their initial difficulties in understanding the SA Canvas. But it is worth looking if more teams fell the same about this second point in order to make a real change to the SA Canvas.

4.10 Threats to Validity

During this study we aimed to answer the research questions and with that prove the validity of our hypothesis. Even though we could answer the research questions, the study had some threats that could influence the results obtained. Based on a list of threats present in [31], we identified some threats:

1. **Response bias**, this refers to answers that do not reflect the true opinion of the participants. In our case there could be some feeling of proximity as we worked directly with the team,

and the team can have fear of responding with the truth when they have a negative perception. To avoid this we tried to explain that we wanted them to answer in the most honest way, even if they have a negative perception of the SA Canvas, because we wanted true results even if negative. For Team Leader, Developer 1 and Junior Developer, we think we were able to mitigate this threat, but for Developer 2 we think it didn't happen, because his answers did not seem to reflect the truth.

2. **Low number of participants and small duration of the study**, as our study was only performed with 4 different participants, all from the same team, and where we collected data from 2 sprints. With this, it is hard to affirm that the results of our study can be generalised to different teams within the company and outside the company and its a threat that we were not able to mitigate.

3. **Non-replicable events**, as during this case study there were some events that can be non replicable in other studies, and can make this study somewhat unique, which was a threat that we were not able to mitigate at all, because it was out of our control. Also, this study was performed during a pandemic situation which difficulted some of our work because all the participants were working remotely.

Chapter 5

Conclusions

5.1 Summary

After reviewing the state of the art, we understood the methods and tools to measure and assess software quality. With this review, we learned about software quality models that provide a good way to define software quality. However, as these models did not provide a way to assess and improve software quality, we looked for methods that did. We discovered different methods that helped us find common problems, strengths and differences between them, which led to our formulated hypothesis present in section 4.3.

During this state of the art review, we were able to identify that, in many cases methods, needed to be easy and useful to use, as many teams had a low amount of time to use in other development activities and constant pressure from the product owners to deliver the changes necessary for the product. That is why we opted to use a continuous assessment method, the SA Canvas, that did not necessarily need an expert team in SA and the method to use it. It provides a good way to organise and use SA in the team's development process.

In order to verify the validity of our hypothesis, we formulated four research questions related to the SA Canvas. To answer these questions, we conducted an industrial study using the SA Canvas to support the analytics activities with a small team. During this study, we asked the participants to answer a couple of surveys to gather data. By analysing this data, we were able to answer the research questions, which can be seen in section 4.9.

5.2 Main Contributions

This dissertation had the following contributions:

- **State of art review** - Where we explored some software quality models and software quality's assessment methods. With this review, we were able to identify the main difficulties and benefits of adopting SA practices and the main issues and advantages of each one of these.

- **Industrial study** - We conducted an industrial study with a small team of 3 members, inserted in a large company. This study helped gather some user perceptions of the adoption of SA practices and SA Canvas. With this, we could verify the usefulness and ease-of-use of the SA Canvas and its impact on software teams' decision-making process and code quality.

5.3 Future Work

In this section, we present how this work could be extended in order to strengthen this study conclusions and answers to the research questions defined in 4.4.

- **Extend the study to a larger number of teams and participants**, as we only performed the study with one team that was composed of 3 members.
- **Involve more areas of the team in the study**, as we only worked with the development team, and it could be interesting to verify the perceptions of other areas involved in the decision-making process, like business analysts and product owners.
- **Extend the time of the study**, because our study lasted only for 12 weeks, in which the SA Canvas was used for 2 full sprints and at the start of the third one.
- **Look for more evidence of usefulness** of the new pattern discovered during this study by applying it in different teams from different companies and areas of development.

Appendix A

Surveys

A.1 Background Survey

Software Analytics Survey

The survey will take approximately 4 minutes to complete.
My name is João Fidalgo, and I'm studying Software Engineering at FEUP. This survey is related with my dissertation topic, that is related with software analytics and continuous assessment of internal software quality, more specifically code quality, using analytics techniques.
Before presenting the artifact to help in the analytics activities, I would like you to answer this short survey to know the current knowledge about software analytics and the usage of some analytics tools.

* Required

1. What is your role in the current project? *

2. How many years of experience you have? *

3. How many years you have in this company? *

4. What is your project? *

5. What is Software Analytics to you? Briefly describe it. *

Software Analytics

Software analytics takes into account the source code, static and dynamic characteristics, and processes related with the development of the product. Software Analytics aims to describe, monitor, predict and improve the efficiency and quality of the software product. The main objective is to produce insightful and actionable information about the project that can help improve it and to prevent some future issues.

6. How often does your team use Software Analytics? *

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

7. How often your team use metrics to inform their decisions? *

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

8. If you answered 3, 4 or 5 in the previous question, which metrics? Do you think that those metrics give you important information for the problems you try to solve? Why? *

9. How does your team finds possible solutions to a issue? *

- ☐ Discussion based on experience
- ☐ Leaves it to the person in charge of that issue
- ☐ Implements and analyses metrics to obtain valuable information about it.
- ☐ Tries different solutions until it's fixed
- ☐ Other

10. How often your team defines quality thresholds concerning code quality? *

Quality thresholds like minimum test coverage, comment frequency, etc.

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

11. How often do you use refactoring in your code? *

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

12. Changes and improvements to the code are based on what? *

- ☐ Product owner decisions
- ☐ User reports
- ☐ Code metrics
- ☐ Members discussion
- ☐ Other

13. What is the importance of code quality in your project? *

From a range of Not Important at all to Very important.

	1	2	3	4	5	
Not important at all	○	○	○	○	○	Very important

A.2 Sprint Survey

Sprint 2 Survey

Ending the second sprint using the Software Analytics Canvas, I would like you to answer a survey about your perceptions about the tool during this sprint. I would like you to answer the questions in the most honest way, so that the results are as true as possible.

* Required

* This form will record your name, please fill your name.

1. It was easy to understand what were the objectives of Software Analytics Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

2. The usage of Software Analytics Canvas was useful. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

3. The SA Canvas helped the team to think about the quality problems in the project. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

4. The SA Canvas helped the team to think about the best metrics for the identified problems. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

5. The implemented metrics in the usage of SA Canvas leaded to improvements in the code quality. *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. The SA Canvas brings greater transparency to the decision-making process. *

In this question greater transparency refers to everyone understanding why metrics are implemented and why decisions and tasks were defined.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

7. The time that SA Canvas took from other development activities was beneficial to the project. *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

8. I found it easy to define Key Issues in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. I found it easy to define the Measurements for the defined Key Issues. *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. I found it easy to define the Methods and Tools that obtain the Measurements defined. *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. I found it easy to define Highlights after implementing the Methods and Tools. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

12. I found it easy to define Insights after analyzing the Methods and Tools. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

13. I found it easy to define Decisions after analyzing the Insights and Highlights *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

14. If you felt any difficulties understanding and using the SA Canvas, explain them here.

Answer this question if you found the SA Canvas hard to understand.
Explain what steps were hard to understand and what concepts were difficult to implement in your project.

A.3 Final Survey

Final Survey

This is the final survey meant to be answered accordingly with the Canvas usage since the beginning and
to get your intentions of keeping the usage of Software Analytics and the usage of Software Analytics
Canvas. I would like for you to answer the most honest way to get the most accurate results.

* Obrigatório

* Este formulário irá registar o seu nome, por favor preencha seu nome.

1. The Canvas' usage was useful *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

2. The issues raised were important to our project *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

3. The metrics implemented led to useful insights of our code quality. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

4. The Canvas' usage has led to improvements in the code quality *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

5. The Canvas represents a good support for the analytics tasks *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

6. The metrics and tools implemented will continue to be used in the future *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

7. Software Analytics represents a valuable method to use in the future of our project *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

8. The Canvas will continue to help us organize and implement analytics in the future *

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

A.4 Refinement Survey

Refinement Survey

This survey has the objectives of gathering feedback about the Software Analytics Canvas' elements and possible refinements to it. I would like for you to answer in the most honest way to gather the best suggestions for the future of the SA Canvas.

* Obrigatório

* Este formulário irá registar o seu nome, por favor preencha seu nome.

1. I think that "Key Issues" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

2. If you didn't found the "Key Issues" important or useful, explain briefly why.

3. I think that "Measurements" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

4. If you didn't found the "Measurements" important or useful, explain briefly why.

5. I think that "Methods and Tools" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

6. If you didn't found the "Methods and Tools" important or useful, explain briefly why.

7. I think that "Highlights" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

8. If you didn't found the "Highlights" important or useful, explain briefly why.

9. I think that "Insights" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

10. If you didn't found the "Insights" important or useful, explain briefly why.

11. I think that "Decisions" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

12. If you didn't found the "Decisions" important or useful, explain briefly why.

13. I think that "Goals" plays an important role in the SA Canvas. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

14. If you didn't found the "Goals" important or useful, explain briefly why.

15. The bottom row of the SA Canvas helps the team to organize the analytics tasks during the development process. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly Agree

16. If you didn't found the "Analytics tasks" important or useful, explain briefly why.

17. If you have any suggestions to do for the SA Canvas write them here.

If you have any write them here with all the details, to help us understand why and what would be that change. This can be very important for possible future refinements in the Software Analytics Canvas.

References

[1] P Antonellis, D Antoniou, Y Kanellopoulos, C Makris, E Theodoridis, C Tjortjis, and N Tsirakis. A data mining methodology for evaluating maintainability according to iso/iec-9126 software engineering–product quality standard. *Special Session on System Quality and Maintainability-SQM2007*, 2007.

[2] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. The goal question metric approach. *Encyclopedia of Software Engineering*, 2:528–532, 1994.

[3] Nigel Bevan. Quality in use: Meeting user needs for quality. *Journal of systems and software*, 49(1):89–96, 1999.

[4] Barry W Boehm, John R Brown, and Mlity Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, pages 592–605, 1976.

[5] Nanette Brown, Robert Nord, and Ipek Ozkaya. Enabling agility through architecture. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2010.

[6] Marcel Bruch, Eric Bodden, Martin Monperrus, and Mira Mezini. Ide 2.0: collective intelligence in software development. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 53–58, 2010.

[7] Raymond P. L. Buse and Thomas Zimmermann. Information needs for software development analytics. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 987–996. IEEE, jun 2012.

[8] Joelma Choma, Eduardo Guerra, Tiago Silva da Silva, Luciana AM Zaina, and Filipe Figueiredo Correia. Towards an artifact to support agile teams in software analytics activities. In *SEKE*, pages 88–122, 2019.

[9] Joelma Choma, Eduardo Martins Guerra, and Tiago Silva Da Silva. Patterns for implementing software analytics in development teams. In *Proceedings of the 24th Conference on Pattern Languages of Programs*, PLoP '17, USA, 2017. The Hillside Group.

[10] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1992.

[11] Jacek Czerwonka, Nachiappan Nagappan, Wolfram Schulte, and Brendan Murphy. Codemine: Building a software development data analytics platform at microsoft. *IEEE software*, 30(4):64–71, 2013.

[12] Thomas H Davenport. Make better decisions. *Harvard business review*, 87(11):117–123, 2009.

[13] Yvonne Dittrich, Kari Rönkkö, Jeanette Eriksson, Christina Hansson, and Olle Lindeberg. Cooperative method development. *Empirical Software Engineering*, 13(3):231–260, 2008.

[14] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999.

[15] Martin Fowler. bliki: Technicaldebt, May 2019.

[16] John E Gaffney Jr. Metrics in software quality assurance. In *Proceedings of the ACM'81 conference*, pages 126–130, 1981.

[17] Alan Gillies. *Software quality: theory and management*. Lulu. com, 2011.

[18] Shrinath Gupta, Himanshu Kumar Singh, Radhika D. Venkatasubramanyam, and Umesh Uppili. Scqam: A scalable structured code quality assessment method for industrial software. In *Proceedings of the 22nd International Conference on Program Comprehension*, ICPC 2014, page 244–252, New York, NY, USA, 2014. Association for Computing Machinery.

[19] Liliana Guzmán, Marc Oriol, Pilar Rodríguez, Xavier Franch, Andreas Jedlitschka, and Markku Oivo. How can quality awareness support rapid software development?–a research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 167–173. Springer, 2017.

[20] Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: a survey of iso/iec 9126. *IEEE Software*, 21(5):88–92, 2004.

[21] Stephen H Kan. *Metrics and models in software quality engineering*. Addison-Wesley Professional, 2003.

[22] Avinash Kaushik. *Web analytics 2.0: The art of online accountability and science of customer centricity*. John Wiley & Sons, 2009.

[23] Chaitanya Kothapalli, S. G. Ganesh, Himanshu K. Singh, D. V. Radhika, T. Rajaram, K. Ravikanth, Shrinath Gupta, and Kiron Rao. Continual monitoring of code quality. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, page 175–184, New York, NY, USA, 2011. Association for Computing Machinery.

[24] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. Technical debt: From metaphor to theory and practice. *Ieee software*, 29(6):18–21, 2012.

[25] Silverio Martínez-Fernández, Andreas Jedlitschka, Liliana Guzmán, and Anna Maria Vollmer. A quality model for actionable analytics in rapid software development. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 370–377. IEEE, 2018.

[26] Jim A McCall, Paul K Richards, and Gene F Walters. Factors in software quality. volume i. concepts and definitions of software quality. Technical report, GENERAL ELECTRIC CO SUNNYVALE CA, 1977.

[27] Steve McConnell. Managing technical debt. *Construx Software Builders, Inc*, pages 1–14, 2008.

[28] P. Murthy, S. K. V, T. Sharma, and K. Rao. Quality model driven dynamic analysis. In *2011 IEEE 35th Annual Computer Software and Applications Conference*, pages 360–365, 2011.

[29] Padmalata Nistala, Kesav Vithal Nori, and Raghu Reddy. Software Quality Models: A Systematic Mapping Study. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pages 125–134. IEEE, may 2019.

[30] Alexander Osterwalder and Yves Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, 2010.

[31] Kai Petersen and Cigdem Gencel. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, pages 81–89, 2013.

[32] Reinhold Plösch, Harald Gruber, Christian Körner, and Matthias Saft. A method for continuous code quality management using static analysis. *Proceedings - 7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010*, pages 370–375, 2010.

[33] R. Plösch, H. Gruber, A. Hentschel, Ch Körner, G. Pomberger, S. Schiffer, M. Saft, and S. Storck. The emisq method and its tool support-expert-based evaluation of internal software quality. *Innovations in systems and software engineering*, 4(1):3–15, 2008.

[34] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, December 2008.

[35] G. Samarthyam, G. Suryanarayana, T. Sharma, and S. Gupta. Midas: A design quality assessment method for industrial software. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 911–920, 2013.

[36] Chris Sterling. *Managing Software Debt: Building for Inevitable Change (Adobe Reader)*. Addison-Wesley Professional, 2010.

[37] Edith Tom, AybüKe Aurum, and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.

[38] Enrique Larios Vargas, Joseph Hejderup, Maria Kechagia, Magiel Bruntink, and Georgios Gousios. Enabling real-time feedback in software engineering. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, pages 21–24, 2018.

[39] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidi, A. Goeb, and J. Streit. The quamoco product quality modelling and assessment approach. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 1133–1142, 2012.

[40] Dongmei Zhang, Yingnong Dang, Jian-Guang Lou, Shi Han, Haidong Zhang, and Tao Xie. Software analytics as a learning case in practice: Approaches and experiences. In *Proceedings of the international workshop on machine learning technologies in software engineering*, pages 55–58, 2011.