# UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering, Mathematics & Computer Science**

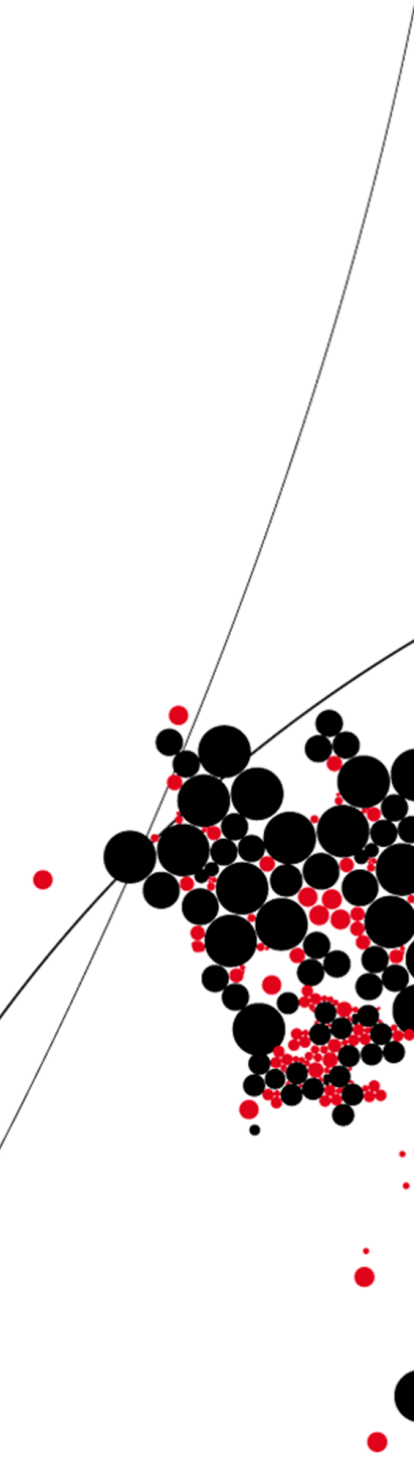
# Design and Implementation of an Assessment Method based on the Capability Maturity Model Integration (CMMI)

**Stylianos Gavriel - s1975412**

**Master Thesis Business Information and Technology**

**At Trading Point Holdings Ltd.**

**March 2023**



**Supervisors:**
Dr. G. Sedrakyan (UT)
Dr. L. Ferreira Pires (UT)
C. Hadjichristodoulou (XM)
W. Zubik (XM)

Faculty of Electrical Engineering,
Mathematics and Computer Science (EEMCS)
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

# Preface

After a journey of five and a half years, my student life at the University of Twente has reached its conclusion. This period has been filled with countless adventures and opportunities for personal growth. It all began with my Bachelor's degree in Business Information and Technology, where I was fortunate to have remarkable professors who guided me towards becoming a more critical thinker and expanding my expertise in various domains. Continuing on this path, I pursued a Master's degree in Business Information and Technology with a specialization in Data Science. The completion of this thesis marks the culmination of seven months of dedicated work, and I am genuinely pleased with the outcome. Numerous individuals have played a significant role in the realization of this thesis, and I would like to express my gratitude to some of them:

I would like to thank my supervisors Gayane Sedrakyan and Luis Ferreira Pires, whose guidance and support were invaluable throughout the entire process. I would like to thank them for the countless hours they must have spent reading my drafts, and their expertise and insightful feedback which have greatly contributed to the quality of this thesis.

I would also like to thank my team leader Charalambos Hadjichristodoulou for the opportunity to work closely at the IT Governance department and gain working experience. I would like to thank my colleagues and company supervisors of Trading Point Holdings Ltd, Charalambos Hadjichristodoulou and Waldemar Zubik, who have been a source of motivation, inspiration and endless support. Their encouragement and camaraderie made this journey all the more enjoyable. I would like to thank them for the time they made for me despite their busy schedules.

I would also like to thank other team members for the introduction and guidance to the processes within the organization. For keeping me motivated throughout both coffee breaks as well as stressful periods. I would like to thank the people who provided me with the tools and data to make this thesis possible.

Most importantly, I would like to thank my friends and family, especially my parents, sister and girlfriend, for always supporting me throughout my studies. This thesis would not have been possible without the support you provided me in hard times.

# Contents

# List of Figures

# List of Tables

# List of acronyms

| | |
|---|---|
| **IT** | Information Technology |
| **CMMI** | Capability Maturity Model Integration |
| **CMM** | Capability Maturity Model |
| **DevOps** | Development and Operations |
| **SW-CMM** | Capability Maturity Model for Software |
| **SECM** | System Engineering Capability Model |
| **IPD-CMM** | Integrated Product Development Capability Maturity Model |
| **BPMM** | Business Process Maturity Model |
| **KPIs** | Key Performance Indicators |
| **BSC** | Balanced Scorecard |
| **KRIs** | Key Result Indicators |
| **RIs** | Result Indicators |
| **PIs** | Performance Indicators |
| **CSFs** | Critical Success Factors |
| **CI/CD** | Continuous Integration and Development |
| **SLAs** | Service Level Agreements |

# Chapter 1

# Introduction

Numerous activities can contribute to effective governance within an organization, spanning across a wide range of internal and external areas. One valuable resource that software development organizations may find indispensable is the utilization of maturity models, such as the Capability Maturity Model Integration (CMMI), in their pursuit of attracting and retaining customers [12]. Maturity models serve as invaluable tools for IT governance, facilitating the assessment of an organization's current state and the identification of practical improvement measures [13]. Maturity models are employed for process improvement, their effectiveness is lying into accurately identifying the appropriate metrics and proposing improvement strategies, allowing organizations to consistently enhance their engineering proficiency across various dimensions of software development. These assessments play a pivotal role in the effective monitoring, management, and enhancement of IT processes and structures specific to software development [14]. This is enabled by facilitating continuous monitoring of software development efficiency and effectiveness [15].

Software development processes are regarded as the main area for quality improvement and can be broken down depending on the organization into five dimensions. Requirements Management is one of these dimensions and can be characterized as the process of gathering, analyzing, verifying, and validating the requirements and needs for a given product or system being developed [16]. It enables to clearly understand stakeholder expectations and confidently deliver a product that has been verified to meet the requirements. Quality Assurance Management is an example of a software development dimension, which refers to the processes and methods used to prevent any kind of mistakes, inconsistencies, and defects within software development processes [17]. Furthermore, many organizations follow set of rules and practices within their development teams (Team Practices dimension), including healthy and consistent code practices, code quality gates and alignment with the organization's enterprise architecture [18]. Numerous organizations adhere to specific sets of rules for software release, such as (Agile) Development Prac-

tices [19] and DevOps Practices [20], enabling the achievement of coherence, standardization, and quality through iterative testing and continuous enhancement of the software development process. However, there is a need to improve these processes and measures for performance. Metrics in the context of system maturity level can be defined as ”An objective indicator or measure which facilitates system maturity level improvement.” [21]. In this thesis, the focus was to improve software development processes applying an assessment method constructed by the CMMI model.

## 1.1 Motivation

To meet the increasing and competing demand of software, many organizations have adopted the CMMI model. CMMI covers 22 key processes which can be overwhelming and opens the question of how to best implement it, considering the specific circumstances of an organization [9]. The vast majority of the organizations, however, seem to have failed because of the high dependence of the quality of a product to the quality of software development processes that are used to develop and maintain it [9]. The CMMI can directly emphasize software development practices, however, there is a lack of transparency and practical usage experience of such models [12]. Ayyagari and Atoum [4] mentioned that CMMI as a reference model does not specify a systematic process of how to implement the model in practice, leaving room for organizational development approaches. Furthermore, they state that small organizations are having difficulties implementing the model due to the high costs that come with it. Moreover, Raul Vidal et al. [8] mentioned that there are differences in performance between CMMI implementations within organizations, depending on not only of the context of the business, projects and teams but also on the methodologies used in the implementation of the model.

## 1.2 Research Goal

The thesis aimed to provide the research community with a method for assessing and benchmarking an organization’s maturity level of software development processes with the CMMI model. By using a consistent set of assessment criteria (five dimensions), the assessment method helped the organization to compare the level of maturity within the organization’s teams or with others in the same industry or with best practices in the field.

In order to enable the above goals, the thesis includes a literature review on the CMMI model. Several software development dimensions are identified according

to the organizational experts and literature review helped to get a comprehensive view. This in turn helped to design and implement an assessment method based on the CMMI model and the organization's context. Consequently, multiple assessment surveys have been deployed, which allowed to assess the teams, evaluate the maturity of development processes across the five identified dimensions, understand what development teams are lacking and what risks they face. The results obtained with the assessment surveys have been analysed and visualized according to the opinion of organizational experts and best practices from the literature. Therefore, the designed assessment method allowed us to guide the organization on how to achieve a higher level of maturity by introducing improvements. This includes recommendations on tools, processes, and techniques that are known to improve development practices.

This thesis questioned the validity of the assessment method by following two methods. Firstly, by deploying a validation survey towards the assessment method for feedback by the participants, and secondly, by examining performance metrics of software development processes, while comparing performance with historical data of the organization.

As such, the implementation of the assessment method and analysis of results, enriches the current literature by introducing an assessment method and steps towards making the method's used more transparent.

## 1.3 Thesis Organization

The remainder of this report is organized as follows. Chapter 2, gives the background of this thesis, in Chapter 3, the research methodology of this paper is described, defining our methodology used, research questions, and a reflection on the methods used. In Chapter 4, literature reviews are performed to underpin the attributes of CMMI models and the software development dimensions found by the experts to accommodate the organization's context. Further, literature helped identify what data presentation techniques help to create an environment of understanding the data collected by the assessment surveys enabling the introduction of improvements, and what metrics are relevant for measuring software development processes. Chapter 5 addresses the design plan, and the designed assessment method. Chapter 6 discusses the analysis and interpretation of the assessment survey results thought data presentation techniques, as well as a discussion of improvements with experts. Chapter 7 includes an analysis of the methods for validating the assessment method, by both validation surveys and by using software development metrics. Finally, Chapter 8 and Chapter 9, present the discussion and conclusion of this thesis, respectively.

# Chapter 2

# Background

This chapter gives a brief overview of the topics of the thesis by providing what IT governance is, the role of maturity models in IT governance, and an introduction of the CMMI model and its implication. This thesis utilizes the CMMI model as a framework for process improvement to design the assessment method.

## 2.1 What is IT Governance?

IT governance involves the readiness to formulate and execute decisions related to objectives, procedures, interested parties, and technology at both tactical and strategic levels [22]. Wim [23] defined IT Governance as ”the organizational capacity exercised by the board, executive management and IT management to control the formulation and implementation of IT strategy and in this way ensuring the fusion of business and IT”. Although many definitions exist most of them focus on the same aspects, mainly achieving the link between business and IT and that the primary responsibility is held by the board of directors [15].

IT governance is an element of corporate governance which aims to improve the overall management of IT and derive improved value from investments in information and technology. The main three crucial IT governance questions are concerned with efficiency, effectiveness and control of IT. Efficiency refers to how efficient business processes enable high return on investment, risk mitigation and fast and reliable delivery of services and products. Effectiveness usually concerns the alignment of business with IT processes. While control in IT refers to the procedures or policies that provide a reasonable assurance that the IT used by an organization operates as intended. IT Governance thus reflects broader corporate governance principles, both pursuing an ongoing questioning of the organization’s governance model’s sufficiency in minimizing risks and maximizing returns [24].

The growing demand for attention in governance responsibility while simultane-

ously managing and integrating service contribution to organizational functioning creates an ever-increasing dependency on effective IT systems and processes [25]. Successful organizations must ensure that they will be able to integrate their IT contributions with strategies, culture and desired ethics in order to achieve their business objectives, while also capitalizing the utilization of technology and information value. An effective IT governance initiative needs to provide mechanisms for IT managers to develop business and IT plans, while allocating responsibilities and accountabilities, prioritise and organise IT initiatives whilst following business requirements, and track performance and outcomes.

## 2.2 IT Maturity Models

Maturity models are tools that facilitate internal and external benchmarking while also identifying future improvements and providing guidelines through evolutionary processes of organizational development and growth [26]. A maturity assessment can be used to measure the current maturity level of several aspects of an organization, enabling concerned stakeholders to clearly identify strengths and improvement points and hence prioritize tasks to increase maturity levels. The term ”maturity” may be defined as ”the state of being complete, perfect or ready” [26]. In the context of software development processes, ”maturity” is defined as the state of software following best software engineering practices and compatibility with internationally accepted standards. Maturity examines the activities of software product development or service provision and the integrity of how well these activities are performed, hence the ability of achieving the defined scope, quality goals, cost, and schedule [27]. Maturity models consist of several maturity levels from lowest to highest depending on the domain and the concerns motivating the model [28]. In this way, they can identify organizational strengths and weaknesses while also providing benchmark information.

### 2.2.1 The role of maturity models in IT governance

Maturity models play a crucial role as a tool that can aid in identifying weak points of an organization towards their efforts to introduce IT governance. IT governance helps businesses prioritise and organise IT initiatives, while maturity models can help achieve certain goals within this domain. Smits and Hillegersberg [7], describe a study to develop an IT governance maturity model. They had built an IT governance model foundation that characterized the sides of IT governance. They mentioned that to advance maturity in an organization, attention is driven on both the

hard and soft sides of IT governance. The IT governance model depicted in Table 2.1, was built as a foundation to develop a maturity model. The hard side is often related to structure and processes, while the soft side is related to social aspects like behaviour and culture in organizations.

| Governance | Domain | Focus area |
|---|---|---|
| Soft | Behavior | Continuous improvement |
| | | Leadership |
| | Collaboration | Participation |
| | | Understanding and trust |
| Hard | Structure | Functions and roles |
| | | Formal networks |
| | Process | IT decision-making |
| | | Planning |
| | | Monitoring |
| Context | Internal | Culture |
| | | Informal organization |
| | External | Sector |

**Table 2.1:** IT governance model foundation [7].

### 2.2.2 Generic structure of maturity models in IS literature

The purpose of maturity models is to outline the path to maturation in an organization, including the definition of stages and relationship between them [29]. The assumption of these models is that enabling a higher score or degree of maturity also increases positive change in a variety of dimensions, so that the model captures the maturation process while providing a construct or an artifact to measure progression.

Lester et al. [30] identified five important components to describe a maturity model: (i) *Maturity levels*: used to describe the overall maturity of an entity and the level of abstraction at the highest level, (ii) *Dimensions*: termed as benchmark variables, process areas, capability and critical success factors [31], (iii) *Sub-categories*: second level variables that the dimensions depend on, (iv) *Path to maturity*: most maturity models following in a linear path from lowest to highest maturity and (v) *Assessment questions*: usually linked to the sub-categories with the maturity score visualized most often as a graphical representation. According to the above components, Lester et al. [30] presented the generic structure of a maturity model depicted in Figure 2.1 and Figure 2.2, which visualize the measurement hierarchy.

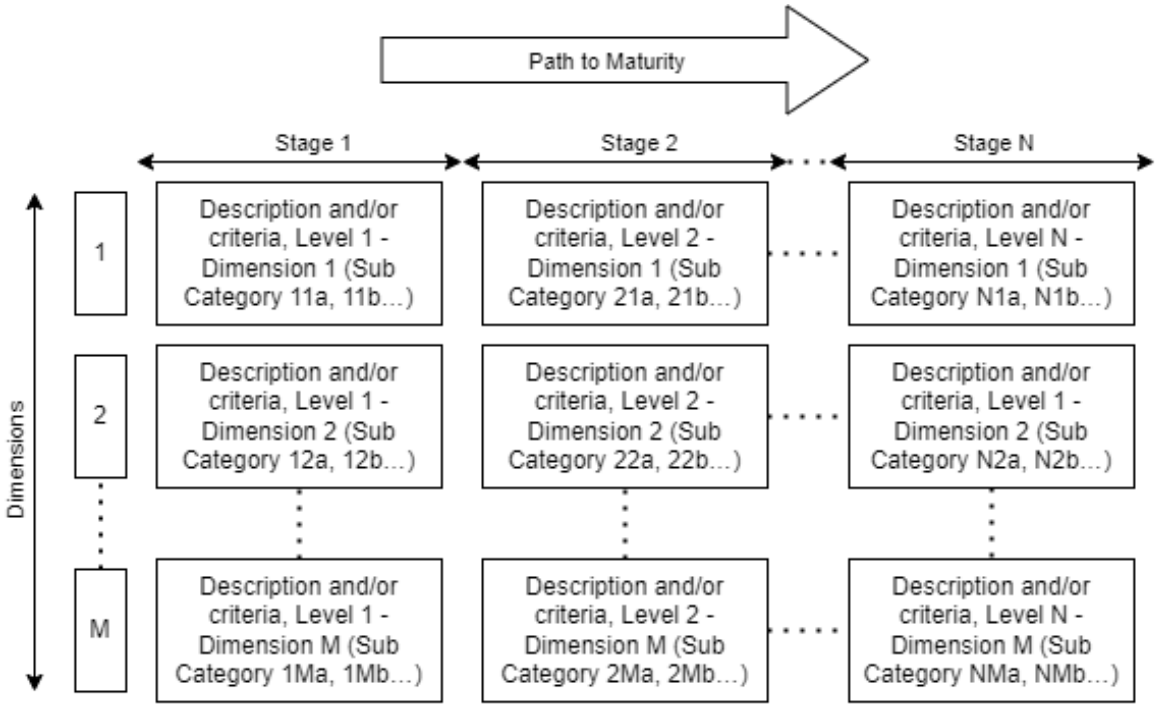



**Figure 2.1:** Design Structure Maturity Model.

Methods for assessing the maturity of an organization acknowledge the importance of measuring quality, control, and management of the software development process, while each technique has elements to measure processes.

### 2.2.3 Capability Maturity Model (CMM)

Capability Maturity Model (CMM) was created in 1986 has been widely accepted as a standard and has been used in a wide array of problem areas [30]. CMM focuses on improving processes in an organization. These models address elements of effective processes for one or more disciplines, describing an improvement path from immature processes to mature processes with improved quality and effectiveness [32]. Further, they can be a tool and a mechanism towards both ensuring the production of high-quality indicators in terms of data, but also for standardizing key processes around the production of Sustainable Development Goals (SDGs), and for facilitating interoperability within the data ecosystem [33]. However, CMM application in software development has sometimes been problematic, by applying multiple models that are not integrated across the organization, which can be costly in training and improvement activities [32].

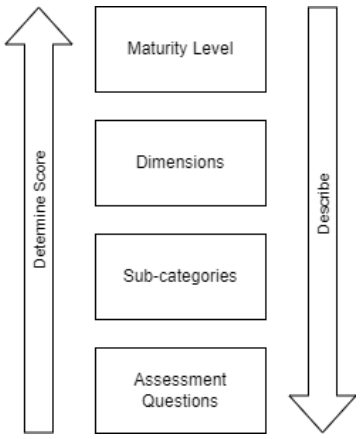



**Figure 2.2:** Measurement Hierarchy.

### 2.2.4 Capability Maturity Model Integration (CMMI)

Following the emergence of CMM, CMMI emerged at the start of 2000s as an improved framework of CMM, in order to improve software development processes to achieve higher quality [34]. CMMI comes with general guidelines and models that transcend disciplines, addressing the entire cycle of a service or product, including development, delivery, and maintenance [35]. CMMI was introduced to avoid the problem of having to use multiple CMMs, with a combination of the Capability Maturity Model for Software (SW-CMM), the System Engineering Capability Model (SECM), and the Integrated Product Development Capability Maturity Model (IPD-CMM) [32]. The purpose is to provide a CMM that covers both product and service development as well as maintenance, together with a framework for adding new bodies of knowledge. In 2007, four main bodies of knowledge have been identified: (i) system engineering, (ii) software engineering, (iii) integrated product and process development and (iv) supplier sourcing them [32]. These knowledge bodies are addressed by the process areas associated with them and by components called discipline amplifications. A process area is a set of best practices in an area that when implemented correctly, should satisfy a set of important goals, which in turn allow for significant improvements in the area [32]. An additional feature of CMMI with respect to CMM model is the introduction of a continuous representation of the model which allows the option to assess and grade each process individually [34]. The success of CMMI inspired many researchers to develop several maturity models in other domains including the Business Process Maturity Model (BPMM).

#### 2.2.4.1 The implications of implementing the CMMI model

Raul Vidal et al. [8] identified three main problems when it comes to implementing the CMMI model in an organization: *Inadequate Metrics Deployment*, *CMMI Pro-*

*gram Management Issues*, and *CMMI Understanding Issues*. Inadequate metrics could result from improperly deploying the indicators to control a project, and frequently CMMI implementation is ineffective, and people misinterpret the process. Table 2.2, lists the problems with their description, according to [8].

| **Type of Problem** | **Description** |
|---|---|
| Inadequate Metrics Deployment | Meaningless, useless and non goal-driven indicators |
| | Inexperienced implementers |
| | Complex solutions, hard to maintain |
| | Same indicators for all situations |
| | Outdated measurement plans |
| CMMI Program Management Issues | Senior management not involved in establishing objectives, policies and the need for processes |
| | Software Engineering Performance Group not managed |
| | Sponsor not playing its role and delegating authority |
| CMMI Understanding Issues | Lack global view of the model |
| | Do not understand several Generic Practices |
| | Do not understand difference between capability and maturity level |

**Table 2.2:** Implications of implementing the CMMI model [8].

## 2.3 Conclusion

In conclusion, maturity assessments play a crucial role into enabling IT governance efforts in an organization. CMMI models can help organizations guide performance of their software development teams to a more capable state. With the continuous representation the model can assess and grade each software development process individually, while also following a continuous improvement plan. However, several implications are identified and need attention when attempting to implement the CMMI model, from understanding, managing and deploying metrics.

# Chapter 3

# Research Methodology

This chapter describes the research objective of this thesis, along with the process of achieving the objective. It also includes the research questions that were designed to assist the entire process, followed by a detailed walk-through and reflection on the methods used.

## 3.1 Research Objective and Motivation

This thesis is performed following the Design Science Research by Wieringa [1]. The objective is to develop an assessment method as an artifact which is essentially a conceptual model designed to evaluate the management of software development processes. Design Science Research provides a structured and systematic approach for creating such artifacts, by designing and refining solutions iteratively using validation cycles.

The need for an assessment method arose from the increasing complexity of software development processes, which resulted by the growing customer base and market size. However, current assessment methods are limited in their ability to provide a systematic and transparent method of how to implement an assessment for an organization. Therefore, this gap hinders the ability to provide accurate and reliable assessments. This thesis fills the gap by creating a robust method that enhances processes management within the software development domain.

To construct the design artifact, namely the assessment method, this thesis draws upon established theories and concepts, such as process improvement, assessment methods, the software engineering concepts behind the five dimensions(Team Practices, Development Practices, DevOps Practices, Requirements Management and Quality Assurance Management), measurement theory, and data-driven decision making. These theories and concepts provide a solid foundation for understanding the underline principles of assessment methods which helped to design

the artifact. The design and validation process is explained further in Section 3.2 which delves into the research questions, and in Section 3.4 which provides a reflection on the methods used.

## 3.2 Design Cycle

The Design Science Research emphasizes on the Design Cycle as stated by Wieringa, who provides an iterative problem-solving process and consists of several tasks: *problem investigation*, *treatment design*, *treatment validation*, *treatment implementation*, and *implementation evaluation*. This thesis follows a structured iterative cycle process where initial designs of the assessment method are created and refined by evaluating against received feedback, which inform the subsequent design adaptations.

The initial design treatment according to the Design Cycle, includes the assessment method as an artifact. In the context of our organization, we utilize five key software development dimensions as a reference framework, which were decided by experts. These dimensions served as the foundation for our designed assessment method, and form the assessment criteria against which the organization's software development processes are evaluated. The assessment method is based on the CMMI model, which is a comprehensive framework for process improvement and is a core component of our assessment approach. The assessment method further includes the assessment surveys in order to collect work information about the organization's processes, an analysis of the results of these surveys with data presentation techniques, and the implementation of improvements.

The treatment validation phase includes two evaluation methods: validation surveys which would provide feedback on the designed artifact and enable the iterative cycle process, and an analysis of software development metrics to measure the impact of the assessment method.

It is worth noting that the evaluation methods played a significant role in redesigning the assessment method. However, conducting additional iterations for treatment implementation and validation of the cycle process was not feasible. This is because implementing improvements often requires a substantial amount of time, sometimes up to a year or two. Therefore, a second iteration of the process was not feasible within the given time-frame.

Figure 3.1 showcases the steps of this thesis following the Design Cycle. A more detailed explanation of the steps and reflection on the methods used can be found in Section 3.4.

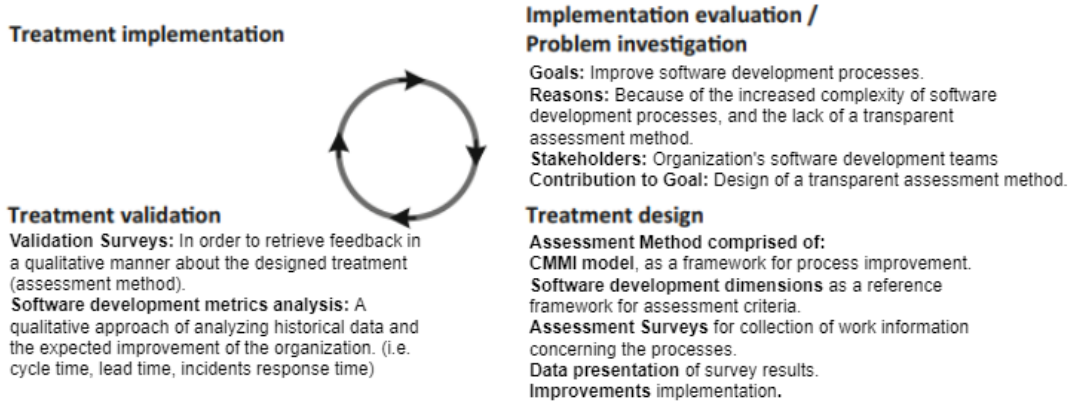



**Figure 3.1:** Design Cycle from Wieringa [1].

## 3.3 Research Questions

The template depicted in Figure 3.2, allowed the formulation of our main research question.

Improve <a problem context>
by <(re)designing an artifact>
that satisfies <some requirements>
in order to <help stakeholders achieve some goals>

**Figure 3.2:** Template for defining design problems [1].

Hence, we aimed to improve the performance of software development processes, in order to cope with the ever increasing complexity of the organization's processes. In order to achieve this, an assessment method that satisfies the organization's context has been carefully designed. Therefore, the main research question is:

*How can we improve the software development processes of the organization by applying an assessment method, in order to help the organization cope with the ever increasing complexity of their processes?*

To address this research question, three sub-questions are answered. The first sub-question focused on understanding the first two components of the assessment method. To design the method, we conducted a literature review to understand the CMMI model's attributes and the five dimensions that serve as assessment criteria for the organization's software development processes. Therefore, the first sub-question is as follows:

*1. What are the attributes of CMMI models, that aid to design an assessment method for software development processes in the context of the organization?*

Based on insights from sub-question one, we proceeded to design the initial assessment method. This method includes criteria from the five software development dimensions, the CMMI model, and input from experts who helped design the assessment surveys. To effectively analyze survey results and implement improvements, we conducted a literature review to explore data presentation techniques. Thus, the second sub-question is as follows:

*2. What data presentation techniques can be used to effectively visualize the assessment surveys results and support the implementation of improvements?*

To evaluate the implementation of the designed assessment method, two approaches were employed. First, we create a validation survey to gather feedback qualitatively of the assessment method and its relevance to the organization. Second, we identified software development metrics from literature to validate the assessment method. Analyzing these metrics enabled us to quantitatively validate the assessment method's effect by comparing the organization's software development performance with historical data. Therefore, the final sub-question is formulated as follows:

*3. How does our assessment method affect the performance of the organization's software development processes?*

The combination of the research and sub-questions lead to the research model presented in Figure 3.3.

## 3.4 Research Method

This section provides a comprehensive overview of the steps taken in this thesis. We further reflect on the methods used for the components of the assessment method, as well as the validation techniques.

### 3.4.1 Designing the Assessment Method

As mentioned earlier, our assessment method is comprised of five software development dimensions, the CMMI model, assessment surveys, survey results visualization, and implementation of improvements. These dimensions were established by experts who possess deep knowledge of the organization's context and software

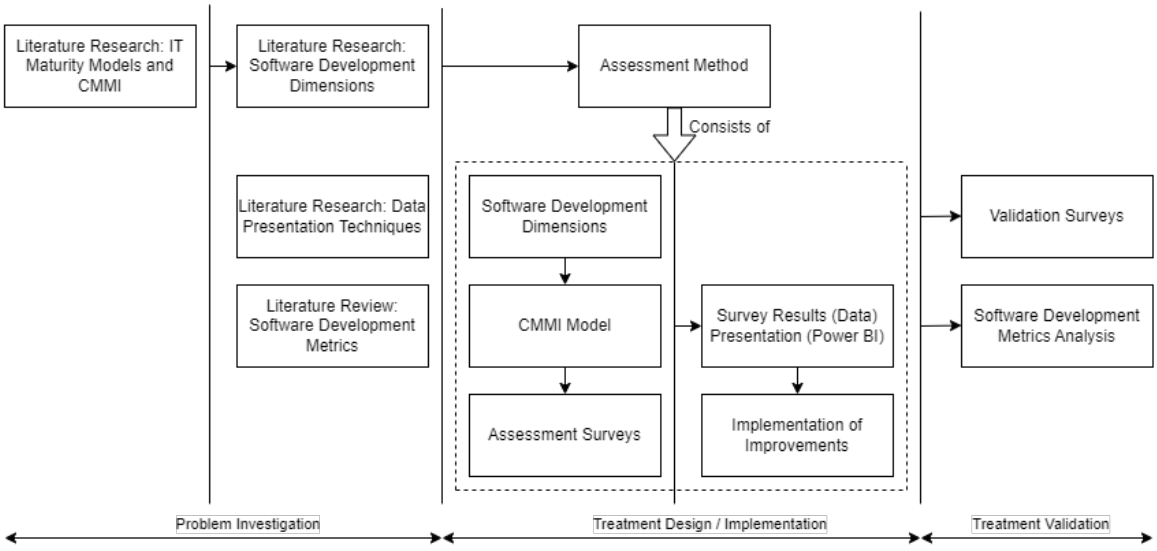



**Figure 3.3:** Research Model.

development processes. To make the assessment process more efficient, we divided these dimensions into categories. The main goal of using these software development dimensions was to ensure that our assessment method aligns with the organization's context and establishes the assessment criteria.

On the other hand, the CMMI model was utilized since it provides a comprehensive framework for process improvement, and guided the assessment method. Khoshgoftar and Osman [36], compared several maturity models, however, most of them either focus on project management or business. CMMI models are primarily designed for software, with a focus on software development processes, making them a suitable framework to follow in this thesis. Literature review enabled us to find two representations of the CMMI model, the staged and the continuous representation which is followed in this thesis. Literature further enabled us to understand the process areas of the model which fall under four categories: Project Management, Support, Engineering, and Process Management, and are used as guidelines for process improvement.

Once an understanding of the five dimensions and process areas of CMMI models was acquired, together with experts we designed assessment surveys. The idea behind the assessment surveys was to help us collect quantitative data regarding the organization's software development processes. Iqbal et al. performed in a similar fashion a survey where participants where asked to score the process areas of the CMMI model using percentages [37]. Pikkarainen and Mäntyniemi discussed about three case studies that have been studied to determine how the CMMI model could be used in assessing agile software development [38]. Amongst these cases, two of them have performed interviews while the third one had a combination of interviews

and surveys. In this thesis we conducted surveys for assessing in a quantitative matter. Using quantitative analysis instead of qualitative analysis for this component allowed us to employ questions in a binary response format, where participants provided a straightforward ”Yes” or ”No” answer. Closed questions provided us with a quick and easy way of collecting a huge number of responses opposed to interviews and open-ended questions, which require much more effort.

The survey results involved the allocation of scores to questions answered as ”Yes”. The scoring process follows the maturity score of each question, which was assigned during the design of the assessment method, which followed the framework depicted by the CMMI model. This is elaborated on further in Chapter 4, which delves into the CMMI model, as well as in Chapter 5, which presents the assessment method. This scoring method is selected to create a quantitative representation of the maturity across the organization’s processes.

To derive actionable insights from the outcomes of the assessment surveys, a data presentation tool (Power BI) was used. Literature review provided us with several other tools that feature similar functionalities, however Power BI is a very powerful tool and provides the means to data pre-processing, analysis and visualization. By utilizing the data presentation tool and its charts, it was possible to effectively communicate the results, and prioritize processes for improvement.

Moving forward, we designed a conceptual map to organize the thesis’ concepts. Conceptual maps visually depict connections between ideas around a central theme. In contrast, a theoretical framework summarizes concepts and theories from established research knowledge [39]. Since this thesis reflects operational elements, we opted for a conceptual framework.

We also identified two types of conceptual variables: independent and dependent. The five software development dimensions and the corresponding CMMI model process areas are independent variables. Introducing improvements allowed us to control these dimensions and observe their impact on dependent variables. These dependent variables, are represented by various software development metrics, which measure the assessment method’s effects on the organization’s performance. Figure 3.4 depicts the conceptual map.

### 3.4.2 Treatment Validation

The final step of Wieringa’s Design Cycle, is treatment validation. Two steps ensure that this thesis follows a proper treatment validation:

Firstly, a validation survey was created as a qualitative approach. Qualitative research, compared to a quantitative research, collects stronger information, because it is possible to pinpoint exact concerns, and details. Surveys facilitate prompt re-

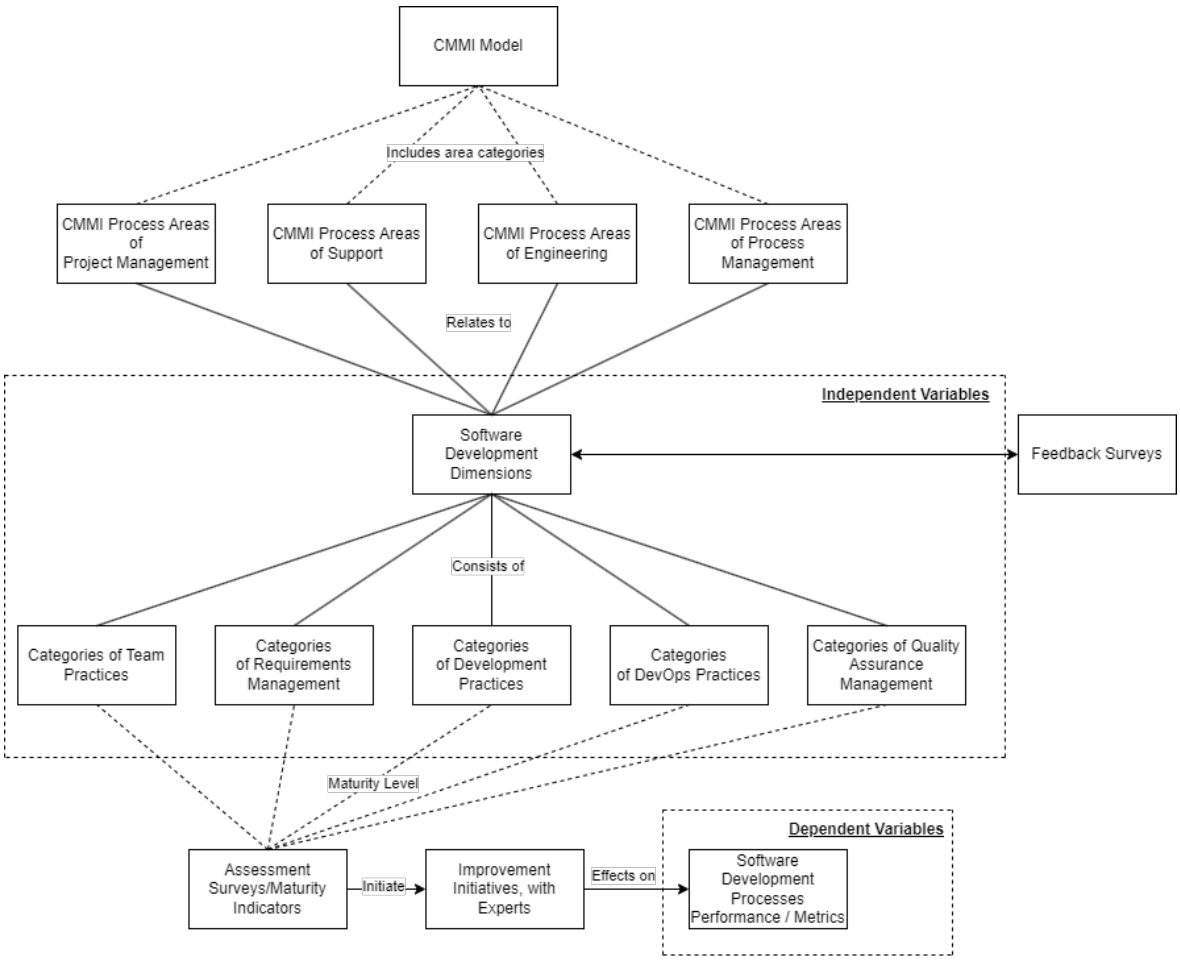



**Figure 3.4:** Conceptual Map.

sponses through open-ended questions opposed to interviews, thereby streamlining the process of collecting the opinions of the participants. Therefore, a validation survey was conducted in order to gain feedback from the participants of the assessment method which mainly included team leaders, as they can provide a big picture of their teams. Team leaders were chosen, in contrast to team members to have a concise response of the team from people who have the whole picture instead of individual responsibilities and thoughts. The use of open-ended questions served the purpose of capturing the participants' perspectives on the assessment method and acquired contextual insights that might remain inaccessible through closed questions [40]. The survey has questions related to the assessment method's structure, practicality, and relevance to the teams and their specific concerns.

Secondly, following the conceptual map this thesis adopted also a quantitative approach with software development metrics, where data is extracted from the organization's existing databases. The advantages of quantitative research over qualitative in this instance is that it reduces research bias, enables repeatability, and facilitates easy comparisons and interpretation of results [41]. This choice was practical

because of the large data volume and the availability of historical data of the organization's processes. The selection of metrics was based on both established best practices in the literature and the limitations of available data from the organization. These metrics, like average lead time, average cycle time, and incident response times of software developers, were then visualized to provide quantitative insights into the impact of the assessment method.

# Chapter 4

# Literature Review

This chapter introduces the literature review, first, we include an overview which describes how this literature review was performed. Then we include the literature review which demonstrates knowledge on the components of assessment methods which helped us bridge the knowledge gap. First, by reviewing the CMMI model, its process areas and representations. Then by gaining knowledge on the five software development dimensions, and on the data presentation techniques. Finally, we include a literature review on software development metrics used to validate the assessment method.

## 4.1 Literature Overview

In order to gain a comprehensive grasp of the relevant literature concerning the subjects addressed in this thesis, a systematic literature review was performed, following the approach outlined by Carnwell and Daly [42]. The approach suggested by Carnwell and Daly follows five steps: defining the scope of the review, identifying the sources of relevant information, reviewing the literature, writing the review, and applying the literature to the proposed study.

The first step is defining the scope of the literature review. In this thesis the scope of the literature review is to get an understanding of assessment methods, which relate to the theories and concepts behind process improvement, concepts of the dimensions, measurement theory, and data-driven decision making. In order to design the assessment method, we get an understanding of the components such as the attributes of CMMI models, the software development dimensions, and methods for data presentation. Finally, for validation purposes we get an understanding of what metrics can be used to measure performance of software development. In this sense this thesis mostly attempts to capture theories and concepts specific to the study. However, some empirical works have been considered but mostly to gain the

perspectives out of these topics. In a nutshell, this literature review tried to collect relations between theory and concepts, rather than attempting to empirically compare outcomes. This was done since concepts used during this thesis have provided wide evidence amongst the research community on the benefits they provide to an organization, therefore it was seemed unnecessary to explore them empirically.

To ensure the quality of the chosen papers, only scholarly databases were used which included Google Scholar, IEEE, Springer Link, and Science Direct. The criteria for inclusion encompassed journal articles, conference papers, books, and papers from esteemed institutions. Furthermore, efforts were exerted to concentrate on up-to-date papers published after 2015. Nonetheless, in certain instances, locating recent literature on specific topics posed a challenge, necessitating the inclusion of older papers to attain an understanding. An example is the article published by the Software Engineering Institute, on CMMI for development version 1.3 [3] in 2010 since this was the last version of the model published as of the execution of this thesis.

The second step of Carnwell and Daly's approach is identifying and selecting the sources of relevant information. The search terms are devised through an exploratory literature review with manual input, and are included in Table A of the Appendix, as a research protocol. In essence, papers included where definitions on the concepts of the thesis, in the IT domain, and papers with little to no counter arguments. On the other hand, we excluded papers that had no theoretical consensus, and papers that have multiple viewpoints or positions regarding the topics.

The third step is reviewing the literature. The exploratory process of choosing research papers went through several phases. First, keywords were selected according to the topics of this thesis: the CMMI model, Team Practices in software development, Quality Assurance Management, Software Development Practices, DevOps Practices, Requirements Management in software development, Data Visualization or Presentation, Software Development metrics. A list of keywords can be found in Table A.2 of the Appendix. Then the outcomes were screened based on their titles to eliminate clearly irrelevant papers. Subsequently, the abstracts and key sections of the remaining papers were reviewed, resulting in the exclusion of additional non-relevant ones. Papers that still remained uncertain in terms of relevance were included in the subsequent step, during which they were carefully read to make a final determination regarding their inclusion or exclusion. The total number of the remaining papers was 50, considering the topics discussed earlier, 15 of them for the CMMI model, 15 for the five dimensions, 9 for data presentation, and 11 for software development metrics. A detailed break down of the number of papers for each step can be found in Table A.3 of the Appendix.

The final step is constructing the review in the subsequent sections, which demon-

strate insights into the current knowledge in the fields and help to answer the sub-questions defined in Chapter 3.

## 4.2 CMMI Representations

According to Chaudhary and Chopra [10], the model follows two sets of representations. The staged representation is well known and follows a structure of five levels of organizational maturity. It thus enables comparison between organizations and offers a proven road map for improvement initiatives. Moreover, the continuous representation of CMMI allows organizations to specify the order of improvement initiatives that best fit the organizational objectives and mitigate areas of high risk at the time [4]. There are usually five levels of maturity within an organizational plateau for the overall capability of that organization, with each level having a predefined set of processes that are assigned to it. This section describes the staged and continuous representations of the model.

### 4.2.1 Staged Representation

The staged representation focuses on best practices that an organization can use to improve their processes in the areas that are within the maturity level it chooses to achieve [43]. Maturity levels organize the process areas, which in turn include generic and specific goals as well as generic and specific practices. While the common features organize the generic practices. It is important however, that organization map their processes to the associated process areas of the model. Therefore, enabling to control process improvement by tracking the organization's maturity level of conformance to the model. It is imperative however that not every organizational process can be mapped to a specific process area [3]. The structure of the CMMI staged representation is depicted in Figure 4.1.

**Specific Goals** apply to a process area and deal with the unique characteristics that describe what must be implemented to satisfy the process area. They are a required model component and are used in appraisals to determine if requirements of a process area are met [2]. **Specific Practices** are activities that are considered important in meeting the requirements of the associated specific goal. They are expected model components and describe activities expected to result into achieving the specific goals of the process areas. **Common Features** organize the generic practices of each process area and they are not rated in any way. **Generic Goals** as the name implies are called generic because the same goal statements can appear in many process areas. In a staged representation each process area includes only

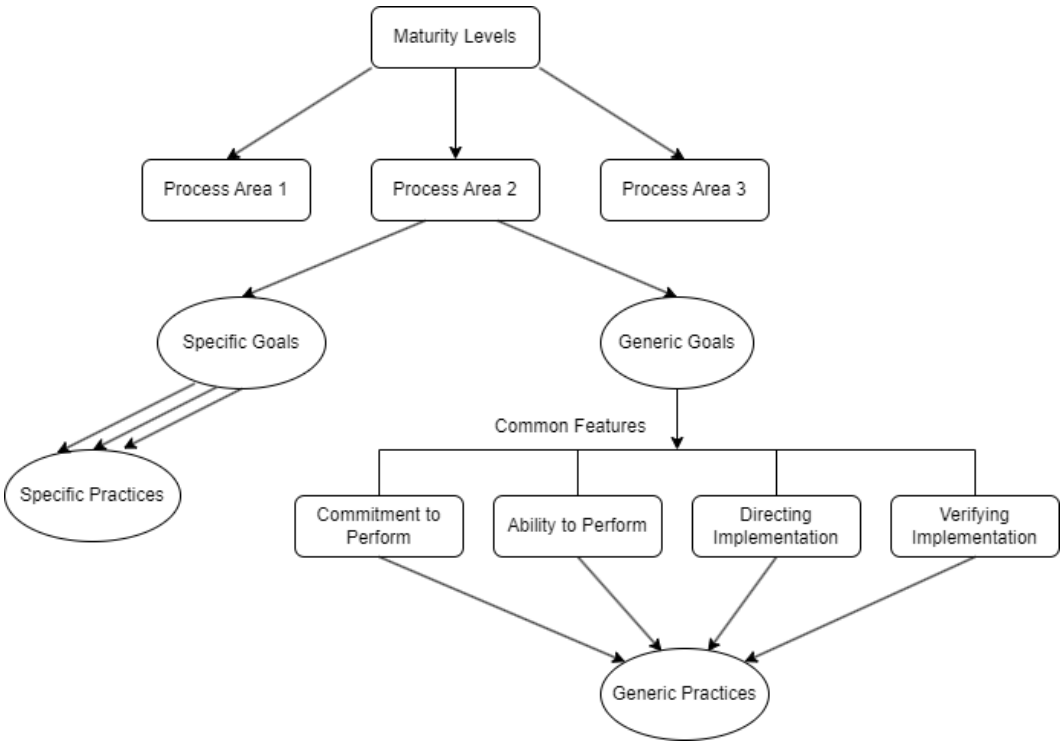



**Figure 4.1:** The structure of a CMMI staged representation [2].

one generic goal. They are required model components and through their achievement signifies improved control in planning and implementation of processes associated with that process area [3]. **Generic Practices** provide categorization to ensure that the processes associated with the process areas will be repeatable, lasting and effective. They are expected components of the maturity models and are categorized by generic goals and common features.

Usually the staged representation of CMMI organizes process areas into five **Maturity Levels**. These levels include, *Initial (ML1)*, *Managed (ML2)*, *Defined (ML3)*, *Quantitatively Managed (ML4)*, and *Optimizing (ML5)* [28] [10]. They describe an evolution of improvements for software development processes, beginning with basic improvement practices and progresses through a predefined set of success levels [28]. For a staged model these levels provide the organizations with a recommended order of approaching process improvement. As the name implies for the staged representation, maturity levels provide a recommended order of approaching process improvement in stages. The maturity levels of an organization can be measured by the achievement of the specific and/or generic goals that do apply to each set of process area. Table 4.1 depicts the focus of each maturity level and the corresponding set of process area.

At maturity level 1 (ML1) processes are usually ad hoc and chaotic, where the organization usually does not provide a stable environment. Competences and hero-

| Maturity level (ML) | Focus | Process area |
|---|---|---|
| ML 1: Initial | ad hoc process | None of the process areas |
| ML 2: Managed | Basic project management | Requirements management<br>Project planning<br>Project monitoring and control<br>Supplier agreement management<br>Measurement and analysis<br>Process and product quality assurance<br>Configuration management |
| ML 3: Defined | Process standardization | Requirements development<br>Technical solution<br>Product integrated<br>Verification<br>Validation<br>Organizational process focus<br>Organizational process definition<br>Organizational training<br>Integrated project management<br>Risk management<br>Decision analysis and resolution |
| ML 4: Quantitatively managed | Quantitative management | Organization process performance<br>Quantitative project management |
| ML 5: Optimizing | Continuous process improvement | Organization innovation and deployment<br>Causal analysis and resolution |

**Table 4.1:** CMMI staged representation levels and areas [9].

ics of people in the organization being the key enabler of its success, rather than the use of proven processes and practices [4]. Organizations at maturity level 1 (ML1) are usually over committing and would abandon processes in times of crisis [3]. At maturity level 2 (ML2) organizations achieve all the generic and specific goals of that process area. Meaning that organizations have ensured that requirements are managed and processes are planned, performed, measured and controlled [2]. Furthermore, commitments are established amongst relevant stakeholders and are revised as needed, while ensuring that requirements, standards and objectives of those stakeholders are met. At maturity level 3 (ML3) all generic and specific goals have been achieved by process areas defined in the previous levels as well as level 3. Processes are well understood and characterized and are described in standards and methods. A critical distinction between level 2 and 3 is the scope of standards, processes descriptions and procedures, which usually are tailored from the organization's set of standards to suit a particular project or organizational unit at level 3 [3]. Another differentiation is that processes at level 3 are typically described in a more detailed manner while also being more rigorously. At maturity level 4 (ML4) organizations have achieved all specific goals of previously defined process areas as well as the generic goals assigned to level 2 and 3 process areas. An establish-

ment of quantitative objectives for quality and process performance are established and used as criteria in managing processes. These objectives are are based on the needs of the customers, end users, and the organization. A distinction form other levels is that the performance of processes is controlled using statistical and other quantitative techniques [9]. Maturity level 5 (ML5) focuses on continually improving software development process performance through both incremental and innovative improvements.

There are perhaps many reasons of selecting the staged representation for an organization. The selection is usually dependent on the familiarity of the CMMI model by the organization [44]. Some of the positive advantages of the staged representation is that it provides a proven sequence of improvements. This sequence begins with basic management practices and progresses through a predefined and proven set of successive levels which serve as a foundation for the next level [10]. The staged representation allows organizations to permit comparisons amongst and across organizations by the use of maturity levels and the level to be achieved. Further it enables to create a single rating that summarizes appraisal results and allows organizations to compare one another.

### 4.2.2 Continuous Representation

For a continuous representaion model generic goals and generic practices apply to multiple process areas, which define a sequence of capability levels that represent improvements in the implementation and effectiveness of the improvement initiatives. Similarly to the staged representation, the continuous also focuses on best practices for improving process performance while also organizational processes need to be mapped to process areas of the CMMI model. The structure of the CMMI continuous representation is depicted in Figure 4.2.

**Specific Goals** are applicable to a process area and cope with the unique characteristics that describe what must be implemented to achieve the requirements of a process area. They are required model components and are used to determine whether a process area is satisfied. Each goal has at least one capability level 1 practice linked to it. **Specific Practices** are activities that are considered important into achieving the associated specific goals. Specific practices can be associated at different capability levels linked to the same goal and are expected components of the maturity model [3]. Each capability level has only one **Generic Goal** that describes the maturation that the organization must achieve at the capability level. Therefore having only a determined amount of generic goals equal to the number of capability levels excluding number zero (five in total). Achieving requirements of generic goals in a process area implies improved control in planning and imple-

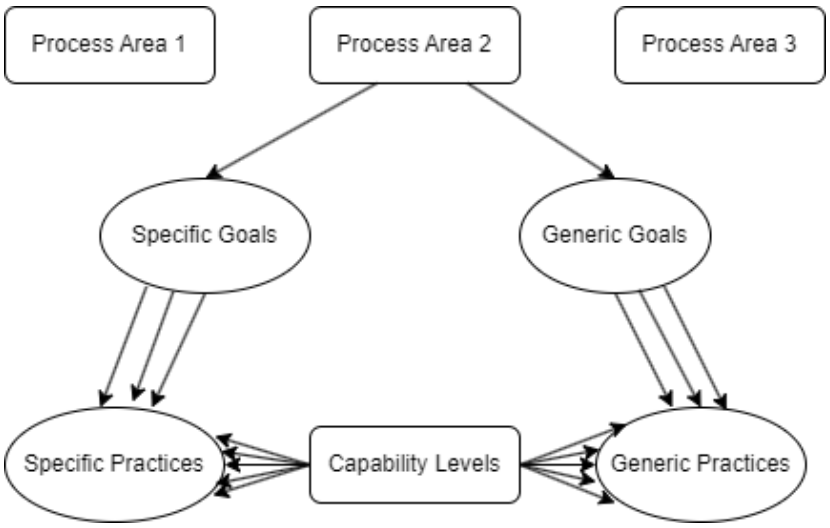



**Figure 4.2:** The structure of a CMMI continuous representation [3].

mentation of the processes associated with that process area. They are required maturity model components and signify the satisfaction of a process area. **Generic Practices** ensure that the processes associated with the process area are effective, lasting and repeatable [44]. They are expected components and are linked one to one to each generic goal.

The continuous representation uses six **Capability Levels** to measure the achievement of a specific process area for an organization [10]. These capability levels are: *Incomplete (CL0)*, *Performed (CL1)*, *Managed (CL2)*, *Defined(CL3)*, *Quantitatively Managed(CL4)*, and *Optimizing (CL5)*. A capability level is comprised of related generic and specific practices for a process area that aid in improving the organization's software development processes associated with that area. These capability levels are build upon one another, providing a recommended order for process improvements [45]. Hence, capability levels represent process improvement paths by illustrating an evolution of improvements for each of the process areas. The continuous representation CMMI consists of the same 22 process areas as the staged representation. However, no process area is categorized into a particular maturity level [44]. All process areas are organized into four categories, which are: *Project management*, *Process management*, *Engineering*, and *Support* as shown in Table 4.2.

At capability level 0 (CL0) the process areas are deemed incomplete, which is a process that is either not performed or partially performed. This is caused by one of the specific goals not being satisfied. At capability level 1 (CL1) processes are characterized as a being performed, which is a process that satisfies the associated specific goal of the process area. It enables and supports the work needed to produce recognized output work products using recognized input work products.

A difference from an incomplete process is that a performed process satisfies all of the specific goals at that particular process area. A capability level 2 (CL2) is characterized as a managed process. These processes are planned and executed in accordance to policies, employee skills and people having adequate resources to produce controlled outputs. These processes further involve relevant stakeholders, are monitored controlled and reviewed, and are evaluated according to their process descriptions [3]. A capability level 3 (CL3) is characterized as a defined process which is a managed capability level 2 process. These level 3 processes are tailored form the organization's set of standard processes according to its guidelines and contributed work products, measures and other process-improvement information [45]. At capability level 4 (CL4) processes are characterized as quantitatively managed, which are defined capability level 3 processes that are controlled using statistical and other quantitative techniques [46]. Their quality and process performance are justified in statistical terms and are managed throughout the life of these processes. At capability level 5 (CL5) processes are characterized as optimized, and are quantitatively managed processes at capability level 4. These processes are changed and adapted to meet relevant current and project business objectives, while also focusing on continually improving the processes performance through both incremental and innovative technological improvements. These address root causes of process variation and measurably improve the organization's processes that are identified, evaluated and deployed as appropriate [44].

An important advantage of the continuous over the staged representation is that it gives software development organization the flexibility to select process areas they want to improve. This in turn enables the organization to select the order that best suits their business objectives [10]. In turn minimizing and mitigating the organization's areas of risk [3]. Furthermore, it enables comparisons amongst and across organizations from a process area to process area basis, or by comparing results throughout the use of equivalent staging. Continuous representation provides an easy comparison of improvement initiatives to the International Organization for Standardization and Electrotechnical Commission (ISO/IEC) 15504, since the categories of process areas are similar to ISO/IEC 15504 [44].

## 4.3 Process Areas

CMMI and similar process capability models have been popularized and long studied, with many papers giving emphasis on the benefits and cost savings [10]. Organizations constantly engage into process improvement which empowers project management all over the projects' lifetime. Decisions are taken based on measurements and analysis of the current situation throughout several process areas that the

| Category | Process area | Maturity level |
|---|---|---|
| Project management | Project planning | ML 2 |
| | Project monitoring and control | ML 2 |
| | Supplier agreement management | ML 2 |
| | Integrated project management | ML 3 |
| | Risk management | ML 3 |
| | Quantitative project management | ML 4 |
| Process management | Organizational process focus | ML 3 |
| | Organizational process definition | ML 3 |
| | Organizational training | ML 3 |
| | Organizational process performance | ML 4 |
| | Organizational performance management | ML 5 |
| Engineering | Requirements management | ML 2 |
| | Requirements development | ML 3 |
| | Technical solution | ML 3 |
| | Product integration | ML 3 |
| | Verification | ML 3 |
| | Validation | ML 3 |
| Support | Configuration management | ML 2 |
| | Process and product quality assurance | ML 2 |
| | Measurement and analysis | ML 2 |
| | Decision analysis and resolution | ML 3 |
| | Causal analysis and resolution | ML 5 |

**Table 4.2:** CMMI continuous representation levels and areas [10].

CMMI focuses on. The CMMI has identified and categorized 22 process areas (PAs) that have to be managed in order to succeed into improving software development practices [9]. These 22 areas are discussed further in this section.

Ayyagari and Atoum [4] differentiated between a set of CMMI interacting areas, which include *Process Management*, *Support*, *Engineering*, and *Project Management*. Figure 4.3 depicts these process areas which finally entail on the overall increase of performance across the organization. A process area is a cluster of related practices in a group that, when performed together would satisfy a set of goals which are important for making significant improvements in that area [3].

### 4.3.1 Process Area Interactions

This section describes the interactions between process areas that aid on understanding the organizational view of process improvement. The categorization of

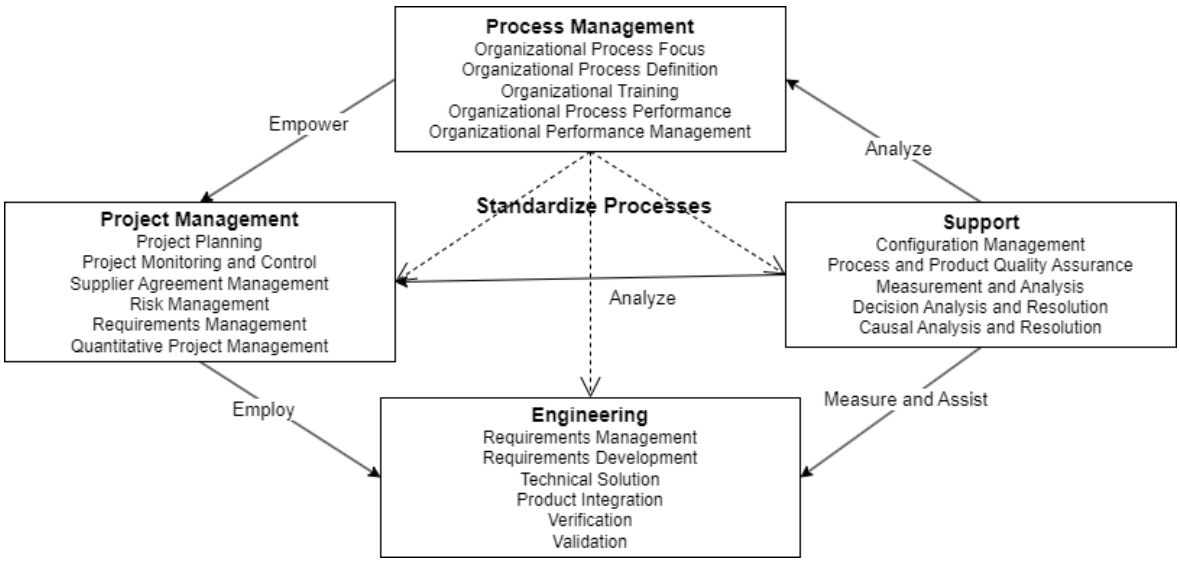



**Figure 4.3:** CMMI process areas by Ayyagari and Atoum [4].

process areas mentioned by Ayyagari and Atoum [4], is showcased in this way to enable clear discussions. However, process areas often interact and have an effect to each other regardless of their categorization.

#### 4.3.1.1 Process Management

The Process Management process area contains activities related to definitions, plans, resources, deployments, implementations, monitoring, controls, appraisals, measurements, and process improvements. As depicted in Figure 4.3, process area that fall under Process Management are: Organizational Process Focus, Organizational Process Definition, Organizational Training, Organizational Process Performance, and Organizational Performance Management. To understand their interactions amongst process areas, it is useful to categorize them into two groups, the **Basic Process Management Areas**, and the **Advanced Process Management Areas** [47].

The **Basic Process Management Areas** provide the organization with a capability document and share best practices, learning across the organization and organizational process assets [3]. The Organizational Process Focus area helps the organizations to plan and implement software development process improvements based on the understanding of the strengths and weaknesses of the organization's processes. Improvement initiatives are obtained thought a set of process improvement proposals, measurement of processes, lessons learned in implementation of processes, and results obtained for process and product evaluation activities [47]. The Organization Process Definition area establishes and maintains the organization's standard processes based on the process requirements and objectives of the

organization [2]. Assets are also established and maintained which includes descriptions of processes, process elements, and life-cycle models, as well as processing tailoring guidelines, process related documentation and data [10]. Performing these defined processes is enabled via experiencing and work products including measurement data, process descriptions, process artifacts, and lessons learned, which are in turn incorporated into the organization's set of standards [44]. While lastly the Organizational Training area pinpoints the strategic training needs of the organization as well as training in part of accommodating across projects and support groups. This is done to acquire the skill required to perform the organization's set of standards.

The **Advanced Process Management Areas** provide the organization with a capability to achieve its quantitative goals for quality and process performance [47]. These areas are strongly dependent on the ability to develop and deploy processes and supporting assets. Quantitative objectives for quality and process performance are derived by the Organizational Process Performance area [10]. Organizations provide teams with projects and support groups that have common metrics, process performance baselines, and process performance models. These support assets provide quantitative project management and decision making of critical processes. The organization analyzes the data collected from these tasks and develops a quantitative understanding of software quality, and software development performance. The Organizational Innovation and Deployment process area identifies and deploys proposed improvements that improve the organizational ability to meet its quality and software development performance objectives [3]. Improvements to deploy are selected according to quantitative understanding of the potential benefits and costs from deploying improvement initiatives.

#### 4.3.1.2 Project Management

The Project Management category covers the project management activities which relate to planning, monitoring, and controlling. As depicted in Figure 4.3, process area that fall under Project Management are: Project Planning, Project Monitoring and Control, Supplier Agreement Management, Risk Management, Requirements Management, Quantitative Project Management. Similarly to Process Management to get a clear understanding of the Project Management areas two categories are identified: **Basic Project Management areas** and **Advanced Project Management areas**.

The **Basic Project Management areas** address basic activities related to establishing and maintaining a project plan, commitments, monitoring progress against the plan, making correct decisions, and managing supplier agreements. Project

Planning area includes processes to develop the project plan, involve the stakeholders appropriately, obtain the commitment to the plan and maintain it. Stakeholders not only represent the technical expertise for software and process development, but also the business implications of the product and process development [10]. Planning includes requirements that define the software and project and covers the various project management and engineering activities that will be performed [48]. The Project Monitoring and Control process area includes monitoring activities and making correct decisions. The plan identifies the level of project monitoring, the number of times progress is reviewed, and the metrics used to monitor progress [3]. The Supplier Agreement Management process area identifies the need for the project to acquire work that is produced by suppliers. Management of the supplier that is going to produce the identified software component is being made possible by the supplier agreement that is established and maintained and will be used to manage the work. Both progress and performance of the supplier is monitored and acceptance reviews and tests are being conducted on the work done [46].

The **Advanced Project Management areas** include activities such as establishing a process that is tailored from the organization's set of standards, as well as coordinating and collaborating with relevant stakeholders. The Quantitative Project Management process area set quantitative and statistical techniques to govern process performance and software quality. The objectives of the quality and processes performance are identified by the organization [48].

#### 4.3.1.3 Engineering

The Engineering process areas cover development and maintenance of activities that are included across engineering disciplines. As depicted in Figure 4.3, process area that fall under Engineering are: Requirements Development, Technical Solution, Product Integration, Verification, and Validation. This areas integrate software engineering and system engineering practices into a product oriented process for improvement initiatives. In order to enable this focus is given on essential business objectives, rather than specific disciplines [49].

The Requirements Development area encompasses the identification of requirements from customers. The software requirements are analyzed to produce a high level conceptual solution [2]. Those requirements are also supplied in the Product Integration process area. They are then combined and interfaces are ensured to meet the requirements supplied by the Requirements Development process area. Requirements Management process area describes activities for obtaining and controlling requirement changes and ensures that data and relevant plans are kept up to date [44]. Moreover, it ensures that changes in requirements are reflected in

project plans, activities and work products. It is a dynamic and often recursive sequence of events and usually its processes impact other Engineering process areas. The Technical Solution area is responsible to develop technical data packages for product components that will be used by the Product Integration area [2]. Alternative solutions are examined with the purpose of selecting the optimal design based upon the established requirements and criteria. The Decision Analysis and Resolution process area under the Support category is responsible for selecting the final solution. The Technical Solution process area is reliant upon specific practices in the Verification process area in order to perform design verification and peer reviews during design [3]. This process area ensures that selected working products are meeting the specified requirements, by selecting work products and verification methods that are used to verify these products against the requirements. Verification further addresses peer reviews which are a proven way for removing defects early on and provide valuable insights. The Validation process area on the other hand validates software against the customer's requirements. It is performed either in an operational environment or a simulated environment [49]. The Product Integration process enables the establishment of specific practices which are associated with generating the best possible integration sequence, integrating product components and delivering the software to the customer [3].

#### 4.3.1.4 Support

The Support process areas cover activities that support the software development processes and maintenance of them, while also address the processes that are used in the context of performing other activities for different process areas. The addressed processes can cover processes that are applied more generally towards the organization [44]. Such process areas include Process and Product Quality Assurance that can be used with all the process areas to provide an objective evaluation of the processes and work products. As depicted in Figure 4.3, process area that fall under Support are: Configuration Management, Process and Product Quality Assurance, Measurement Analysis, Decision Analysis and Resolution, and Causal Analysis and Resolution. To understand their interactions amongst process areas, it is useful to categorize them into two groups, the **Basic Support Areas**, and the **Advanced Support Areas** [50].

The **Basic Support Areas** address the basic supporting functionalities that are used by all process areas. The Measurement and Analysis area aids all process areas by providing specific practices that direct projects and organizations into aligning measurement needs and goals with a measurement approach that will provide these objective results [2]. These results aid on taking the appropriate decisions

and actions. The Process and Product Quality Assurance area aids all process areas by providing practices for evaluating performed processes, work products and services. It ensures that any concerns identified during these evaluations are appropriately resolved. This process area facilitates the delivery of superior-quality products and services by granting the project team and management at all levels suitable insight and input regarding the processes and related work products throughout the project's duration [50]. The Configuration Management process area reinforces all other process areas by establishing and upholding the integrity of work products through configuration identification, configuration status accounting, configuration control, and configuration audits. Configuration Management process area encompasses a wide range of work products, including deliverables for customers, internal work products, acquired products, tools, and various items used in the creation and description of these work products. Examples of work products that can be subject to configuration management include plans, process descriptions, requirements, design data, drawings, product specifications, code, compilers, product data files, and technical publications for products [44].

The **Advanced Support areas** offer the projects and organization an enhanced support capability, where each of these process areas depends on specific inputs or practices derived from other process areas. By utilizing the Causal Analysis and Resolution area, the project endeavors to comprehend the underlying causes of variation that naturally occur within processes and eliminate them from the project's operations. Moreover, this knowledge is applied to continuously enhance the organization's processes. The improvement activities encompass both the defined processes specific to the project and the organization's established set of standard processes [3]. The Decision Analysis and Resolution process area reinforces all other process areas by implementing a formal evaluation process. This process guarantees that alternatives are thoroughly assessed and the most optimal one is chosen to effectively achieve the goals set by the process areas [50].

### 4.3.2 Modifying the CMMI Model

This section explores the significance of making modifications to the CMMI model to better align with an organization's specific context, goals, and industry requirements. Modifying any CMMI model can be done for the purpose of utilizing a subset of the model in order to fit with the needs of specific domains within the organization. This includes modifying evaluation methods that involve a selection of options to use in evaluating performance. This is done to aid the organization in aligning the model with its business needs and objectives, by focusing on aspects of software and services that are beneficial to the organization [46].

As described by the Software Engineering Institute [44], there are two distinct perspectives when it comes to modifying the model to the organization for: **internal process improvement**, and **benchmarking purposes**. An internal process improvement addresses disciplines, maturity levels, process areas and capability areas. Modifying the model in the context of the organization rises attention on identifying the process areas and practices that support the organization throughout it's business needs and objectives. As an example a maturity model modification has been proposed by Wangenheim et. al. [51] in order to meet certain aspects and requirements of the health domain. The model focuses on the essential elements of an effective process, and it is generally recommended to address most of the process areas and practices in the model. Therefore, it is important to exercise caution when it comes to deciding of whether to exclude parts of a CMMI model. Excluding fundamental processes or specific practices entirely is discouraged because evidence suggests that following CMMI based improvement efforts significantly enhances the achievement of business objectives, such as meeting cost and schedule targets [4]. It is acknowledged that organizations may not be able to adopt every aspect of the model due to various factors. However, organizations and projects can still derive significant value from the model even if they implement a partial set of process areas, goals, or practices.

The utilization of CMMI models for benchmarking purposes enables the comparison of process performance evaluation across an organization, either through state-of-the-practice reports or amongst a group of organizations such as potential suppliers. However, when applying modifications in this context, it is crucial to ensure consistency in the ratings obtained from the use of models in multiple appraisals. Consequently, the ability to modify models for benchmarking is significantly restricted, especially when maturity levels resulting from appraisals are publicly disseminated for marketing purposes. It should be noted that the chosen scope of a performance evaluation also has an impact on the context of benchmarking. If one organization decides to evaluate solely software engineering, while another chooses to appraise both software and systems engineering, a fair or accurate comparison between the two would not be feasible [44].

## 4.4 Software Development Dimensions

Identifying the dimensions that construct initiatives of software development is of high importance aiding both in identifying areas for improvement and the configuration of the assessment method.

### 4.4.1 Team Practices

Organizations use a set of teams to perform the processes mentioned above and it is important to note that many practices within a team context can contribute to performance. The abundance of teams within an organization focuses on the importance of team performance measurement. Effective teamwork creates knowledge, minimizes errors, enhances productivity, promotes innovation, increases job satisfaction, and ensures success [18]. Deployed teams that have the adequate training and led correctly can be very powerful. However, ensuring that teams perform, learn, and develop is not an easy task. In order to enable these goals, performance measurement tools can be really useful. These tools allow measurements of collective and individual performance. A paper by Fabrice et al. [11] reviewed several articles through a combination of teams and four processes within a conceptual categorization of attitude, behavior, and cognitive team constructs. The reviewed articles identified several attitudes, behaviors and cognition that can be measured included in Table 4.3.

| **Categories** | **Constructs** |
|---|---|
| *Attitudes* | Openness |
| | Trust |
| | Cohesion |
| | Team Viability |
| *Behaviors* | Collaboration |
| | Communication |
| | Leadership |
| *Cognition* | Transactive Memory System |
| | Shared Mental Model |
| | Information Sharing |
| | Knowledge Exchange |

**Table 4.3:** Team Practices Constructs identified in reviewing several articles by Fabrice et al. [11].

### 4.4.2 Requirements Management

Requirements management in the context of software development is the process of identifying what is required of the software when it is developed. It is usually executed before the development of that software. Requirement is a condition or capability needed by the users to solve a problem or achieve an objective [52]. Re-

quirements must be met with the system or components of a system that satisfy a contract or a standard, specification set by users or the organization [53]. Software developed must conform to the set of requirements defined by a set of requirements which in turn define the success and failure of that software. It must further specify what requirements are, write the requirements in an organized way to elicit them, and document them. Moreover, requirements are changing throughout the development of software hence requirement management is a systematic way to elicit those requirements and document them while also establishing and maintaining agreements between customers and stakeholders [53]. Requirement engineering activities include requirements elicitation, requirements analysis, requirements specifications, requirements verification, and requirements management. Where requirements management belongs to planning and controlling of these activities [52].

### 4.4.3 Development Practices

Software development practices play a crucial role in the quality of the final product [12]. Software development practices is the process of dividing software development into smaller, parallel, or sequential steps or sub-processes to improve design and product management [12]. Development practices refer to the methodologies, frameworks, techniques, and standards deployed during the software development life-cycle. These practices encompass an array of activities and approaches that software development teams use to design, code, test, and deliver software solutions [54]. Therefore, following best software development practices is a way to improve performance and productivity of an organization. Keeping best practices in mind also enhances efficiency, enables better decision making and provides employees with an internal knowledge base [54].

#### 4.4.3.1 Agile Methods

Agile methods have been widely used in software development practices over the last decades [55]. Agile development is used as a term that advocates working software over comprehensive documentation, responding to change over following static plans, and customer collaboration over contract negotiation. Teams are opted to work closely together while having more frequent communication, being aware of other team member's work efforts and being able to split workload equally among peers. Agility requires that best architectures, requirements, and design emerge from self-organizing teams [12]. Collaboration and communication are key in agile literature [56]. The agile manifesto is supported behind the following processes:

- Teamwork quality: refers to the performance of interactions, task strategy, task

process, and task activities.

- Communication: describes the quality of communication throughout team members [57].
- Coordination: management of dependencies between activities, such as shared resources, task assignments, and task relationships [55].
- Balance of contribution: contribution of task relevant knowledge of all members to the decision-making process [55].
- Competitive attitude: self interest in the expense of the team's performance [19].
- Effort: prioritization of team's tasks [55].
- Cohesion: the ability of the team to stay together and remain united for the completion of its goals [19].

### 4.4.4 DevOps Practices

DevOps are a set of practices, tools and cultural philosophy which automates and integrates the processes between software development and IT teams. DevOps stands for development and operations, and is a practice that aims at merging development, quality assurance and operations deployment and integration into a single and continuous set of processes and procedures [58]. This methodology is an extension of Agile and Continuous delivery approaches from the software engineering paradigm [59]. The advantage of this approach is that it does not require fundamental technical changes and is being rather oriented to changing a way a team works. The main principles of DevOps are automation, continuous delivery and fast reaction to feedback. DevOps require a delivery cycle which comprises of planning, development, testing, deployment, release, and monitoring with an active cooperation between the different stakeholders of a team [59]. Continuous delivery, is a technique which merges development, testing and deployment into a streamline process [60].

### 4.4.5 Quality Assurance Management

Quality Assurance refers to a systematic process which ensures the excellence of software and services [17]. Nowadays, almost every organization uses software in running its business operations. This has increased the need to have software dependent products and services that are reliable, useful, and secure every time during their operations [61]. It is reported that attaining the required software quality within

the dynamic and short duration of deployment is a challenge to many organizations. Organizations that fail to overcome these challenges translate to organizations that accept high risks of product failures which leads to considerable impacts to not only the organization's image, but also to its business operations. Furthermore, it has consequences to business operations, or the organization has accepted a lower speed implementation of current technologies, which shows that the organization risk to lose in the current competitive industry [61]. It is therefore imperative that software and services have the appropriate quality checks before deployment.

## 4.5 Data Presentation

Data presentation is a very important step towards analyzing the data and identifying points for improvement for software development processes. Visualizing of an organization's performance is the basis for monitoring, controlling, and improving of the operations within. Hence it is the foundation of our collective scientific knowledge therefore, figures are important because they allow showcasing the data that supports key findings [62]. In the following sections visualization tools and state of the art data presentation techniques are presented.

### 4.5.1 Visualization Tools

Various tools have emerged during the years to aid presentation and analysis of information. One of the most important features a visual must have, is that it should be interactive, it must be able to display relevant information when hovered over it, zoom in and out panel should be there. Some of these tools are discussed in a paper by Ali et al. [63], these include:

- *Tableu*: is an interactive visualization tool which is focused on Business Intelligence, it provides custom techniques, it is fast, flexible and supports most data formats and connections to servers.
- *Microsoft Power BI*: is a powerful cloud-based business analytics service, interactive, rich, flexible and persuasive with the consisting elements: Power BI Desktop, Software as a Service (SaaS) and Apps.
- *Plotly*: is build using Python and Django framework, free to use but with limited features, can be used in Ipython, jupyter notebook and panda library.
- *Gephi*: is an open-source network analysis tool, build to handle large, and complex data sets.

- *Excel 2016*: it is a spreadsheet developed by Microsoft, can be used for Big Data, statistical analysis, and visualization purposes, and can be connected to many services such as SaaS and HDFS a distributed file system that handles big data sets.

### 4.5.2 Key Considerations for Effective Visualization

Information collection just for the sake of displaying information is very time consuming and expensive [64]. Data can sometimes have a barrier on displaying useful information. Meaningful information must be able to show a pattern over time and the user must be able to understand the displayed information [65]. Furthermore, the way measurements have been derived must be completely understandable by the viewers, while also providing enough insights for viewers to identify important items that need attention. Firstly, attention should be given to the data that can aid in decision making when designing dashboards avoiding any redundant information that can complicate processes and remove user friendliness. The dashboard should display information in a single page allowing viewers to gain the big picture [64].

Visualizing survey results is a critical step in harnessing the power of data and transforming it into actionable insights within the context of the CMMI model. Exploring the key considerations that can greatly enhance the effectiveness of visualizing survey results for the CMMI model is of great importance. Effective visualization goes beyond simply presenting data in visually appealing formats, which involves understanding the nuances of the data, identifying meaningful patterns, and uncovering valuable insights [66]. By adopting a strategic and purposeful approach to visualization, organizations can make informed decisions, drive improvements, and align their efforts with the model's objectives.

According to Aigner et al. [67], there are three major criteria that have to be met in order to utilize the outstanding capabilities of the human visual perception and the computational power of computers to analyse, understand, and communicate results. These criteria are expressiveness referring to the requirement of showing nothing less or more of what the data contains, effectiveness which is that it considers the degree of cognitive capabilities of the human visual system. Finally appropriateness which considers the cost-value ratio to assess the benefits of the visualization process in respect to achieving a given task [67]. Given the meaningful insights that survey results can offer, it is essential to consider various factors when creating visualizations from the data, including: clarifying the goals of the visualization, and determining the most appropriate visual representation techniques as explored.

Grappling with the processing and interpretation of the collected data, particu-

larly when dealing with a substantial number of responses, presents a formidable challenge. Visualizing survey questionnaire responses addresses this challenge by presenting data in a concise and easily comprehensible format. This approach allows organizations to explore trends, identify areas of strength and improvement, and facilitate effective decision-making [65]. By converting raw data into visual representations such as charts, graphs, and infographics, stakeholders can swiftly discern the software development processes of the organizations [68].

#### 4.5.2.1 Dashboards

Dashboards are often used for the purpose of visualizing data as they are a powerful tool to interpret information in a single view providing a graphical representation of the current situation [5]. A dashboard is defined by Staron [5] as an easy way to read real time user interfaces, depicting graphical representations of the current status and any historical trends of an organization's Key Performance Indicators (KPIs) to enable decision making. The development process of a dashboard is contacted iteratively in close collaboration with the users the dashboards or the people representing the users. Figure 4.4 included showcases the stages of the dashboard development. Dashboards provide mechanisms to visualize information in an operationally oriented measurement system, that measures performance against targets and thresholds using right time data [64].


**Figure 4.4:** Dashboard Development Process [5].

The process includes:

- Requirements elicitation: to collect stakeholder expectations and create the first mock-ups of the dashboard.
- Dashboard type selection: find the technology to obtain the dashboard.
- Impact evaluation: to observe what is the impact of the dashboard for the organization.
- Dashboard maintenance: to monitor the correct operation of the dashboard and that it shows the information required.

The process of selecting the right type of dashboard follows the selection model described in a paper by Staron [69], it includes the following:

- Visualization type: defining the type of charts and graphs to be used.
- Data acquisition: determine how the data is input in the visualization tool by making an assessment.
- Stakeholders: defining who are the stakeholders.
- Delivery: define how to provide the data to the users.
- Update: define how often the data will be updated.
- Aim: defining what the purpose of the dashboard is.
- Data flow: determining how much processing of the data will be executed within the dashboard.

## 4.6 Performance Measures

Measuring the performance of software development processes has become a central issue in both the scientific as well as the business world, since organizations are opting to achieve effective and efficient results. The recognition of performance measurement models for this purpose ensures both alignment with a business strategy and processes hence determining that the choice of performance indicators is organizational dependent [70]. The Balanced Scorecard (BSC) among other performance measurement models aid to translate an organization's strategy into operational indicators or otherwise metrics and objectives.

Metrics are used to measure performance, although most organizations seem to have a poor understanding of what constitutes a good metric [65]. Not all metrics have the same time frames, which in turn impacts the frequency at which they are measured. Most of the times performance does not improve immediately and can fluctuate throughout a project's life cycle [68]. Some can be measured in real time while others once per week or even a month. Harold [65] has categorized metrics according to the following time frames for measurement purposes:

- Full project duration metrics: exist throughout the duration of a project while collecting data in a weekly or monthly basis.
- Life cycle phase metrics: exist through a particular phase within a project.
- Limited life metrics: exist during an element of work or work package.
- Moving window metrics: the starting and finishing dates can shift as the project progresses.

- Alert metrics: exist until a condition is met.

David [71], categorized different performance measurements into four categories:

- (i) Key Result Indicators (KRIs): Giving an overall summary to the board as to how the organization is performing.
- (ii) Result Indicators (RIs): Which tells how teams are combined to produce results.
- (iii) Performance Indicators (PIs): Which tells management what teams are delivering.
- (iv) KPIs: Which tells how the organization is performing in their Critical Success Factors (CSFs), and by monitoring them it enables to increase performance.

However, many organizations that have operated with key performance indicators have found that little or no difference has been made. In most cases this was caused by a fundamental misunderstanding of the lack of preparations indicated in the seven foundational stones proposed by David [71]. The foundational steps are used as a preparation in implementing the six stage process that an organization can follow in a sixteen week process and it includes: (i) Getting the CEO and senior management committed to change, (ii) Up-skill in house resource to manage the KPI project, (iii) Leading and selling the change, (iv) Finding your organization's operational CSFs, (v) Determining measures that will work in your organization, (vi) Get the measures to drive performance [71]. In this section stages (iv), (v) and (vi) are of interest. The seven foundational stones as a preparation for implementing performance indicators include: (i) Partnership with staff, unions, and third parties, (ii) Transfer of power to the front line, (iii) Measure and report only what matters, (iv) Source KPIs from CSFs, (v) Abandon processes that do not deliver, (vi) Appointment of a home-grown chief measurement officer, (vii) Organization wide understanding of the winning KPIs definition [70].

### 4.6.1 Determining measures that will work in an organization

Agile methods like Scrum, Lean Software Development, and Kanban have become popular in the software industry [72]. These methods support lightweight practices, continuous deliveries, and customer collaboration instead of lengthy planning, extensive documentation, and rigid development phases. Table 4.4, illustrates a variety of common software development metrics identified throughout several papers used in these methods.

| Metric | Description |
|---|---|
| Velocity [72] [73] | Indicates the amount of work a software development team can complete in a given timeframe. |
| Effort estimate [74] [72] | Forecasts how much effort is required to develop or maintain a software application. |
| Defect count [75] | Shows the number of bugs found . |
| Technical debt [76] | The number of features and functionality that is postponed, taking shortcuts, or accept less-than-optimal performance to advance the project. |
| Lead time [77] [72] [78] | Is the time required from creating a work item until it is completed. |
| Cycle time [65] [72] | Is the time required from staring to work on an item until it is completed. |
| Test coverage [72] | The number of tests compared to the existing amount of code. |
| Number of unit tests [72] | The number of tests performed on code. |
| Incident response time [72] | The time required for a bug to be fixed once it is acknowledged. |

**Table 4.4:** Identified metrics throughout literature.

### 4.6.2 Get the measures to drive performance

Getting measures to drive performance usually requires a reporting framework to be developed at all levels within the organization [71]. The reporting framework must fit in with the requirements of different levels in the organization and the real time data that supports decision making. As suggested by David the key tasks for developing such a framework includes: providing teams with the appropriate training on reporting, establishing meaningful visuals that are easy to understand, developing a hierarchy of reports to management, the stuff and the board of directors. However, performance measurements are directly involved with the unresolved problems of defining measures, their quality, and the culture of both cataloging and reporting [79].

## 4.7 Conclusion

In conclusion, the CMMI model has two main representations, the staged and the continuous representation. The staged representation has five maturity levels, guiding organizations from chaotic to mature processes. However, the continuous representation has six capability levels, offering flexibility in selecting process areas for improvement. In the staged representation, maturity levels range from Initially (chaotic processes) to Optimized. On the other hand, continuous representation's capability levels span from Incomplete to Optimizing. Both representations focus on process improvement, however the staged representation offers a structured approach, while the continuous representation provides flexibility. CMMI models cover 22 process areas, divided into Process Management, Support, Engineering, and Project Management categories. These process areas interact between them to contribute to the organization's performance. Modifications on the model serve internal improvement or benchmarking, with a caution against excluding crucial parts.

Moreover, software development dimensions play a pivotal role in enhancing

software development processes. Team practices are crucial for knowledge sharing, effective collaboration, and innovation within an organization. Requirements management ensures that software aligns with user objectives and needs. Development practices, including Agile and DevOps methods, contribute to efficient design, coding, testing, and delivery of software solutions. Quality assurance management guarantees reliable and excellent software performance. These dimensions collectively foster improved teamwork, optimized software development life cycles, and enhanced software quality, addressing the evolving demands of the industry and ensuring successful outcomes.

Finally, effective data presentation techniques are essential for converting information into insights through visualization. Dashboards offer real-time snapshots, requiring a simplistic design. Performance measurement in software development is crucial for optimization. Several software development metrics are identified which range from project duration to limited life metrics.

# Chapter 5

# Designing the Assessment Method

This chapter delves into the critical steps involved in designing the assessment method according to the CMMI model and software development dimensions. This chapter encompasses two essential components: the formulation of a design plan and the designed assessment method that suits the organization's five software development dimensions.

## 5.1 Design Plan

This section proposes the design plan for the assessment method. As discussed previously in Section 3.3, the CMMI model serves as a framework for process improvement. With the principles and guidelines outlined in the previous sections, we have established a structured approach to improve the organization's software development processes and practices. We therefore follow the plan outlined in Figure 5.1.

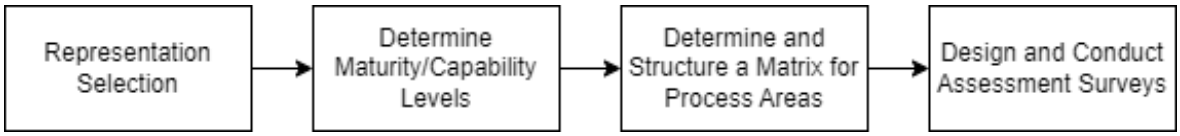



**Figure 5.1:** Configuration Plan

The initial phase is Representation Selection, where the suitable CMMI representation is chosen. This decision involves opting for either the staged representation, focusing on maturity levels, or the continuous representation, which emphasizes on capability levels. The choice is based on the organization's objectives, context, and desired level of process improvement. Subsequently, we determine the appropriate maturity or capability levels that align with their context. Afterward, the step titled "Determine and Structure a Matrix for Process Areas", we identify the relevant process areas within the CMMI model that align with the organization's business

objectives and priorities. This step entails mapping the identified process areas to the corresponding maturity or capability levels. This entails a framework which aids to design the assessment surveys discussed later on in Chapter 6

### 5.1.1 Continuous Improvement Plan

The organization considers a continuous improvement plan as a pivotal component, as it fosters ongoing enhancements of an organization's processes and practices. This plan acts as a well-structured roadmap, enabling the systematic identification of areas that need improvement, the implementation of changes, and the measurement of performance growth [80]. The organization has set its sights on utilizing continuous improvement by introducing assessment surveys to be completed by team leaders for various software development dimensions. These surveys will then be used to generate reports, propose improvement recommendations, and subsequently implement them. Therefore, embracing the path of continuous improvement, the organization progresses through iterative quartiles, each spanning four months, within a yearly cycle. The proposed continuous improvement plan for the organization is illustrated in Figure 5.2.


**Figure 5.2:** Continuous Improvement.

**Step 1:** Team leaders are provided with maturity assessment surveys, carefully crafted to encompass closed-type questions that span across the five vital di-

mensions of software development: Requirements Management, Team Practices, Development Practices, DevOps Practices, and Quality Assurance Management. The possible responses to these questions are either ”Yes,” indicating a positive response where the aspect being asked is true, or ”No,” indicating a negative response where the aspect being covered is false. These thoughtfully categorized questions within each dimension accurately reflect the maturity levels, ranging from one to five.

**Step 2:** Based on the responses received from the teams, a maturity level is calculated. These results are transferred from Microsoft surveys to SharePoint, and subsequently, captivating visualizations are generated using PowerBI. Figure 5.3, depicts the data flow.

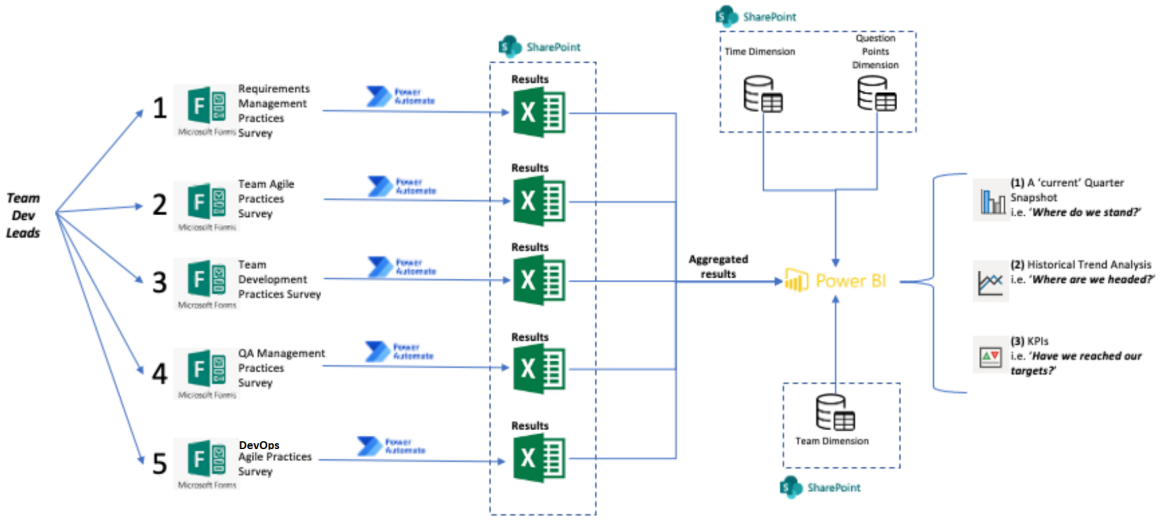



**Figure 5.3:** Data Flow.

**Step 3:** Thorough analysis of the results is conducted, leading to the creation of tailored recommendations for each team. These recommendations are designed to be incremental, considering the current impact on team productivity, as well as the short-term value and quality they bring. Moreover, potential risks that may arise during the implementation phase of these recommendations are carefully considered.

**Step 4:** The implementation phase encompasses the creation, prioritization, estimation, and clarification of all recommendations. Regular catch ups are diligently scheduled to monitor the team's progress, ensuring transparency and alignment. Once a significant amount of work has been accomplished, the next iteration is executed, paving the way for continuous improvement.

## 5.2 The Designed Assessment Method

Based on the organization's goal of implementing a continuous improvement plan, the most suitable CMMI representation would be the continuous. This representation aligns with the organization's objective of fostering a culture of ongoing enhancement and iterative progress. The upcoming sections present the design of the assessment method according to the CMMI continuous representation.

### 5.2.1 Capability Levels

Drawing upon the knowledge presented in Section 4.2.2 and considering the organization's context, the identified capability levels include: **Undefined**, **Initial**, **Developing**, **Defined**, **Managed**, and **Leading**. These capability levels provide a structured framework to assess and gauge the organization's maturity and progress in its continuous improvement journey. Table 5.1, includes the description and resulting range of each capability level.

| **Capability Level** | **Description** | **Result Range** |
|---|---|---|
| Undefined | No assessment was executed so far for this area. | Not Available |
| Initial | At this capability level, engineering practices are characterized as unpredictable, poorly controlled, and reactive. They lack stability, which can result in increased inefficiency and hinder the overall effectiveness of the team. | $<$40% |
| Developing | At this capability level, engineering practices are defined by development teams and often exhibit a reactive approach. | 40-60% |
| Defined | At this capability level, all engineering practices are thoroughly defined and well-understood. Development teams demonstrate a proactive approach rather than a reactive one. Furthermore, the organization has established comprehensive standards that offer guidance across all five dimensions, fostering a culture of consistency and continuous improvement. | $>$60-75% |
| Managed | At this capability level, all engineering practices are meticulously measured and controlled. The organization relies on quantitative data to implement changes in a predictable manner, ensuring they align with organizational standards. | $>$75-90% |
| Leading | At this capability level, all processes within the development organization are both stable and flexible. The organization maintains stability by establishing reliable and consistent processes, while also fostering flexibility to adapt and respond effectively to changes in the environment. | $>$90% |

**Table 5.1:** Capability levels, their description and their result range.

### 5.2.2 Correlation between Software Development Dimensions and CMMI Process Areas

This section describes the correlation between the software development dimensions and the process areas of the CMMI model. The purpose of this section is to ensure that we follow the process areas of the CMMI model while creating the assessment criteria.

Team Practices refer to the collective efforts and coordinated actions undertaken by software development teams. Effective teamwork, communication, and coordination within the team enable increased productivity, enhanced creativity, and improved problem-solving capabilities [18]. These practices are designed to aid teams in executing their tasks and responsibilities effectively and efficiently. While the specific team practices may vary depending on the process area, they generally align with the goals and objectives of the CMMI model. The process area of the CMMI model that involve Team Practices are:

- **Organizational Training:** Involves conducting training needs assessments, creating training plans, delivering training sessions, and evaluating the effectiveness of training programs [18].

- **Organizational Process Focus:** This area emphasizes the importance of process improvement, standardization, and measurement to enhance the overall performance and productivity of teams [81].

- **Organizational Process Definition:** This area involves identifying, documenting, and standardizing processes to ensure consistency and repeatability across projects and teams within the organization [82].

- **Configuration Management:** It involves ensuring that the correct versions of software artifacts are used, maintaining configuration baselines, and conducting configuration audits [81].

- **Project Planning:** This area is a critical aspect of team practices that involves defining project objectives, determining the scope, identifying tasks and resources, establishing timelines, and creating a roadmap for successful project execution [81].

- **Project Monitoring and Control:** This area is an essential component of team practices which involves tracking project progress, comparing it against the project plan, identifying deviations, and taking corrective actions to keep the project on track [83].

- **Risk Management:** It is an essential part of team practices as it aids identifying, assessing, and mitigating potential risks that could impact the team's objectives and success [83].

- **Requirements Management:** It involves identifying, mitigating, and monitoring risks throughout the project life cycle. Furthermore it aids teams to anticipate and proactively address potential issues that could impact project success [83].

- **Quantitative Project Management:** This area emphasizes the use of data, measurements, statistical and mathematical models to plan, monitor, and control projects [11].

Requirements Management aims at capturing, analyzing, and documenting the software requirements from stakeholders. An effective Requirements Management process ensures that software solutions align with customer needs and business objectives, reducing the risk of duplicating work, scope creep, and misunderstandings [16]. The linked process areas are:

- **Requirements Development:** Requirements Management within this area includes conducting stakeholder interviews, using collaborative techniques to capture requirements, and maintain traceability between requirements and project deliverables [18].

- **Requirements Management:** This area is direclty linked to this dimension as the name suggests, which is the set of best practices for process improvement in software development [83].

- **Project Planning:** Project planning does not solely focus on requirements management but rather incorporates the considerations of requirements within the project planning activities [16].

- **Verification and Validation:** These areas are closely linked to Requirements Management as they are essential activities within the broader context of ensuring the quality and correctness of a software or a system [84].

Development Practices integrate the methodologies, frameworks, and techniques employed during the software development life cycle. These practices include the selection of appropriate programming languages, architectural patterns, coding standards, code reviews, testing methodologies, and debugging techniques [85]. The related process areas are:

- **Requirements Development:** It involves selecting appropriate programming languages, architectural patterns, and design principles that align with the specified requirements [86].
- **Technical Solution:** In this area they might involve performing architectural design, implementing coding standards and best practices, conducting code reviews, and using tools for version control for code and configuration management [87].
- **Verification and Validation:** Development Practices play a crucial role in test planning, test case design, and test execution [86]. These areas include creating test plans, executing test cases, performing inspections and walk throughs, conducting peer reviews, and analyzing test results [88].
- **Configuration Management:** It establishes version control mechanisms, defining branching and merging strategies, and ensuring the integrity and consistency of the software configuration items [87].
- **Product Integration:** This area is a crucial aspect of development practices as it involves combining individual components or subsystems into a complete, functioning product, while ensuring that the different parts of the product work together seamlessly and as intended [89].
- **Decision Analysis and Resolution:** This area is a development practice that helps teams make informed decisions by systematically analyzing and evaluating various alternatives. It is particularly useful in complex projects where multiple options need to be considered before making a decision [85].
- **Causal Analysis and Resolution:** This area is a development practice that aims to identify the root causes of problems or issues and develop effective resolutions to prevent their re-occurrence, it involves analyzing the underlying causes rather than just addressing the symptoms of a problem [82].

DevOps Practices aim on streamlining the collaboration between software development and IT operations by fostering a culture of continuous integration, delivery, and deployment [90]. The linked process areas are:

- **Project Management:** It ensures collaboration between development and operations teams, which facilitates better project planning, resource allocation, and risk management, resulting in more effective project management [20].
- **Verification and Validation:** It enhances the verification and validation process by enabling the automation of testing and quality assurance activities [91].

- **Technical Solution:** It fosters a seamless integration of development and operational activities, enabling the development of technical solutions that are optimized for efficient deployment, scalability, and performance [87].
- **Configuration Management:** DevOps enhance the use of version control systems, configuration management tools, and infrastructure as code approaches [81].

Quality Assurance Management aims at the implementation of strategies, processes, and activities to guarantee that the developed software meets the required standards and customer expectations [17]. The related process areas are:

- **Organizational Process Focus:** It focuses on establishing and maintaining effective organizational processes and involves defining quality objectives, implementing quality assurance practices, and ensuring compliance with organizational standards and policies [61].
- **Organizational Process Definition:** It includes establishing quality criteria, guidelines, and templates to be followed throughout the development life cycle.
- **Project Planning:** It defines quality goals, identifies quality risks, and establishes quality assurance processes and techniques.
- **Technical Solution:** It implements processes to assess and improve the quality of the technical solution [87].
- **Verification and Validation:** It includes developing test plans, conducting functional and non-functional requirements, and ensuring that software products meet the specified quality criteria [86].
- **Measurement and Analysis:** Involves defining key performance indicators (KPIs), establishing data collection processes, conducting data analysis, and sharing insights with teams to drive improvements [70].
- **Process and Product Quality Assurance:** This area involves translating requirements into design, code, and test cases for quality. It ensures that it enables engineering practices to follow specific activities to requirements [53].

### 5.2.3 Dimensions Subcategories

This section aims to outline the subcategories of the dimensions as assessment criteria, enabling the design of survey questions. Subdivision aids on achieving greater clarity, organization, and granularity within the dimensions. By subdividing the dimensions into smaller, more specific subcategories, it becomes easier to capture

and analyze nuanced aspects of the subject being assessed. This further aids to design targeted and focused survey questions, ensuring both that the data collected is relevant and meaningful. Subsections of each dimension are included below:

**The Team Practices dimension includes:**

- **Roles and Team Composition:** Assesses the team's organizational structure to ensure that all roles are properly defined and present.
- **Artefacts & Definitions:** Verifies the presence and adherence to crucial definitions such as Definition of Ready and Definition of Done, as well as the establishment of key artifacts like agile dashboards, burn-down charts, and release scope.
- **Events:** Ensures that the team has established and regularly conducts essential events, including stakeholder meetings, retrospectives,sprint reviews, demos, and daily stand-ups.
- **Collaboration:** Evaluates the presence of comprehensive communication channels and the implementation of a structured collaboration approach between the team and other teams.
- **Continuous Learning:** Verifies the existence of individualized learning plans for team member and assesses their alignment with relevant technologies.

**The Requirements Management dimension includes:**

- **Backlog Management (including acceptance criteria):** Examines the structure and ongoing maintenance of the backlog, ensuring that it is well- organized, regularly updated, and refined. Additionally, verifies if user stories and epics have clear acceptance criteria that are measurable, and if stories are appropriately estimated.
- **Documentation Management:** Verifies the comprehensive documentation and continuous updating of all requirements to ensure clarity and accuracy throughout the project life cycle.

**The Development Practices dimension includes:**

- **Knowledge Sharing:** Assesses the presence of structured documentation, encompassing both software-related and architectural documentation, within the team's practices.
- **Code Quality Management:** Verifies the implementation of quality gates for code, which include adherence to guidelines, best practices, and the utilization of static analysis tools to evaluate aspects such as cyclomatic complexity, coupling, code duplication, and other relevant metrics.

- **Code Review Management:** Evaluates the team's adherence to mandatory code review practices during commit or pull requests,including the presence of a review strategy, checklist,and coding standards. Additionally, verifies if code reviews encompass the validation of business logic, ensuring that the code aligns with the intended functional requirements.

- **Unit Testing:** Verifies whether the team is actively writing unit tests and if these tests are considered as one of the quality gates for code.

- **Technical Debt Management:** Assesses whether the team is actively tracking and estimating technical debt, and if they are allocating development time specifically for addressing and managing technical debt within their schedule.

- **Secure Development:** Evaluates the team's adherence to security standards by examining their implementation of practices such as static and dynamic code analysis, tracking vulnerabilities, and the establishment of defined security coding practices.

- **Architecture Foundations:** Verifies whether the team is adhering to company architecture standards, patterns, and best practices, including the presence of architecture reviews. Additionally, assesses if the code is designed to be cloud-ready, considering cloud architecture principles and requirements.

**The Quality Assurance Management dimension includes:**

- **Test Organization:** Checks if Quality Assurance organization is structured.

- **Testing Strategy:** Checks if team has defined strategy.

- **Test Design:** Evaluates the team's implementation of defined test cases, including positive and negative scenarios, as well as the establishment of test suites. Additionally, checks if test cases are appropriately prioritized and cover both functional and non-functional requirements.

- **Defect Management:** Verifies if the team is effectively tracking, prioritizing, and estimating all defects encountered during development. Additionally, assesses whether there are specific Service Level Agreements (SLAs) in place for different defect types and priorities, ensuring timely resolution and adherence to defined response times.

- **Automation:** Assesses the presence of test automation within the team's practices and verifies if it is integrated into the Continuous Integration and Development (CI/CD) pipeline.

- **Reporting:** Verifies the existence of reporting mechanisms for testing activities and assesses if the reporting follows a unified template, ensuring consistency and standardized documentation of testing results and findings.

**The DevOps Practices dimension includes:**

- **CI/CD**: Verifies the implementation of a CI/CD pipeline within the team's workflow. Additionally, checks if there is a well-defined process in place to handle and address any disruptions or issues when the pipeline breaks or fails.
- **Release Management:** Assesses whether the team is adhering to the release standards established by the company.
- **Monitoring and Alerting:** Verifies if the team has implemented monitoring and alerting systems and assesses whether they align with the company's standards.
- **Incident Management:** Assesses whether the team has implemented an incident management process and has defined escalation policies.
- **Infrastructure Management:** Verifies the presence of a well-established environment infrastructure and the availability of infrastructure deployment scripts.
- **Access Management:** Assesses whether the team has a defined access management process in place. This involves evaluating if there are established procedures and protocols for granting, revoking, and managing access to systems, applications, and sensitive data.

# Chapter 6

# Assessment Survey Results

This chapter describes the assessment survey process, along with visualizing the assessment survey results and determining the capability levels of the organization across the five software development dimensions. Finally, we describe the discussions made with team leaders of the organization and agreements made for introducing improvements to processes that exhibit low performance.

## 6.1 The Assessment Survey Process

The design of assessment survey is made according to the design of the assessment survey in Chapter 5. Currently the organization uses internal as well as external tools such as Release and Task Tracking systems, Code Quality systems, Build and Pipeline systems, and Source Code Management systems. The Release and Task Tracking systems include software tools or platforms that aid organizations to track and coordinate tasks, bugs and other project-related activities [92]. The organization uses *Jira* as the main system for Release and Task Trucking activities. Sonar Qube used by the organization is a Code Quality system which includes static code analysis tools that analyse source code to identify potential issues, coding violations, and adherence to coding standards [93]. Build and pipeline systems such as Jenkins and GitLab CI/CD used by the organization are software tools or platforms that automate the process of building, testing and deploying software applications. Source Code Management systems such as GitLab used by the organization, is a version and control system or revision control system which aid development teams manage changes to source code files over time. The surveys are designed according to the concepts of these systems.

Each team out of 23 in total within the organization is led by a team leader who serves as a representative. We acquired responses from team leaders which are the representatives of the of their team. The surveys are designed according to ex-

perts and the knowledge acquired from the literature review. The survey questions can be found in Appendix B. Each question in the assessment surveys is assigned a capability level, and it is scored accordingly. The assessment surveys questions are evaluated using a binary system, where respondents can provide either a ”Yes” or ”No” answer. This binary system simplifies the evaluation process by categorizing the responses into two distinct options, allowing for straightforward analysis and interpretation of the data. A ”Yes” answer indicates the representative level associated with that question, while a ”No” answer receives a score of zero. After the assessment surveys are conducted, team leader's responses are recorded and stored in a PowerShare Excel file for each dimension. The subsequent chapter focuses on presenting the results of the collected data, following best practices for visualization and analysis.

## 6.2 Visualization of Survey Responses

Introducing the surveys to team leaders is a crucial step-by-step process that takes into consideration the inherent complexity of team structures across different brands within the organization. Recognizing the diverse composition of teams and their unique characteristics, a systematic approach is adopted to ensure the surveys are completed effectively and accurately. The goal of visualizing is to gain a comprehensive understanding of maturity across teams and dimensions of software development processes. The key visualizations to be considered therefore are:

- Survey progress: Monitoring the progress of surveys completed by teams within brands, this provides valuable insights into the level of survey participation and completion.
- Timeline of responses: A timeline of responses provides valuable info for tracking the progress and evolution of survey completion over time.
- Average score per dimension: Analyzing the average score per dimension provides a perspective on survey responses by dissecting and evaluating data at a dimensional level.
- Team score across dimensions: Examining team results across dimensions offers a comprehensive and detailed overview of the performance of each team.
- Team score across dimensions with brands: This approach allows for a comprehensive understanding of how teams perform across dimensions within each specific brand.

- Team-Brand comparison per dimension: This approach allows for a detailed assessment of how each brand performs in specific areas or aspects measured by the survey.
- Team deviation from average: This approach provides insights into the extent to which teams' performance differs from the average.
- Average score per category: Analyzing the average score per category provides a valuable perspective on survey responses by dissecting and evaluating data at a deeper level.
- Frequently answered questions as No: Highlighting common negative responses in the surveys.
- Frequently answered questions as Yes: Highlighting common positive responses in the surveys.

Timelines are used when studying the execution and evolution aspects of any process. Although visualization of survey progress and the timeline of responses may not directly pinpoint areas for improvement within software development teams, they serve a vital purpose in monitoring the survey process itself. These visuals provide insightful information on the overall progress and completion rate of surveys, ensuring that the survey is on track and that teams are actively participating and completing the survey [94].

By assessing the capabilities along the CMMI maturity level on various parts of the organization and take steps to improve areas of weaknesses and capitalize on their strengths [6]. Hierarchies, structures, and relationships are frequently used and play a significant role [66]. In data presentation complexity and hierarchical nature of the assessment surveys it is important to introduce visuals that can cope with hierarchically structured data. Amongst others, graph based data presentation techniques are in line with hierarchical structures, where color, shape and size usually play the role of presenting categorical data [66].

### 6.2.1 Assessment Survey Progress

This chart aids in underpinning the completion of the surveys that has been achieved and the representative population depicted in Figure 6.1.

By examining the table on the right-hand side of the figure, we can gain insights into the survey completion status of 23 teams. Each team is categorized under two distinct brands, namely the United States (Trading.com) and European-based (XM) teams. By leveraging the dynamic features of Power BI, it becomes possible to select a specific team from the table and visualize its survey completion progress

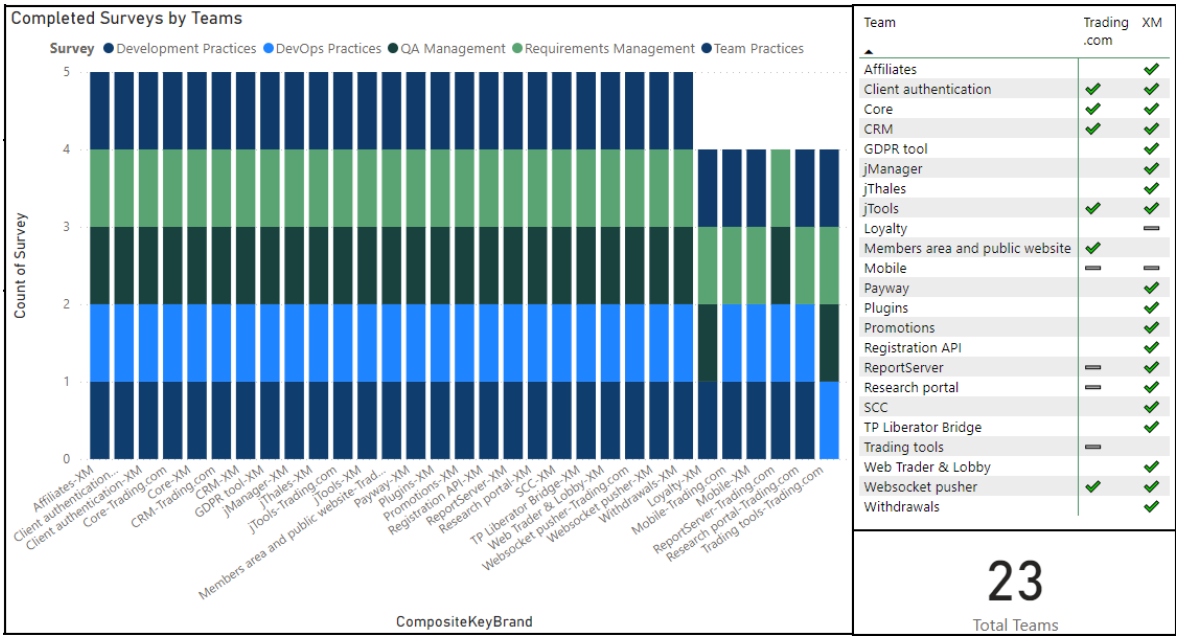



**Figure 6.1:** Survey Progress.

on the bar chart displayed on the left side. The bar chart not only showcases the completion status of surveys but also provides a clear distinction between the two brands. Teams with two green check marks across the table indicate that they have successfully completed all five surveys for both brands. On the other hand, the grey dash showcases that teams have completed a specific number of surveys based on their relevance and applicability to the respective area.

### 6.2.2 Timeline of Responses

The timeline of responses provides an insightful view of the cumulative number of surveys completed by the teams over a designated period. This timeline is represented as a percentage of the total surveys that need to be completed. Conducting the survey followed a circular approach, where teams were engaged in the survey process throughout the span of a month, as depicted in Figure 6.2.

### 6.2.3 Average Score per Dimension

The bar chart and radar chart serve a crucial role in determining the overall performance of the organization by providing the average score for each dimension across all teams. As showcased by Figure 6.3 and Figure 6.4, mostly all dimension scores are above 50%, and the average of the organization is 54.15%.

The organization currently operates at the ”Developing” capability level, as evidenced by an average score of 54.15% across the five dimensions. This score

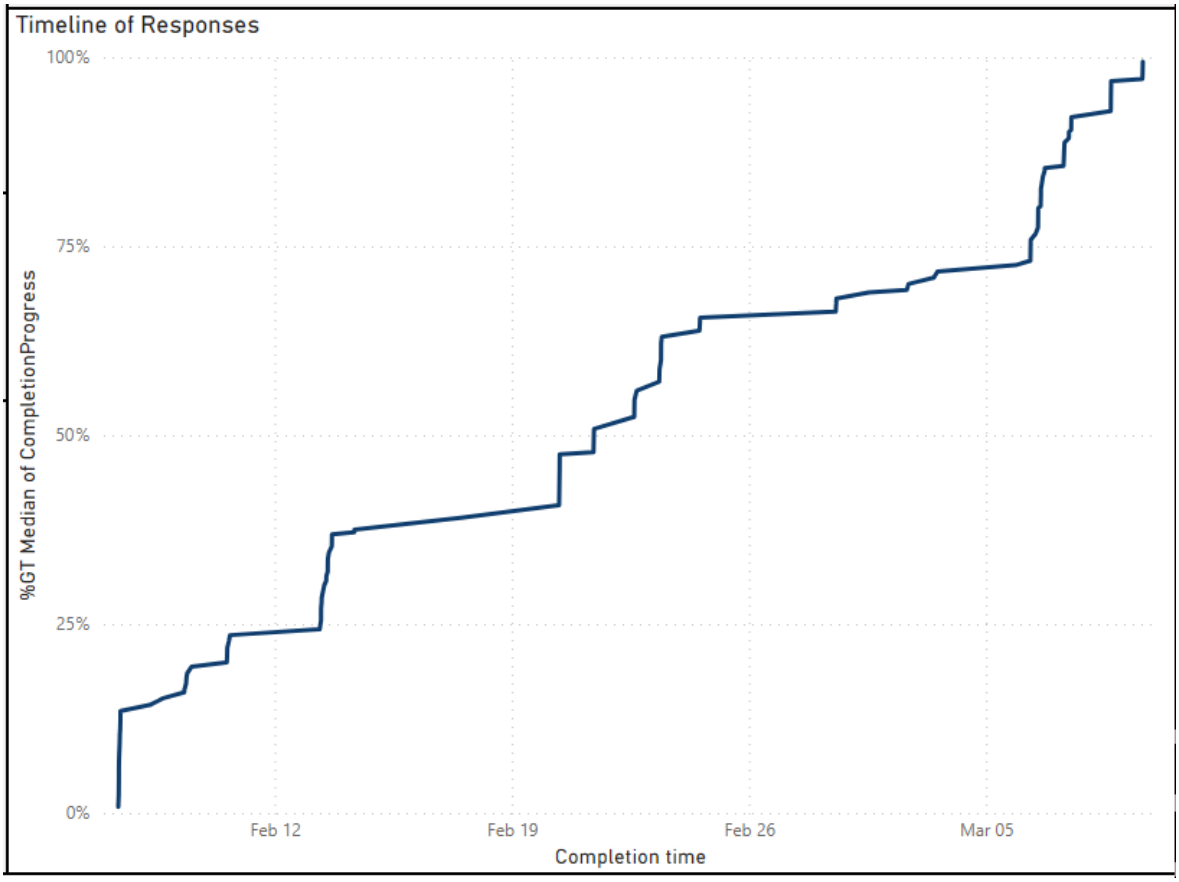



**Figure 6.2:** Timeline of Responses.

indicates that there is significant room for improvement, placing the organization at the third capability level according to Table 5.1. The ”Developing” capability level proposes that the organization has made progress in implementing certain practices and processes but still has ample opportunity to enhance its overall maturity.

### 6.2.4 Team Score Across Dimensions

The visualization in Figure 6.5, offers a comprehensive overview of all teams across the dimensions. Teams such as Plugins, and jTools have the lowest score in Requirements Management and Quality Assurance Management respectively. Appendix C, includes a drill down view of each dimension for more details, including the maximum, average and lowest scores.

### 6.2.5 Team Score Across Dimensions with Brands

This visualization in Figure 6.6, provides a comprehensive overview of all teams across their respective brands, encompassing multiple dimensions.

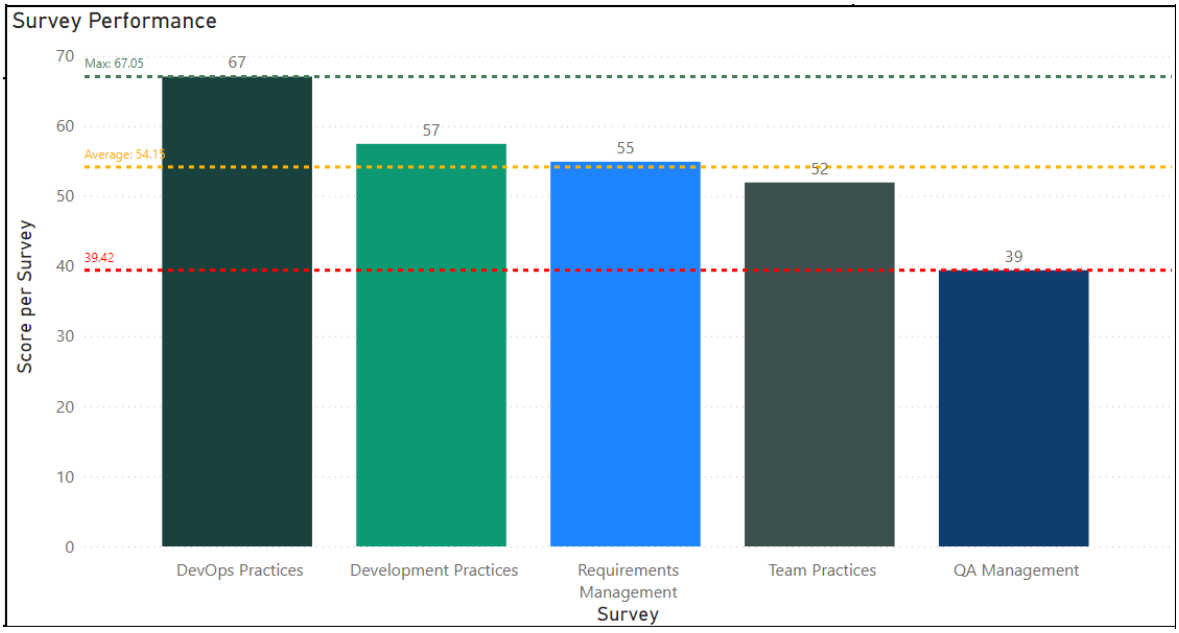



**Figure 6.3:** Average Score per Dimension.

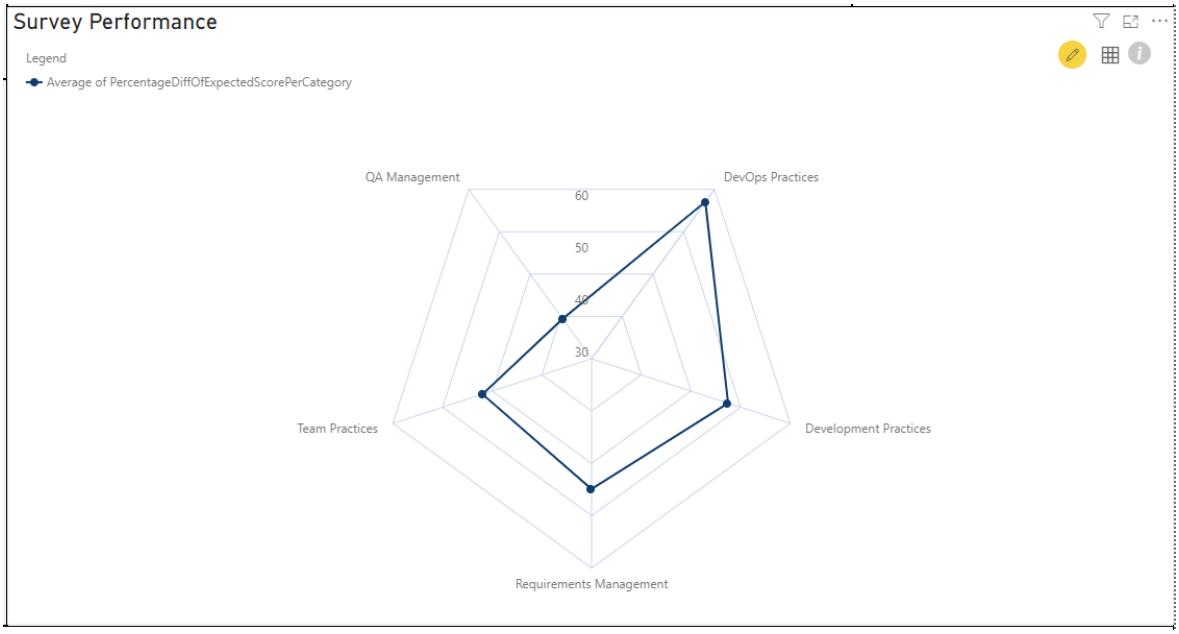



**Figure 6.4:** Radar of the Average Score per dimension, taking inspiration by Akkiraju et al. [6].

### 6.2.6 Team-Brand Comparison per Dimension

In Figure 6.7, a deeper analysis of the relation between brands for each team is presented, with the Websocket pusher team serving as an example.

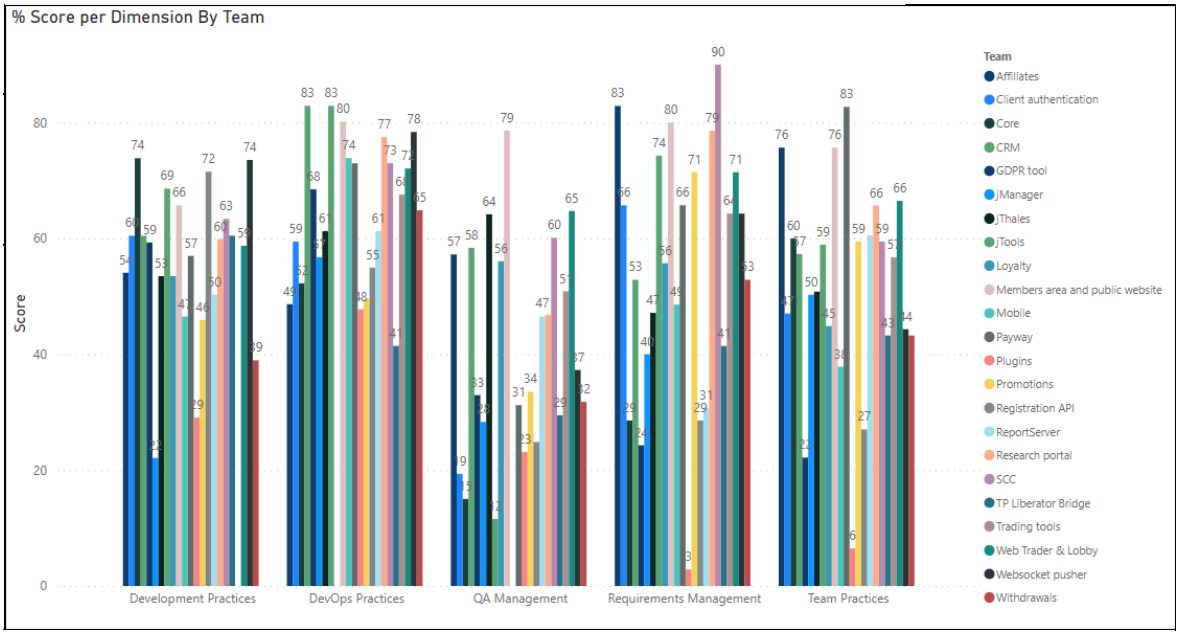



**Figure 6.5:** Team Score Across Dimensions.

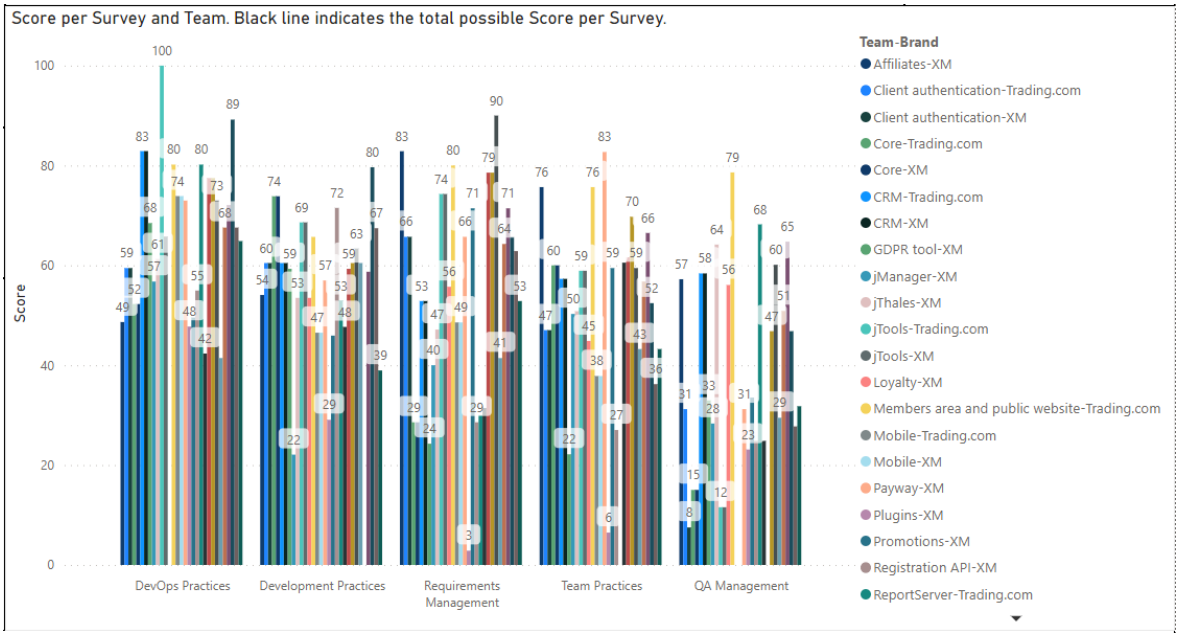



**Figure 6.6:** Team Score Across Dimensions with Brands.

### 6.2.7 Team Deviation from Average

The visualization depicted in Figure 6.8, illustrates the deviation of each team's score from the overall average. Teams located on the right hand side with negative values are below average, while teams on the left-hand side are above average. Notably, the Plugins team stands out as it consistently under performs across all dimensions.

The chart on the right-hand side further illustrates the deviation of scores, con-

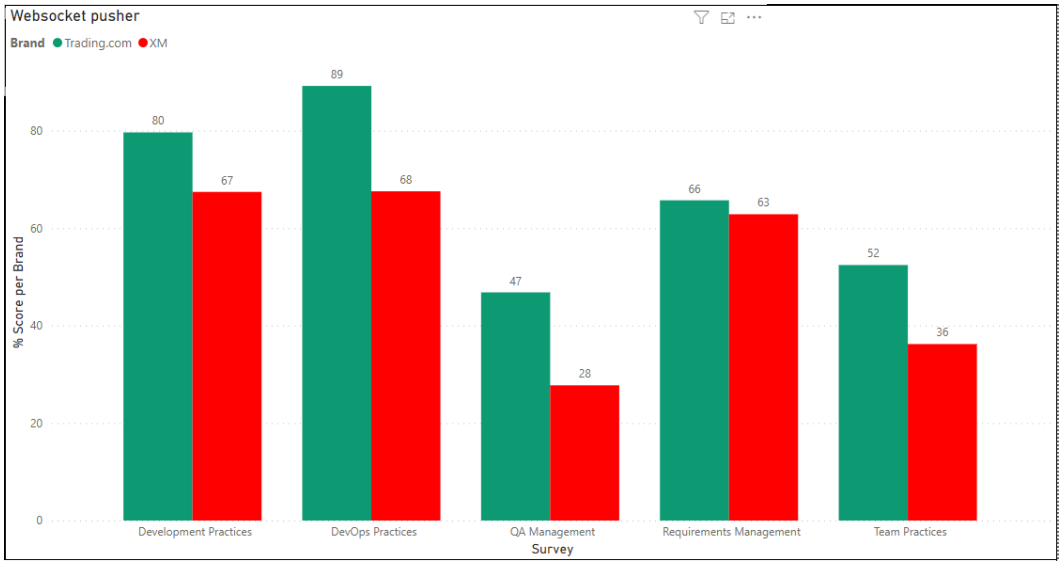



**Figure 6.7:** Team-Brand Comparison per Dimension.

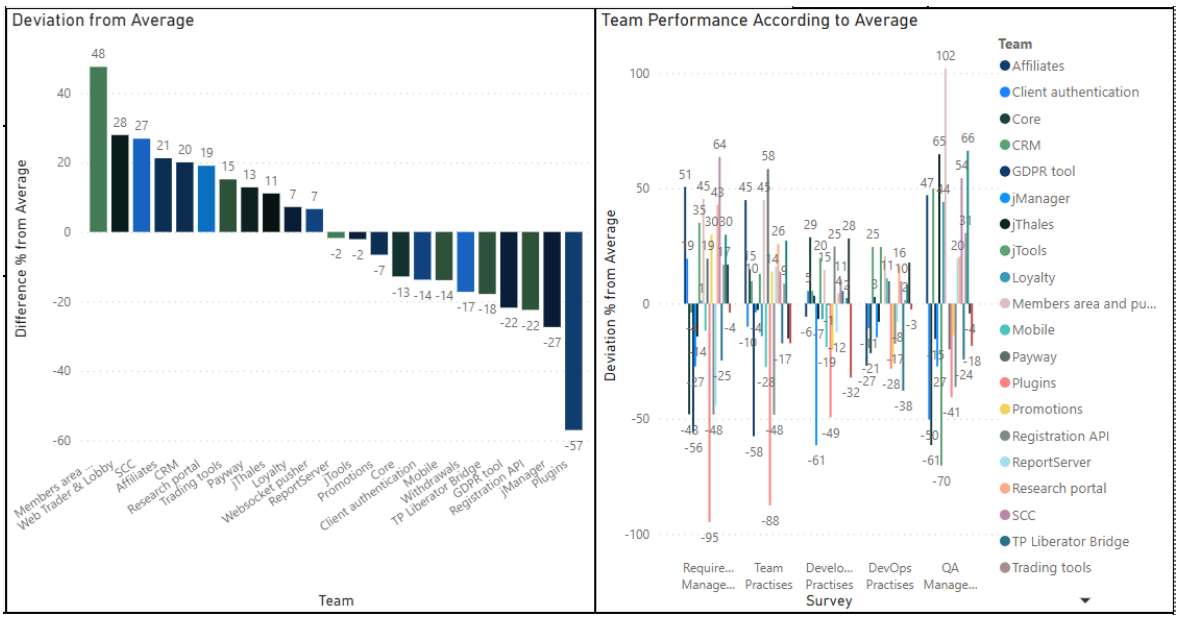



**Figure 6.8:** Team Deviation from Average.

sidering the brands associated with each team.

### 6.2.8 Average Score per Category

Figure 6.9 illustrates the average scores per category within their respective dimensions, throughout the organization's software development teams.

In the initial stages of implementing the assessment method, it is generally considered ideal for teams to strive for a score of 50% in each category. This score suggests that teams have achieved a moderate level of adherence to the practices

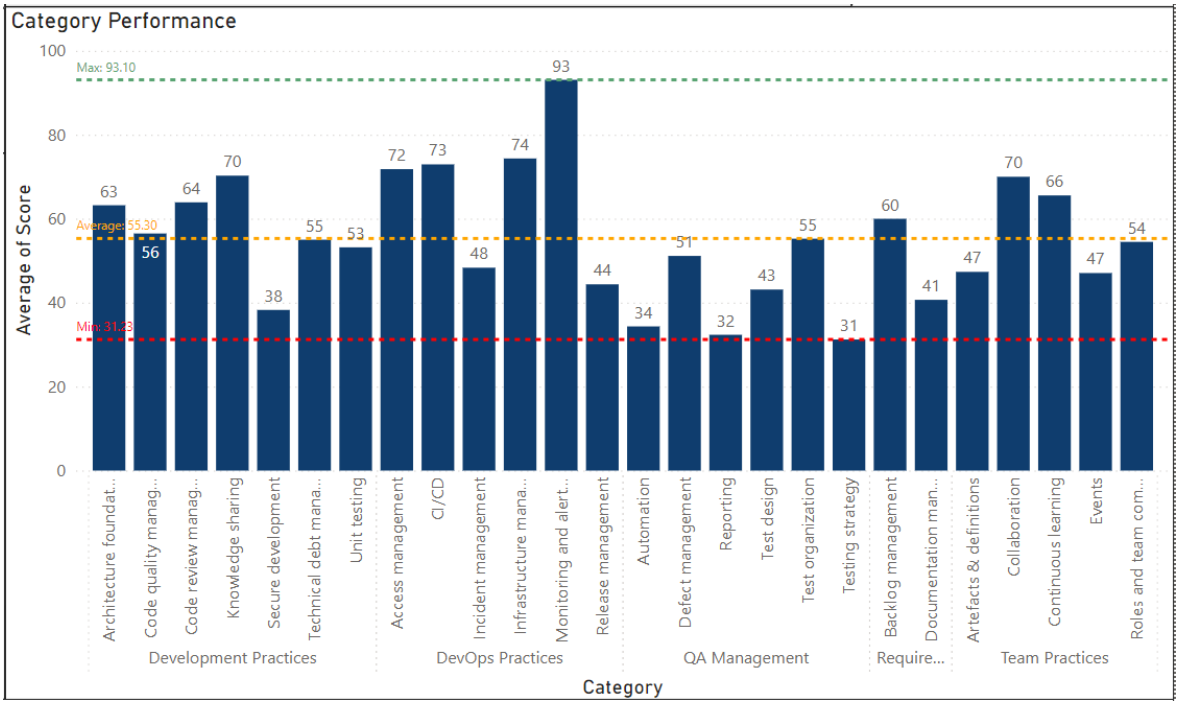



**Figure 6.9:** Average Score per Category.

and processes outlined in the CMMI model. The analysis of the figure reveals that nine out of twenty-six categories have a score below 50%, making them a good starting point for the organizations improvement initiatives. The ultimate aim of implementing the assessment method is to continuously improve and advance beyond the initial target scores. A more precise identification of specific questions and corresponding areas of interest is carried out through further analysis in the following sections.

### 6.2.9 Frequently Answered Questions as No

Figure 6.10 showcases a table, which focuses on the questions that received the highest number of ”No” responses within each category of the dimensions, where the corresponding scores are below the desired threshold of 50%.

Although Team Practices have received an average score of 53%, its category Artifacts & Definitions had questions that had the most negative answers. In Development Practices the category Secure Development is under performing. In Quality Assurance Management most categories under perform, these include Automation, Reporting, and Testing Strategy. Automation in quality assurance for software development refers to the use of automated tools, frameworks, and processes to streamline and enhance the testing and quality assurance activities throughout the software development life cycle. Reporting in Quality Assurance includes the process of doc-

| Survey | Category | Question | Number of NOs |
|---|---|---|---|
| Team Practises | Artefacts & definitions | Do you have documented Definition of Ready? | 28 |
| Team Practises | Artefacts & definitions | Do you keep Definition of Ready up-to-date? | 28 |
| Team Practises | Artefacts & definitions | Do you validate each user story against your Definition of Ready? | 28 |
| Development Practises | Secure development | Do you have SLAs for fixing security issues? | 26 |
| Team Practises | Events | Are key stakeholders reviewing iteration output and providing feedback to the team? | 25 |
| Team Practises | Events | Do you execute at least 2 improvements from each retrospectives during the next iteration? | 25 |
| QA Management | Testing strategy | Do You evaluate test strategy regularly and, where necessary, adapted for future use according to business needs? | 24 |
| Development Practises | Secure development | Do you have a list of security coding practices adopted on the project and published? | 24 |
| QA Management | Testing strategy | Do You include test design techniques (https://confluence.app.trading-point.com/display/TRAQA/Test+design+techniques) in test strategy? | 24 |
| QA Management | Automation | Do You regularly review critical areas for auto tests coverage? | 24 |
| QA Management | Testing strategy | Do You consider product risk analysis when creating test strategy? | 23 |
| Team Practises | Events | Do you have demo meetings for each iteration (i.e. a meeting where you present new api, new features, etc)? | 23 |
| Development Practises | Secure development | Do you obfuscate your code? | 23 |
| DevOps Practises | Incident management | Do you have a process for keeping your runbooks up to date? | 22 |
| DevOps Practises | Incident management | Do you have SLAs in place for responding to incidents? | 22 |
| QA Management | Automation | Do You include coding standards and test-data handling methods in automated test guidelines? | 22 |
| Team Practises | Events | Do you include key stakeholders in planning meetings? | 22 |
| QA Management | Reporting | Do You include metrics in bi-weekly reporting? | 22 |
| QA Management | Testing strategy | Do You keep Your production snapshot data up to date (i.e. frequently and when extra data is needed)? | 22 |
| Team Practises | Artefacts & definitions | Do you perform a production release at the end of every iteration? | 22 |
| QA Management | Testing strategy | Do You regularly evaluate test strategy against metrics from incidents that occurred in production? | 22 |
| QA Management | Testing strategy | Do You create snapshot of production like data? | 21 |
| QA Management | Testing strategy | Do You define test coverage and test depth according to analysed risks? | 21 |
| QA Management | Testing strategy | Do You define test levels and test types according to analysed risks? | 21 |
| Development Practises | Secure development | Do you have Security Quality Gates that can stop the release? | 21 |
| QA Management | Automation | Do You prioritise test cases for the automation in order to meet the test execution schedule? | 21 |
| QA Management | Reporting | Do You receive QA Sign off in written form for releases? | 21 |

**Figure 6.10:** Frequently Answered Questions as No.

umenting and communicating information about the quality of software being tested. For Testing Strategy it is ideal that teams have to adhere to key components of testing. Testing involves determining the scope and objectives of testing, which includes identifying the specific features, functionalities, and scenarios that will be tested [95].

In Requirements Management, Documentation Management has a score of 41%. It includes whether teams document all requirements in the release and task tracking system, whether teams defines non-functional requirements, and whether requirements are kept up to date. Key aspects of Requirements Management discussed previously in Section 4.4.2.

In DevOps Practices categories that under perform are Incident Management, and Release Management. Incident Management concerns activities for incident monitoring, such as defects and bugs found through out the software. These activities as seen by Figure 6.10 are, having standard Service Level Agreements (SLAs) for the processes, and policies for escalating incidents. Release Management concerns planning, coordinating, and overseeing the release of software products or updates [96].

### 6.2.10 Frequently Answered Questions as Yes

Similarly, Figure 6.11 displays a table extracted from Power BI, highlighting the top questions that received the most ”Yes” responses across dimension categories iden-

tified with scores lower than 50%.

| Survey | Category | Question | Number of Yes |
|---|---|---|---|
| Team Practises | Artefacts & definitions | Do you track all team work (tasks, defects, user stories, improvements, spikes) in a task management tool (i.e. jira)? | 29 |
| Team Practises | Artefacts & definitions | Do you track progress of items on a board (might be physical board or jira one)? | 29 |
| Team Practises | Artefacts & definitions | Do you track blockers on your board (special status, flags, blocked by task relationships etc.)? | 26 |
| Team Practises | Artefacts & definitions | Do you have a guide or handbook that contains information about setting up development environment to work? | 25 |
| Team Practises | Events | Do you have daily meetings? | 25 |
| QA Management | Testing strategy | Do You create testing data in a way is the same or similar to production data? | 23 |
| Team Practises | Events | Do you always perform planning meeting for iteration (i.e. sprint planning session)? | 22 |
| Team Practises | Events | Do you always use a board to faciliate daily meetings? | 22 |
| DevOps Practises | Incident management | Do you have runbook defined in case alert policy would be triggered? | 22 |
| DevOps Practises | Release management | Do you store release artefacts in artefact repository or are in a position to deploy any version of your software at any point of time without unnecessary delays (i.e. nexus, firebase, etc.)? | 22 |
| DevOps Practises | Incident management | Do you have alerting escalation policies defined? | 21 |
| Team Practises | Events | Do you include bug fixing when planning capacity for iteration? | 21 |
| QA Management | Test design | Do You create test cases in a way they are directly reflecting all requirements? | 20 |
| Team Practises | Artefacts & definitions | Do You estimate all work that the team committed to? | 20 |
| Team Practises | Events | Do you plan iteration in accordance to team capacity (member leaves etc....)? | 20 |
| Team Practises | Artefacts & definitions | Do you use versioning for releases? | 20 |
| Team Practises | Artefacts & definitions | Do you have a guide or handbook for on-boarding newcomers (i.e. project materials to be read, introduction to technology, etc...)? | 19 |
| DevOps Practises | Incident management | Do you have primary and secondary contact defined for each escalation policy? | 19 |
| Development Practises | Secure development | Do you perform regular static security scan? | 19 |
| Development Practises | Secure development | Do you scramble production data (remove sensitive info and names) when creating a test data? | 19 |
| Team Practises | Artefacts & definitions | Do you track release scope for each iteration (are you able to explain at any point in time what is Your plan for next release)? | 19 |
| Team Practises | Events | Do you always include qa members in planning meetings? | 18 |
| QA Management | Test design | Do You document test cases? | 18 |
| Team Practises | Artefacts & definitions | Do you update frequently your guide or handbook so it's always reflecting current project situation? | 18 |
| Team Practises | Events | Do you use specific tool for retrospectives? | 18 |

**Figure 6.11:** Frequently Answered Questions as Yes.

This table can be a useful tool for guiding less mature teams to adopt practices followed by teams with higher maturity levels. This approach can help bridge the gap between different maturity levels and facilitate knowledge sharing and growth across the organization.

## 6.3 Improvements

This section provides an overview of the key issues identified and suggests possible improvements for teams to consider. These findings and recommendations were derived through coordinated discussions involving team leaders, and by referencing the results of the assessment surveys. The proposed improvements incorporate best practices from relevant literature reviewed and insights shared by experts within the organization. Additionally, improvements have been tailored to align with each team's specific processes and ways of working. We first prioritize subcategories of dimensions that have scores less than 50%, and second by examining questions with high number of negative responses.

Testing strategy with a score of 31% shows that teams actually do not follow standard techniques, do not perform risk analysis, and fail to frequently update production snapshots. They further seem to fail on evaluating their test strategy, define

a test coverage and test depth according to analyzed risks. Together with these items we would suggest the team to clearly document requirements, and test procedures.

Reporting has a score of 32% and shows that teams do not include bi-weekly reports, do not have regular reviews for quality assurance reports in terms of informational value, transparency and ease of use. We would recommend that teams try to improve these things together with introducing templates for Quality assurance reports, as this could introduced a standardized way of reporting avoiding any inconsistencies and ensuring all tests are covered.

Automation with a score of 34% shows that teams are lacking when it comes to reviewing areas for autonomous testing, including coding standards, and prioritizing test cases to meet test execution schedules. The quality assurance teams do not develop test cases in parallel to the development cycle. Automated testing tools and frameworks can significantly enhance efficiency, accuracy, and coverage, freeing up valuable time for testers to focus on more complex scenarios [95].

Secure development with a score of 38%, shows that teams have a high number of negative answers when asked about the inclusion of service level agreements (SLAs) for fixing security issues. We suggest that teams create a list of security coding practices that should be adopted by projects and make it publicly available to developers providing them the appropriate training.

Documentation management has a score of 41% and shows that teams do not use any formal language to describe business requirements such as sequence diagrams. Most of the teams also fail to keep their requirements up to date, and have not defined their non-functional requirements for their projects.

Test design has a score of 43%, shows that teams are lacking the ability to link test cases with requirements, and teams do not evaluate and adjust test design techniques for further use. It was observed that teams often rely on manual testing methods, such as checking if a button works or verifying changes after deployment, without a comprehensive approach to testing. To address this issue, it is recommended that teams adopt more robust testing procedures and embrace automation where applicable.

Release Management has a score of 44% showing that teams do not have the appropriate documentation as a checklist that serves as a guide for decision making about releases. The teams further do not write down any release notes, or a rollback strategy written down. Therefore, it is crucial for teams to have well-defined procedures, transparency, and standardized templates, which can be introduced by a motivated senior person.

Artefacts and definitions show a score of 47% and determines that teams do not document, keep up to date and validate their definition of ready. Teams further fail

to perform a production release at the end of every iteration or track releases in Jira. Moreover, they do not have a definition of done, or keep it up to date. However, during the discussions, teams highlighted a common practice of starting work and facing unexpected issues. Teams can aim to provide approximate time estimates for completing work items and establish requirements that closely align with actual expectations. It was also noted that the problem lies in the absence of agreed and documented requirements, rather than attempting to overly specify every detail.

Events with a score of 47% shows that team does not have feedback on their deliverable. Teams do not put effort on improving, or have demo meeting for every sprint. Without receiving feedback from customers, teams are unable to learn from their experiences and address potential errors effectively. It is essential to incorporate customer feedback loops to foster continuous learning and improvement. To enhance the testing process, several improvement points were identified, including scope identification, approach enhancement, risk assessment, and addressing limitations in requirements.

A suggested overall improvement is to establish a minimal viable guideline that teams should follow while allowing for flexibility and decentralization in certain aspects. This approach recognizes that each team may have unique implications and ways of working, and it promotes a culture of adaptability and ownership. The guideline can provide a baseline set of principles and best practices that all teams are expected to adhere to. This could include overarching agile principles, such as collaboration, iterative development, and continuous improvement. The purpose of the guideline is to ensure a common understanding and alignment across teams, while still allowing room for autonomy and customization. The responsibilities of these guidelines can be assigned to teams in order to introduce motivation and further address team needs.

# Chapter 7

# Validity Analysis

This chapter provides a comprehensive overview of the validation surveys conducted, which determined the relevance and validity of the assessment method. Moreover, this chapter establishes a concrete understanding of the validity of the assessment method, by examining several software development metrics historically and determining any performance gains.

## 7.1 Validation Surveys

In order to ensure the validity of this thesis and facilitate a potential redesign of the assessment method, a targeted survey has been distributed among participants of the assessment process. This validation survey serves as a critical tool to validate the accuracy and applicability of the assessment method. This iterative process plays a crucial role in strengthening the foundation of the research, as it allows for a thorough examination and validation of assessment method and criteria.

Based on the feedback received, respondents provided insights on various aspects of the assessment process. They shared their opinions on the overall perception of the assessment method, clarity and comprehensibility of the questions in the assessment surveys, and whether the questions effectively capture the relevant aspects of the organization.

Due to the number of responses acquired, we decided to showcase the overall context and group the questions with similar context. The subset was selected based on the answers comprehensibility, their perceived usefulness, and how insightful they are.

The feedback from participants regarding their overall experience can be categorized as follows: There were 10 promoters who expressed positive satisfaction, 5 passives who were neutral in their assessment, and 3 detractors who had concerns or negative feedback. The answers in this regard were:

*"It is a very good measure to see what processes we are lacking and could be improved."*

*"I think is a nice check we have to do once or twice a year. Sometimes our reminders getting ignored for many of these issues"*

*"I dont see the value, what we need to do is have direct line from the IT Governance team with each of the teams. These kind of everybody answer-all-questions are pointless"*

Overall, there is a positive opinion on the value that this assessment process can yield, others however where skeptical on the number of improvement points that could be discussed and the guidance that the teams should have.

Questions in maturity assessment surveys where clear and easy to understand: with 8 promoters, 7 passives and 3 detractors. The open-question answers, to specify what was unclear are:

*"The questions where clear enough."*

*"Would prefer if some examples were also mentioned in some questions. For instance, it was not clear to me what 'key stakeholders' meant."*

Questions in maturity assessment covering the right aspects had 9 promoters, 6 passives and 3 detractors. With open-question specifying which questions are not valid answered with:

*"Quality Assurance questions to teams that do not have a Quality Assurance. Questions about scrum to teams that do not use scrum."*

*"The questions about releases as our team is not working with releases but continuous deploys."*

Another question for this survey included was whether team leaders would include any other questions in the maturity assessment. Answers are:

*"I think we should include less questions not more."*

*"I think this covers most important topics."*

*"You should include questions such as: Is there a standard process for making change requests to teams?"*

In summary, the survey results highlight a generally positive perception of the value provided by the assessment process, although there were concerns and differing viewpoints. The feedback received has been invaluable in refining the survey questions and ensuring they effectively capture the relevant aspects of the organization. Moving forward, these insights will guide further improvements to enhance the accuracy, applicability, and efficiency of the maturity assessment surveys.

## 7.2 The Software Development Performance.

The goal of this section is to determine whether the implemented assessment method aids the organization into achieving better performance. The analysis of these metrics show a quantitative performance timeline and hence the applicability of the assessment method in the organization's software development teams. While the CMMI model itself outlines a set of best practices, it is essential to assess its impact and effectiveness within the organization to ensure that the intended benefits are being realized. To assess the success of the assessment method, the organization can rely on various software development metrics. These metrics serve as quantifiable indicators, providing valuable insights into the performance and progress of the organizations improvement initiatives.

Teams agreed to work with their own practices according to the specific requirements and goals of each team. The purpose of the thesis is to introduce a minimal set of procedures for each team to follow in order to achieve higher performance in software development processes. With improvement initiatives discussed previously a transition has led to performance increase as shown in Figure 7.1. The average lead time showcases the time that software development task has been committed to the backlog until the time that it has been completed. Lead time is a commonly used metric which showcases the amount of time it takes to complete a process or service from the moment it is requested or initiated. It is a metric used by organizations to measure software development performance [77]. Ideally, the lead time of tasks should decrease which can be seen by the bar chart in a monthly basis.

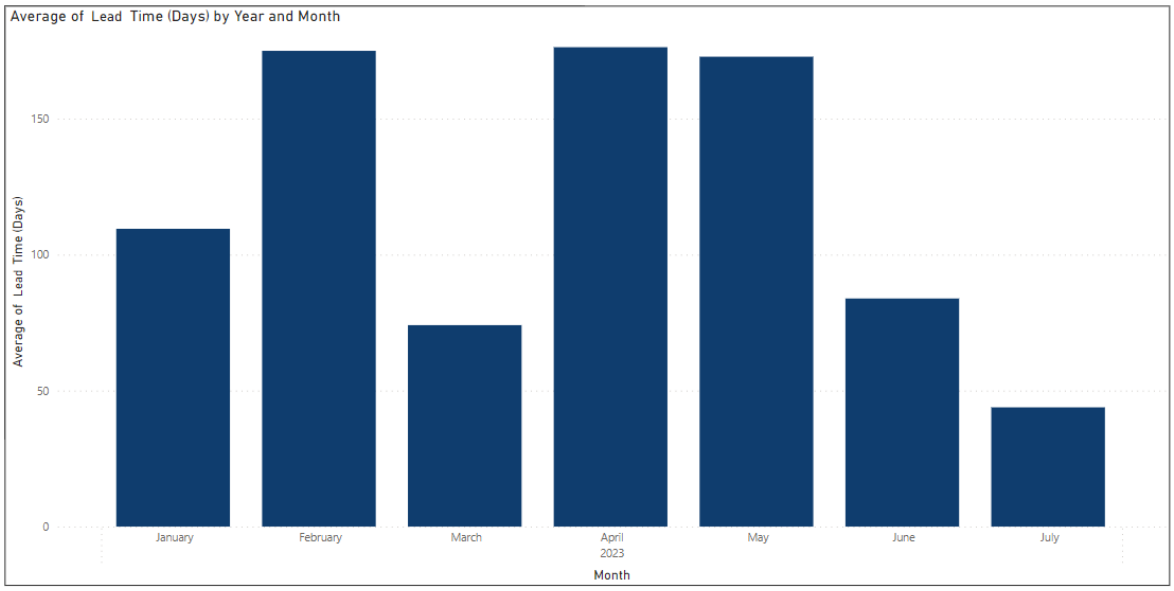



**Figure 7.1:** Average Lead Time in Days

Moreover, Figure 7.2 depicts the average cycle time which is the time it takes to

complete a single cycle of a task. It starts from the beginning of one unit of work and ends with its completion. Cycle time is a measure of process efficiency and is often used to evaluate and improve productivity [65]. The figure showcases that the cycle time has been decreased overall, confirming that improvement initiatives have played a role.

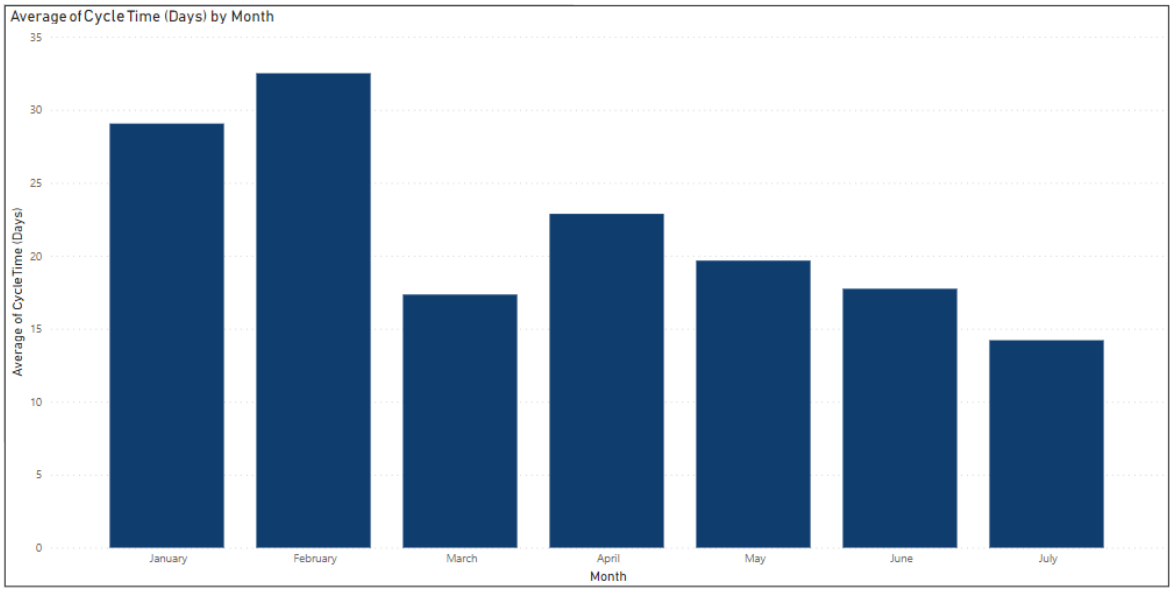



**Figure 7.2:** Average Cycle Time in Days

Incidents refer to unexpected events or issues that occur during the development process or after the software has been deployed. Ideally the number of incidents for an organization should stay low. Figure 7.3, showcases the time of responding and resolving an unexpected event. Although, the bars in the figure fluctuate between months there is a noticeable decrease in incidents in July. The number of incidents include bugs, security vulnerabilities, performance issues, and integration failures. This directly impacts software development performance and management of incidents is a requirement.

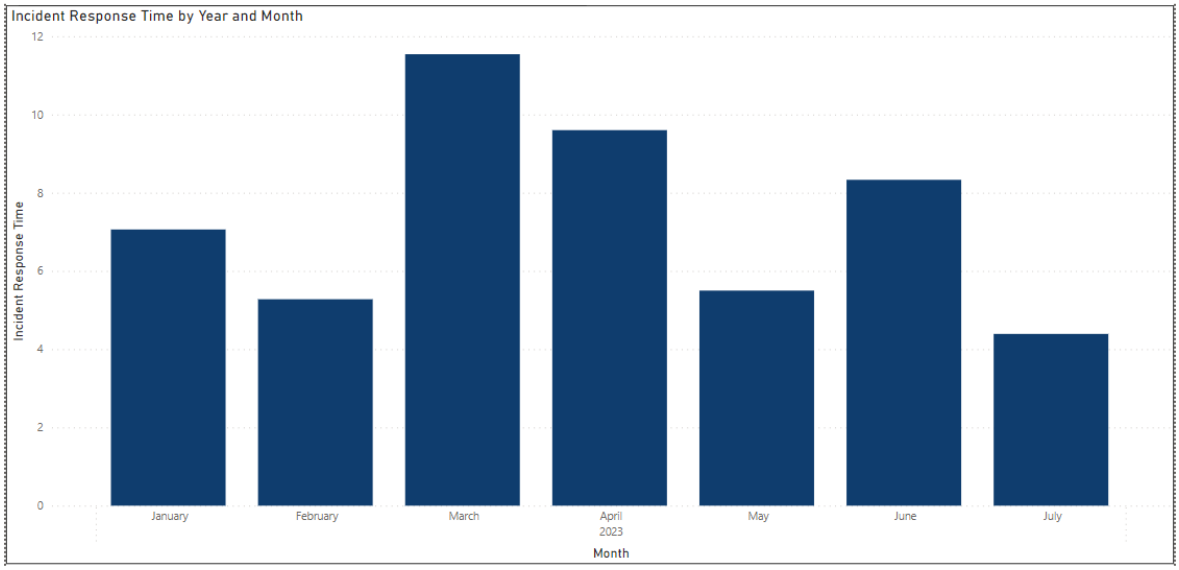



**Figure 7.3:** Incident Response Time

# Chapter 8

# Discussion

This chapter describes the objective of this research, core values of the designed assessment method, the validity of this method as well as the thesis contribution to research.

## 8.1 Implications

The objective of this research was to design an assessment method as an approach to help organizations improve their software development processes. The need for an assessment method arose from the increasing complexity of software development processes. However, current assessment methods are limited in their ability to provide a systematic method of how to assess an organization. This gap hinders the ability to provide accurate and reliable assessments, therefore, this thesis fills this gap by providing a transparent assessment method.

### 8.1.1 Core values of the assessment method

The assessment method designed as an artifact consists of the CMMI model, the software development dimensions, data presentation, and the introduction of improvements. The CMMI model is a framework for process improvement and constitutes of two representations with 22 process areas. Ultimately, the continuous representation was chosen. The selection process took into consideration the specific business context, and goals for continuous improvement of the organization. While, the software development dimensions have been identified by experts of the organization and have been used as a reference framework for the assessment criteria. The software development dimensions used in this thesis are Team Practices, Requirements Management, Development Practices, DevOps Practices, and Quality Assurance Management. The objective was to ensure that the software development dimensions align closely with the process areas outlined in the CMMI

model. We then create the assessment surveys to assess the organization processes. Consequently, we collect the data and visualize it with the use of Power BI, giving a path towards implementing the improvements. Finally, we ensure validity of the assessment method with two methods. Firstly, we conduct a validation survey which provided feedback on the designed artifact, and an analysis of software development metrics to measure the impact of the assessment method.

When it comes to applying this assessment method other organizations should not ask themselves how they can implement a fixed assessment method but rather on how they can tailor this method to their needs in order to leverage their improvement processes through the CMMI model and experience the maximum amount of benefits via minimizing risks out of change. While the selection of software development dimensions has been associated with specific areas, it is crucial to acknowledge that these dimensions may not be universally applicable to every organization. The process of selecting these dimensions must consider the unique context and goals of each organization.

The thesis provides an assessment method that allows organizations to effectively assess their software development processes, even in the absence of universally defined software development dimensions. The assessment method outlined in the thesis aligns with the process areas defined by the CMMI model, highlighting the importance and value of such an approach. Additionally, the thesis acknowledges the significance of various domains, such as soft, hard, and context governance, as identified in the IT governance model foundation by Smits and Hillegersberg [7]. These domains play a crucial role in the success of improvement initiatives, further emphasizing the nature of the method presented in the thesis.

However, this thesis has its limitations due to the fact that the Design Cycle by Wieringa provides an iterative process of (re)designing an artifact, and has not been followed on its full potential. An initial design was created and one cycle was performed which included the treatment validation step, which could ideally provide the means to proceed with a second iteration. However, due to the complex human factor, change management, and the time required for improvements to take effect, it was not possible.

## 8.2 Thesis Validity

Validity refers to whether the artifact that is designed to measure a certain concept really does measure the indented concept [1]. According to Wieringa, the object of study that supports an inference also supports the validity requirements of this object of study. Wieringa uses several types of validity and categorizes them as follows.

### 8.2.1 Construct Validity

Wieringa defined construct validity as the degree at which the implementation of constructs to phenomena is warranted with respect to the research goals and questions [1]. Construct validity in general is a matter of degree, they can be valid up to a certain level and never be totally valid.

Considering this thesis construct to improve the software development processes of an organization, validation is warranted by four criteria. Firstly, the literature review helps to identify and incorporate relevant practices and findings from existing literature into the assessment method. By aligning with established frameworks, the research methodology becomes more rigorous and systematic, enhancing the construct validity of the thesis findings. This approach ensures that the assessment method developed in the research accurately reflects the current knowledge and practices identified in the literature. Pattern matching was utilized by studying the software development dimensions and the CMMI model's process areas across several research papers, as well as their effects and best practices. Mainly speaking the CMMI model provides a framework for process improvement, and the software development dimensions are concepts which are widely known in the research community and industry.

Secondly, experts were constantly in close proximity contributing their thoughts and expertise in the identification of the software development dimensions and their categories as well as designing the questions for the assessment surveys.

The third criteria was the validation surveys that where directed to the assessment method. This was to ensure that the assessment method captured the right essence, making it relevant to improving the processes of the organization.

Finally, this thesis is utilizing the concept of goal setting theory, which determines goals that are specific, measurable, achievable, relevant and time bound [97]. By measuring previous performance and tracking performance after the implementation of the assessment method. Indicators such as lead time, cycle time, and the number of incidents are used to ensure that poor operationalization is mitigated in a quantitative manner. This was done with the comparison of the software development metrics throughout a period of six months.

### 8.2.2 Internal validity

Andrade [98] determined internal validity as the way to examine whether the subject of study that was designed, conducted, and analyzed allows trustworthy answers to research questions in the study. It checks whether it is based on judgement and is not a computed statistic and whether there is systematic bias. There is a need to contextualize the use of the assessment method within the context of the

organization.

However, the internal validity of this thesis is somewhat uncertain due to concerns regarding the control of confounding variables. In particular, many variables can contribute to the observed increase of performance in the organization's software development processes, which are not controlled by this thesis. These variables could include the nature of a large organization, the extensive number of employees and teams the organization encompasses and the changes in team size, or team dynamics. Moreover, the months that software development metrics were measured could have specific attribute such as bank holidays where work is decreased. A way to accommodate this would be to measure performance throughout several years.

### 8.2.3 External validity

Wieringa [1] defines external validity as the degree of support for the generalization of a theory beyond the cases or population on which it was studied. An object of study should be repeatable in the sense that other researchers will be able to acquire similar results. For the object of study to support external validity it should ensure that it satisfies the population predicate [1]. The object of study might include more than one predicates, hence it should be enabling representative sampling. The treatment's ability to generalize, can be ensured by following similar treatments [1].

The literature review throughout the research project ensures external validity to a degree. The design of the assessment method is studied as well as techniques for data presentation. This would ensure that reapplying the methods used in this research for analyzing the maturity of an organization would be as effective and applicable to a degree in other organizations as well. By incorporating the process areas of the CMMI model, we enhance the thesis' credibility and relevance to the broader community.

However, the dimensions identified and questions of the assessment surveys are directly applicable to the organization's context. The external validity to this extend is questionable and needs to be researched in other organizations to get a final observation. Furthermore, the survey sample does include multiple teams within the organization however these teams are part of the same organization. However, due to the international structure of the organization one can argue that the diversity of participants is met.

## 8.3 Contributions to Research

The existing literature on maturity assessment models is extensive, but there is a lack of a transparent method and practical usage experience for their implementation. While research on assessment methods has increased significantly, it often remains focused on case studies or conceptual analysis rather than analytical implementations. This study provided a systematic process of how to implement an assessment method based on the CMMI model as a framework for process improvement in practice. Consequently practical experience necessitates the ability of assessment methods to align with the organization's context and internal structures giving rise to the software development dimensions which are used as assessment criteria and provide relevance to the CMMI model's process areas. In addition to designing the assessment surveys and applying improvements, this study also examines the impact based on quantitative data from the organization, achieved by introducing an analysis of software development metrics.

# Chapter 9

# Final Remarks

This Chapters covers the final remarks of the thesis, what conclusions were made answering the research question and sub-questions, as well as a discussion about the future work.

## 9.1 Conclusions

With the ever increasing complexity of software development processes, management becomes more and more challenging. IT governance is a corporate governance body which aims to improve the overall management of software development processes, while utilizing a plethora of methods. Such methods are maturity assessments which serve as invaluable tools to assess an organization's current maturity state and consequently introduce improvements. In order to advance maturity and overall introduce changes we need to drive attention on both the hard and soft governance. Therefore, this thesis designed an assessment method as an approach to help organizations improve their software development processes. An assessment method was necessary due to the gap of current literature which does not provide a systematic way and practical use of how to assess an organization.

Therefore, the main research question of this thesis was:

*How can we improve the software development processes of the organization by applying an assessment method, in order to help the organization cope with the ever increasing complexity of their processes?*

To answer the main research question, we answered three sub-questions, all of which are addressed in the subsequent discussion. The first sub-question was:

*1. What are the attributes of CMMI models, that aid to design an assessment method for software development processes in the context of the organization?*

To address this sub-question we got an understanding of the first two components of the assessment method. Namely, the attributes of the CMMI model which is used as a reference framework for process improvement, and the software development dimensions as a guiding framework for assessment criteria. The literature has shown that CMMI models use four categories which are comprised of 22 process areas in total. These process areas represent key practices and activities that need to be performed effectively to achieve a particular level of maturity. We have further recognized the significance of modifying the CMMI by utilizing a subset of the model to better align with the organization's specific context and goals. This gives rise to the software development dimensions which are used as a reference framework to determine assessment criteria, and are key building components for any assessment method, indicated by Table 2.1 from Chapter 2. Experts possessing expertise and knowledge about the organization were involved to identify the five software development dimensions. Literature has provided us with the knowledge about the dimensions which in turn enabled us to follow the CMMI model, ensuring that we cover as many process areas as possible. In the assessment method, we covered 19 out of 22 process areas, leaving behind the, ”organizational performance management”, ”organizational process performance”, and ”supplier agreement management” areas. This is due to the fact that these process areas are not covered by the software development teams.

The second sub-question was:

*2. What data presentation techniques can be used to effectively visualize the assessment surveys results and support the implementation of improvements?*

Upon conducting the assessment surveys, answers were visualized according to best practices. To accomplish this, the thesis utilized insights from literature review that provided guidance on best practices for visualizing. This entails selecting appropriate chart types and layouts that effectively convey the intended message. Additionally, scales were utilized that accurately represent the magnitude of the data being presented. Proper scaling ensured that the visualizations accurately reflected the relationships and patterns within the data. Color and size were also employed strategically to convey information. Careful color choices were used to differentiate categories or highlight specific data points, while size variations were utilized to represent magnitude.

Following the analysis, software development processes were identified with a particular focus on low scores for dimensions Team Practices, and Quality Assurance Management. However, the areas for improvement extend beyond these specific dimensions and encompass various categories within each dimension. In response to identifying process areas we have set a goal of 50% score in order to

prioritize subcategories of the dimensions. Furthermore, we looked at individual questions to determine the root cause of low scores, by prioritizing questions with a high number of negative answers. Team leaders were responsible to implement improvements on testing strategy, reporting, automation, secure development, documentation management, test design, release management, artefacts and definitions, and events. A key improvement was the introduction of guidelines which provide a baseline set of principles and best practices that teams are expected to follow.

The third sub-question was:

*3. How does our assessment method affect the performance of the organization's software development processes?*

The final sub-question focused on validating the assessment method, firstly by examining the feedback received from the validation survey, and secondly by analyzing the software development metrics. The validation surveys provided us with an overall positive feedback regarding the applicability of the assessment method, the opinions of stakeholders (team leaders) about the perception and values delivered by the assessment method, and the clarity of the assessment criteria and questions. The assessment method was perceived as valuable by the participants in most cases, in others they where skeptical and instead proffered to have a direct guidance from the governance bodies of the organization. In terms of the clarity of questions participants have found it clear enough, while some would have preferred including some examples. In terms of the relevance there was an overall positive response, while others pointed out that some questions were not applicable to their team's context. With all this in mind the validation surveys provided us with feedback and the means to redesign the assessment method.

To assess the impact of the assessment method on software development performance, various metrics were identified based on best practices from the literature. Lead times, cycle times, and incident response times showed some impact as indicative measures. It was observed that the assessment method had contributed to the improvement of software development processes to an extend. However, it was found that establishing a strong causal relationship with the independent variables (software development dimensions) on the dependent variables (software development metrics). External variables, such as bank holidays, changes in team size, or team dynamics, may have contributed to the observed improvement. These factors were beyond the control of the thesis and could potentially influence the measured outcomes. Recognizing the influence of these external variables is important for a comprehensive understanding of the findings and constitute the limitations of the thesis. It highlighted the need to consider and account for such variables in the analysis and interpretation of the results in future research.

## 9.2 Future Work

The focus of this thesis was to design an assessment method, according to the organization's context and by utilizing the CMMI model. We have used the continuous representation of the CMMI model, and therefore a pertinent question arises regarding the applicability of a similar assessment method using the staged representation. Mainly, it is worth considering how organizations that do not follow a continuous improvement approach can apply this assessment method. Further research, would involve analyzing and comparing the effects of implementing the assessment method using the continuous and staged representations of the CMMI model, as a comparative approach.

Furthermore, this thesis was conducted within a specific organization that operates in the trading industry. The external validity of the thesis is indeed subject to scrutiny due to the contextual constraints of this particular scenario. In order to address this limitation and broaden the generalizability of the findings, future research endeavors could undertake an analysis that compares and examines the applicability of the assessment method across multiple organizations and industries.

Further, an examination of the implementation was performed utilizing the existing quantitative data of the organization. This analysis was performed with limited number of software development metrics. Further research could give more insights about the effects of this assessment method with other software development metrics, such as agile velocity, sprint goal success rate, number of software releases, throughput, response time, and reliability, availability and serviceability.

Moreover, external variables, including factors like team members' annual leave, changes in team size, and team dynamics, may have played a role in the observed improvement of software development processes. The nature of the dynamic work environment, bank holidays and many more variables were not under the control of the research and had the potential to impact the measured outcomes. Taking into account and exercising control over these variables is a crucial step in ensuring both the validity of the thesis and accurately attributing the assessment method's contribution to the observed improvements.

# Bibliography


[1] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin, Heidelberg, 2014.

[2] C. P. Team *et al.*, "Cmmi for software engineering, version 1.1, staged representation (cmmi-sw, v1. 1, staged)," *Software Engineering Institute, Pittsburgh, PA*, 2002.

[3] S. C. P. Team, *CMMI for Development v1. 3*. Lulu. com, 2010.

[4] M. R. Ayyagari and I. Atoum, "Cmmi-dev implementation simplified," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, 2019.

[5] M. Staron, "Dashboard development guide how to build sustainable and useful dashboards to support software development and maintenance," 2015.

[6] R. Akkiraju, N. Nayak, R. Torok, and J. von Kaenel, "A practitioner's tool for enterprise risk management capability assessment," in *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*. IEEE, 2010, pp. 369–374.

[7] D. Smits and J. van Hillegersberg, "It governance maturity: Developing a maturity model using the delphi method," in *2015 48th Hawaii International Conference on System Sciences*, 2015, pp. 4534–4543.

[8] I. L. Margarido, J. P. Faria, M. Vieira, and R. M. Vidal, "Challenges in implementing cmmi high maturity: lessons learnt and recommendations," 2013.

[9] S. Mahmood and D. C. Kundian, "Capability maturity model integration cmmi," *The Virtual University of Pakistan*, 2015.

[10] M. Chaudhary and A. Chopra, *CMMI for Development*. Springer, 2017.

[11] F. Delice, M. Rousseau, and J. Feitosa, "Advancing teams research: What, when, and how to measure team dynamics over time," 2019.

[12] F. Selleri Silva, F. S. F. Soares, A. L. Peres, I. M. de Azevedo, A. P. L. Vasconcelos, F. K. Kamei, and S. R. de Lemos Meira, "Using cmmi together with agile software development: A systematic review," *Information and Software Technology*, vol. 58, pp. 20–43, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914002110

[13] J. Becker, R. Knackstedt, and J. Poeppelbuss, "Developing maturity models for it management," *Business & Information Systems Engineering*, vol. 1, pp. 213–222, 06 2009.

[14] M. Simonsson, P. Johnson, and H. Wijkström, "Model-based it governance maturity assessments with cobit," 2007.

[15] S. De Haes and W. van grembergen, "It governance and its mechanisms," *Information Systems Control Journal*, vol. 1, pp. 27–33, 01 2004.

[16] "Ieee standard computer dictionary: A compilation of ieee standard computer glossaries," *IEEE Std 610*, pp. 1–217, 1991.

[17] G. Kaur, I. Kaur, S. Harnal, and S. Malik, "Factors and techniques for software quality assurance in agile software development," *Agile Software Development: Trends, Challenges and Applications*, pp. 257–272, 2023.

[18] E. Salas, D. L. Reyes, and A. L. Woods, "The assessment of team performance: Observations and needs," *Innovative assessment of collaboration*, pp. 21–36, 2017.

[19] Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå, "Teamwork quality and project success in software development: A survey of agile development teams," *Journal of Systems and Software*, vol. 122, pp. 274–286, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016412121630187X

[20] I. Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying devops practices of continuous automation for machine learning," *Information*, vol. 11, no. 7, p. 363, 2020.

[21] R. Gove and J. Uzdzinski, "A performance-based system maturity assessment framework," *Procedia Computer Science*, vol. 16, pp. 688–697, 2013, 2013 Conference on Systems Engineering Research. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050913000732

[22] M. Simonsson and P. Johnson, “Defining it governance-a consolidation of literature,” in *The 18th conference on advanced information systems engineering*, vol. 6. Citeseer, 2006.

[23] W. Van Grembergen, “Introduction to the minitrack ”it governance and its mechanisms” hicss 2002,” in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2002, pp. 3097–3097.

[24] S. Hamaker and A. Hutton, “Principles of it governance.” *Information Systems Control Journal*, vol. 2, 2004.

[25] N. Kakabadse and A. Kakabadse, “Is/it governance: Need for an integrated model,” *Corporate Governance*, vol. 1, pp. 9–11, 12 2001.

[26] T. Mettler, P. Rohner, and R. Winter, “Towards a classification of maturity models in information systems,” in *Management of the Interconnected World*, A. D’Atri, M. De Marco, A. M. Braccini, and F. Cabiddu, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 333–340.

[27] S. Peldzius and S. Ragaisis, “Comparison of maturity levels in cmmi-dev and iso/iec 15504,” *Applications of Mathematics and Computer Engineering*, pp. 117–122, 2011.

[28] D. Proença and J. Borbinha, “Maturity models for information systems - a state of the art,” *Procedia Computer Science*, vol. 100, pp. 1042–1049, 12 2016.

[29] M. Röglinger, J. Pöppelbuß, and J. Becker, “Maturity models in business process management,” *Business process management journal*, 2012.

[30] L. A. Lasrado, R. Vatrapu, and K. N. Andersen, “Maturity models development in is research: a literature review,” in *IRIS Selected Papers of the Information Systems Research Seminar in Scandinavia*, vol. 6, no. 6. IRIS New York, NY, USA, 2015.

[31] T. Mettler and P. Rohner, “Situational maturity models as instrumental artifacts for organizational design,” *http://www.alexandria.unisg.ch/Publikationen/67758*, 01 2009.

[32] R. Constantinescu and I. M. Iacob, “Capability maturity model integration,” *Journal of Applied Quantitative Methods*, vol. 2, no. 1, pp. 31–37, 2007.

[33] I. Marcovecchio, M. Thinyane, E. Estevez, and P. Fillottrani, “Capability maturity models as a means to standardize sustainable development goals indicators data production,” *Journal of ICT Standardization*, vol. 6, no. 3, pp. 217–244, 2018.

[34] F. Yucalar and S. Z. Erdogan, “A questionnaire based method for cmmi level 2 maturity assessment.” *Journal of Aeronautics & Space Technologies/Havacilik ve Uzay Teknolojileri Dergisi*, vol. 4, no. 2, 2009.

[35] A. Tarhan, O. Turetken, and H. A. Reijers, “Business process maturity models: A systematic literature review,” *Information and Software Technology*, vol. 75, pp. 122–134, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584916300015

[36] M. Khoshgoftar and O. Osman, “Comparison of maturity models,” in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 297–301.

[37] J. Iqbal, R. B. Ahmad, M. H. N. M. Nasir, M. Niazi, S. Shamshirband, and M. A. Noor, “Software smes’ unofficial readiness for cmmi®-based software process improvement,” *Software Quality Journal*, vol. 24, pp. 997–1023, 2016.

[38] M. Pikkarainen and A. Mäntyniemi, “An approach for using cmmi in agile software development assessments: experiences from three case studies,” in *SPICE 2006 conference, Luxemburg*, 2006, pp. 4–5.

[39] C. Kivunja, “Distinguishing between theory, theoretical framework, and conceptual framework: A systematic review of lessons from the field.” *International journal of higher education*, vol. 7, no. 6, pp. 44–53, 2018.

[40] C. Neuert, K. Meitinger, D. Behr, and M. Schonlau, “The use of open-ended questions in surveys,” *Methods, data, analyses: a journal for quantitative methods and survey methodology (mda)*, vol. 15, no. 1, pp. 3–6, 2021.

[41] N. Basias and Y. Pollalis, “Quantitative and qualitative research in business & technology: Justifying a suitable research methodology,” *Review of Integrative Business and Economics Research*, vol. 7, pp. 91–105, 2018.

[42] R. Carnwell and W. Daly, “Strategies for the construction of a critical review of the literature,” *Nurse education in practice*, vol. 1, no. 2, pp. 57–63, 2001.

[43] D. Proença, “Methods and techniques for maturity assessment,” in *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2016, pp. 1–4.

[44] C. Team, “Capability maturity model® integration (cmmi), version 1.1–continuous representation,” *Software Engineering Institute, Pittsburgh, PA*, 2002.

[45] N. Kerzazi, “Conceptual alignment between spem-based processes and cmmi,” in *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*. IEEE, 2015, pp. 1–9.

[46] J. Aguiar, R. Pereira, J. Braga Vasconcelos, and I. Bianchi, “An overlapless incident management maturity model for multi-framework assessment (itil, cobit, cmmi-svc),” *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 13, pp. 137–163, 2018.

[47] wibas, “Process management (cmmi-dev),” 2021, retrieved: 08/06/2023. [Online]. Available: https://www.wibas.com/cmmi/process-management-cmmi-dev

[48] ——, “Project management (cmmi-dev),” 2021, retrieved: 08/06/2023. [Online]. Available: https://www.wibas.com/cmmi/project-management-cmmi-dev

[49] ——, “Engineering (cmmi-dev),” 2021, retrieved: 08/06/2023. [Online]. Available: https://www.wibas.com/cmmi/engineering-cmmi-dev

[50] ——, “Support (cmmi-dev),” 2021, retrieved: 08/06/2023. [Online]. Available: https://www.wibas.com/cmmi/support-cmmi-dev

[51] C. G. von Wangenheim, A. von Wangenheim, F. McCaffery, J. C. R. Hauck, and L. Buglione, “Tailoring software process capability/maturity models for the health domain,” *Health and Technology*, vol. 3, no. 1, pp. 11–28, 2013.

[52] D. Ionita, C. van der Velden, H.-J. K. Ikkink, E. Neven, M. Daneva, and M. Kuipers, “Towards risk-driven security requirements management in agile software development,” in *Information Systems Engineering in Responsible Information Systems: CAiSE Forum 2019, Rome, Italy, June 3–7, 2019, Proceedings 31*. Springer, 2019, pp. 133–144.

[53] M. A. Ahmed, M. Anwar, and T. Hassan, “Requirements management processes; an insight into the rmp activities in software industry and literature (a case study),” 2023.

[54] A. A. Khan, J. Keung, M. Niazi, S. Hussain, and M. Shameem, “Gsepim: A roadmap for software process assessment and improvement in the domain of global software development,” *Journal of software: Evolution and Process*, vol. 31, no. 1, p. e1988, 2019.

[55] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis,” *arXiv preprint arXiv:1709.08439*, 2017.

[56] D. E. Strode, S. L. Huff, B. Hope, and S. Link, “Coordination in co-located agile software development projects,” *Journal of Systems and Software*, vol. 85, no. 6, pp. 1222–1238, 2012.

[57] S. Wagner and M. Ruhe, “A systematic review of productivity factors in software development,” *arXiv preprint arXiv:1801.06475*, 2018.

[58] S. Badshah, A. A. Khan, and B. Khan, “Towards process improvement in devops: a systematic literature review,” in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, 2020, pp. 427–433.

[59] M. Zarour, N. Alhammad, M. Alenezi, and K. Alsarayrah, “A research on devops maturity models,” *Int. J. Recent Technol. Eng*, vol. 8, no. 3, pp. 4854–4862, 2019.

[60] G. Rong, H. Zhang, and D. Shao, “Cmmi guided process improvement for devops projects: an exploratory case study,” in *Proceedings of the International Conference on Software and Systems Process*, 2016, pp. 76–85.

[61] A. Mishra and Z. Otaiwi, “Devops and software quality: A systematic mapping,” *Computer Science Review*, vol. 38, p. 100308, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013720304081

[62] T. L. Weissgerber, N. M. Milic, S. J. Winham, and V. D. Garovic, “Beyond bar and line graphs: time for a new data presentation paradigm,” *PLoS biology*, vol. 13, no. 4, p. e1002128, 2015.

[63] S. M. Ali, N. Gupta, G. K. Nayak, and R. K. Lenka, “Big data visualization: Tools and challenges,” in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016, pp. 656–660.

[64] H. Kerzner, *Project management metrics, KPIs, and dashboards: a guide to measuring and monitoring project performance*, 2017.

[65] ——, *Project management metrics, KPIs, and dashboards: a guide to measuring and monitoring project performance*. John Wiley & Sons, 2022.

[66] A.-L. Mattila, P. Ihantola, T. Kilamo, A. Luoto, M. Nurminen, and H. Väätäjä, “Software visualization today: Systematic literature review,” in *Proceedings of the 20th International Academic Mindtrek Conference*, 2016, pp. 262–271.

[67] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of time-oriented data*. Springer, 2011, vol. 4.

[68] A. Shamim, V. Balakrishnan, and M. Tahir, "Evaluation of opinion visualization techniques," *Information Visualization*, vol. 14, no. 4, pp. 339–358, 2015. [Online]. Available: https://doi.org/10.1177/1473871614550537

[69] M. Staron, K. Niesel, and W. Meding, "Selecting the right visualization of indicators and measures–dashboard selection model," in *Software Measurement: 25th International Workshop on Software Measurement and 10th International Conference on Software Process and Product Measurement, IWSM-Mensura 2015, Kraków, Poland, October 5-7, 2015, Proceedings 25*. Springer, 2015, pp. 130–143.

[70] A. Van Looy and A. Shafagatova, "Business process performance measurement: a structured literature review of indicators, measures and metrics," *SpringerPlus*, vol. 5, no. 1, pp. 1–24, 2016.

[71] D. Parmenter, *Key performance indicators: developing, implementing, and using winning KPIs*. John Wiley & Sons, 2015.

[72] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development – a systematic literature review of industrial studies," *Information and Software Technology*, vol. 62, pp. 143–163, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095058491500035X

[73] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, "Advances in using agile and lean processes for software development," in *Advances in computers*. Elsevier, 2019, vol. 113, pp. 135–224.

[74] A. Idri, F. azzahra Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Information and Software Technology*, vol. 58, pp. 206–230, 2015.

[75] M. K. Thota, F. H. Shajin, P. Rajesh *et al.*, "Survey on software defect prediction techniques," *International Journal of Applied Science and Engineering*, vol. 17, no. 4, pp. 331–344, 2020.

[76] W. N. Behutiye, P. Rodríguez, M. Oivo, and A. Tosun, "Analyzing the concept of technical debt in the context of agile software development: A systematic literature review," *Information and Software Technology*, vol. 82, pp. 139–158, 2017.

[77] P. Olivia H, "Development of a framework for achieving internal control and effectively managing risks in a devops environment," 2019.

[78] S. Tyagi, A. Choudhary, X. Cai, and K. Yang, “Value stream mapping to reduce the lead-time of a product development process,” *International Journal of Production Economics*, vol. 160, pp. 202–212, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527314003521

[79] W. Van Dooren, G. Bouckaert, and J. Halligan, *Performance management in the public sector*. Routledge, 2015.

[80] H. M. Alfaraj and S. Qin, “Operationalising cmmi: integrating cmmi and cobit perspective,” *Journal of Engineering, Design and Technology*, 2011.

[81] L. Linsbauer, A. Egyed, and R. E. Lopez-Herrejon, “A variability aware configuration management and revision control platform,” in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 803–806.

[82] B. Fitzgerald and K.-J. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[83] A. Tereso, P. Ribeiro, G. Fernandes, I. Loureiro, and M. Ferreira, “Project management practices in private organizations,” *Project Management Journal*, vol. 50, no. 1, pp. 6–22, 2019.

[84] A. Chahin and K. Paetzold, “Planning validation & verification steps according to the dependency of requirements and product architecture,” in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2018, pp. 1–6.

[85] T. W. Malone and K. Crowston, “The interdisciplinary study of coordination,” *ACM Computing Surveys (CSUR)*, vol. 26, no. 1, pp. 87–119, 1994.

[86] L. Dubé and D. Robey, “Software stories: three cultural perspectives on the organizational practices of software development,” *Accounting, Management and Information Technologies*, vol. 9, no. 4, pp. 223–259, 1999.

[87] P. Abrahamsson, A. Hanhineva, and J. Jäälinoja, “Improving business agility through technical solutions: A case study on test-driven development in mobile software development,” in *Business Agility and Information Technology Diffusion: IFIP TC8 WG 8.6 International Working Conference May 8–11, 2005, Atlanta, Georgia, USA*. Springer, 2005, pp. 227–243.

[88] K. Kakimoto, K. Sasaki, H. Umeda, and Y. Ueda, “Iv&v case: empirical study of software independent verification and validation based on safety case,” in *2017*

*IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2017, pp. 32–35.

[89] R. G. Cooper, "The drivers of success in new-product development," *Industrial Marketing Management*, vol. 76, pp. 36–47, 2019.

[90] L. Zhu, L. Bass, and G. Champlin-Scharff, "Devops and its practices," *IEEE software*, vol. 33, no. 3, pp. 32–34, 2016.

[91] M. Gokarna and R. Singh, "Devops: a historical review and future works," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2021, pp. 366–371.

[92] S. Saito, Y. Iimura, A. K. Massey, and A. I. Antón, "How much undocumented knowledge is there in agile software development?: Case study on industrial project using issue tracking system and version control system," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 194–203.

[93] M. Seiler and B. Paech, "Using tags to support feature management across issue tracking systems and version control systems: A research preview," in *Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27–March 2, 2017, Proceedings 23*. Springer, 2017, pp. 174–180.

[94] S. Bresciani and M. J. Eppler, "Choosing knowledge visualizations to augment cognition: The managers' view," in *2010 14th International Conference Information Visualisation*. IEEE, 2010, pp. 355–360.

[95] W. E. Lewis, *Software testing and continuous quality improvement*. CRC press, 2017.

[96] S. M. Mohammad, "Devops automation advances it sectors with the strategy of release management," *International Journal of Computer Trends and Technology (IJCTT)–Volume*, vol. 67, 2019.

[97] E. A. Locke and G. P. Latham, "Goal setting theory," in *Motivation: Theory and research*. Routledge, 2012, pp. 23–40.

[98] C. Andrade, "Internal, external, and ecological validity in research design, conduct, and evaluation," *Indian journal of psychological medicine*, vol. 40, no. 5, pp. 498–499, 2018.

# Appendix A

# Research Protocol: Literature Protocol

## A.1 Inclusion and Exclusion Criteria

| Inclusion Criteria | Exclusion Criteria |
|---|---|
| Papers about the structure of CMMI models. | Papers about maturity models outside the domain of software development. |
| Papers that investigate the implementation of a staged and continuous representation of CMMI. | Papers about other maturity models. |
| Papers about Team Practices in software development. | Papers of the dimensions that have very specific concepts not used in this thesis. |
| Papers about Requirements Management in software development. | Papers about data presentation techniques with no reflection on presentations used. |
| Papers about Software Development Practices. | Papers about software development metrics which require special access or permissions for data collection. |
| Papers about DevOps Practices. | |
| Papers about Quality Assurance Management. | |
| Papers about data presentation techniques. | |
| Papers about software development performance metrics. | |

**Table A.1:** Inclusion and Exclusion criteria.

## A.2 Keywords

To perform the literature review we searched on the key words found in Table A.2.

| Keywords | |
|---|---|
| CMMI model | Software Development Practices |
| CMMI in IT governance | DevOps Practices |
| CMMI staged representation | Agile Software Development |
| CMMI continuous representation | Quality Assurance Management |
| CMMI implementation | Data presentation techniques |
| CMMI in software development | Data visualization techniques |
| Team Practices in software development | Software development metrics |
| Requirements Management in software development. | Software development performance |

**Table A.2:** Keywords List

## A.3 Search Process

The search key for CMMI was:

*"CMMI model" AND ("Staged Representation" OR "Continuous Representation" OR "Implementation" OR "Representations" OR "IT Governance")*

For data presentation techniques the search key was:

*"Software development" AND "Improvement" AND "Performance" AND ("data presentation techniques" OR "Data Visualization techniques")*

| | CMMI model | Software Development Dimensions | Data Presentation | Software Development Metrics | Total |
|---|---|---|---|---|---|
| **Unique results** | 1,180 | 285 | 261 | 277 | **2,003** |
| **After filtering on title** | 38 | 41 | 25 | 26 | **130** |
| **After filtering on abstract** | 25 | 23 | 14 | 19 | **81** |
| **After filtering on full text** | 15 | 15 | 9 | 11 | **50** |

**Table A.3:** Search Results

# Appendix B

# Assessment Surveys Questions

| Category | Question | Maturity level |
|---|---|---|
| Backlog management | Do you estimate your work in backlog? | CMMI 3 |
| | Do you group work into epics in your backlog when necessary (i.e. large topics should be grouped into epic)? | CMMI 2 |
| | Do you have backlog in task management tool? | CMMI 2 |
| | Do you link all stories/epics in backlog with requirements? | CMMI 4 |
| | Do you link defects (i.e. bugs) with stories/epics/tasks if such relation exist? | CMMI 5 |
| | Do you refine backlog before each iteration? | CMMI 3 |
| | Do you review priorities in backlog in each iteration? | CMMI 5 |
| | Do you set acceptance criteria for all stories/epics in backlog? | CMMI 4 |
| | Do you set priority for all issues in backlog? | CMMI 4 |
| | Do you set single responsibility for all issues in backlog (one person is responsible and accountable for output)? | CMMI 4 |
| | Do you specify relation between issues if such exists? | CMMI 5 |
| | Do you use man-days to estimate work in backlog? | CMMI 3 |
| | Do you use poker planning for estimation? | CMMI 4 |
| | Do you use story points to estimate work in backlog? | CMMI 4 |
| Documentation management | Do you document all business requirements in confluence? | CMMI 2 |
| | Do you have non-functional requirements defined for Your project/team? | CMMI 3 |
| | Do you include other specific non-functional requirements for each story/epic (i.e. non-functional reqs that are only valid for particular story/epic and they are not generic)? | CMMI 4 |
| | Do you keep business requirements up to date? | CMMI 4 |
| | Do you use any formal language to describe your business requirements (i.e. BDD, sequence diagrams, etc)? | CMMI 5 |

**Table B.1:** Requirements Management Survey Questions.

| Category | Question | Maturity level |
|---|---|---|
| Roles and team composition | Do you have defined team leader role in your team? | CMMI 2 |
| | Do you have onboarding buddy (mentor) for each new comer in your team? | CMMI 3 |
| | Do you have defined role of devops champion in your team? | CMMI 2 |
| | Do you have a Security Champion assigned in your team? | CMMI 3 |
| | Do you have a dedicated scrum master? | CMMI 4 |
| | Do you have a dedicated product owner? | CMMI 4 |
| | Do you have people rotation plan defined for incident management? | CMMI 3 |
| | Do you have process defining deputies in case people are absent? | CMMI 5 |
| Artefacts & definitions | Do you have documented Definition of Ready? | CMMI 2 |
| | Do you validate each user story against your Definition of Ready? | CMMI 5 |
| | Do you keep Definition of Ready up-to-date? | CMMI 4 |
| | Do you have documented Definition of Done? | CMMI 2 |
| | Do you validate each user story against your Definition of Done? | CMMI 5 |
| | Do you keep Definition of Done up-to-date? | CMMI 4 |
| | Does your Definition of Done include criteria that testing is done and at least all critical defects are fixed and verified? | CMMI 3 |
| | Do you track all team work (tasks, defects, user stories, improvements, spikes) in a task management tool (i.e. jira)? | CMMI 2 |
| | Do you estimate all work that the team committed to? | CMMI 3 |
| | Do you track progress of items on a board (might be physical board or jira one)? | CMMI 3 |
| | Do you track blockers on your board (special status, flags, blocked by task relationships etc.)? | CMMI 4 |
| | Do you track release scope for each iteration (are you able to explain at any point in time what is your plan for next release)? | CMMI 3 |
| | Do you use versioning for releases? | CMMI 3 |
| | Do you track releases in a task management tool (i.e. do you use releases feature in jira)? | CMMI 4 |
| | Do you link relevant team work (tasks, issues, etc.) with particular release in jira? | CMMI 4 |
| | Do you perform a production release at the end of every iteration? | CMMI 5 |
| | Do you have a guide or handbook for on-boarding newcomers (i.e. project materials to be read, introduction to technology, etc...)? | CMMI 2 |
| | Do you have a guide or handbook that contains information about setting up development environment to work? | CMMI 2 |
| | Do you update frequently your guide or handbook so it's always reflecting current project situation? | CMMI 4 |
| Events | Do you have daily meetings? | CMMI 2 |
| | Do you include qa members in daily meetings when required? | CMMI 3 |
| | Do you include product stakeholders in daily meetings when required? | CMMI 4 |
| | Do you always use a board to faciliate daily meetings? | CMMI 4 |
| | Do you always perform planning meeting for iteration (i.e. sprint planning session)? | CMMI 2 |
| | Do you have review meetings for each iteration (i.e. a meeting with stakeholders to discuss output from sprint)? | CMMI 3 |
| | Do you have demo meetings for each iteration (i.e. a meeting where you present new API, new features, etc)? | CMMI 5 |
| | Do you have retrospective meetings for each iteration? | CMMI 4 |
| | Do you plan iteration in accordance to team capacity (member leaves etc.. . . )? | CMMI 3 |
| | Do you include key stakeholders in planning meetings? | CMMI 3 |
| | Do you always include qa members in planning meetings? | CMMI 4 |
| | Do you include bug fixing when planning capacity for iteration? | CMMI 5 |
| | Do you use specific tool for retrospectives? | CMMI 3 |
| | Do you track retrospective improvements? | CMMI 4 |
| | Do you execute at least 2 improvements from each retrospectives during the next iteration? | CMMI 5 |
| | Are key stakeholders reviewing iteration output and providing feedback to the team? | CMMI 5 |
| Collaboration | Do you have dedicated communication channel(s) for the team (i.e. slack, teams; can be multiple channels as well)? | CMMI 2 |
| | Do you have external channel(s) for communication so other teams can reach you at any point of time? | CMMI 3 |
| | Do you have a process defined for communicating impediments to other teams (i.e. escalation via Project Management Organization (PMO), Business Analysts (BA), etc. to report and discuss impediments with other teams)? | CMMI 4 |
| | Do you actively monitor your dependencies with other teams if such were discovered (i.e. have dedicated person that tracks the status, sync-ups, discussing deadlines, etc.)? | CMMI 4 |
| | Do you specify and communicate risks to stakeholders that are related with impediments that were discovered? | CMMI 5 |
| Continuous learning | Do you as a team leader create list of training needs for all your team members at least on a yearly basis? | CMMI 4 |
| | Do you create personal development plan (grow plan, succession plan) for all team members? | CMMI 4 |
| | Do you actively track personal development plan for all team members (i.e. regular goals evaluation, helping with materials, answering questions, etc)? | CMMI 5 |
| | Do you evaluate regularly products and services that are used by the team (i.e. new frameworks, new libraries, new development approach or architecture patterns)? | CMMI 5 |

**Table B.2:** Team Practices Survey Questions.

| Category | Question | Maturity level |
|---|---|---|
| Knowledge sharing | Do you have knowledge base (e.g. wiki, confluence)? | CMMI 2 |
| | Does knowledge base have an up-to-date dedicated section for software related documentation (i.e. info about software we write, purpose, etc.)? | CMMI 3 |
| | Does knowledge base have an up-to-date dedicated section for architecture documentation (i.e. deployment diagrams, component diagrams, etc.)? | CMMI 4 |
| | Does knowledge base have an up-to-date dedicated section for coding standards, best practices, Dos, Don'ts, etc.? | CMMI 4 |
| | Do you use pair programming? | CMMI 3 |
| | Do you distribute tasks between developers in a way they can learn all areas of the system? | CMMI 4 |
| | Do you have practice of mentoring inside of the team? | CMMI 5 |
| Code quality management | Do you have coding standards as a set of guidelines, best practices, programming styles and conventions that all developers adhere to when writing source code? | CMMI 2 |
| | Do you enforce coding standards automatically with tool? | CMMI 3 |
| | Do you use static code analysis tools? | CMMI 4 |
| | Do you measure code quality metrics (e.g. cyclomatic complexity, coupling, class hierarchy, code duplications, method cohesion)? | CMMI 5 |
| | Do you have quality metric targets to identify whether code quality is on an accepted level? | CMMI 4 |
| Code review management | Are you in a position to quickly assess that particular code committed to repo is linked with task in Jira (i.e. by adding task number to commit, name feature branches with task numbers, etc.)? | CMMI 3 |
| | Do you have commit message template defined? | CMMI 2 |
| | Do you enforce commit message standards with automated tool? | CMMI 4 |
| | Do you practice mandatory peer code review on Pull/Merge Request basis? | CMMI 2 |
| | Do you have code review strategy, which identify how to choose and assign reviewers? | CMMI 3 |
| | Do you have a code review checklist as a guidance what must be reviewed? | CMMI 3 |
| | Do you validate that the team follows coding standards as part of code review process? | CMMI 2 |
| | Do you validate security aspects as part of code review process? | CMMI 4 |
| | Do you validate business logic as part of code review process? | CMMI 3 |
| | Do you validate unit tests for changed logic as a part of code review process? | CMMI 3 |
| | Do you measure technical debt (e.g. code smells) using static code analysis tools? | CMMI 4 |
| Technical debt management | Do you regularly allocate time to address technical debt found by static code analysis (e.g. every iteration, every several iterations)? | CMMI 5 |
| | Do you track intentional technical debt (reasoned architecture and development short-cuts to meet project deadlines)? | CMMI 4 |
| | Do you estimate your intentional technical debt? | CMMI 4 |
| | Do you regularly allocate time to reduce intentional technical debt? | CMMI 5 |
| Unit testing | Do you create unit tests as a part of working on development task? | CMMI 2 |
| | Do you create unit tests for part of code that was fixed? | CMMI 4 |
| | Do you measure code coverage for unit tests? | CMMI 3 |
| | Do you have unit test coverage below 50%? | CMMI 2 |
| | Do you have unit test coverage between 50 and 75%? | CMMI 3 |
| | Do you have unit test coverage between 76 and 90%? | CMMI 4 |
| | Do you have unit test coverage above 90%? | CMMI 5 |
| Secure development | Do you perform regular static security scan? | CMMI 3 |
| | Do you perform regular dynamic security scan? | CMMI 3 |
| | Do you have SLAs for fixing security issues? | CMMI 3 |
| | Do you have Security Quality Gates that can stop the release? | CMMI 4 |
| | Do you have a list of security coding practices adopted on the project and published? | CMMI 5 |
| | Do you have a backlog of security issues tracked in bug tracking system? | CMMI 4 |
| | Do you scramble production data (remove sensitive info and names) when creating a test data? | CMMI 3 |
| | Do you obfuscate your code? | CMMI 2 |
| Architecture foundations | Do you perform architecture review as part of preparation for implementation (i.e. checking if services are following standards, if the communication protocol is following standards, etc.)? | CMMI 3 |
| | Do you have architecture practices/patterns defined in your team (i.e. you follow IOC, synchronous vs asynchronous communication, using MVP, MVVM, MVC etc.)? | CMMI 3 |
| | Do you build your solution in a way it has cloud native architecture? | CMMI 4 |
| | Do you engage your team in dedicated architectural activities (review of new API standards, etc)? | CMMI 5 |
| | Are you aware of what is the desired enterprise architecture in the company? | CMMI 3 |
| | Is your project compliant with desired enterprise architecture? | CMMI 5 |
| | Is your project at the minimum supported language version and framework that is defined by company policy? | CMMI 2 |
| | Do you feel that your project is at desired supported language version and framework and no upgrades are needed? | CMMI 3 |

**Table B.3:** Development Practices Survey Questions.

| Category | Question | Maturity level |
|---|---|---|
| Automation | Do you have test automation strategy that defines what test scope will be automated to what degree, when, by whom, by which methods, by what test tools, in what kind of environment? | CMMI 3 |
| | Do you have guidelines on how to design and execute automated tests? | CMMI 3 |
| | Do you include coding standards and test-data handling methods in automated test guidelines? | CMMI 4 |
| | Do you include processes for reporting and storing test results in automated test guidelines? | CMMI 4 |
| | Do you include general rules for test tool usage and/or information on how to access external resources in automated test guidelines? | CMMI 4 |
| | Do you cover all critical areas defined by business with automated tests? | CMMI 3 |
| | Do you achieve pass rate of over 95% for valid cases with automated tests? | CMMI 3 |
| | Do you prioritise test cases for the automation in order to meet the test execution schedule? | CMMI 4 |
| | Do you create automation tests in parallel to the relevant development cycle? | CMMI 4 |
| | Do you track status and progress of automation testing in task management tool? | CMMI 3 |
| | Do you track all the test executions? | CMMI 3 |
| | Do you generate report for different test results to create a big picture of automation testing? | CMMI 4 |
| | Do you regularly review critical areas for auto tests coverage? | CMMI 5 |
| Defect management | Do you have defined defect lifecycle (there are step by step rules how we handle every defect including jira workflow)? | CMMI 2 |
| | Are all defects following the same lifecycle? | CMMI 3 |
| | Do you assign priority to every single defect? | CMMI 3 |
| | Do you assign severity to every single defect? | CMMI 3 |
| | Do you include information about the subsystem, program and version, root cause in every single defect? | CMMI 4 |
| | Do you analyse defects for common properties and make recommendations to avoid same defect in future? (CAPA) | CMMI 5 |
| | Do you include leaked defects (i.e. defects leaked to production, not discovered before prod) into regression suite? | CMMI 5 |
| Reporting | Do you create Bi-weekly testing reports? | CMMI 2 |
| | Do you receive QA Sign off in written form for releases? | CMMI 3 |
| | Do you include test progress, test results and product risks in the bi-weekly reporting? | CMMI 2 |
| | Do you include metrics in bi-weekly reporting? | CMMI 3 |
| | Do you have templates defined for all QA reports? | CMMI 3 |
| | Do you regularly review QA reports in terms of informational value, transparency and easy to use characteristics? | CMMI 4 |
| Test design | Do you create test cases in a way they are directly reflecting all requirements? | CMMI 2 |
| | Do you document test cases? | CMMI 2 |
| | Do you create test cases according to company standards, including steps to reproduce, test data, expected result? | CMMI 3 |
| | Do you use formal test design techniques to design test cases? | CMMI 3 |
| | Are test cases checked and evaluated independently on validity and maintainability by QA team? | CMMI 3 |
| | Do you use test cases peer review process? | CMMI 4 |
| | Do you trace/link test cases with requirements? | CMMI 4 |
| | Do you evaluate and adjust test design techniques for further re-use? | CMMI 5 |
| Test organization | Do you have dedicated QA resources in your team? | CMMI 3 |
| | Do you know as a team who is responsible for testing the part of the system you are working on? | CMMI 2 |
| | Do you know the scope and approach for test activities in your team? | CMMI 2 |
| | Do you include QAs in assessing the impact and risk analysis of change requests and defects? | CMMI 2 |
| | Do you track QA tasks in task management tool with corresponding links to related sources? | CMMI 3 |
| | Do you have QA documentation standards defined and introduced to team? | CMMI 3 |
| Testing strategy | Do you have test strategy defined and documented? | CMMI 2 |
| | Do you consider product risk analysis when creating test strategy? | CMMI 2 |
| | Do you define test levels and test types according to analysed risks? | CMMI 4 |
| | Do you define test coverage and test depth according to analysed risks? | CMMI 4 |
| | Do you include test design techniques in test strategy? | CMMI 3 |
| | Do you evaluate test strategy regularly and, where necessary, adapted for future use according to business needs? | CMMI 3 |
| | Do you create testing data in a way is the same or similar to production data? | CMMI 4 |
| | Do you create snapshot of production like data? | CMMI 3 |
| | Do you keep your production snapshot data up to date (i.e. frequently and when extra data is needed)? | CMMI 4 |
| | Do you regularly evaluate test strategy against metrics from incidents that occurred in production? | CMMI 4 |
| | Are all functional tests automated? | CMMI 5 |
| | Are all non-functional tests automated (i.e. performance, accessibility, security, etc)? | CMMI 5 |

**Table B.4:** Quality Assurance Management Survey Questions.

| Category | Question | Maturity level |
|---|---|---|
| Access management | Do you restrict access to sensitive information (i.e. artefacts, api keys, private keys, usernames and passwords)? | CMMI 2 |
| | Do you use tools dedicated to store sensitive information (i.e. azure key storage, AWS kms)? | CMMI 3 |
| | Do you have roles defined in code repository? | CMMI 2 |
| | Do you have roles defined in CI/CD tool? | CMMI 2 |
| | Do you restrict access to software artifactory? (i.e. docker repository, etc) | CMMI 2 |
| Continuous Integration/ Continuous Development | Do you use centralized continuous integration tool? | CMMI 2 |
| | Do you use centralised continuous delivery tool? | CMMI 2 |
| | Are your builds triggered automatically upon pull request? | CMMI 3 |
| | Do you have CD pipelines defined for lower environments (DEV, TEST)? | CMMI 3 |
| | Do you have CD pipelines defined for higher environments (STAGING, PROD)? | CMMI 4 |
| | Are unit tests executed automatically as part of the CI pipeline? | CMMI 3 |
| | Are integration tests executed automatically as part of the CD pipeline? | CMMI 4 |
| | Are security tests executed automatically as part of CD pipeline? | CMMI 5 |
| | Is CI pipeline blocked in case of unit test failure? | CMMI 2 |
| | Is CI pipeline blocked in case of static code analysis error (criteria is met also if this is part of linking orcompilation phase)? | CMMI 3 |
| | Is CI pipeline blocked in case of a security test (i.e. twistlock, owasp) error? | CMMI 4 |
| Incident management | Do you have alerting escalation policies defined? | CMMI 3 |
| | Do you have primary and secondary contact defined for each escalation policy? | CMMI 4 |
| | Do you use a dedicated tool for defining escalation policies (i.e. PagerDuty)? | CMMI 4 |
| | Do you have runbook defined in case alert policy would be triggered? | CMMI 4 |
| | Do you have a process for keeping your runbooks up to date? | CMMI 5 |
| | Do you have SLAs in place for responding to incidents? | CMMI 2 |
| Infrastructure management | Do you create infrastructure for lower environments via automation scripts? | CMMI 3 |
| | Do you create infrastructure for higher environments via automation scripts? | CMMI 3 |
| | Do you store infrastructure management scripts in repository? | CMMI 3 |
| | Do you have 4 separate environments set-up for your system (Development/ Test/Staging/Prod)? | CMMI 4 |
| Monitoring and alerting | Do you use tool for automated logs collection (i.e. elastic, kibana, etc.)? | CMMI 2 |
| | Do you use tool for infrastructure and application monitoring? | CMMI 3 |
| | Do you have alert policies defined for infrastructure? | CMMI 3 |
| | Do you have alert policies defined for errors in the app (i.e. 5xx issues)? | CMMI 3 |
| | Do you have alert policies defined for application performance monitoring (CPU, memory, disk usage, etc.)? | CMMI 3 |
| Release management | Do you have rollback strategy defined and written? | CMMI 2 |
| | Did you test your rollback strategy? | CMMI 3 |
| | Do you have release readiness check process defined (i.e. checklist that you are going through before making decision about release)? | CMMI 3 |
| | Do you have notification policy for your release process (i.e. email to business, other teams, etc)? | CMMI 3 |
| | Do you store release artefacts in artefact repository or are in a position to deploy any version of your software at any point of time without unnecessary delays (i.e. nexus, firebase, etc.)? | CMMI 2 |
| | Do you use release notes for each release? | CMMI 3 |

**Table B.5:** DevOps Practices Survey Questions.

# Appendix C

# Assessment Survey Results Analysis

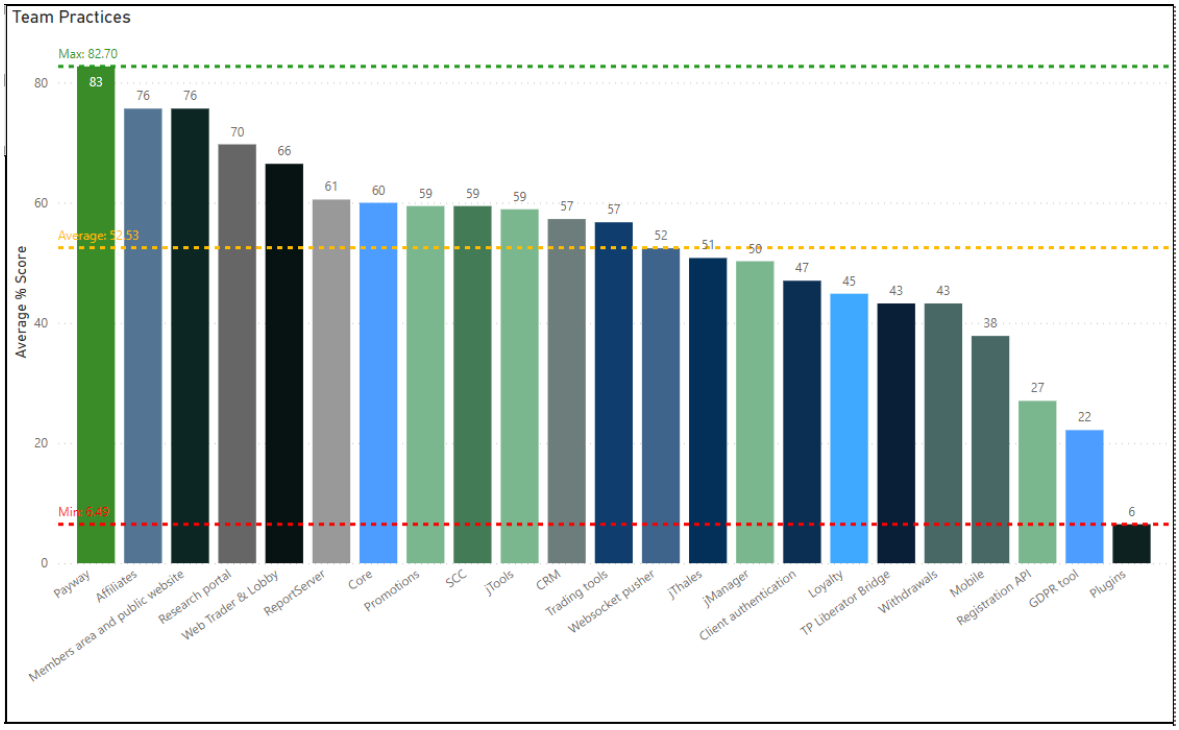



**Figure C.1:** Team Performance per Dimension (Team Practices).

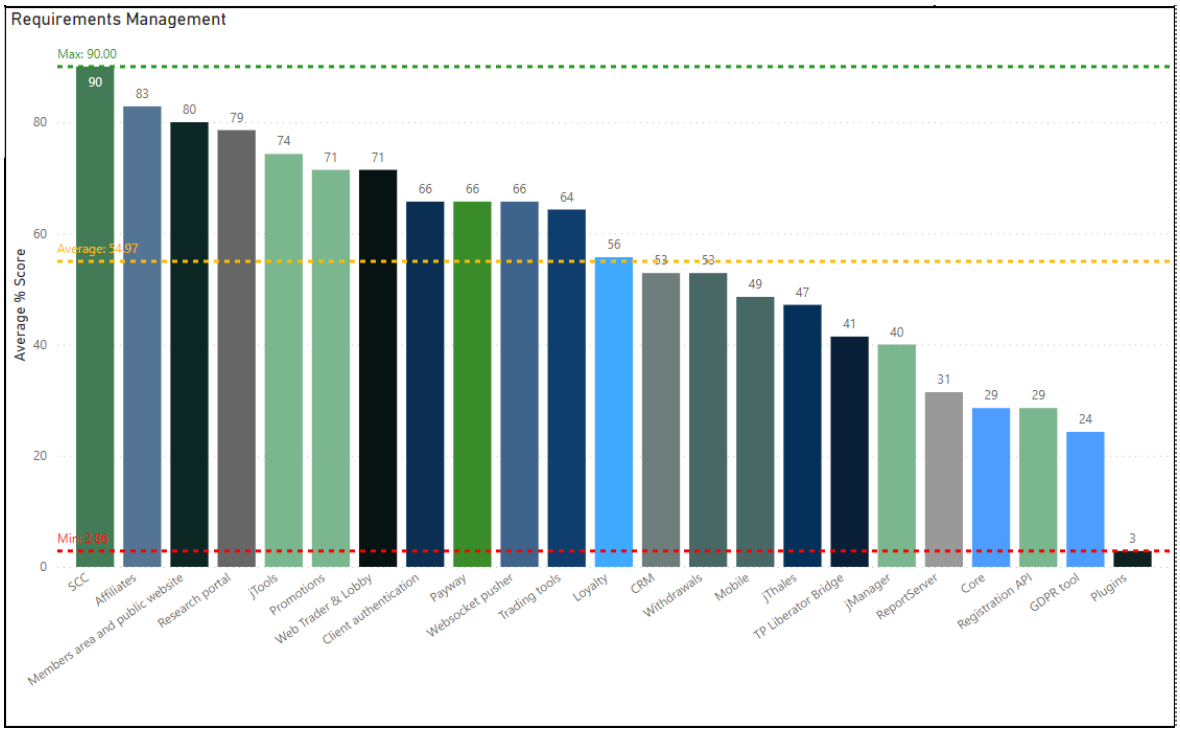



**Figure C.2:** Team Performance per Dimension (Requirements Management).

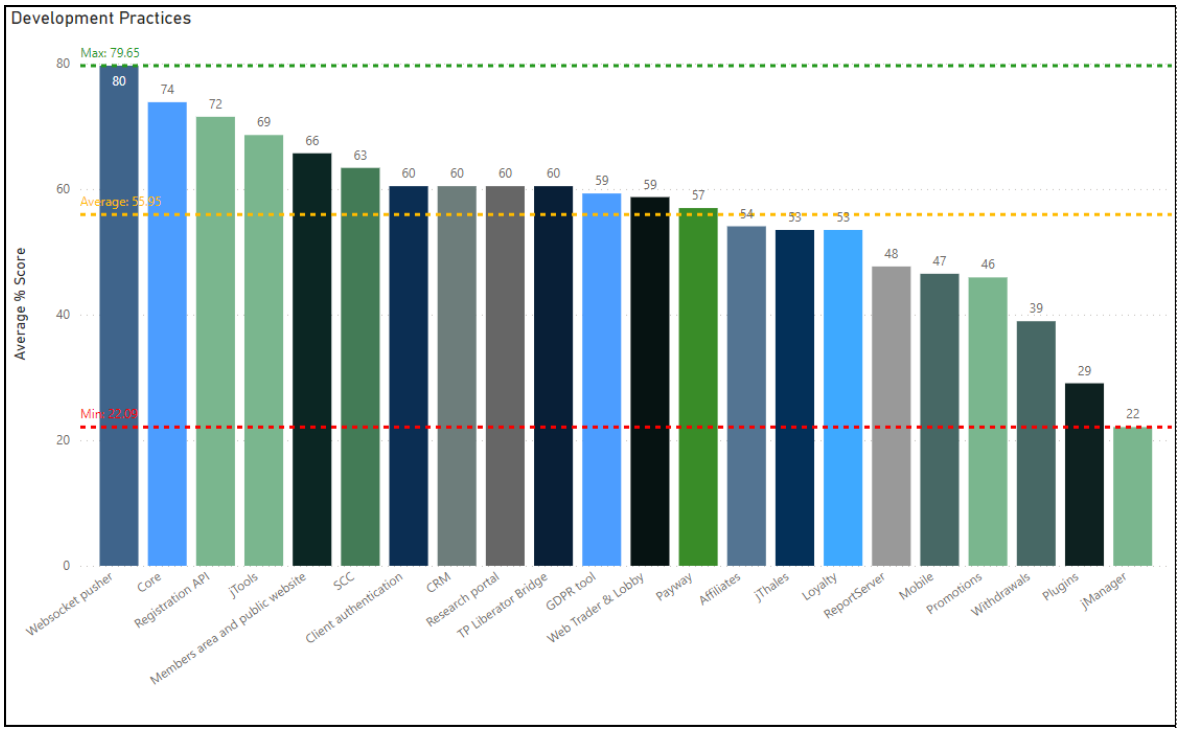



**Figure C.3:** Team Performance per Dimension (Development Practices).

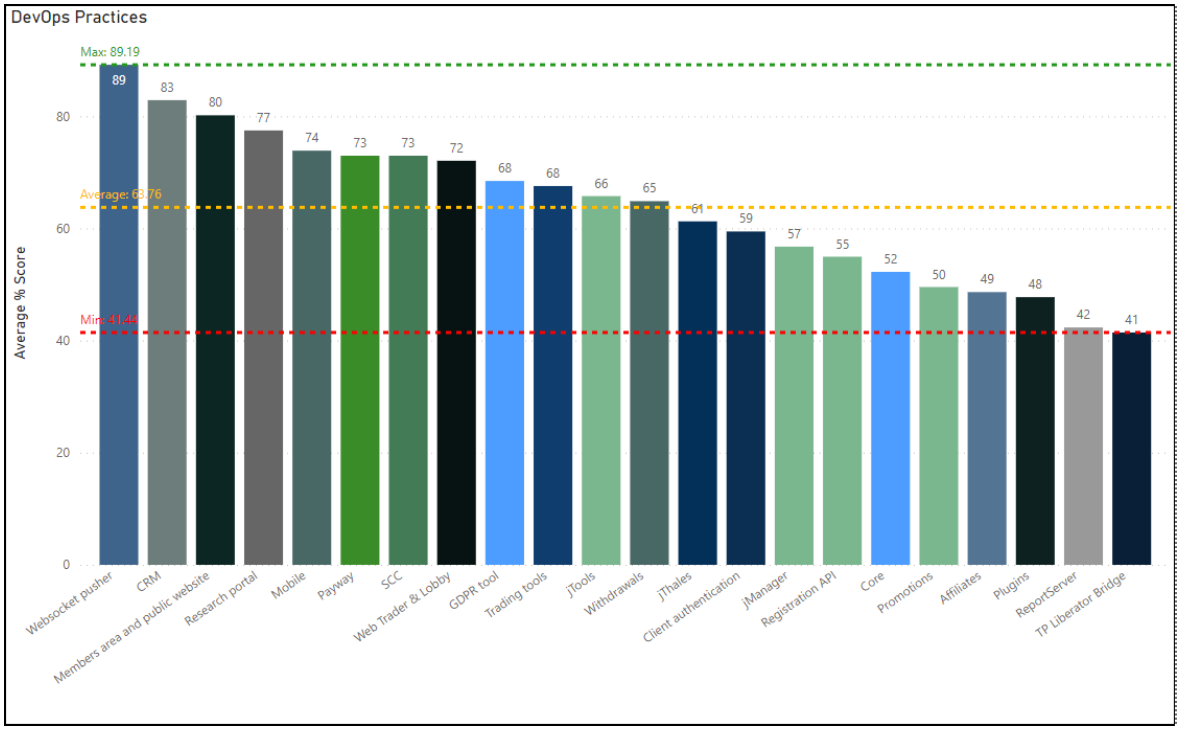



**Figure C.4:** Team Performance per Dimension (DevOps Practices).

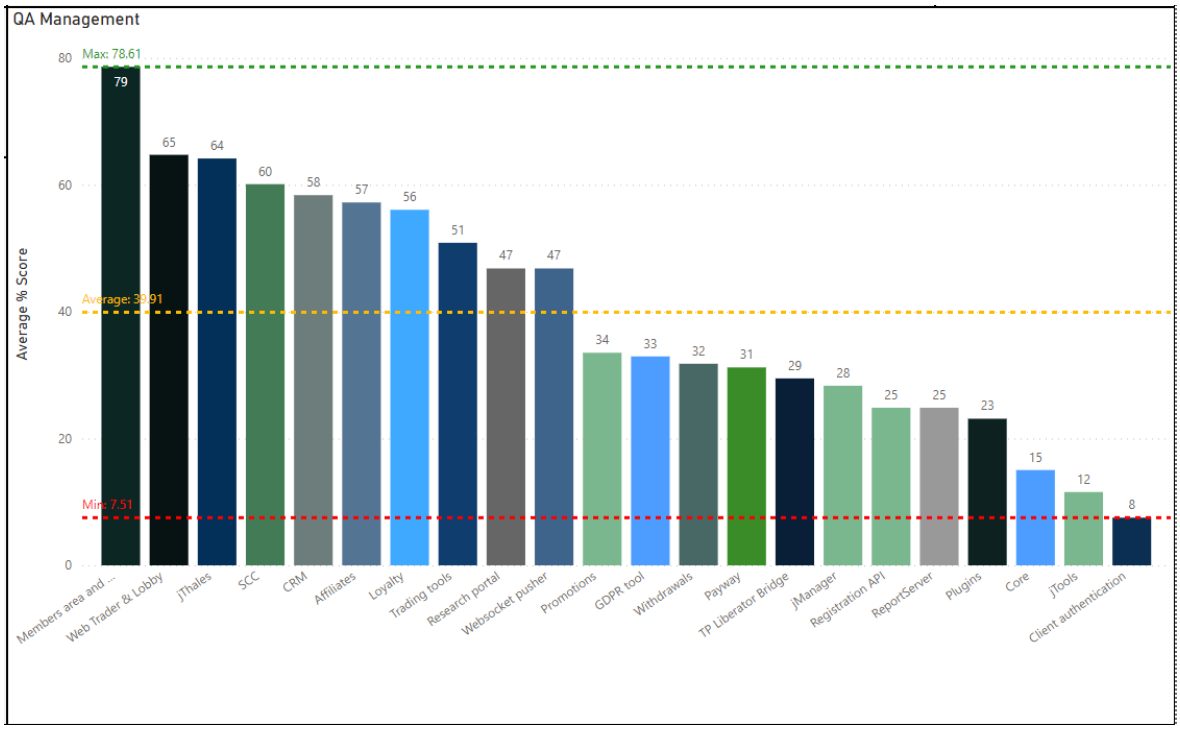



**Figure C.5:** Team Performance per Dimension (Quality Assurance Management).