Scaled Agile Maturity Model

MSc Thesis Revathi M Chandrasekaran

August 2016
Scaled Agile maturity model

Thesis submitted to
Delft University of Technology

In partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

In System Engineering Policy Analysis and Management
Faculty of Technology, Policy and Management

By
Revathi M Chandrasekaran
Student Number: 4385276
COHORT: 2014-2016
Date: 17th August, 2016

Graduation Committee
Chair: Prof. Dr. Ir. Marijn Janssen, TU Delft
First Supervisor: Asst. Prof. Drs. Jolien Ubacht, TU Delft
Second Supervisor: Assoc. Prof. Dr. Martijn Warnier, TU Delft
External Supervisors: Ms. Yasmina Beithia, Mr. Pascal Kersic, TomTom Group, The Netherlands.
This page intentionally left blank.
Preface

This thesis report written in the period of March-August 2016 is the final facet of my master studies and concludes my MSc education in System Engineering Policy Analysis and Management at the Faculty of Technology, Policy and Management, TU Delft. It has been a challenging period of extensive learning not only in the academic arena, but also on a personal level. I would like to take this opportunity and thank all those who have supported my research in different occasions and made my research experience cherish forever.

I would like to express my sincere gratitude to my first supervisor Jolien Ubacht who helped me in shaping my ideas towards a clear goal. Starting point of the thesis represented a challenging situation when the scope was unclear with a broad research context. She believed in my ideas and provided flexibility in building up the research according to my interests. Her approach towards partnering in learning and open mindset towards all perspectives demonstrated her perfect role as a coach. Furthermore, I am grateful to my chair, Professor Marijn Janssen and Second supervisor Martijn Warnier for their constructive feedbacks and support during our meetings. And I would like to show my appreciation to entire academic group of TU Delft in helping me achieve great heights.

I would also like to show my deepest gratitude to my TomTom mentors Pascal Kersic and Yasmina Beithia in demonstrating a positive attitude and taking personal interests towards my work, during my internship period. Even before the start of internship, they extended constant support and guidance in analyzing the problem and research context. They constantly appreciated my ongoing effort and empowered me through constructive feedback and reinforcement. They extended willingness to go out of their comfort zone in different occasions, in spite of their busy schedules. This research would have not been possible without their support. I have had many chances to work in closer collaboration with Pascal, who acted as a positive role model in demonstrating specific behaviors and actions required, to be successful in the professional world.

I would like to thank the experts who helped me in giving constructive inputs during the interview and expert evaluation phase which was extremely helpful in finalizing my design. I would also like to thank the rest of my TomTom colleagues who made my stay in Drive team such a pleasant experience.

Finally, my deepest thanks must to be my family and friends who supported me with confidence and motivation in the most crucial moments of the journey. Saving the most important to the last, I wish to give my heartfelt thanks to my beloved husband, whose continual support in my academic endeavors helped me to complete this thesis.

Revathi M Chandrasekaran
August 2016
ABSTRACT

In today’s world agile software development has been embraced more and more in software service industry. Though the agile practices have gained widespread popularity in the recent years, there are quite a number of concerns in scaling the agile practices from team level to the entire enterprise. Few frameworks such as Scaled Agile Framework (SAFe, Disciplined Agile Delivery (DAD) and Large Scale Scrum (LeSS) have been developed to address these concerns in scaling agile practices. Although these frameworks provide a template in scaling agile in large enterprises, currently there is a lack of a holistic method which would help them in implementing scaled agile practices or adapting to scaled agile software development. Before or after adopting such a framework, organizations require a structured model for assessing the level of completeness of adoption or find areas of improvements in the scaled agile practices, which would also help them in developing a roadmap for further progress and initiatives.

This research offer guidance for the IT organizations towards scaled agile software development by providing a maturity model. This maturity model is composed of six stages as rows and five scaled agile principles in columns, in which each stage and column forms a matrix of scaled agile practices. Each of the practices consists of indicators which help in assessing the level of adoption of the practices. Once having identified the lacking criteria in the adoption of scaled agile practices, organizations can start focusing on the lacking criteria and other areas of improvement. The research also strongly suggests the collaborative spirit in adaptation of scaled agile practices, by using this model as a discussion tool in the team, program and portfolio levels of scaled agile environment. Future research in this arena would aim at researching on the dynamics of emergence on scaled agile practices and the notion of such an emergence on multitude of stakeholders involved in a scaled agile process.

Key words: Scaled agile framework, Agile software development, Maturity model, Scaled agile practices, ambidexterity.
This page intentionally left blank.
# Table of Contents

ABSTRACT ................................................................................................................................. i

List of tables .................................................................................................................................. 7

List of figures ................................................................................................................................. 8

Chapter 1 Introduction .................................................................................................................. 10
  1.1 Introduction ............................................................................................................................ 10
  1.2 Research context – Scaled agile software development ......................................................... 10
  1.3 Problem at stake .................................................................................................................... 11
  1.4 Study relevance ..................................................................................................................... 12
    1.4.1 Societal relevance ........................................................................................................... 12
    1.4.2 Scientific relevance ....................................................................................................... 13
  1.5 Research Questions ............................................................................................................... 13
    1.5.1 Main research question ................................................................................................. 13
    1.5.2 Sub research questions ................................................................................................. 13
  1.6 Thesis outline and research planning .................................................................................... 15
  1.7 Delineations .......................................................................................................................... 17

Chapter 2 Agile and scaled agile software development ............................................................ 19
  2.1 Literature review method ....................................................................................................... 19
  2.2 Agile software development .................................................................................................. 19
    2.2.1 Software development .................................................................................................... 20
    2.2.2 Need for agile software development ............................................................................. 20
    2.2.3 Agile principles and values ........................................................................................... 23
    2.2.4 Scrum process ............................................................................................................... 24
    2.2.5 Need for scaled agile methodologies ............................................................................. 26
  2.3 Scaled agile methodologies .................................................................................................... 26
    2.3.1 Sealing software development ......................................................................................... 27
    2.3.2 Scaled agile software development ............................................................................... 27
    2.3.3 Scaled agile principles and core values ......................................................................... 33
  2.4 Chapter summary .................................................................................................................. 34
Chapter 3 Maturity models of agile software development ................................................................. 36
3.1 Introduction ................................................................................................................................. 36
3.2 Maturity model ........................................................................................................................ 36
   3.2.1 Definition and concepts ......................................................................................................... 36
   3.2.2 Selection criteria .................................................................................................................. 38
3.3 Exploration on maturity models .............................................................................................. 38
   3.3.1 Capability maturity model integration for development (CMMI-DEV)................................. 38
   3.3.2 Agile maturity model (AMM) ............................................................................................... 40
   3.3.3 ISO/IEC 15504 – SPICE ................................................................................................... 43
   3.3.4 Scrum Maturity model ......................................................................................................... 44
   3.3.5 Agile adoption framework .................................................................................................. 45
   3.3.6 Robert Benefield’s model .................................................................................................. 46
3.4 Characteristics of agile maturity models .................................................................................... 49
3.5 Requirements of scaled agile maturity model ........................................................................... 51
3.5 Chapter summary ....................................................................................................................... 52
Chapter 4 Design and Development of Scaled agile maturity model ........................................... 54
4.1 Introduction ................................................................................................................................. 54
4.2 Translation of requirements into the model .............................................................................. 54
4.3 Design pattern of the model ...................................................................................................... 55
   4.3.1 Stages of the maturity models ............................................................................................. 57
   4.3.2 Agile/Scaled agile principles ............................................................................................... 61
   4.3.3 Agile and Scaled agile practices ......................................................................................... 66
   4.2.4 Initial design of Scaled agile maturity model ................................................................. 76
   4.2.6 Participatory observation ................................................................................................. 77
   4.2.6 Assessment criteria .......................................................................................................... 81
4.3 Chapter summary ....................................................................................................................... 81
Chapter 5 Evaluation of the scaled agile maturity model .............................................................. 83
5.1 Introduction ................................................................................................................................. 83
5.2 Evaluation approach .................................................................................................................. 83
   5.2.1 Type of evaluation used in expert interviews .................................................................... 83
   5.2.2 Expert selection method ................................................................................................. 84
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2 Emphasis on people</td>
<td>133</td>
</tr>
<tr>
<td>5.3.3 Emphasis on project</td>
<td>133</td>
</tr>
<tr>
<td>6 Conclusion</td>
<td>133</td>
</tr>
<tr>
<td>References</td>
<td>134</td>
</tr>
</tbody>
</table>
List of tables

Table 1 Principles of agile software development (Highsmith & Cockburn, 2001) .................................. 11
Table 2 Differences between traditional and agile software development methods ............................... 22
Table 3 Definition of software development agility in literature .......................................................... 22
Table 4 Core values of agile methods (Highsmith & Cockburn, 2001) .................................................. 23
Table 5 Principles of SAFe ...................................................................................................................... 33
Table 6 Translation of requirements into the model ................................................................................. 54
Table 7 Mapping of core values ............................................................................................................. 58
Table 8 Final mapping of core values .................................................................................................... 60
Table 9 Principles used in maturity models ............................................................................................ 62
Table 10 Initial design of maturity model ............................................................................................... 65
Table 11 Practices involved in studied maturity models .......................................................................... 67
Table 12 Practices involved in stage 1 .................................................................................................... 69
Table 13 Practices involved in stage 2 .................................................................................................... 70
Table 14 Practices involved in stage 3 .................................................................................................... 71
Table 15 Practices involved in stage 4 .................................................................................................... 73
Table 16 Practices involved in stage 5 .................................................................................................... 74
Table 17 Initial design of the model ........................................................................................................ 76
Table 18 Initial design after participant observation .............................................................................. 80
Table 19 Matrix of experts on technical and process dimensions ......................................................... 85
Table 20 Evaluation criteria .................................................................................................................... 86
Table 21 Expert evaluation results .......................................................................................................... 87
Table 22 Suggestion for improvement .................................................................................................... 91
Table 23 Adaptation of practices in Stage 2 ........................................................................................... 93
Table 24 Scaled agile maturity model design after expert evaluation .................................................... 96
Table 25 Requirements of the design ..................................................................................................... 100
Table 26 Initial design ............................................................................................................................ 102
Table 27 Criticisms on the model ........................................................................................................... 104
Table 28 Expert evaluation interview probes .......................................................................................... 120
Table 29 Indicators for the assessment criteria ....................................................................................... 121
List of figures
Figure 1 Research design flow.......................................................... 16
Figure 2 Phases of traditional project management methods(Palmquist, Lapham, Miller, Chick, & Ozkaya, 2013) ..................................................... 20
Figure 3 Scrum process (Hicks & Foster,p.3,2010)................................... 25
Figure 4 Product and Platform mode of team scaling.............................. 27
Figure 5 SAFe Big picture framework (Leffingwell, 2011)....................... 28
Figure 6 Team, program and portfolio levels....................................... 29
Figure 7 Portfolio level events.............................................................. 30
Figure 8 Different forms of ART (Leffingwell, 2011)............................... 31
Figure 9 Portfolio level events.............................................................. 32
Figure 10 CMMI-DEV Maturity stages ................................................. 39
Figure 11 Design flow of CMMI-DEV .................................................. 40
Figure 12 Agile maturity model (Chetankumar Patel & Ramachandran, 2009) 41
Figure 13 Process improvement roadmap for AMM ............................ 43
Figure 14 SPICE Maturity stages(Connor & Eds, 2013)........................... 44
Figure 15 Matrix with agile levels and principles(Sidky et al., 2007)........ 46
Figure 16 Benefield’s technical maturity model ................................... 48
Figure 17 Adaptive systems.................................................................. 50
Figure 18 Design flow.......................................................................... 55
Figure 19 Design pattern ...................................................................... 55
Figure 20 Initial Design pattern.............................................................. 56
Figure 21 Design pattern of the model.................................................. 56
Figure 22 Mapping of core values.......................................................... 59
Figure 23 Stage 1 mapping .................................................................. 60
Figure 24 Stage 2 and 4 mapping.......................................................... 61
Figure 25 Stage 3 and 5 mapping.......................................................... 61
Figure 26 Agile and scaled agile principles............................................. 63
Figure 27 Mapping of principles............................................................. 64
Figure 28 Scrum practices .................................................................... 66
Figure 29 Scaled agile practices............................................................. 67
Figure 30 Design pattern ..................................................................... 101
Figure 31 Revisiting design guidelines .................................................. 106
Chapter 1 Introduction

1.1 Introduction
This chapter represents the introductory part of the research by placing the research context – Scaled agile software development (Section 1.2), stating the research problem and formulating the research objective (Section 1.3) and stating the relevance to the conduct of the research (Section 1.4). Further, the research questions are provided (Section 1.5) and an overview of the research planning in investigating the research context is given (Section 1.6). Finally, the delineations are presented in section 1.7.

1.2 Research context – Scaled agile software development
In the early stages, strategy management mostly revolved around finding the best problem-solution fit by top management people, with little or no involvement of the team who will implement the strategies. The shift in strategic thinking from a mechanistic perspective to a perspective which takes complexity and uncertainty into account, is evident from various strategic management approaches in organizational context (Nerur & Balijepally, 2007). This trend in strategic thinking, transitioning from a deterministic problem solving approach towards a dynamic process, defined by active involvement of all the stakeholders and iterative cycles, is also reflected in software development. To this end, the Agile manifesto was proposed in the year 2001 which introduced several principles to “uncover better ways of developing software by doing it and helping others do it” (Highsmith & Cockburn, 2001, p.121). Ever since the manifesto was created, it has brought remarkable changes in the Agile software development field, through introducing a multitude of light development tools such as scrum, lean software development, Kanban and Extreme Programming (XP) to name a few (Dingsøyr, Nerur, Balijepally, & Moe, 2012). These unprecedented changes in a decade is attributed to the fact that software development continuously needs to improve its quality, while increasing productivity, which indeed drives an organization to look for innovative practices to develop software.

Many organizations which followed traditional software development methodologies started moving towards agile, due to its benefits in introducing agility across the teams and also due to various demerits of traditional methodologies. Major problems faced by the organizations which followed traditional project management approaches include changing project goals along the timeline of the project and discrepancy in the end product and proposed end product. (Highsmith & Cockburn, 2001). Customer may attribute to change in the requirements as the project proceeds towards the end goal. Being unresponsive to these changes, might lead to customer dissatisfaction and business failure. In addition, traditional methodologies focus extensively on documentation and needs a lot of rework if there is a change introduced in the middle of the project. In contrast, Agile methodologies focused on

---

1 Referencing my own paper submitted to Master Thesis definition course work, which relates to agile software development.
responding to change as a default factor and stressed mainly on the agile principles, which reads as follows.

**Table 1 Principles of agile software development (Highsmith & Cockburn, 2001)**

<table>
<thead>
<tr>
<th>Principles of agile software development (Highsmith &amp; Cockburn, 2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals and interactions</td>
</tr>
<tr>
<td>Working software</td>
</tr>
<tr>
<td>Customer collaboration</td>
</tr>
<tr>
<td>Responding to change</td>
</tr>
<tr>
<td>Small and Frequent releases</td>
</tr>
</tbody>
</table>

After a decade of innovative trends in agile practices which focused heavily on a team level, organizations started looking for scaled agile practices in order to scale the agile practices from team level to enterprise level. This is mainly due to criticisms for agile practices, which can be applied to only one team rather than several teams working on a project (Reifer, Maurer, & Erdogmus, 2003). In large-scale development organizations, agile breakdown into smaller tasks poses a serious challenge due to complex dependencies between the teams which work on same project. These dependencies can be either the dependencies between workflow in the development process, dependencies due to different software artefacts, or dependencies due to tasks assigned to team members of different teams (Sekitoleko, Evbota, Knauss, Sandberg, & Ab, 2014). Complexity of dependencies increases with the size of the organization where several scrum teams work on a project. In addition, organizations often discover misalignments between methods used by different teams when using agile methodologies in a team level, in a large-scale setting (Tourtoglou & Virvou, 2012). Furthermore, unsolved interdependencies between the teams may lead to long lead times and increase business failure (Heikkila, Paasivaara, & Lassenius, 2013). Using agile on a team level empowers agility only across a team. In order for the whole organization to embrace agility, agile methodologies need to scale up from team level to higher levels. Thus there developed a need for scaled agile software development among the organizations.

**1.3 Problem at stake**

In order to scale agile methodologies across the enterprise, various frameworks such as Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD) and Large Scale Scrum (LeSS) were developed in the recent years, with SAFe being popular among the crowd which has gained widespread attention among several large scale organizations (Laanti, 2014). However these frameworks only act as a template, in which current agile practices that are used in the real world for a team level are extrapolated or scaled to an enterprise level. Thus the template contains all the essential practices needed for an organization to scale up their agile practices. As such, the templates contain several practices which needs careful attention before being implemented or needs continuous improvement after being implemented. None of the frameworks give a description on assessment criteria of certain scaled agile practices and which practices need to be in place before evolving in a particular practice. To summarize, these frameworks tend to focus only on describing the best practices, but makes fewer
attempts to describe the implementation strategy or the maturation paths of these practices in adopting the framework.

The problem of this research is formulated as such: Despite the availability of several frameworks, which described the best practices in scaling agile across the enterprise, there is no clear understanding on the maturation paths of these practices and assessment criteria in assessing the level of adoption of these practices. Organizations lack a scaled agile maturity model that could help them in identifying the areas of improvement of a certain practice and list of standard practices which needs to be in place in evolving towards a scaled agile practice.

The problem owners are the large scale organizations that have several teams working towards a common goal and could tap the advantage of such a model in adopting towards a scaled agile practice. Though the model is aimed at being evaluated by experts in one organization from a specific industry sector, we claim this as our ground work in bridging the gap discussed above. Our objective of this research is:

“To develop a scaled agile maturity model that can help organizations which have adopted scaled agile methodologies, in finding the areas of improvements in adapting to scaled agile practices and maturity path of those practices in stages”

As such, the maturity model will identify the stages of maturity of practices and assessment criteria in evaluating the scaled agile practices used in those stages.

1.4 Study relevance
This section gives an overview of the main drivers with respect to academic contribution and communal perspective, to conduct this research.

1.4.1 Societal relevance
The suggestion of assisting the organizations for adapting to scaled agile practices is not new. The idea that organizations with large number of teams working on a similar end goal should adapt scaled agile practices for scaling their agility across the enterprise, has been promoted in the recent years (Leffingwell, 2011). Nonetheless, novelty of this research lies in the concept of “Maturity model”, along with the assessment criteria which can be used in assessing the adoption level of those practices. This work represents an exploratory design oriented research, given the fact that the end goal is to build a scaled agile maturity model for large scale organization. This model should provide more insights into the areas of improvements that play a role in successful adaption to scaled agile practices. In addition, the research context is a practical problem faced by large scale organizations. Therefore actual relevance of this research translates into direct benefits for the organizations in terms of successful adoption to scaled agile methodologies. So clearly the research aims in improving the quality of adoption and maintenance of scaled agile methodologies in a large scale IT organization, which forms the part of Software community.
1.4.2 Scientific relevance
Next to the practical significance of this study, academic motivation also plays as a major driver to conduct this research. This study will serve to the development of a new theoretical body of knowledge contributing to the existing literature on scaled agile methodologies by introducing a scaled agile maturity model to be used by the large scale organizations. Furthermore, this research aims in bridging the gap in literature regarding the articles that address the problem of adaption of scaled agile methodologies. A lot of research was conducted to investigate the scaled agile software development and the implications of the framework available in the market, while the topic of maturity model is only limitedly discussed in literature. Therefore, through this research, we would like to contribute to the research community in the following domains: Scaled agile adoption and assessment criteria.

1.5 Research Questions
This section introduces the research questions and the description on why these research questions are relevant concerning the research context and problem statement.

1.5.1 Main research question
To illustrate the design of scaled agile maturity model, the following central research question needs to be answered:

“How does the scaled agile maturity model look like?”

The answer to this question will translate into a scaled agile maturity model, based on which large scale organizations could find the areas of improvements in adapting to scaled agile practices and maturity path of those practices in stages.

1.5.2 Sub research questions
To come up with a viable scaled agile maturity model, the central research question leads to some sub-questions. The flow of the sub research questions is based on the research design cycle proposed by (Peffers et.al, 2007). The method embraces six stages for conducting design research in information systems: a) Problem identification and motivation, b) Definition of objectives for a solution, c) Design and development, d) Demonstration, e) Evaluation, and f) Communication.

The very first stage of the design cycle is the problem identification and motivation for constructing the new IS artefact. The second stage focusses on describing objectives of the solution, sequentially after identifying the problem at stake. This step is translated as chapter 1 Introduction, in which we introduced the problem statement, motivation and objectives of this research.

The third stage of the design cycle focusses on the design and development of our IS artefact, the scaled agile maturity model. As a first step in design phase, we introduce the first sub-question which focusses on gathering knowledge on the research context: Scaled agile software development and reads as follows:

RQ1. What does the process of scaled agile development look like?
This chapter represents the domain description and supports the development of scaled agile maturity model by looking at the agile and scaled agile software development in general, identifying the need for scaled agile methodologies in large scale organizations and scaled agile practices. To answer this question, two research methods are used. First a literature review of academic sources in addressing the topic was made, and then participatory observation on the practices followed by a large scale organization in real life context, will be performed.

The second research question aims to focus on the maturity models available in literature and design constructs used by the models in designing them, mainly investigating the merits and demerits of the design strategy employed by the models.

**RQ2. What are the basic requirements which have to be considered in designing a scaled agile maturity model?**

By answering this question, the requirements of designing the scaled agile maturity model will be identified. After identifying the requirements, the research will aim at designing the model, and the sub-question reads as follows:

**RQ3: How does the initial design of scaled agile maturity model look like?**

The answer to this question will provide the first version of the scaled agile maturity model. After the design and development of the model, the next stage proposed by Peffers.et.al is the demonstration stage in which the designed artefact is evaluated to observe and measure how well the artefact supports the solution proposed in the design. Therefore the forth sub-question focusses on this concept and reads as follows:

**RQ4. What is the value of the scaled agile maturity model in a real life context?**

This last question will serve as a way to validate the initial design of the model. Expert evaluation will be used as a research method in answering this sub-question. Experts will be selected based on their relevance to the proposed research context. TomTom was chosen as a company for the evaluation stage, due to its situation in software industry which is relevant to our research objective - a large scale organization with recent adaption to scaled agile methodologies. Since 2014, TomTom is engaged in developing scaled agile practices across the enterprise with more than 200+ certified and trained scaled agilists, which makes it suitable for such an evaluation.
1.6 Thesis outline and research planning

With regards to the outline, the final research project consists of a document of six chapters. As a thumb of rule, each sub question relates to the chapter and corresponds to the sequential order of the research questions. This section provides an overview of each chapter and their content.

Chapter 1. Introduction. This chapter provides the thesis definition, by introducing the problem statement, research questions and methodology.

Chapter 2. Agile and Scaled agile software development. This chapter presents an extensive overview of the agile and scaled agile software development methodologies in literature. Furthermore, process involved in agile and scaled agile software development will be studied in detail, yielding the main implications for the research context.

Chapter 3. Maturity models of agile software development. This chapter presents an overview on the agile maturity models in literature. The main design constructs used in designing a model are discussed. The possible merits and demerits of including certain design constructs will be investigated. Then, based on the findings, requirements for initial design will be derived.

Chapter 4. Design and Development of scaled agile maturity model. This chapter presents the initial design of the model with the requirements identified in Chapter 3. The requirements are then translated into practical requirements for the model. Apart from the findings on requirements from chapter 3, it will also include the findings on participatory observation on the practices followed by TomTom.

Chapter 5. Evaluation of scaled agile maturity model. This chapter presents the findings on the expert evaluation phase explored in TomTom. This is done by presenting the initial design of the model to various key actors in TomTom and validated for its initial objectives, consistency and completeness of the model.

Chapter 6: Conclusions. This chapter answers the central research question, reflects on the research and initial objectives proposed in the introductory chapter. Furthermore, it also includes the future research questions and reflection in general on the research work.

Figure 1 below depicts the research design flow and helps to visualize the design flow as discussed above. In addition it also provides an indication of how the sub-questions are organized as chapters. As a rule of thumb, each sub-question has its own chapter.
Figure 1 Research design flow
1.7 Delineations
Delineations imposed by this research are mainly related to the adaptations to the original design cycle. This study follows the six steps of design cycle as proposed by Peffers et.al. (Peffers et al., 2007). However some of the steps had to be adjusted to fit the requirements and scope of this thesis, as follows.

The first step, problem identification and motivation and second step, definition of objectives are described in chapter 1. The third step, Design and development of the model are answered through three sub-questions and along three chapters (2, 3, 4). After the design phase, Peffers et.al introduce the demonstration phase, in which the use of the artefact designed, is being demonstrated and compared with initial objectives of a solution, which focused on ex-post level of evaluation. However at this stage, due to exploratory nature of the research, we consider using ex-ante level of evaluation with experts before finalizing the design. This is required at this stage to avoid the feedback loops, in the further stages of evaluation. The results of expert evaluation are discussed in Chapter 5. Based on this assessment final version of the scaled agile maturity model will be realized, and final conclusions will be drawn in Chapter 6. However, ex-post evaluation proposed by Peffers et.al (2007) cannot be achieved at this stage and further research is required to accomplish the stage.
Chapter 2
Chapter 2 Agile and scaled agile software development

This chapter represents the literature review on the research context as shown in the research flow diagram Figure 1, in and answers the first research question:

**RQ1. What does the process of scaled agile development look like?**

The objective of this chapter is to provide the background on which the research is based: Agile and scaled agile software development processes and practices which help the organization in reaching particular maturity stage. This chapter is exploratory in nature and provides the first insights into what an agile software development are principles and practices involved in ideal agile and scaled agile software development.

To answer this sub-question 1, a systematic literature survey was conducted. This chapter is structured as follows. The first section includes the method of literature survey used in selection of articles and key words (Section 2.1). Next to that, a brief introduction of software development is given to get acquainted with the practices and principles of software development in general (Section 2.2). Further in section 2.3 agile software development and its processes are elaborated. Next to that, scaled agile software development and its processes are described in section 2.4.

### 2.1 Literature review method

In this section, the literature review method that is used to find the literature regarding the scaled agile and agile software development is briefly described. Articles are collected through a systematic search on the scientific databases such as Scopus, web of science and google scholar. The collected articles and publications are carefully studied to achieve the above mentioned objectives.

Throughout the literature survey, Scopus was mainly used as a search database to collect the publications on the research context. The keywords such as agile software development, scaled agile were used. The queried results were huge if a single query is used to collect the articles. Hence a combination of keywords such as, ‘software development’, ‘agile’ and ‘scaled agile’ were used, as such it will contain the main concepts of this research. The search resulted in several articles, most of them relating to the category of Computer Science and Information Systems. However when the key word “scaled agile” is used, it resulted in very few articles, describing the newness of the topic in the academic world. The resulted articles were mostly related to the challenges faced in a large organization in scaling up agile rather than the practical implications faced by the organization in adapting to scaled agile practices. This research makes the first effort in bridging the aforementioned gap in literature.

### 2.2 Agile software development

The purpose of this sub-chapter is to give a short description on agile software development and the processes involved in it. Before moving on to agile software development, we will give a brief introduction to software development; further we will briefly explain the need of agile software development and thereon move towards the description on agile software development. This will also
include the discussion on agile principles which forms the backbone of agile software development practices. Next to that, the need for approaches in scaling agile across the entire enterprise will be discussed.

2.2.1 Software development
Software development is a process in which the experts from different technical domains collaboratively work together to create a software artefact or product (Souza, Redmiles, Mark, & Field, 2002). The approaches used in managing the process of software development, helps in structuring, planning and organizing the process of developing software artefacts. A wide variety of such approaches has been evolved in the recent years, each with its own strengths and weaknesses (Lee & Weidong Xia, 2009). One such approach is the agile methodology, which has a set of principles for an adaptive software development process.

2.2.2 Need for agile software development
To get a clear picture on agile software development, there is a necessity to research on the theoretical background on why there is a need for such an approach to software development. Before the introduction of agile practices in the year 2001, traditional project management strategies remained as a popular means of developing software in several organizations (Lee & Weidong Xia, 2009). Traditional project management strategies follow a waterfall approach in which a new phase is introduced only after the previous phase is fully completed. Output of one phase becomes the input for the next phase and the software development process is sequential in nature, rather than being iterative, as shown in the figure below:

![Figure 2 Phases of traditional project management methods](Palmquist, Lapham, Miller, Chick, & Ozkaya, 2013)
As seen from the figure 2, traditional project management methodologies have a series of sequential steps in which the requirements are collected upfront before the start of the project. Also from this representation, it can be pointed out that waterfall-based approaches presume that the requirements will remain unchanged during the course of the project. Next to that, the stage on analysis is done only once in the course of the project and precedes with the design phase. Followed by coding, testing and integration which occurs single time during the phase of the project.

Problems arise when the goals of the project are changing due to new requirements from the customer and each phase needs to be revisited several times. Since project management methodologies aim at sequential process flow, revisiting the stages may lead to several time delays and customers may be left unsatisfied either due to time creep or due to unavailability of their new requirements (Highsmith & Cockburn, 2001). Being unresponsive to the changes required by the customer might prove ineffective and therefore lead to customer dissatisfaction and business failure. In addition, traditional methodologies focus extensively on documentation and need a lot of rework if there is a change in requirements in the middle of the project (Palmquist et al., 2013).

An effective solution for this would be to follow agile response in every phase of software development by using agile methods, which assumes requirements are changing and there always might be a need to revisit the previous stages of the project. To bridge this gap on demerits of traditional waterfall-based methods, agile software development approaches emerged which according to Palmquist et al.,

“an iterative and incremental approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with “just enough” ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders” (Palmquist et al., 2013)

Agile’s philosophy mainly emphasizes the formation of holistic teams who are multifaceted and exhibit both specialized and general skills. This redundant role would help them in reacting to emergent situation and uncertainty, which makes agile unique in contrast to traditional approaches (Nerur & Balijepally, 2007). Furthermore, agile approaches are quick and adaptive to business changes when new requirements emerge from the customer. This helps in more customer participation during the course of project, ultimately leading to more customer satisfaction and business success. Palmquist et. al thus describes agile software development approaches as a parallel world to waterfall-based approaches, in which the systems evolve through a series of short iterations in which the stages on analysis, design, code, test, and integration occurs at each iteration, as opposed to sequential waterfall-based approaches (Palmquist et al., 2013).

The key differences highlighted by Palmquist et. al and Nerur et. al are described in the table below..

---

2 Referenced from my own paper submitted to Master thesis definition course related to Agile software development
Table 2 Differences between traditional and agile software development methods

<table>
<thead>
<tr>
<th>Content</th>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fundamental assumptions</td>
<td>Systems are fully specifiable, predictable and can be built through meticulous and extensive planning</td>
<td>Adaptive software development, using the principles of continuous design, continuous learning and frequent reflective cycles.</td>
</tr>
<tr>
<td>Communication</td>
<td>Formal</td>
<td>Informal</td>
</tr>
<tr>
<td>Management style</td>
<td>Centralized</td>
<td>Decentralized among different teams, leadership and collaboration</td>
</tr>
<tr>
<td>Role assignment</td>
<td>Individual roles assigned by management</td>
<td>Self-organizing teams and encourages role interchangeability</td>
</tr>
<tr>
<td>Customer’s role in the process</td>
<td>Important</td>
<td>Critical</td>
</tr>
<tr>
<td>Development model</td>
<td>Phase by phase - incremental</td>
<td>Evolutionary, reflective and adaptive - Iterative</td>
</tr>
<tr>
<td>Responsive to business changes</td>
<td>Slow</td>
<td>Quick and adaptive</td>
</tr>
</tbody>
</table>

With respect to agile software development, scholars have different definitions on the term agility which is highlighted in the table below.

Table 3 Definition of software development agility in literature

<table>
<thead>
<tr>
<th>Literature</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Conboy &amp; Fitzgerald, 2004)</td>
<td>“It is defined as the continual readiness of an entity to rapidly or inherently, pro-actively or reactively, embrace change, through high-quality, simplistic, economical components and relationships with its environment the continual readiness of an entity”</td>
</tr>
<tr>
<td>(Highsmith, 2004)</td>
<td>“Agility is the ability to both create and respond to change in order to profit in a turbulent business environment; it is the ability to balance flexibility and stability”</td>
</tr>
<tr>
<td>(Qumer &amp; Henderson-Sellers, 2008)</td>
<td>“Agility is a persistent behavior or ability of an entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, and uses economical, simple, and quality instruments in a dynamic environment; agility can be evaluated by flexibility, speed, leanness, learning, and responsiveness”</td>
</tr>
<tr>
<td>(Lee &amp; Weidong Xia, 2009)</td>
<td>“Software development agility is a team’s capability to efficiently and effectively respond to and incorporate user requirement changes during the project life cycle”</td>
</tr>
</tbody>
</table>
Analyzing the definitions from various scholars, software development agility mainly stresses on the team’s capability and culture to effectively respond to changes and efficiently collaborate with the users to develop the software product. In order to make a consistent effort towards analyzing the software development agility, four core values and twelve agile principles were formally introduced and documented in the publication of the *Agile manifesto* in the year 2001 (Lee & Weidong Xia, 2009).

### 2.2.3 Agile principles and values

Before moving onto the practices and processes involved in the agile software development, it is essential to get a better understanding on the agile principles, which form the backbone of the practices and frameworks designed for agile methodologies. The agile principles enlisted in the agile manifesto include (Highsmith & Cockburn, 2001):

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

#### Table 4 Core values of agile methods (Highsmith & Cockburn, 2001)

<table>
<thead>
<tr>
<th>Individuals and interactions</th>
<th>over</th>
<th>Processes and tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working software</td>
<td>over</td>
<td>Extensive documentation</td>
</tr>
<tr>
<td>Customer collaboration</td>
<td>over</td>
<td>Contract negotiation</td>
</tr>
<tr>
<td>Responding to change</td>
<td>over</td>
<td>Following a plan</td>
</tr>
</tbody>
</table>
According to the Agile Manifesto, agile software development values individuals and interactions over processes and tools, working software over documentation, customer collaboration over contract negotiation and responding to change over following a plan. It promotes smaller and frequent delivery of working software to effectively manage the emerging requirements of users. With these principles and core values as backbone, several agile methods were developed in the recent years (Campanelli & Parreiras, 2015). Some of the mainstream agile methodologies include Extreme programming (XP), Scrum, Kanban, Lean, Feature-driven development (FDD), Dynamic systems Development Method (DSDM), and Adaptive software development (ASD), Crystal and Rational Unified Process (RUP). Each method focuses on a specific core value of agile and there is no particular standard on how a methodology should implement its agile features. With Scrum being the most popular and currently the most adopted agile methodology in various organizations (Campanelli & Parreiras, 2015).

2.2.4 Scrum process
In order to better understand the process of agile methodologies, we present a short explanation of Scrum. Scrum is an agile methodology, which helps in adaptive software development process for smaller teams (Rising & Janoff, 2000). Scrum, rather than only as methodology, but also acts as a framework of defined practices and roles, which helps the team focus on agility during the course of the project. Most of the practices of Scrum are based on the principle that "the change in requirements along the project is acceptable" and therefore Scrum focuses on increasing the ability of the team to deliver software product to the customer without any time delays, while adapting to changing circumstances. In order to do so, the Scrum describes a process which is illustrated in Figure 3

The process starts with a Scrum iteration called a sprint, which is the basic unit of workflow in Scrum. The sprint is a time-boxed effort that is prominent for a specific period of time. The duration is fixed for each sprint, and is normally between one or two weeks, with two weeks being the common duration among many organizations. Before each sprint, there is an initial planning phase called sprint planning event that aims to define and identify the work for each sprint, and create a backlog of tasks that has to be finished by the end of the sprint (Rising & Janoff, 2000). These tasks come from a prioritized list of tasks called product backlog, which contains all the potential features of the software in general. Features which are selected to be implemented for a specific sprint forms the Sprint backlog. Each sprint aims at delivering a potentially shippable increment to the application and delivering a valuable functionality to the customers, where potentially shippable increment denotes a piece of working software that can be used by the customers. During the sprint, the team conducts daily Scrum meetings to discuss the progress made by the team towards the end goal. However, requirements remain fixed for that Sprint (time-boxed), meaning that no change is allowed in the sprint backlog once it is finalized and development is started. New requirements are sent to the product backlog and queued up to the subsequent sprint sessions. At the end of each sprint, there is a review session to review the work done during the sprint to the respective stakeholders and sprint retrospective session, which identifies lessons to be learned and improvements for the next sprints (Hicks & Foster, 2010).
In order to effectively follow the scrum process, few key roles are established:

1. **Product owner – Owner of backlogs**
   Product owner is the owner of sprint backlog, which contains all the tasks that needs to be done for a particular sprint. The item in the backlog is user-centric and takes the format of user stories, which is an agile approach that helps in writing the requirements from the lens of customer, rather than from the team. (Rising & Janoff, 2000). He/She acts as a point of communication to the development teams for any doubts regarding the sprint backlog. It is the responsibility of the product owner to prioritize the tasks in the sprint backlog, which helps the team in identifying the tasks of high and low priority and which tasks needs to be picked up next. Product owner also helps in communicating the product priorities to the team and team’s priorities to the external stakeholders.

2. **Scrum master – Owner of processes**
   Scrum master leads the scrum meetings such as Daily scrum meetings, sprint retrospective, sprint planning and sprint review. In addition, he/she empirically measures the progress of the development team towards the goal planned at the start of the sprint, in delivering the working software. He/She also ensures that the team members make progress, and also record
the decisions made during the scrum meetings and initiate the actionable results for the decisions (Rising & Janoff, 2000).

3. Development team – Owner of development
Development team is responsible for delivering the software increment of the product proposed and scoped at the start of each sprint. Usually the team is made of 3-10 individuals and cross-functional in nature made up of developers, documenters and quality control staffs.

2.2.5 Need for scaled agile methodologies
Core values and principles of agile methodologies revealed that organizations look for more adaptive frameworks for developing software rather than traditional project management strategies which lack in taking uncertainty into account. The development of different agile frameworks in the recent years also shows that the software industry has an increasing interest towards the deployment of agile methods. By emphasizing more on the use of iteration level development through small features, agile methodologies have increased the ability of organizations to quickly respond to changing customer requirements. However, agile practices/methodologies have a history of being applicable only to smaller teams (Kaehkoenen, 2004). Further it has been widely acknowledged by various scholars that the agile methodologies such as scrum, XP etc. were originally developed only for team-level process improvements (Tourtoglou & Virvou, 2012).

Furthermore, it has been highlighted that using an agile framework such as scrum, only on the team level can lead to dependency and synchronization problems (Kaehkoenen, 2004). This is widely noticed in large scale organizations, which has dependencies between several teams working on a similar project. As defined by Sekitoleko et.al (2014), these dependencies can be either the dependencies between workflow in the development process, dependencies due to different software artefacts, or dependencies due to tasks assigned to team members of different teams (Sekitoleko et al., 2014). Complexity of dependencies increases with the size of the organization, due to increase in teams working on a project. In addition, organizations often discover misalignments between methods used by different teams when using agile methodologies in a team level, in a large-scale setting (Tourtoglou & Virvou, 2012). Furthermore, unsolved interdependencies between the teams may lead to long lead times and increase business failure (Heikkila et al., 2013). Using agile on a team level empowers agility only across a team. In order for the whole organization to embrace agility, agile methodologies need to scale up from team level to higher levels.

2.3 Scaled agile methodologies
The purpose of this subchapter is to give a description of the landscape on scaled agile methodologies and the process followed in scaling the agile practices across an organization. Before moving on to the scaled agile software development, we would like to give a brief description on “scaling” in general. Next, a description on scaled agile methodologies and its process are given.
2.3.1 Scaling software development

Large-scale software development is often large in scope in terms of projects, number of team working on project, diversity of customers and end goal. Ozkaya et al introduces 2 types of scaling, when the scope increases in a large scale software development: Team scaling and Time scaling.

Team dimensions of scale relates to multiple teams working towards the end goal, but with different tasks. In a large-scale software development, there are often multiple teams within an organization that must collaborate and provide input to the software product development. It needs careful orchestration and effective processes for seamless collaboration of these teams in working towards the end goal (Ozkaya, Gagliardi, & Nord, 2013). This can either be multiple teams working on a project (product mode) or multiple teams working on multiple products that is part of a program (Platform mode).

![Figure 4 Product and Platform mode of team scaling](image)

In contrast, time dimension of scaling relates to duration of the software development and lifecycle of the software development process on the whole. Over time, technology and the system may undergo changes in various ways. In response to the changes, the system needs to rhythmically scale up taking lifecycle budgeting, planning and milestone planning into account (Ozkaya et al., 2013). Among the two dimensions, team dimension is typically the very frequent perspective often being concerned in the software development from team level to enterprise level.

2.3.2 Scaled agile software development

As already discussed, team level scaling is a popular dimension to scale the agile practices across the enterprise. Few frameworks exist till date, which takes team dimension into account in scaling the agile practices such as Disciplined Agile delivery (DAD), Large Scale Scrum (LeSS), Leading agile and Scale Agile Development (SAFe). With SAFe being the popular framework adopted by many large-scale,
multinational organization (Laanti, 2014). For sake of understanding the scaled agile software development, we provide a short description of SAFe framework and its process in scaling agile from team levels to higher levels.

Developed and created by Dean Leffingwell, the scaled agile framework is a highly structured and prescriptive template that supports large-scale IT organizations to embrace agility across the whole enterprise. SAFe framework uses agile practices which are used in the real world and extrapolates the agile practices from team level to enterprise level (Leffingwell, 2011). The framework is developed under version currently and is publicly available as a big picture format as shown below in Figure 5 (Leffingwell, 2011). This framework is studied for its principles, practices and processes, which will represent general body of scaled agile methodologies in this research in developing a scaled agile maturity model.

![SAFe Big Picture Framework](image)

**Figure 5 SAFe Big picture framework** (Leffingwell, 2011)

SAFe Big Picture represents the holistic visual representation of the framework. And uses already existing agile methodologies such as Scrum and lean (Leffingwell, 2011). In simple words, the framework is divided into three levels, namely *team, program, and portfolio levels*. It follows team dimension in separating the practices followed in these 3 levels.
Figure 6 Team, program and portfolio levels

**Team level**
On the team level, the framework proposes scrum as the basic element in delivering a potentially shippable increment towards the end product. Individually, teams practice scrum principles and ceremonies such as Sprint planning, retrospective, daily stand ups etc and the team members collaboratively work on the tasks/items enlisted in their team’s backlog. As such, every team will have its own backlog of tasks in delivering a potentially shippable increment to the product level goal. The team level goals of multiple teams will form the product goals at the program level. Hence scaling to higher levels is achievable, if only the team level has 100% achievement of agile practices. Thus it can be well noted that ‘being agile at the team level acts as a main pillar in scaling up agile’

**Program level**
The program level supports in organizing several teams working on a similar project or having multiple dependencies, in a collaborative manner in order to scale up the agile practices to a step higher than the team level. Program level helps in aligning the teams with a similar vision and goals for the end task. Each product in the program level will have its own prioritized program backlog/items which had to be delivered to the customer at periodic intervals. Together all the teams working on a specific product in the program level, form Program increment (PI) objectives. These objectives are the
summarized description of the specific business and technical goals intended by the teams to achieve in the program increment. Thus, similar to Sprint planning in team level, program level has Program increment planning. Similarly the program level has its own program backlog compared to sprint backlog in the team level. In the team level, only one team works towards providing working software increment, whereas in the program level, several teams work together to develop a Program increment (PI).

In order to contribute to the PI objectives at the program level, all the teams and team members should collectively come together for planning and discussion. Such an event is named as Program increment (PI) planning, can be put in simple words as program level sprint planning with different teams. The goal of agile/ scaled agile software development is to frequently deliver valuable increments of a solution at periodic intervals, which is accomplished via stream of releases. With every release, the organization delivers more value to the customer. SAFe uses the name of “Agile release train (ART)”, for such releases, in which 5-12 teams plans, commits, and executes the release together at periodic intervals. ART forms the major construct of SAFe framework, which helps in providing cadence and synchronization, which forms the pillar in moving towards a scaled agile environment. The releases are planned during a 2-day planning event (PI planning), which involves all the stakeholders forming the program level.

It should also be noted that, ART differs in accordance with the company size and distribution, as shown below in the figure.
As can be seen in the figure, organizations differ in their products and number of teams working towards the product. Each team creates a value stream. There can be either multiple value streams in a single ART, or multiple ARTs forming a single value stream. However SAFe big picture, illustrates the case of single ART, in which the team value streams fit well within the limit.

Similar to the scrum roles in the team level, program level also includes essential roles in order to effectively perform the portfolio level events such as ART in an organization. This is listed as follows:

1. **Release train engineer - Owner of processes**
   In simple words, release train engineer is the scrum master in the program level. He/she establishes and communicates the agenda for PI planning events. He facilitates the readiness for PI planning, by fostering the preparation of program backlogs. He acts as a bridge in communicating program level objectives to the teams and vice versa. He reports the overall status of the agile release train to the concerned stakeholders in the program level.

2. **Product management – Owner of backlogs**
   Product management has the feature backlog of all items which needs to be addressed in the PI planning, collected through the voice of the customer. Product managers of the product management team works along with product owners of each team in prioritizing the backlog in the PI planning event.

3. **User experience designers and system architects – Owner of designs**
   Though the features in the backlog are clear, it needs huge effort to visualize the features in real time. User experience design (UX design) plays a major role in visualizing the features in order to make it
more clearly to the developers as what the functionality is, before they start to develop the features. UX designers take the responsibility of visually describing how the feature will improve the quality of interaction between the users and software. They provide the relevant technical infrastructure to support the implementation of PI objectives. System architects have the technical responsibility in creating the overall architecture for a new feature implementation in PI objectives. Architecture refers to the definition of abstract higher-level requirements and definition of interfaces between different sub-systems (black boxes), when a new feature is introduced in the application (Leffingwell, 2011).

**Portfolio level**

Portfolio level is the highest level of concern in the framework. The portfolio level includes basic governance mechanisms that help in assuring that the investments across different agile release trains at the program level, provides necessary returns to the enterprise. The essential events of the level are highlighted in the figure below.

![Figure 9 Portfolio level events](image)

Portfolio level consists of a program portfolio management system which looks after the activities at the portfolio level and takes the responsibilities for strategy and investment funding across different ARTs. They also calculate the financial, technological and business constraints, and further implement a portfolio solution strategy for ensuring perfect running of agile release trains. In addition, portfolio level will contribute to *Business epics*, which capture and reflect the new business ideas that can be used as an input across different product units in the program level. These epics form the backlog at the portfolio level and acts as a highest priority backlog in the framework.
2.3.3 Scaled agile principles and core values

With scaled agile levels being defined, it is also essential to investigate the principles SAFe follows in defining the framework. The practices encouraged in different levels of the SAFe framework, takes its shape from its core principles (Leffingwell, 2011). The principles include:

Table 5 Principles of SAFe

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take an economic view</td>
<td>In order to achieve best value and quality at shorter lead times, it is essential to develop a value stream economic framework which illustrates the costs of delay, operational and development costs etc.</td>
</tr>
<tr>
<td>Apply systems thinking</td>
<td>It is essential to apply systems thinking in the scaled agile environment, which mainly focusses in understanding the system by investigating the interactions among the components that make up the system</td>
</tr>
<tr>
<td>Assume variability: Preserve options</td>
<td>Responding to emerging requirements</td>
</tr>
<tr>
<td>Build incrementally with fast, integrated learning cycles</td>
<td>Promoting to build solutions incrementally in a series of short iterations.</td>
</tr>
<tr>
<td>Base milestones on objective evaluation of working systems</td>
<td>Each integration should provide an opportunity to evaluate the solution, frequently and throughout the life cycle of the project. This in turn helps in financial, technical and fitness for purpose.</td>
</tr>
<tr>
<td>Apply cadence, synchronize with cross domain planning</td>
<td>Alignment and transparency across different teams working collaboratively on the project. Cadence transforms unpredictable events to predictable and provides a rhythm for development.</td>
</tr>
<tr>
<td>Decentralized decision making</td>
<td>Distributed decision making across different levels: team, portfolio, and program</td>
</tr>
</tbody>
</table>

From the principles, it can be noted that the characteristics on alignment, transparency and incremental building of technical solution plays an essential role among the practices involved in scaling the agile practices across the enterprise. Though SAFe helps in achieving substantial improvements in scaling agile to higher levels than team levels, we believe that there is truly no one solution to every challenge an organization faces in scaling agile. With its principles and practices SAFe provides a prescriptive method of how scaling agile across the organization can look like and its artifacts which has to be in place for the successful implementing practices at scale. However given the complexity of unique challenges each and every organization faces, it needs an orchestrated
tailoring and customization before being implemented in an organization. In addition to that, SAFe provides a big picture of what has to be done in scaling agile, but misses the factor on describing the step by step implementation of those practices, which is highly relevant for an organization to find areas of improvement (if already adopted to scaled agile methodologies) or areas of achievement (if not adopted to Scaled agile methodologies).

2.4 Chapter summary

Agile and scaled agile software development processes, principles, core values are investigated in this chapter. We answered the first sub-research question in this chapter, which is

*RQ1. What does the process of scaled agile software development look like?*

Using literature review, we determined that scaled agile software development takes a process of scaling the agile practices from team level to program and portfolio level, by taking team dimension as the pillar of scaling factor. We also identified different artefacts and practices used in scaled agile methodologies and its level of abstraction between different levels. We also found that, most of the agile/scaled agile frameworks defined in the literature are backed up with agile/ scaled agile principles and core values as the foundation for defining the practices in different levels. It is also identified that, scaling of agile practices to higher level than team level can occur, only when the team level has clearly an agile element in the way of working. And number of unique challenges across different organizations, indicates that the scaled agile practices enlisted by the frameworks cannot be a perfect solution, unless the solution is tailored and customized according to the needs of the organization. By combining these findings, along with literature on agile maturity models, we will be able to provide the description on potential pillars of the scaled agile maturity model.
Chapter 3
Chapter 3 Maturity models of agile software development

3.1 Introduction

This chapter introduces the design guidelines has to be taken into account to design the scaled agile maturity model and answers the second sub research question:

RQ2. What are the basic requirements which have to be considered in designing a scaled agile maturity model?

We have seen in the previous Chapter 2 Agile and scaled agile software development that core principles and core values form the backbone of the practices enlisted in agile and scaled agile methodologies. Therefore, design of the maturity model which takes agile/scaled agile methodologies into account should also employ core principles and values into model. Besides core principles and values, there are certain other elements which have to be considered in designing the model. Thereby a systematic literature review is conducted with the goal of identifying main requirements and characteristics of agile and scaled agile software development using agile maturity models. In addition to that, scaled agile methodologies take their roots with agile software development; hence it is essential to analyze already existing agile maturity models in the academic world. Furthermore, by using this approach and answering the second research question, the main pillars for designing scaled agile maturity model can be identified.

This chapter is structured as follows. First a literature review on definitions of “maturity” in terms of software development is studied. Then, maturity models for agile software development are studied, by looking at the related studies in this area. The aim of this exercise is to investigate the main factors used by various maturity models in designing the model. The potential merits and demerits of different maturity models are studied to analyze the elements that have to be considered in the design. Due to large number of publications available in literature, a selection process is defined as why only few models were considered in the review. In the final part of this chapter, key findings from the exploration of maturity models and main design guidelines of the model are drawn.

3.2 Maturity model

3.2.1 Definition and concepts

To get a better understanding on what maturity models are and the benefits of using such a model, the following section discusses the basic characteristics of maturity models. As a first step, a concise definition of maturity model is required. Various definitions used by different authors are explained below.

Glimko et.al (2001) gives a description on maturity models as “Maturity models describe the development of an entity over time. This entity can be anything of interest: a human being, an organizational function, etc.”(G.Klimko, 2001). Willem et.al (2007) defines maturity model as a “Structured collection of elements that describe the characteristics of effective processes at different
stages of development. It also suggests points of demarcation between stages and methods of transitioning from one stage to another” (Willem, P, 2007). In addition to that, many publications refer to the description provided by CMMI as a basic definition in defining a maturity model, which is described as “A framework representing a path of improvements recommended for software organizations that want to increase their software process capability”.

Though these definitions give us the bottom line on the usage of maturity models, we need a clear understanding on elements of such a model and meaning of maturity in general. The term maturity is defined as “the state of being mature: fullness or perfection of development or growth” in oxford English dictionary. Further in the field of Information technology, maturity is normally referred to capabilities, in which capability is described as “the power or ability in general to fulfill specified tasks and goals” (Simpson & Weiner, 1989). Hence from a linguistic perspective maturity models will serve to ensure the conditions of growth when a specific element reaches the perfect state in their intended purposes (Wendler, 2012).

Next characteristic which has to be investigated when such a fullness of growth/maturity is reached is the perspective on the end goal of designing such a maturity model. Tom et.al (2009) defines two perspectives in defining the purpose of maturity models: Life cycle perspective and Potential performance perspective (Tom McBride, 2009). Life cycle perspective measures the state of the growth of the process areas, taking the evolution of an organization into account in evaluating the maturity. This perspective names the final stage as “Fully mature” and represents the ideal stage of maturity. This perspective aims at fulfilling certain characteristics at a particular stage of maturity and can evolve to next stage only if the characteristics are fully performed. Whereas, potential performance perspective defines the maturity stages through means of a development path which focusses more on the potential improvements which might occur by moving along the model. This perspective gives user the final decision to decide which stage of maturity is the “level of completeness” according to their situation (Wendler, 2012).

To summarize, maturity models define and explain the state of perfection or completeness of certain elements or objects and ensures a desired evolution or performance path along clear discrete stages. Though life cycle perspective provides a hard rule in evolving towards a maturity stage, we claim that the user should have final decision to decide the completeness of the element. This is in line with the argument that, every organizations face multitude of challenges and there can’t be a unique solution to all kind if challenges faced by an organization. Hence we claim taking the “Potential performance” perspective into account in our design. Our idea comes in intersection with the definition provided by Becker et.al (2009) “A maturity model consists of a sequence of maturity levels for a class of objects. It represents an anticipated, desired, or typical evolution path of these objects shaped as discrete stages. Typically, these objects are organizations or processes or practices” (Becker, Knackstedt, & Pöppelbuß, 2009) This definition forms the basis of the research in exploring the maturity models, which forms the following section.
3.2.2 Selection criteria

During the process of searching for a range of publications in the domain of agile maturity model, it was noticed that there are more than 40 maturity models published in the literature to date (May 2016). Among the models published, only few models were considered popular in the business world and have proven benefits, through case studies. Since the evaluation of the agile maturity models plays an important role in this research, few additional inclusion criteria’s are established in selecting a maturity model for literature review. The inclusion criteria’s include,

1) The model/ framework should entail a detailed description of the model and should have detailed analysis on the maturation path enlisted in the model.

2) The research/study should have been enlisted in one of the major conference proceedings/journals relevant to IS research domain (such as International conference on Information systems (ICIS), European conference on Information systems (ECIS) etc.), which is considered as an indicator from the academic perspective of the model.

3) The model/framework should entail a discussion on the procedural issues.

Search was based on electronic databases such as Scopus, IEEE and google scholar. The research domains such as Information systems, computer science or business and management are included during the search. Search strings such as “maturity model”, “capability model”, “and process improvement model” were used. Following section provides a literature review on the publications which resulted after the initial scanning of the publications through the selection criteria

3.3 Exploration on maturity models

In the previous section we took a closer look at definition and purpose of maturity models in general. From the literature review, we found two different perspectives in describing the maturity models in information systems domain. In this section, we will take a closer look at the maturity models in literature.

3.3.1 Capability maturity model integration for development (CMMI-DEV)

The Capability Maturity Model Integration is an approach to process improvements in the software development process areas that allow organizations to develop effective development processes (CMMI Product Team, 2010). CMMI includes several models, in which CMMI-DEV is considered as a suitable version of CMMI for software development practices. CMMI-DEV describes best set of practices that has to be included in developing software applications and artifacts from their inception to the stage of departure/deployment. It uses five levels to recommend organization a clear way to carry out the software development processes and has 22 process areas.

CMMI-DEV recommends two evolutionary paths in proceeding with the maturity levels: Staged and continuous (CMMI Product Team, 2010). The staged path focusses on a holistic maturity level in an organization, in which by means of each maturity level, the organization improves an essential set of practices/processes which helps in preparing to the next stage. CMMI-DEV includes a mandatory set of end goals for each level, which has to be achieved to get a certain level of maturity. Though there are definite goals described by CMMI-DEV, it is prescriptive in nature and the organization has to figure out as how to achieve those goals. The continuous representation of CMMI-DEV describes
the assessment of the capability for each process areas such as requirements management, software design, software delivery etc. Thus an organization can choose a set of already named processes which needs improvement, particular to the selected process areas (CMMI Product Team, 2010).

Staged representation of the model describes 5 levels of process improvement from a level of poorly developed process level to a highly mature process level. As discussed, CMMI-DEV has a strict schedule of progress towards the end goal before maturing to the next stage of maturity.

![Figure 10 CMMI-DEV Maturity stages](image)

CMMI-DEV has an advantage of including 2 representations for the model and models of each representation also have their own advantages. However, the criteria on which representation has to be employed in what kind of situation, is missed out in the design. Furthermore, CMMI-DEV describes 22 process areas, which may add overhead in terms of documentation. In order to go through the vast set of documented requirements, organizations may require additional resources for implementation which may additionally pose additional time and effort. The characteristics of the CMMI-DEV makes it more of a waterfall or traditional approach, due to its structured way of progress towards a maturity level and software development teams might oppose such a model as it is not fluid enough to collaborate with teams working in agile culture. For instance when looking at the design flow (Figure 11), we recognize that the organizations implementing CMMI-DEV, should go through three levels of gates in a particular stage. A goal of the process area has practices, and further practices have sub-practices. Level of completion takes a bottom up approach in which, sub-practices has to be fulfilled for the practice to be named as “complete” and again the goals are evaluated for completeness against the completed practices.
Figure 11 Design flow of CMMI-DEV

Following a stage gate model along the maturity paths represents the model is strictly waterfall and lacks agility when there is a change in the requirements along the course of maturity. This clearly follows the life cycle perspective of maturity models and considered as a disadvantage.

3.3.2 Agile maturity model (AMM)
With inspiration from CMMI-DEV, Patel et.al describes agile maturity model, which helps in software process improvement to suit agile way of working more. Agile software practices are more business oriented and therefore AMM aim at aligning software process improvement practices with business objectives. AMM is based on agile principles and is designed to improve the adoption of agile software development methodology in IT organizations, through following dedicated goals at each stage of the model (Chetankumar Patel & Ramachandran, 2009). This follows a maturity path, which gives a high level view on how agile practices mature from an initial or ad-hoc level to a higher level, which entails continuous improvement in the process. Each level has a predefined goal to help the organization or the practitioner to focus only on their improvement activities alongside the particular goal (Chetankumar Patel & Ramachandran, 2009).
The model consists of 5 stages, as illustrated in the figure below,

- Level 1: Initial Level (no goals defined)
- Level 2: Explored (project planning, stakeholders orientation practices)
- Level 3: Defined Level (Customer relationship management, software quality)
- Level 4: Improved (Project management)
- Level 5: Mature level (Performance Management)

Figure 12 Agile maturity model (Chetankumar Patel & Ramachandran, 2009)

**Level 1** Company does not have any defined agile software development practices and follow the traditional waterfall based approaches.

Patel et.al defines an organization with **level 2** capability experiencing fewer problems in software development than level 1. Main goals in maturing to level 2 would be project planning, Improving agile requirements engineering, customer and stakeholder’s orientation practices and enhancing value, collaboration and planning practices.
Level 3 is denoted by more focus on customer relationship management, frequent deliveries, pair programming, communication, coding, testing and quality of software (Chetankumar Patel & Ramachandran, 2009). After enhancing processes involved in planning and requirement analysis in level 2, level 3 focusses more on the technical practices like coding/testing practices, and also on the practices related to customer relationship management. As such, it also represents that when an organization matures to level 3, most of the technical problems are solved but the problems related to human factor such as team level challenges and organization problems exists. This is in contrast to what we assume as agile mindset/culture. Organization with traditional waterfall based approaches will need to know what agile methodologies are, as a first step towards adapting to agile. With line of argumentation that the agile methodologies are a parallel world to traditional waterfall approaches, we argue that the initial stage should focus on illustrating the differences between agile and waterfall, through means of proper training and getting people acquainted to agile way of working.

Level 4 denotes the maturity in project management and people orientation practices. The goals of this level are mostly related to teams. The AMM at level 4 maturity aims to help stakeholders respect for the co-workers, identify and improve problems related to team level such as self-organizing skills, self-assignment tasks and collaboration.

Level 5 maturity helps in continually improving the software development processes involved in an organization through project performance management (Chetankumar Patel & Ramachandran, 2009). The main goals of this level include faster response to changes and defect prevention.

In addition to the maturity levels, Patel et.al also defines a software process improvement roadmap for agile software development practices. They describe an agile adaptability and suitability assessment which helps in effective implementation of agile practices through AMM. The purpose of involving this process is to ensure the readiness of different team members, in adapting to certain level of maturity (Chetankumar Patel & Ramachandran, 2009).

Identification of KPAs (Key process areas) through assessing the maturity level forms the next step in the roadmap. Depending on the goals indicated in the individual maturity level, key process areas of improvement has to be analyzed. AMM assessment questionnaires play an important role at this stage, which would help in identifying the KPAs. Validation of the model is conducted with group of experts; however evaluation of the model in the business world still remains.
To summarize, AMM implementation has a similar structure to CMMI and defines the agile maturity through 5 levels: from initial to sustained mature level. The model has been developed with the purpose of adapting business objectives with SPI (software process improvement) practices and enhances the adaptability of agile software development across the organization. For each maturity level, key process areas, objectives and assessment criteria are defined to help in assessing the maturity level. However, AMM lacks to identify all the essential practices used in agile software development and lacks argumentation behind why the practices included at each level and basis of including certain practices in certain process areas. This makes us question the inclusion of certain practices on certain levels of the model and the consistency of levels. For instance, we argue that the team level should be considered at the initial stage in the model, but not in the stage 4.

3.3.3 ISO/IEC 15504 – SPICE
Moving on to the next model, SPICE (Software Process Improvement and Capability determination) is a set of international standards for software process improvements and provides a maturity model for the capability determination of a particular process in software development. Similar to CMMI-Dev, SPICE includes process areas and six capability levels that can be reached through maturity in process areas. These capability levels are similar to CMMI’s capability levels but are different from the maturity levels of CMMI. For instance, capability and end goals to reach a
particular level resonates CMMI’s structure, whereas the maturity stage is different from CMMI (Connor & Eds, 2013).

![SPICE Maturity stages](image)

Figure 14 SPICE Maturity stages (Connor & Eds, 2013)

Each of the capability levels contains one or more process attributes. Each of the process attributes has one or more practices which need to be achieved in order to achieve a particular process attribute. So clearly it follows the same design flow of CMMI (Figure 11 Design flow of CMMI-DEV). But there are slight differences in the process areas. SPICE and CMMI-DEV, both address continuous process improvement. SPICE defines 6 capability levels whereas CMMI-DEV defines 4 capability levels. This means that a process in SPICE can continue to improve up to the 6th capability level independent of the other processes, whereas in CMMI-DEV a process area which reaches 4th capability level cannot improve itself further independently. To further improve a process area, some high maturity process areas need to be deployed which help in improving all other process areas. This means that even in CMMI-DEV’s continuous process improvement, processes cannot continue to improve independently after reaching 4th level. However the concerns of being waterfall exist in SPICE as well.

### 3.3.4 Scrum Maturity model

Scrum maturity model is defined by Yin et.al which presents a maturity model to help IT organizations to improve the agile software development processes, concentrating on the user’s involvement in the processes through scrum (Yin, Figueiredo, & Silva, 2011). The Scrum Maturity mainly aims to guide organizations through self-improvement and continuous learning, which the scholars think is mandatory on this competitive IT environment. Furthermore, this model explicitly focusses on scrum practices and wishes to help the organization willing to adapt scrum on their team level development processes.
SMM defines 5 levels,
- Level 1 - Initial
- Level 2 - Managed
- Level 3 - Defined
- Level 4 - Quantitatively managed
- Level 5 - Optimizing

**Level 1** is the first and lowest level of maturity, defines an organization that uses Scrum, but lacks any structured process in achieving the goals. In the sense, the organizations which lack in any kind of scrum methodology is classified as belonging to this level (Yin et al., 2011).

**Level 2** is almost similar to that of AMM defined by Patel et.al, with similar goals focused on software requirements engineering and basic scrum management.

**Level 3** is focused more on the customer relationship management and Iteration management, with both goals leading to best customer satisfaction.

**Level 4** of scrum maturity aims in boosting the practices in standardizing the project management and process management.

**Level 5** focus on continuous self-improvement and self-learning practices to help compete with the competitive IT market and helps in achieving higher levels of customer satisfaction and also from other stakeholders.

To summarize, the characteristics in assessing the adaption level of the maturity model is limited to scrum. It provides a perspective for the assessing the scrum maturity in an organization. However, there it lacks in defining a structured process to tell the users an assessment criteria to validate the responses. This model includes seven goals which refer to the process areas and software processes. Since the maturity model focusses on the scrum management principles, certain technical practices are not covered in evaluating the maturity level of the agile process.

### 3.3.5 Agile adoption framework

The Agile Adoption Framework is described as a prescriptive process to guide agile adoption across the organization. AAF has two main elements: The Sidky Agile Measurement Index (SAMI) and 4-stage roadmap flow that utilizes SAMI to determine the organization readiness in implementing agile software development approaches (Sidky, Arthur, & Bohner, 2007).

As described by the scholars, the motivation behind developing a SAMI is mainly due to the main concerns of the organizations to determine the readiness of the team members to adapt to agile. But the agile potential (i.e, the degree to which the entity can adopt agile practices) is determined not only by its practices and team members, but also influenced by the circumstances surrounding them. There is a need for a measurement index to assess the potential agility of an organization (Sidky et al., 2007).

The framework defines 5 levels of agile practices,
- Level 1: Collaborative
- Level 2: Evolutionary
Level 3: Effective
Level 4: Adaptive
Level 5: Encompassing

Level 1 focuses on the team level dynamics and aims at improving the self-organizing nature of agile teams, and aims at improving the people factor.

Level 2 aims at improving the evolutionary nature of the artefact developed, through software configuration management strategies.

Level 3 helps in improving the effective software development practices across the enterprise and tends to focus more on the technical perspective of the model.

Level 4 focusses on the practices which help in uncertainty situations and continuous tracking of emerging requirements.

Level 5 aims at the highest level of maturity, in which the organization is assumed to be completely mature and include all the practices which represent the ideal situation for an agile environment.

The authors ensured that the agile levels had an agility factor into consideration and agile principles play a major role in design. The agile principles introduced in the measurement index directly reflect the agile manifesto introduced in the year 2001 (Highsmith & Cockburn, 2001).

![Figure 15 Matrix with agile levels and principles](Sidky et al., 2007)

The Agile Practices are grouped according to the agile levels and the agile principles. Each of the practices adheres to the agile principles introduced in the columns (Sidky et al., 2007). Once the matrix is populated with the agile practices, couple of approaches is used to assess the achievement of those practices in a particular level. These approaches can be either a single factor or many factors which can be grouped together to collaboratively help in assessing the level of achievement of a particular practice. (Sidky et al., 2007). However the model does not include best practices in the software industry, which guides the software process improvement. For instance the sprint ceremonies such as daily stand up, sprint planning etc acts as a backbone on the team level to embrace agility. These practices are not incorporated in the model, which makes it less usable in the real world scenario.

3.3.6 Robert Benefield’s model

Robert Benefield defines a maturity model which helps in mapping the maturity of processes/practices for independent teams which develop software through agile methodologies. Benefield
defines agile maturity with seven dimensions: Automated Regression Testing, Code Quality Metrics, Automated Deployment and Backout, Automated Builds and Configuration, Management best practices, Interlocked Delivery and Interface Integration Testing, Test Driven Development (TDD), Performance and Scalability Testing (Benefield, 2010). This is one of the few models which look explicitly into technical considerations and its maturity in defining the maturity of the agile practices. In addition Benfield also introduces several definitions in his model, which are considered essential in describing the characteristics of the model.

The following terms are the exact definitions of Benefield used in his publication, pertaining to the seven dimensions of maturity (Benefield, 2010):

1) “Automated Regression Testing – identifying a regression in functionality soon after committing code reduces the cycle time required to find and fix the offending code. Automated regression testing that runs alongside a build allows the process to accelerate and be less intrusive

2) Code Quality Metrics – establishing code standards and building a culture of design and code reviews allows for better management of code maintainability and complexity

3) Automated Deployment and Blackout – it is important to ensure uniformity of packaging and deployment to guarantee consistent behavior between development, test and production environments. Automation further removes the ability for inconsistency, and also allows for deployments to be done automatically as part of a larger build and test automation suite.

4) Automated Builds and Configuration Management best practices – build consistency is important, as is making builds so straightforward to run that they can and are run all the time. Having an effective configuration management strategy, including a standard and known source code version control system and a sensible repository layout, both aids automated builds as well as allows for effective code management.

5) Interlocked Delivery and Interface Integration testing – required to provide cross component delivery transparency of the progression of development work within each component team, and identify important touch points between components. This allows for component teams to better align their work so that they do not hold up integration efforts due to cross component misalignments

6) Test Driven Development (TDD) – practice of having developers writes and executes unit tests to ensure they fail before they write the code to ensure that when code is written, it is only that which is necessary to pass the tests. This ensures that the code is tested effectively to ensure it behaves as the customer would expect it to behave, rather than at best testing the code as it was written than how it was intended to be used.

7) Performance and Scalability Testing- ensuring there is production quality built in, as well as building and improving the capacity and scaling model for the component. This both ensures that development teams are thinking through the performance characteristics of what is being built, as well as provides a potential early indicator if the solution will be cost effective.”
After defining the seven dimensions on maturity, Benefield defines a maturity model with 5 levels. Level 1: Emergent engineering best practices Level 2: Continuous practices at component level Level 3: Cross component continuous integration Level 4: Cross journey continuous integration Level 5: On demand just in time releases.

The goal of **Level 1** is to improve series of inconsistencies among the teams by establishing a common base of software development practices, which will act as a solid foundation for the team to build further practices upon. For this, Benefield introduces certain practices such as unit testing, code reviews, code quality management and common coding standards at this level (Benefield, 2010).

The goal of **Level 2** is to introduce a rhythm in the development activities within the development team. Practices such as automation and build–deploy test scripts are introduced at this level to ensure the same kind of practices are consistently used both in the developmental activities and software production launches.

Within **Level 3**, Benefield introduces practices for continuous integration and to reduce the length of cycle releases. For instance, practices such as continuous delivery, test driven development, Model driven development are introduced to improve frequent releases of software to the customers, in contrast to bigger releases with longer release cycles.

**Level 4** aims at bringing the effective engineering practices as a routine among the development team. Practices such as refactoring, test automation, shorter release cycles and continuous integration are
acquainted and there exists a closer and more organized collaboration between the organization and the customers.

At Level 5, Benfield defines the team to be highly mature in the technical practices and team quickly understands any deviation from the development environment. He defines the level at which the teams continuously produce software which can be released to the customer any time (Benfield, 2010).

To summarize, the Benfield’s model, describes agile maturity in a more technical perspective than a managerial perspective. The main drawback of this model is the lack of description on practices which has to be included in every other stage of maturity, in order for the organization to come up with the assessment of maturity level and key process areas which needs further improvement. Furthermore, the model does not come up with detailed characteristics or practice based goals, which might be helpful for an organization to validate the process improvements areas. Next to that, it lacks visualization of the model which makes it less convenient to look up for stages or practices involved in the stages.

3.4 Characteristics of agile maturity models

The previous sections mainly dealt with investigation of selected maturity models in literature. This section helps in analyzing the characteristics of studied maturity models and builds the primary design guidelines for next phase of the research (Design and Development).

As discussed earlier, maturity models acts as a tool in defining the user how an element matures. In software development field, the element may refer to the agile team, which may use the proclamation used in the maturity models to assess its own situation and find the specific areas of improvement to achieve highest level of maturity (Fontana, Meyer, Reinehr, & Malucelli, 2015). In the past years, process improvement is based mainly on the principles and guidelines stated by the Capability Maturity Model Integration (CMMI-DEV) and International Standard ISO/IEC 15504 SPICE (CMMI Product Team, 2010).

Various other maturity models take its inspiration from CMMI for its design and implementation (C Patel & Ramachandran, 2009). However CMMI-DEV suffers from criticisms for its waterfall approach and its method of documenting various process initiatives. As a matter of fact, when creating any model or framework for agile software development, agility should be at the goal of the model. It has been studied that agility cannot be sustained to the highest maturity levels if the improvement path suggests various process definitions. In addition, it has also been studied that the model will have the highest possible ability to hinder sustaining agility at increased process definition levels (Luksiewicz & Miler, 2012) (Paulk, 2001). It highlights the point that the models which take Life cycle perspective into account through strict regulations to move from stage A to stage B, suffers from serious agility issues. This is due to higher process definitions at one stage which has to be fully accomplished before moving to a succeeding stage. This in turn is contrast to the agile environment which places people over processes, and focusses on adaptive environment instead of a predefined system. Therefore
taking potential performance perspective into account which defines the maturity stage as a development path with focus on the potential improvements or outcomes can help in sustaining agility factor throughout the model. In addition, this perspective gives the choice of final decision to the user, stressing on the fact that there can’t be a unique solution to all kind of challenges faced by the organization.

Furthermore, the agile manifesto describes “placing people and interaction over process and tools”, which forms the major point of discussion at this point (Highsmith & Cockburn, 2001). The Agile manifesto, which forms the backbone of all agile models/frameworks, describes placing people (driven by self-organizing behavior) over process and tools (driven by documentation) and places uncertainty over following a plan. This core value of being adaptive to external changes should also be reflected in the agile models/frameworks developed. If the principle is not stressed enough in a model, then the model cannot have its name as Agile, but rather a generic model.

Further studies also represent “agile teams as complex adaptive systems, which respond to each other’s behavior in order to improve their behavior and thus the behavior of the system they comprise” (D.Stacey, Griffin, & Shaw, 2002). Complex adaptive systems such as an agile or scaled agile team tend to adapt to the changes introduced in the environment in the due course of the project, instead of following a predefined process from the start till the end. This is one of the essential features, which differentiates agile software development approaches to traditional software development methodologies. In line with this argumentation, if the agile and scaled agile teams aim at being mature in this perspective of adapting to external changes through a maturity model, the model should explicitly consider exploration and exploitation of practices into account in adapting to the system. It is in our view there is an essential need to shift the focus from prescribing predefined processes in defining maturity to ambidextrous nature of taking a broader subjective view, in which people play an essential role instead of predefined processes. With this line of argumentation, we claim that maturity of an adaptive system such as an agile system cannot be based on a continuous predefined process, rather a continuous process of combining exploitation and exploration of practices involving agility (Fontana et al., 2015).

Figure 17 Adaptive systems

Based on these characteristics, we build upon three design guidelines for the design. Further these guidelines are translated into practical requirements for the initial design of the scaled agile maturity model.
Design guideline 1 directly corresponds to the criticisms on strict regulation based waterfall approach used in agile maturity models.

Maturing of agile systems cannot be based on waterfall based strict regulations in maturing from one stage to another.

Design Guideline 2 stresses on the importance of practices in the design, as suggested by various other maturity models in literature. The group of essential practices (so called best practices), has been implemented in each of the maturity models. And also be stressed on the fact that, organizations vary in culture and teams are too context dependent. With respect to these practices, the model should rather act as a point of initial discussion or guide to start exploring certain unadopted practices or start aiming at improving the practices through assessment criteria. We also argued about the necessity of ambidextrous nature of the teams in adapting to external changes in contrast to following a predefined process flows. This line of argumentation is translated as design guideline 2 as follows:

Maturing in agility is too context dependent and models should act as a guide leaving space to emergency of behaviors.

Quite a few maturity models lack in argumentation behind choosing certain process areas and inclusion of certain practices in a particular level. This made us question the consistency of the model across different levels. This line of argumentation is translated as design guideline 3 as follows:

Stages/levels of the model should be backed up with an agile factor.

These three guidelines arrived at in this section act as a baseline in the design of the model. The following section deals with the adaptation of these guidelines into practical requirements needed for the design.

3.5 Requirements of scaled agile maturity model

The objective of this section is to determine the requirements in designing a scaled agile maturity model. In order to reach this objective, we explored various agile maturity models existing in literature and came up with 3 essential guidelines to be considered for the same. The first 2 guidelines take the design of the maturity model into account, whereas the third guideline describes the agile factor which needs to be considered in the design. In this section we analyze the design guidelines and translate these guidelines to practical requirements in building the model (Table 6).

Design Guideline 1 stresses on the importance of not being waterfall by introducing strict regulation for each stage. Practically this guideline can be translated into a model only when the model doesn't have any stages of maturity. However, this in turn is in contrast to the notion of a maturity model which helps in denoting how an agile team evolves through stages of maturity. We have two kinds of interesting notions on maturity model at this point. One is that the maturity model should focus on the stages, but should not entail a strict regulation in moving from one stage to another. On the other hand the maturity model, definition per se should claim to have stages of maturity to be called as a
maturity model. In order further discuss on these contrasting opinions, we also consider the design guideline 2 in this argumentation. As a guideline 2, we added the perspective of developing the ambidextrous ability of the teams to mature in scaled agile software development, which is a challenge, given that there is not such a unique recipe for such ambidexterity for every unique problems faced by the organizations. By combining the guidelines 1 and 2, we propose a different notion on maturity models which will act as a guide to the team, in getting to know about the scaled agile practices and its potential outcomes better. This will aim to help in fostering alignment (exploitation) with prospective outcomes but leaving space for the emergence of variety (exploration) by not prescribing or forcing the practices the team should implement, however giving an idea on how the practices can be implemented in stages. In this way, the model supports the discussion for the agile team in exploring the practices. We also claim by this research that the notion on maturity model focusing on establishing specific process areas should change in which teams should focus on adapting practices which remain supportive to their working environment rather than predefined processes and practices established as stages.

Design guideline 3 stresses on the agile factor, which is directly translated into principle, core values and practices corresponding to both agile and scaled agile software development.

3.5 Chapter summary

To summarize, we investigated various maturity models existing in literature and reviewed different aspect of the models such as design flow, merits and demerits of the design constructs, in this chapter. We answered the sub-question,

RQ2. What are the requirements which have to be considered in designing a scaled agile maturity model?

We determined that the main guidelines for the model will include: 1) Agile and Scaled agile practices, principles and core values 2) Ambidextrous nature of the practices 3) Discussion tool in evaluating the nature of practices.

We arrived at these requirements by means of three design guidelines resulted from the literature review of five agile maturity models. The guidelines mainly stressed on avoiding strict regulations in the model, taking context dependent nature of practices into account and agility factor which has to be considered in building a model for agile/scaled agile software development. In addition, we also claimed that agile systems are adaptive systems, and maturity model for such a system cannot be based on a continuous process, rather should be a discontinued process of combining exploitation and exploration of practices involving agility. By combining these findings and other aspects of the model we will focus on the design of the maturity model in the preceding chapter.
Chapter 4
Chapter 4 Design and Development of Scaled agile maturity model

4.1 Introduction
This chapter develops the design of the maturity model for scaled agile development and answers the third sub-question:

**RQ3: How does the initial design scaled agile maturity model look like?**
Based on the sub-question, this chapter is structured as follows. First, the requirements which form the basis of the maturity model is discussed (Section 4.2). After which the requirements are studied in-depth in different sections. Section 4.2.1 discusses the translation of the requirements into the model by explaining the stages of the maturity model. Section 4.2.2 discusses about the agile and scaled agile principles which plays a major role in driving the design towards following scaled agile methodologies. Further, section 4.2.3 deals with the discussion on the scaled agile practices which are included in each stage of the maturity model and the rationale behind choosing specific practices for a particular maturity stage. Section 4.2.4 discusses the indicators which help in determining the agility of the practices at each stage. Finally, section 4.2.5 discusses the assessment criteria which have to be followed in assessing the maturity stage of an organization.

4.2 Translation of requirements into the model
We identified the basic guidelines and requirements of the design in chapter 3. In this section, we provide a brief summary on these requirements and translation of these requirements into the design of the model.

Table 6 Translation of requirements into the model

<table>
<thead>
<tr>
<th>Design guideline</th>
<th>Requirements</th>
<th>Translation into model</th>
</tr>
</thead>
</table>
| 1. Maturing of agile systems is and cannot be based on waterfall based strict regulations in maturing from one stage to another.  
2. Maturing in agility is too context dependent and models should act as a guide leaving space to emergency of behaviors. | Model will act as a discussion tool in stressing the ambidextrous nature of practices. Though the model would not consider the strict process based regulations, it should provide an idea on the possible stages of the scaled agile practices to act as a discussion tool per se. | Each and every stage of the model, should provide improvement measures with clear level of granularity for the teams to discuss on the improvement measures.                                                                                   |
| 3. Stages/levels of the model should be backed up with an agile factor.          | • Agile values  
• Scaled agile values  
• Agile principles  
• Scaled agile principles  
• Agile practices  
• Scaled agile practices | Definition of levels should incorporate the core values or principles of the addressed methodology.                                                                                                                                                                                                  |
4.3 Design pattern of the model

With the requirements of the model discussed, we discuss further on the design pattern of the model in this section. This section helps in analyzing the aspect for building the baseline by using the requirements derived in the previous phase of the research.

As a first step towards constructing the design pattern, we take the design guideline 3, in acknowledging the core values or principles in defining the levels. This is due to the fact that it directly discusses on the design pattern of the model. However, we need a careful orchestration on how we should include the requirements gathered from guideline 3 in constructing the design pattern of the model. The guideline stresses on the necessity of including the agile factor as a point of argumentation in defining the stages or levels of the maturity model. As proposed by Highsmith et.al (2001), the agile factors are composed of agile principles, agile values and agile practices. Hence we claim to use the scaled agile principles, core values and practices in the definition of stages or levels in the model. However we still need to analyze on how these agile or scaled agile factor will form the design pattern of the maturity model.

For this analysis, we confront the design pattern agile methodologies such as scrum, lean or XP used in literature. They were developed, by incorporating agile principles and core values from the agile manifesto. Agile principles and core values formed the subset of these agile practices, which gave a holistic view on how an agile methodology would work when implemented. This helped in stressing more on the agility factor of the design. Following the same line of argumentation, scaled agile maturity model should also focus on agile/scaled agile principles and core values as a subset. As seen in the figure below, the scaled agile maturity model will include scaled agile/agile practices, which will further point to the core values and principles of agile and scaled agile methodologies.
With the pattern of the design being figured out, we still need to analyze the position of these agile factors in the model. We again take inspiration from the design pattern of the agile methodologies. Each practice is developed by taking the agile principles and core values into account. We apply same kind of logic in the design in which a scaled agile/agile practice will take the scaled agile/agile principle or value into account, thus forming a AxB matrix of principles and values, where A represents number of rows and B represents number of columns.

![Figure 20 Initial Design pattern](image)

Populating the model will give rise to a X*Y matrix (Figure below), where X represents the rows (Core values) and Y represents the columns (Principles). This will form the initial design pattern of the model.

![Figure 21 Design pattern of the model](image)
4.3.1 Stages of the maturity models

With the design pattern in hand, it is essential to discuss the elements used in populating the model. First being the stages which has to be used in defining the maturity path in the model and also forms the primary element to be considered in a so called maturity model.

At this point of discussion, we would like to bring into consideration the contrasting design constructs about evaluating the stages. As already seen, our first design guideline in designing a scaled agile maturity model is,

“Maturing of agile systems is not a predefined continuous process and cannot be based on waterfall based strict regulations in maturing from one stage to another”.

This entails that, organizations should not be forced to adopt a particular practice in order to move onto the next stage of the maturity model. However, this is in complete contrast to the notion of maturity models in general. Maturity models act as a guide in describing the audience how an element matures. Element in this case is a practice, which is placed in subset of principle and core values. When a practice evolves, it will follow a general stage of maturation from the beginning till the end of maturation. Therefore if a practice has to evolve, it has to follow a maturation path in stages. Organization cannot simply have a matured practice, unless the practice has gone through a path of maturity. In order to avoid the contrasting opinions on stages being a continuous process and discontinued process, this research proposes the process improvements as desired state and not a mandatory improvement, thereby giving space for the organization to explore the practices. Hence the stages will form a guide in establishing the maturation path for stages and not force the organizations to adopt a certain practice. As already discussed, agile software teams are complex adaptive systems and the practices have to be explored against the new outcomes, which will in turn generate new practices in the future (Fontana et al., 2015). That being said, this research will not propose that the organization is fully scaled agile when it reaches the end maturity level proposed with this study. There is always a road for improvement and this study is a guide on how an organization can continuously learn to adapt to new practices and outcomes. Hence the fully scaled agile environment can never be reached at the organization level as the entire system of agile and scaled agile is always evolving.

4.3.1.1 Defining the stages

The next challenge after establishing the relation between the practices and the levels is to define the maturity stages of the model. As already mentioned, each of the agile levels would introduce an essential scaled agile core value. Hence the starting point of defining the level, should take the core values mentioned in the agile manifesto (agile software development) and scaled agile process into account.

The agile manifesto states as follows: (Highsmith & Cockburn, 2001, p.p 11)
“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

In addition to the values proposed in the manifesto, review of body of literature through several research studies helped in identifying the core values of agile and scaled agile software development in Chapter 2 Agile and scaled agile software development, as described below.

Table 7 Mapping of core values

<table>
<thead>
<tr>
<th>Core value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Responding to change (Sidky &amp; Arthur, 2007)</td>
<td>The paradigm shift from the traditional software development methodologies are mainly motivated for its characteristics to respond to frequent changes in the goals.</td>
</tr>
<tr>
<td>2. Communication and collaboration (Highsmith &amp; Cockburn, 2001)</td>
<td>This value represents the quality of fostering communication between different stakeholders involved in the software development.</td>
</tr>
<tr>
<td>3. Producing quality software (Highsmith &amp; Cockburn, 2001) (Sidky et al., 2007)</td>
<td>This value seeks to employ practices that help in developing quality software, in order for the team to adapt quickly to change.</td>
</tr>
<tr>
<td>4. Agile mindset and environment</td>
<td>Agile environment that helps in increasing the characteristics of the agile nature in the development process.</td>
</tr>
<tr>
<td>5. Continuous delivery of software (Sidky et al., 2007)</td>
<td>This value focuses on continuous, incremental and evolutionary development of software.</td>
</tr>
<tr>
<td>6. Alignment (Leffingwell, 2011)</td>
<td>Alignment of teams with the enterprise business objectives: This starts from the portfolio level through program backlogs to the team level.</td>
</tr>
<tr>
<td>7. Transparency (Leffingwell, 2011)</td>
<td>Enterprise level transparency starting from portfolio level to team level.</td>
</tr>
</tbody>
</table>

Rows in **blue** represent the core values of both agile and scaled agile development process, whereas the rows in **green** explicitly represent the core values of scaled agile software development. It has to be noted that the values representing the scaled agile process, takes the core values of team, portfolio and program level into account. After identifying the key values of agile and scaled agile software development, the next step is to convert them into stages.
Though there are seven core values sorted out, few core values have similar kind of characteristics and support similar practices. For instance the core value on communication and collaboration presents a core idea on fostering communication between the stakeholders in the organization. It is similar to the core idea on increasing the transparency within all the levels of the scaled agile system such as program, portfolio and team levels. In the sense, the core value on transparency attributes to fostering collaboration and communication among all the stakeholder, but in an indirect notion. Hence mapping the core values based on similar notion will help in next stage of sorting out the similar practices that can be grouped in a particular stage/level.

Figure 22 Mapping of core values

The values “communication and collaboration” and “Transparency” are mapped together to form an agile level. Reason behind is that, value on communication and collaboration stresses a proper collaboration with different stakeholders involved in the process. Whereas the value on transparency, requires enterprise level transparency and building trust between different stakeholders involved in the process, and that they have clear understanding of the goals of each business objective. Hence the underlying characteristic of being able to have a collaborative environment has been mapped out from these two values. We chose to use the

The values “Agile mindset” and “Alignment” are mapped together to form an agile level. Agile mindset is the environment in which the stakeholders involved in the whole process are fully agile and engages in supporting the agility of development processes. In a similar fashion, the value on Alignment stresses on the foundation for the enterprise where business objectives and outcomes are fully aligned and supportive of the scaled agile nature of the processes. Hence the underlying characteristic of the agile/scaled agile mindset of being supportive to the process has been mapped out from these two values.

This leads to five core values, which will be translated into five maturity stages in the next section.
Table 8 Final mapping of core values

<table>
<thead>
<tr>
<th>Agile value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication and collaboration;</td>
</tr>
<tr>
<td>Transparency</td>
</tr>
<tr>
<td>Responding to change</td>
</tr>
<tr>
<td>Producing quality software</td>
</tr>
<tr>
<td>Agile mindset; Alignment</td>
</tr>
<tr>
<td>Continuous delivery of software</td>
</tr>
</tbody>
</table>

4.3.1.2 Maturity stages

After the overlapping core values in translating into stages are discussed, the next step is to provide an appropriate sequence of the stages, which provides the staged approach in adopting Scaled agile methodologies. The order of maturity stages takes inspiration from the SAMI (Sidky agile measurement index) studied in 3.3.5 Agile adoption framework. SAMI takes the agile manifesto and various other organizational agile articles into account, in ordering the stages. These studies set the basis of structuring the stages. The first stage focuses on supporting clear communication between the stakeholders involved in the process, which focuses mainly on the “people and interaction” factor mentioned in the agile manifesto. This forms the basic pillar of adoption of agile and scaled agile methodologies. Without proper communication at place, none of the agile/scaled agile practices can be considered to be efficient. Hence, the value on communication and collaboration forms the stage 1 of the model.

![Stage 1 mapping](image)

Figure 23 Stage 1 mapping

Next to the value of a communication and collaborative environment, agile and scaled agile practices stress more on incremental approach, rather than big upfront design proposed by the traditional methodologies. They focus on creating small increments through time boxed effort called as sprint and potential increments throughout the process. Hence the agile and scaled agile practices claim to achieve continuous development and delivery of the software which makes it distinct to traditional methodologies of building software. Hence we claim to propose the core value on “continuous delivery of software” in stage 2. There are two different reasons on why other values are not considered in stage 2. Value which produces quality software cannot be introduced before the value
on responding to change, as the practices involved in producing quality software also depends on the progressive nature of software development in responding to changes. Hence the value on “responding to change” should logically follow the value on “producing quality software”. Nevertheless value on producing quality software cannot be introduced before the value on “continuous delivery of software” is in place. Hence the value on “continuous delivery of software” takes the stage 2, value on “producing quality software” takes the stage 3 and value on “Responding to change” take the stage 4 in the model.

![Figure 24 Stage 2 and 4 mapping](image)

Moreover, the value, which stresses on quickly adapting to constant changes, cannot be introduced before the value on responding to change, to make sure the practices involved in software development are of good quality and progressive in nature. This should be considered to make sure that changes in the middle of the process will not collapse the software and thereby lead to complete jeopardy. This forms the ordering of stages, as indicated in the following figure.

![Figure 25 Stage 3 and 5 mapping](image)

4.3.2 Agile/Scaled agile principles
Agile and scaled agile principles form one of the essential building components of the maturity model and have to embody into a model on agile/ scaled agile. This section describes about the principles considered in populating the model and relation between the maturity stages and the principles.
4.3.2.1 Relation between maturity stages and principles

Before moving on to describing the agile principles, it is necessary to elucidate the relation principles would entail in the model with respect to the stages illustrated in the previous section. Definition of relation takes inspiration from the analogy used by Wake in defining the user stories, in which the scholar illustrates the user as a multi-layer cake and each layer represents the point of view of a stakeholder in the development process. In this way, when a particular part of user story is sliced, it will have all the layers in it (Wake, 2003). The same concept is applicable in defining the relation between principles and stages in the model. Each of the agile levels, made up of scaled agile practices, is adhered to the agile principles. Hence agile levels and principles form a multilayer cake, in which the practices corresponding to each level will encompass both the adhered agile principles and corresponding maturity stage.

4.3.2.2 Agile and Scaled agile principles

After defining the relation of principles across the maturity stages, it is essential to identify the principles that the model will include in the design. Principles used in various maturity models and agile/scaled literature were studied in the previous chapters (Chapter 2 and Chapter 3). We provide a list of all principles derived from the literature and investigate the principles in this section. This section will also illustrate the mapping of principles into different groups, which can be included in the design. Literature review helped in getting a holistic picture of the principles included in various maturity models and agile/scaled agile methodologies. First, we introduce the principles we derived at from the maturity models, as shown in table 10. Most of the maturity models stress the principles on team level management (Human-centric principles), Customer relationship management (Stakeholder collaboration) and Technical skills (Expertise needed for effective software development methodologies). Benfield’s maturity model explicitly focusses on the technical expertise solutions needed for an organization in improving the effective software quality and practices, illustrates the importance of including the technical expertise in the model. Further human centric factor has been stressed upon in each of the model, but along various stages of the maturity model. For instance, AMM includes people factor in stage 4, whereas the agile adoption framework introduces the people factor in stage 1. This helps in determining the people factor plays an important role in agile and scaled agile systems.

Table 9 Principles used in maturity models

<table>
<thead>
<tr>
<th>Maturity model</th>
<th>Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMM – Agile Maturity model (Chetankumar Patel &amp; Ramachandran, 2009)</td>
<td>Requirements engineering, Customer relationship management, frequent deliveries, pair programming, communication, coding, testing and quality of software, Motivation of team</td>
</tr>
<tr>
<td>SMM - Scrum maturity model (Yin et al., 2011)</td>
<td>Customer relationship management, Iteration management, software releases, team and people management.</td>
</tr>
</tbody>
</table>
Agile adoption framework, SAMI (Sidky et al., 2007) | Plan and deliver software frequently, Human-centric (relationship management), Uncertainty management, Changing requirements, requirements engineering, Technical expertise, pair programming, stakeholder management, Dependency management.


SPICE (Connor & Eds, 2013) | Team level management, customer relationship management.

Benefield’s model (Benefield, 2010) | Automated testing, coding standards, pair programming, frequent software releases, scrum, reduced length of release cycles, dependency management, code refactoring practices, High speed development environments.

Next to analyzing the principle involved in the maturity models, we analyze the principles involved in scaled agile and agile methodologies in general.

Agile principles
- Responding to change
- Frequent releases
- Incremental building
- People factor
- Technical excellence
- Continuous delivery
- Working software
- Uncertainty management
- Fast integrated learning cycles

Scaled agile principles
- Develop in cadence, cross domain planning
- Dependency management
- Economic view
- Systems thinking
- Alignment
- Responding to change
- Incremental building
- People factor
- Working software
- Decentralize decision making

**Figure 26 Agile and scaled agile principles**

After the initial set of principles obtained from literature, we now map the principles in different arenas, so as to introduce the principles as the columns in the design, as shown in the figure below.
Figure 27 Mapping of principles
**Principle 1: People and interactions**
As defined in the agile manifesto, agile and scaled agile principles rely on people and interactions and this forms the cornerstone in defining the processes and models, which take agile methodologies into account. In addition the people factor has been emphasized in almost all the agile and scaled agile models and methodologies.

**Principle 2: Technical expertise**
Agile manifesto and Scaled agile principles state the commitment towards producing the highest quality of software code possible, which is an essential characteristic described in most of the literature involving agile methodologies.

**Principle 3: Incremental building**
Agile and scaled agile processes focus on frequent and incremental software releases and continuous learning process, when the requirements arise as a continuous phase. This also includes synchronizing the business objectives across the team, program and portfolio units and embracing change in delivering value to the customer.

**Principle 4: Customer and stakeholder collaboration**
Next to that, customer and stakeholder collaboration plays an important role in adapting to agility across the organization. If the stakeholders and customers are not involved in the process, it may lead to huge customer dissatisfaction and may further lead to business failure. Hence customer and stakeholder collaboration plays an important role in ensuring that product is being developed according to the needs of the stakeholders. At this point after identifying the principles and maturity stages, the maturity model has two dimensions, which will form the layers of the model as shown in figure x. To complete the design, the layers will be populated in the next stage with agile/scaled agile practices, adhering to the principles in each of the layers.

**Table 10 Initial design of maturity model**

<table>
<thead>
<tr>
<th>Maturity stages</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.3.3 Agile and Scaled agile practices

This section highlights the practices populating the model in adherence to the principles and the maturity stages. Maturity stages form the basic cornerstone in deciding the maturity of the scaled agile methodologies followed in an organization. As discussed before the scaled agile and agile practices forms the subset of scaled agile maturity model, which is further composed of scaled agile/agile practices. Thereby each stage will be composed of set of practices used in the development activities of agile and scaled agile software development. The list of agile and scaled practices studied in the previous chapters forms as an input at this stage of the design. As a first step, we identify the practices followed in the agile methodologies; explicitly looking at scrum process defined in Chapter 2 Agile and scaled agile software development.

Next to that, we identify the scaled agile practices followed in scaled agile methodologies, explicitly looking at the SAFe process identified in Chapter 2 Agile and scaled agile software development. Some of the highlighted practices in the figure below represent the extensive scaled agile practices used in the scaled agile methodology in addition to agile practices examined in the scrum process. Since scaled agile practices takes team level agile practices into account, it also includes the team level scrum practices such as scrum ceremonies, people factor and software quality practices into account. The practices enlisted in the figure below represents the practices performed at the program and portfolio level in an organization.

![Figure 28 Scrum practices](image-url)
Next to identifying the practices involved in agile and scaled agile methodologies, it is necessary to analyze the practices included in various maturity models investigated in Chapter 3. This is shown in the table below:

### Table 11 Practices involved in studied maturity models

<table>
<thead>
<tr>
<th>Maturity model</th>
<th>Agile Practices</th>
</tr>
</thead>
</table>
| AMM – Agile Maturity model (Chetankumar Patel & Ramachandran, 2009) | 1. Collaborative planning  
2. Collaborative teams  
3. Test driven development  
4. Continuous integration  
5. User stories  
6. Coding standards  
7. Customer commitment to work with developing team  
8. Frequent (co-located) face to face interactions |
| SMM - Scrum maturity model (Yin et al., 2011) | 1. Iteration planning  
2. Release management  
3. Reflect and tune process  
4. Requirement analysis  
5. Client driven iterations  
6. Estimate the scope of the project  
7. Release planning schedules  
8. Acceptance testing  
9. No overtime  
10. Project planning  
11. Story cards  
12. Mutual interaction |
| Agile adoption framework, SAMI (Sidky et al., 2007) | 1. Agile project estimation  
2. Risk driven iterations  
3. Maintaining a list of all features (features prioritization)  
4. Continuous delivery  
5. Collaborative planning  
6. Adaptive planning  
7. Plan features, not tasks  
8. Evolutionary requirements  
9. Continuous customer satisfaction feedback  
10. Planning at different levels  
11. Reflect and tune process  
12. Unit tests  
13. Software configuration management  
14. Tracking iteration progress  
15. Kanban systems |
|---|---|
| Benefield’s model (Benefield, 2010) | 1. Unit tests  
2. Test automation  
3. Coding standards  
4. Code refactoring  
5. DevOps  
6. Continuous integration  
7. Process automation  
8. Feedback to customer  
9. Change management  
10. Frequent releases to production  
11. Pair programming |
| CMMI – DEV process areas (CMMI Product Team, 2010) | 1. Configuration management  
2. Project planning  
3. Product and process quality assurance  
4. Requirements management |
| SPICE (Connor & Eds, 2013) | 1. Requirements management  
2. Configuration management  
3. Change management  
4. Continuous improvement practices  
5. Process management  
6. Software configuration management  
7. Adaptive planning  
8. Customer engagement practices |

**4.3.3.1 Identifying practices for stage 1**

After identifying the list of practices from different sources, the next step is to populate the model with the practices which is considered essential to recognize the objective of stages. Firstly, we aim to
populate the model for stage 1, which focusses on core value of increasing communication, collaboration and transparency between the stakeholders involved in the process. As already discussed, the stages form a guide to the practices which helps in increasing agility. As such, this stage of the model helps in introducing to few practices which helps in increasing the effective collaboration among the stakeholders. The practices which can be grouped under this category is shown below. The argumentation on the placement of the practices in certain columns can be seen in the following sections pertaining to principles.

Table 12 Practices involved in stage 1

<table>
<thead>
<tr>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-organizing teams</td>
<td>Pair programming</td>
<td>Collaborative planning</td>
<td>Customer commitment to work with developing team</td>
</tr>
<tr>
<td>Empowered and motivated teams</td>
<td>Knowledge sharing</td>
<td>Epic Kanban systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coding standards</td>
<td>Status reporting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adaptive and reflective tuning</td>
<td></td>
</tr>
</tbody>
</table>

Principle 1: People and interactions
The first agile principle, human-centric practices in which the organization realizes human factor of the agile software processes. However taking the objective of this level into account, it is necessary to define those practices which will help in establishing a supportive and a collaborative environment. At this level it is of utmost important to be able to have empowered and self-organized teams, which is helpful in establishing a collaborative environment.

Principle 2: Technical expertise
The practices relevant in enhancing a collaborative environment are pair programming, knowledge sharing and coding standards. Pair programming helps in collaborative behavior of the team since it involves two or more people jointly programing a specific piece of software. In this way, it helps in increasing the collaborative nature of the agile teams. Knowledge sharing tools such as wikis and blogs helps in elevating the collaborative environment and transparency by supporting the learning behavior among the people. Coding standards facilitate in establishing a common language among the stakeholders, thereby enhancing the collaborative environment.

Principle 3: Incremental building
With the objective of frequent and incremental software builds and continuous learning process, it is essential for the stakeholders to gather collaboratively come up for planning activities. Hence collaborative planning is a powerful tool in proper communication and rationale behind choosing this in the first maturity stage is that collaborative planning helps in step forward towards a collaborative
environment, which forms a base for other practices in the next stages. In addition incremental building requires tool-wise improvements such as Kanban systems and status reporting functionalities increase the project visibility to the stakeholders. Further, it is also essential for the team to adapt the lessons learned in the previous process to improve the collaborative and reflective environment. This plays a major role in collaboration because of its goal in planning activities together as a team.

Principle 4: Customer and stakeholder collaboration

The agile principle, “customer and stakeholder collaboration,” supports a collaborative environment, so that the stakeholders are well informed of the preferences of the customer. Hence involving “customer commitment” as a practice in stage 1 is considered essential, to stress the core value of agile in getting inputs from the customer at every stage. This helps in building a routine among the stakeholders to respond to the changes attributed by the customers in the course of the project.

4.3.3.2 Identifying practices for stage 2

The objective of stage 2 is to deliver continuous delivery of software through progressive and evolutionary techniques which help in delivering the software in shorter cycles. The following section gives a brief discussion about the maturity stage 2 and practices used in the particular stage.

Table 13 Practices involved in stage 2

<table>
<thead>
<tr>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 2</td>
<td>• Automated testing</td>
<td>• Smaller and more frequent releases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Acceptance testing</td>
<td>• Requirements engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tracking iteration progress</td>
<td>• Release planning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Progressive requirement discovery</td>
<td>• Continuous delivery</td>
<td></td>
</tr>
</tbody>
</table>

Principle 2: Technical expertise

Continuous delivery of software greatly emphasizes the technical excellence measures that have to be in place for improving the software quality. Automated testing is another practice which helps in comparison of actual outcomes with the predicted outcomes. Manual testing in some cases might not be effective in finding the defects of the software. Test automation helps in automated testing of these activities in an effective manner. This entails continuous delivery of software in smaller and frequent releases. Next to that acceptance testing helps in determining whether or not the software system has met the requirements specifications. Acceptance testing helps in improving continuous software
releases to a great extent and therefore improves the progressive nature of the release (Benefield, 2010). Continuous improvement can occur only when the requirements are involved iteratively in each every step of the process. Hence the requirements should be incrementally evolved instead of being fully coded and designed in one go. This stresses on the practice of including progressive requirement discovery in stage 2. As the requirements are bound to change, progressive requirements discovery helps in yielding the best value to the customer and helps in embracing change. From a more technical perspective, it is crucial to track the progress of each iteration and the development effort, which enhances the progressive nature of the maturity level. By applying these practices in a continuous fashion for each and every iteration, the teams might get acquainted to the continuous fashion of delivering software, which plays a significant role in the practices involved in the further stages.

**Principle 4: Incremental building**

Incremental building can be effective only when the software development cycle takes the notion of smaller and more frequent releases into account. In addition to that, at each every cycle the team should implement a requirements engineering span in which the requirements are revised and new requirements are defined. This helps in continuous delivery of software in response to change in requirements. The purpose of release planning is to commit to a plan for delivering an increment of software for a specific time period (Leffingwell, 2011). It is a progressive effort involving many of the stakeholders for ranking the product backlog, gathering inputs from the team about overall capabilities and technical impacts and high level vision of the business objectives. This planning helps in organizing smaller and more frequent releases, which helps in improving the continuous delivery of the software.

### 4.3.3.3 Identifying practices for stage 3

Now that the transparent and collaborative environment is instilled in the development process, and teams focus on building systems incrementally, the objective of stage 3 is to increase the quality of the software and productivity of the development processes. Since stage 4 focuses on robust practices (embracing change), stage 3 should ensure that the practices are effective and stable. The following section gives a brief discussion about the maturity stage 3 and practices used in populating the model.

**Table 14 Practices involved in stage 3**

<table>
<thead>
<tr>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
</table>
| Stage 3                 | ● Scrum of Scrum    | ● Continuous integration  
                          | ● Unit tests  
                          | ● Refactoring and continuous improvements  
                          | ● Agile release train  
                          | ● Roadmap  
                          | ● Product backlog  
                          | ● PSI  
                          | ● DevOps |


**Principle 1: People and interactions**

Scrum of scrum plays an important role in scaling scrum to large project teams and scaling up agile methodologies across various projects. In simple words, it helps in scaling the agile ceremony on daily stand up to program level. This meeting is organized at the program level in which one representative from each scrum team collaboratively come together and discuss the technical issues encountered during the Program increment. Hence it helps in feedback cycle and increase in efficiency of the developed software. Rationale behind choosing SOS in stage 3 is that the practices involved in stage 4 consists of even more stable and robust practices, which needs effective team building as base.

**Principle 2: Technical expertise**

The first practice supporting this principle is “continuous integration”. This helps in continuously integrating the new changes to the system, so that the system is always ready to be launched to the customers (Leffingwell, 2011). After writing the code for developing an increment of the software, it has to be integrated to the existing code base. It is an ideal situation in which the team employs automated integration process at every increment of code developed. Automated build tools are employed to verify such integration, in order to detect any errors in the integration process. This helps in continuous improvement of the software code base and helps in embracing change at shorter lead times (Leffingwell, 2011). In addition to that, refactoring can be considered at this stage. Refactoring the code is a practice done by the developers to restructure the codebase, without any disturbances to the interfaces the codebase is in contact with. This helps in responding to change, without collapsing other parts of the system. Next to that, Unit tests are introduced in stage 3, to employ the testing of those units that have integrated to the code base. This helps in maintaining the superior quality of the code.

**Principle 3: Incremental building**

The practices included in stage 3 of the model focus on the synchronization factor of the scaled agile process. Agile release train is the primary construct which needs to be considered after the team’s ability to deliver frequent releases. As discussed before, agile release train helps in creating common vision on the program level and helps in achieving alignment and synchronization across the teams. The rationale behind including these practices in this stage is that trains provide architectural, user experience and engineering guidance to the teams, which forms the basis of stage 4. Agile release train plays an essential role in building the potentially shippable increment at each stage of the project. Roadmap helps in stimulating the vision of the software development life cycle for the next few sprints and helps in giving an overall picture on how the software should evolve in the future. This helps in improving the efficiency of the practices. Product backlog in addition helps in synchronization of team level backlog with program level backlogs.

**Principle 4: Customer and stakeholder collaboration**

With the principle of producing quality software in mind, we introduce the practice of DevOps in this section of the model. DevOps is a practice in which the developers (who develop the code) and operations team (who launch the code to the production environment) need to go hand in hand in their activities. For every integration made by the developers into the code base, deployment readiness of the code to the customers has to be checked as an automated process, instead of doing larger releases (Leffingwell, 2011). This process helps in developing a collaborative environment between the
stakeholders (developers and operation team) and also stresses on the software quality. This forms the rationale behind choosing this practice in this stage.

### 4.3.3.4 Identifying practices for stage 4

The practices in maturity stage 3 focused on increasing the software quality by introducing software development practices that help in automating the development practices. With these practices in hand, we now focus on stage 4 which helps in embracing change to the process. This section briefly describes the practices in creating such an environment.

**Table 15 Practices involved in stage 4**

<table>
<thead>
<tr>
<th>Stage 4</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Managing highly distributed teams</td>
<td>• Building Architecture runway (intentional architecture)</td>
<td>• Adaptive planning</td>
<td>Close collaboration with the customers</td>
</tr>
</tbody>
</table>

**Principle 1: People and interaction**

Highly distributed teams are the teams which are globally distributed and not in the same physical location and work remotely. Distributed teams pose challenges such as time differences and delay in process due to its very nature of being distributed. Effective communication plays an important role in creating an effective distributed environment which is prone to several changes during the course of the project. This practice is relevant only to organizations who have highly distributed teams in different ends of the globe.

**Principle 2: Technical expertise**

Building architectural runway is an essential practice at this stage, because of its true ability of the practice to respond to frequent changes in the goals. Software development activities involve large number of interfaces with the external environment. These interfaces and their connections are illustrated by means of software architectural design. This practice of identifying the interfaces usually occur before the start of the project and left untouched after the development activities are scheduled. Architectural runway is a process in which the underlying architecture of the software being developed is also developed incrementally, without big upfront design of the architecture. This helps in increasing the agility also across the architectural level.

**Principle 3: Incremental building**

Adaptive planning helps in building a robust process. This practice helps in following a process which takes multiple feedback loops during the course of software development. Adaptive planning helps in including people first, in contrast to predictive planning in which process comes first (waterfall approaches). Adaptive planning requires emergent architecture and it is possible only through few engineering practices such as continuous integration, refactoring, and automated tests. This is the rationale behind choosing adaptive planning at this stage which requires the above mentioned
practices to be fulfilled (Leffingwell, 2011). Measuring the business performance plays an important role in quality software due to its importance in viewing the success of the development efforts as a success of the business. Only when the efforts in measuring the business performance is effectively distributed across the development efforts, the team could claim an effective development practice.

Principle 4: Customer and stakeholder collaboration
Scaled agile development is highly dependent on the customer engagement to a greater extent. Closer means of engagement helps in robust software development process when building a general solution that will be used by a significant number of customers. It is highly unlikely to include customer for all the planning sessions involved in the development process. However to build a robust environment which focusses on the changing requirements, closer collaboration with customers is essential. It can be achieved by means of a customer proxy, who can involve in various planning sessions.

4.3.3.5 Identifying practices for stage 5
Maturity stage 4 is built upon practices which help in responding to change and technical foundations, established from maturity stage 1 to 3. Stage 4 takes the development process helps in improving the scaled agility, by supporting practices that help in responding to change. The next level of maturity focusses on the internal environment that is supportive of the scaled agile nature of the software development process. This section presents the scaled agile practices that enable this maturity stage to attain its objective.

Table 16 Practices involved in stage 5

<table>
<thead>
<tr>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 5</td>
<td>Low process ceremony</td>
<td>Scaled agile estimation</td>
<td>Product and solution management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System and solution engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Inspect and adapt workshop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PI Planning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Process management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Innovation and planning</td>
</tr>
</tbody>
</table>

Principle 1: People and interactions
The first agile practice on low ceremony suggests the true agile nature of the practices. Process ceremony refers to the level of strict regulations and paperwork needed in an organization. This also stresses on decentralized decision making across the product arenas. In order to increase agility,

---

3 Customer proxy represents a person who represents the customer and presents the view of a customer. This practice of using a customer proxy is practices in organizations for quicker feedbacks and shorter lead times.
decision making responsibilities should be decentralized and strict regulations should be smoothened out.

**Principle 2: Technical expertise**

Most of the practices involved in the earlier stages are related to technical practices, whereas maturity stage 5 focuses on the most fundamental point of estimating the agility of the plans and practices. Most of the technical practices involve estimation of results at the continuous phase. Though estimation of results can happen at the continuous phase, it is irrelevant to estimate when some practice is not practiced in the organization. Hence only when the practice is at place and the team acknowledges the necessity of the practice in increasing the agility, estimation will prove effective. This forms the rationale behind introducing scaled agile estimation in stage 5. This is an important practice to estimate the agility of the practices involved in the previous stages, because plans and practices can be measured for the level of achievement, only the metric estimates are in place. At the same time highly responsive systems, need flexible estimating perspective to measure the agility. Hence metrics which are used in estimating the scaled agility, are context-specific and depends on a specific team.

**Principle 3: Incremental building**

Incremental building helps in reaching the objective of smaller and more frequent releases with shorter lead times. In order to fully reach this objective, it is essential for an organization to involve the specific group of people to take the responsibility in validating the continuous delivery of software. These teams are identified as Product/solution management team and System/solution engineering team. These team of people works closely with the product owner of the development team in imparting vision and backlog of tasks that needs to accomplished for a particular iteration (Leffingwell, 2011). System and solution engineering plays a critical role in aligning the development teams towards a common technical direction. This helps in creating an internal environment of scaled agility across the organization. In addition, the practice on inspect and adapt helps in achieving the highest maturity in incremental building. This is due to the fact that inspect and adapt workshop helps in discussing the next step for improvements and acknowledge the failures in the previous cycles. This helps the team in achieving ideal results in a smaller and more frequent basis. Next to that, as already discussed in the scaled agile process, PI planning helps in building a scaled agile environment through alignment of several teams working on a project. As denoted by few scaled agile practitioners “If there is no PI planning, then you are not doing Scaled agile, it plays a vital role in scaling the agile practices to the entire organization” (Leffingwell, 2011).

**Principle 4: Customer and stakeholder collaboration**

Scaled agile process defines various roles and responsibilities across the team, portfolio and program level. Clarity on the process it needs to be followed across different teams that work together, forms the primary ingredient in scaling agility across the organization. Closer collaboration and clear identification of roles and responsibilities helps in achieving the highest level maturity, adherent with this principle. Innovation and planning is a practice in which the stakeholders conjointly involve in discussing new initiatives and strategies in expanding the business. This can happen only when the current initiatives are effective and robust. Hence innovation and planning plays an important role in stakeholder collaboration and effective business agility.
4.2.4 Initial design of Scaled agile maturity model

The previous sections dealt with the rationale behind choosing a particular practice in their respective stages of maturity. The complete design of the model is shown below:

Table 17 Initial design of the model

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Self-organizing teams</td>
<td>P3 Pair programming</td>
<td>P6 Collaborative planning</td>
<td>P10 Customer commitment to work with the development team</td>
<td></td>
</tr>
<tr>
<td>P2 Empowered and motivated teams</td>
<td>P4 Knowledge sharing</td>
<td>P7 Epic Kanban systems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5 Coding standards</td>
<td>P8 Status reporting</td>
<td>P9 Adaptive and reflective tuning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 2</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Automated testing</td>
<td>P5 Smaller and more frequent releases</td>
<td>P10 Customer commitment to work with the development team</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2 Acceptance testing</td>
<td>P6 Requirements engineering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3 Tracking iteration progress</td>
<td>P7 Release planning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4 Progressive requirement discovery</td>
<td>P8 Continuous delivery</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 3</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Scrum of Scrum</td>
<td>P2 Continuous integration</td>
<td>P5 Agile release train</td>
<td>P9 DevOps</td>
<td></td>
</tr>
<tr>
<td>P3 Unit tests</td>
<td>P6 Roadmap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4 Refactoring and continuous improvements</td>
<td>P7 Product backlog</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P8 PSI</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 4</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Managing highly distributed teams</td>
<td>P2 Building Architecture runway (intentional architecture)</td>
<td>P3 Adaptive planning</td>
<td>P5 Close collaboration with the customers</td>
<td></td>
</tr>
<tr>
<td>P3 Product and solution management</td>
<td>P4 Measuring business performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4 System and solution engineering</td>
<td>P5 Inspect and adapt workshop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5 Inspect and adapt workshop</td>
<td>P6 PI Planning</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 5</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Low process ceremony</td>
<td>P2 Scaled agile estimation</td>
<td>P3 Product and solution management</td>
<td>P7 Process management</td>
<td></td>
</tr>
<tr>
<td>P4 System and solution engineering</td>
<td>P5 Inspect and adapt workshop</td>
<td>P8 Innovation and planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P6 P1 Planning</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

4 The practices for stage 2 on principles “people and interactions” and “customer and stakeholder collaboration” are left unpopulated in the matrix for review during the expert evaluation phase.
4.2.6 Participatory observation

The practices are positioned in the matrix according to the stages and principles derived from literature. In addition to literature review, participant observation is used as research tool in getting deeper knowledge in the research context. Various other studies in agile software development employ this approach in exploratory research of the subject (Moe, Aurum, & Dybå, 2012) (Nerur, Mahapatra, & Mangalaraj, 2005). Further, McAvoy et.al claim the importance of participant observation in their publication as an appropriate tool to investigate the issues in software development domain (McAvoy & Butler, 2007). In this research, the master thesis student acted as a participant observant in a large scale organization, which adapted to SAFe practices in the year 2014. Author observed daily meetings, scrum ceremonies, planning meetings and release meetings, to understand and analyze the process and practices used in a specific team in the organization. Observation is limited to a particular team due to time and scope restrictions in the research. Below are the observations by the author as a participant in the team for the period of 4 months (March-June 2016). Participatory observation played a significant role in understanding the process of scaled agile methodology. Being a comparatively newer topic of research when compared to agile methodologies, scaled agile methodologies are limitedly explored in literature. In addition, as already argued there is no unique solution to the problems faced by the organization and there is no unique challenge faced by all the organizations in this world, with regards to scaled agile methodology. Hence participatory observation helps in understanding how the team or the organization reacts to context-specific challenges. As already discussed, the nature of ambidexterity in exploring and exploitation of practices helps in agility across the teams. With the help of participatory observation we would also like to gather the results on how the teams employs ambidexterity in employing the practices and their notion towards such an ambidextrous environment.

4.2.6.1 Scaled agile practices

At the team level, the organization follows scrum practices. As a first step towards following scrum, the team members were given considerable authority for many aspects of developmental activities such as, scheduling, self-assigning tasks and also to team member and decentralized decision making. Scrum master played a major role in eliminating the impediments or solve problems that occurred through the course of the project. It is also noted that scrum master acted only as a guide in solving the problems and not as a problem solver himself, in the sense he let the team organize and make decisions on the problems concerned. This emphasized the factor of self-organization playing a major role in the agile/scaled agile team. The team followed time-boxed sprints to develop increments of working software. Each sprint starts with a planning session, followed by daily stand up to track the progress on the sprint tasks and ends with sprint retrospective and review. Planning session helps in demonstrating all the tasks that needs to be done in a sprint session. Team members use task boards as a visual management tool in selecting the items from the product backlog that they can complete during an upcoming sprint. These are represented by a task card and are placed in the scrum board (Photo source).
The team members coordinate on a daily basis through daily stand up and continuously make decisions. The team follows a sprint backlog, which is a part of product backlog. The backlog is transparent to the entire team and progress of every team member towards a task is visible to every other member. Product owner plays a significant role in prioritizing the backlog and presenting a prioritized backlog during the planning meetings.

In addition to the scrum ceremonies, the team organize backlog grooming sessions, in which the team members along with developers, UX designer and product manager take part in making improvements to the backlog. Grooming sessions are significantly used to breakdown the user stories to smaller parts, Estimate the backlog items and look deeper into the backlog for any impediments. At the portfolio level, product management plays a significant role in prioritizing the product backlog necessary for the program increment planning meetings. Program increment planning sessions are conducted among the teams which needs to be aligned for the achieving the end goal. PI planning sessions are a two day meet, for which the agenda is prepared and communicated to the stakeholders by the release train engineer. In addition, customer requirements are analyzed through means of a forum, in which there exists a frequent discussion with the team and the customers.

4.2.6.2 Observation on Ambidextrous nature
The organization followed an ad-hoc software development process, with a top-down approach before adapting to scaled agile practices. New roles and responsibilities emerged along with the
adoption of scaled agile practices. The evolvement of agile teams towards a scaled agile environment shows that, the team started using scaled agile methods by book. They focused on scaled agile learning through workshops and training programs. Though the scaled agile learning was established through workshops, the team didn’t follow all the practices enlisted as ideal practices for a scaled agile environment. The act of sense-making the practices is accomplished only after the need for such a practice is established. For instance, the practice on requirements analysis by getting inputs from the customer is considered as an essential practice in agile and scaled agile software development approaches (Highsmith & Cockburn, 2001). However the team directly didn’t involve in customer requirement analysis on a regular basis. But when there was a need to allow themselves to work in closer collaboration with the customers, the team dedicated a specific time slot to go through the inputs and feedbacks given by the customers in a local community forum, where customers chat about their products and its features. The practice therefore was named as “Community input analysis” and was regularly scheduled every other week for team discussion with all the involved stakeholders. But even before the adaptation of such a practice, the team involved in an exploration phase to experiment the benefits of such a practice scheduled on a regular basis. Only when the practices bring efficiency to the team, it is considered for further discussion for the exploitation. This is in line with the argumentation on Ambidexterity, where organizations involve in a balance between exploitative and exploratory nature of practices (Turner, Swart, & Maylor, 2013). It can also be noted from this characteristic, that the scaled agile teams evolve towards agility through applying ambidexterity behavior, in the sense it improves self-organizing behavior of the teams in responding to business changes, which is the core value of agile and scaled agile systems. It also helps the agile teams evolve towards maturity from agile learning towards ambidextrous behavior, in responsive or adaptive to complex situations.

Another practice which involved a high percent of ambidexterity can be seen in the practice of creating a Minimum viable product. In order to test the new strategy or feature, the team dedicates a specific time span starting to create a product restricted features to gather a validated learning about the product across the customers, with least effort. The response of the customers are tested across the problem and solution hypothesis the team had in mind and then the further initiatives are developed with maximum effort to satisfy the need of the customers. Hence the team involve themselves in a predefined period of exploitation to gather the inputs from the customer. This practice of testing a minimum viable product is explored through brainstorming sessions from the team. Due to unavailability of a model per se, which can be used as a discussion tool in such brainstorming sessions, the team took a longer time span to look into relevant practices which might help in achieving this objective. With this line of argumentation, we claim that scaled agile maturity model will not only act as a primary source for such a discussion tool but also highlights the stages of those practices in the level of maturity.

With inputs from the participant observation, the practices highlighted in brown are added in the model.
Table 18 Initial design after participant observation

<table>
<thead>
<tr>
<th>Stage</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Principles</strong></td>
<td></td>
<td></td>
<td><strong>Principles</strong></td>
</tr>
<tr>
<td>Stage 1</td>
<td>P1 Self-organizing teams</td>
<td>P3 Pair programming</td>
<td>P8 Collaborative planning</td>
<td>P13 Customer commitment to work with the development team</td>
</tr>
<tr>
<td></td>
<td>P2 Empowered and motivated teams</td>
<td>P4 Knowledge sharing</td>
<td>P9 Epic Kanban systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P5 Coding standards</td>
<td>P10 Status reporting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P6 Task volunteering</td>
<td>P11 Adaptive and reflective tuning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P7 Task boards</td>
<td>P12 Backlog grooming session</td>
<td></td>
</tr>
<tr>
<td>Stage 2</td>
<td>P1 Daily stand up</td>
<td>P3 Automated testing</td>
<td>P7 Smaller and more frequent releases</td>
<td>P12 Community inputs</td>
</tr>
<tr>
<td></td>
<td>P2 Sprint review</td>
<td>P4 Acceptance testing</td>
<td>P8 Requirements engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P5 Tracking iteration progress</td>
<td>P9 Release planning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P6 Progressive requirement discovery</td>
<td>P10 Continuous delivery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P11 Sprint retrospective</td>
<td></td>
</tr>
<tr>
<td>Stage 3</td>
<td>P1 Scrum of Scrum</td>
<td>P2 Continuous integration</td>
<td>P5 Agile release train</td>
<td>P9 DevOps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P3 Unit tests</td>
<td>P6 Roadmap</td>
<td>P10 Minimal viable product</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P4 Refactoring and continuous improvements</td>
<td>P7 Product backlog</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P8 Potentially Shippable Increment</td>
<td></td>
</tr>
<tr>
<td>Stage 4</td>
<td>P1 Managing highly distributed teams</td>
<td>P2 Building Architecture runway (intentional architecture)</td>
<td>P3 Adaptive planning</td>
<td>P5 Close collaboration with the customers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4 Measuring business performance</td>
<td></td>
</tr>
<tr>
<td>Stage 5</td>
<td>P1 Low process ceremony</td>
<td>P2 Scaled agile estimation</td>
<td>P3 Product and solution management</td>
<td>P7 Process management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4 System and solution engineering</td>
<td>P8 Innovation and Planning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5 Inspect and adapt workshop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P6 P1 Planning</td>
<td></td>
</tr>
</tbody>
</table>
4.2.6 Assessment criteria
Assessment criteria play a main role in assessing the level of achievement of each of the practice enlisted in the maturity stages. As already discussed, the model forms a discussion tool in exploring the practices involved in the scaled agile methodology. In order for the discussion to be more effective, the model should include couple of factors in addition to practices. The model should also entail the progressive outcomes of adopting to such a practice (incase if the practice is being explored) or the indication of effective adaption (incase if the practices is adopted and the team is looking for improvement measures). Hence we claim that the model should entail the concept of indicators, which will provide intersubjective criteria for level of achievement on the practices included in the model. Without the knowledge on the outcomes a particular practice might create, discussion might prove ineffective as teams should involve themselves to learn more on the practices before even to start discussion on a practice. This idea on including the assessment criteria through progressive outcomes can be fulfilled only when the expert evaluation is complete. This is due to the fact that the next phase on expert evaluation might include a vast change in the initial design of the model, and the notion on assessment criteria also needs a validation before including it in the model.

4.3 Chapter summary
In this chapter we developed the initial design of scaled agile maturity model and answered the third sub-question:

RQ3 “How does the scaled agile maturity model would look like?”

To reach this objective, we used the inputs derived from previous chapters into account while designing the model. By analyzing and interpreting this data, we derived at the design pattern of the model, which consists agile core values and principles forming the rows and columns of a matrix. The core values and principles form a matrix of scaled agile practices, thus forming a 5*4 matrix populated with scaled agile and agile practices. The practices are backed up with the rationale behind populating the particular practice along the stages. After arriving at the initial design of the model, we came up with the idea on assessment criteria for the practices. We introduced the concept of indicators which would help in overall assessment of the practices introduced in the model. As a next step towards finalizing the design, we employ expert evaluation as a research tool to evaluate and validate the model for completeness, consistency and general discussion, across different experts in TomTom. This forms the basis of the preceding chapter.
Chapter 5 Evaluation of the scaled agile maturity model

5.1 Introduction
This chapter represents the evaluation of the designed scaled agile maturity model through experts. A heuristic evaluation is required at this stage to review the goal and end product by specialists in the field of scaled agile from the business world, which forms the basis of forth sub-question:

RQ4. What is the value of the scaled agile maturity model in a real life context?

This chapter is meant to answer this question and goal of this chapter it to fetch systematic and analytical inputs from the experts before finalizing the final version of the scaled agile maturity model. The quality of the prototype is assessed with the help of expert evaluation. The remainder of the chapter is structured as follows: in the next section the evaluation approach is discussed, and then the evaluation and main conclusions are provided. At the end, final design of the scaled agile maturity model and its adapted version after evaluation results are described.

5.2 Evaluation approach

5.2.1 Type of evaluation used in expert interviews
The design of scaled agile maturity model was build using an adapted design approach by Pferss et.al (2007). With regards to the design step phase after the design and development of a model, these scholars introduce a design step to demonstrate the use of the designed artifact to the relevant stakeholders and communicate the solution through demonstrating the use of the artifact. They also describe various case studies in which a model has been developed through the design steps of Pferss et.al, in various organizations. The case studies are carefully studied as how the demonstration phase has been achieved. In most of the case studies, an individual structured / semi-structured expert interview was used as a research tool in demonstrating the design and end goal. It enabled them to gather rich data collected by a widely representative sample of experts, which helped in analyzing the usability of the model(Pffers et al., 2007). It is also noticed that, this design research step differed in goals and methodologies among different case studies. For instance, few of the case studies stressed in evaluating the quality of the design, before being implemented in a real world scenario (Ex-ante), whereas few other case studies stressed in evaluating how well the design goals have been achieved through implementing in a real case(Ex-post). This helped them in presenting an initial proof-of-concept level validation to the organization.

Taking into account that this research is exploratory in nature and focusses on early stages of development, we consider going for ex-ante evaluation methodology, in which the design is evaluated before being implemented in a real world scenario. In addition, an ex-ante expert evaluation would help in reducing the number of iterations needed when developing initial proof of concept when implemented in a real-life context.
5.2.2 Expert selection method

The design of the scaled agile maturity model is proposed in Chapter 4 is at a great extent based on the data collected from the literature and participatory observation of the scaled agile practices involved in a single organization TomTom.

TomTom was chosen as the company from which range of experts are selected. TomTom implemented scaled agile framework in the year 2014 and since then, a range of scaled agile practices have been implemented and adopted for software development in various business units of TomTom (Leffingwell, 2011). Scaled agile practices are practiced by all of TomTom’s large product teams representing navigation software, online services, map creation and sport software. Transition towards a large scale paradigm shift from waterfall based approaches to scaled agile practices was a greater challenge with more than 4,600 employees worldwide. As a first step in moving towards the challenge, TomTom started organizing scaled agile training programs for people at higher hierarchical level such as Chief Information Officers (CIOs), Chief Product Officers (CFOs), Chief scrum masters (CSMs), Product owners, Managers, Directors, and Vice presidents. From there, TomTom began to reorganize the scrum team across the team level, and started scaling up, thereby forming different product clusters at the program level. Specific to this research, TomTom headquarters located at Amsterdam is analyzed for experts, which has more than 200+ scaled agile certified and trained employees. Before selecting the experts, it is essential to discuss on whom we call as experts. We take the definition provided in the publication of Bogner et al., which takes an extensive research in the field of expert evaluation to identify the real experts.

“If the expertise (that is specific interpretive knowledge (“know-why”) and procedural knowledge (“know-how”) in a particular occupational or professional field) is central to the area of research, then the interview can be regarded as an expert” (Bogner et al., p.p108, 2009)

Following the definition from Bogner et.al and focusing on the research domain, we claim that an expert should have both interpretive knowledge and procedural knowledge in the field of scaled agile methodologies. This led to selection of experts in the following two dimensions,

- Technical knowledge (know-how) – Very specific knowledge in the field and has detailed opinions on the operational practices pertaining to scaled agile practices.
- Process knowledge (know-why) – Very specific knowledge on the routines, processes, and stakeholder interactions pertaining to scaled agile.

Based on the two dimensions of expert knowledge, we gathered experts from a different department other than that of the department used in participatory observation. As can be seen in the matrix (Table 22) the experts differ in both roles and dimensions of expert knowledge. Senior product owner, Chief scrum master and Transition manager represent as an expert in both technical know-how and procedural know-why. They helped in contributing to the knowledge on both technical and process perspective in the model. Whereas the General Manager and the Q&P control manager represent as an expert only in Process know-why dimension. Hence they contributed mostly on the procedural perspective of the model. They were not approached for explicit feedback on columns such as technical expertise and on technical challenges.
Table 19 Matrix of experts on technical and process dimensions

<table>
<thead>
<tr>
<th>Experts</th>
<th>Technical knowledge</th>
<th>Process knowledge</th>
<th>Agile function</th>
<th>Experience in TomTom in the described agile function</th>
</tr>
</thead>
<tbody>
<tr>
<td>General manager</td>
<td></td>
<td></td>
<td>Responsible for release of a software increment to the customer. Maintains and coordinates the agile team for continuous software delivery.</td>
<td>5 years</td>
</tr>
<tr>
<td>Senior product owner</td>
<td></td>
<td></td>
<td>Represent the team across different external stakeholders. Backlog owner and acts as a point of communication on behalf of the team.</td>
<td>7 years</td>
</tr>
<tr>
<td>Transition manager</td>
<td></td>
<td></td>
<td>Coach and helped the organization move towards scaled agile from traditional waterfall based approaches</td>
<td>7 years</td>
</tr>
<tr>
<td>Chief scrum master</td>
<td></td>
<td></td>
<td>Helps in removing the impediments for the agile team to deliver project goals and objectives.</td>
<td>5 years</td>
</tr>
<tr>
<td>Vice president</td>
<td></td>
<td></td>
<td>Manages and supports several agile teams to deliver the project goals.</td>
<td>7 years</td>
</tr>
<tr>
<td>Quality and process control manager</td>
<td></td>
<td></td>
<td>Helps in controlling the quality of the software processes across different teams.</td>
<td>5 years</td>
</tr>
</tbody>
</table>

5.2.3 Evaluation procedure
As the initial step in the evaluation process, meeting invites are developed in which a clear and brief description on the goal of the project, prototype of the model, description on need for expert evaluation and brief explanation on what to expect during the session. The meeting invites are sent to
the selected experts in TomTom, requesting for a face-face session in the Amsterdam office at TomTom. The length of each session was approximately 75 minutes and only one interviewee is present during a session Appendix 1).

At the start of each session, the objectives and initial design of the maturity model are shortly explained. After which we used random set of semi-structured questions in order to inquire the interviewee’s opinions on the model. The questions majorly covered the consistency, relevancy, and discussion on model in general (misplacement of practices in the stages, indicator and assessment criteria). It can be noticed that other evaluation aspects such as neutrality and interrelatedness are not included in the evaluation criteria. The neutrality of the model refers to the generalizability of the model with respect to different industries and IT community. This is not included due to couple of generalizability reasons addressed in section 6.7 Limitations. Interrelatedness reveals whether the individual aspects of the model are dependent on each other or contradictory. This aspect is explicitly not included as an evaluation criteria, however indirectly addressed via expert interview while discussion about the model in general. In order to eliminate the subjective opinions on a certain statement, we used advocacy to clarify the argumentation of the interviewee.

Table 20 Evaluation criteria

<table>
<thead>
<tr>
<th>Evaluation criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>According to Hevner et al. (2004, p. 82) the designed model “must be described effectively, enabling its implementation and application in an appropriate domain (Hevner, March, Park, &amp; Ram, 2004). This characteristic on determining the consistent behavior of the model in accordance to present scenario in IT community is considered as the notion of consistency in this research.</td>
</tr>
<tr>
<td>Relevance</td>
<td>With various challenges faced by the organization in adopting to scaled agile methodologies, we claim to check the relevance of challenges confronted by this model.</td>
</tr>
<tr>
<td>General discussion</td>
<td>Interrelatedness of the components in the model, missing components, design pattern, notion of including agile components in the model.</td>
</tr>
</tbody>
</table>

5.3 Evaluation results

The expert evaluation yielded various insights which will be described in this session. This will also influence the further development of the model. For each of the themes, experts provided positive remarks (marked by green) and opportunities for improvement (marked by blue) and areas in which adaptation is required (marked in red).
Table 21 Expert evaluation results

<table>
<thead>
<tr>
<th>Category</th>
<th>Overall remarks/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall relevance</td>
<td>✓ Most of the respondents acknowledged the great relevance of the framework in finding areas of improvement across for scaled agile development.</td>
</tr>
<tr>
<td></td>
<td>“This is greatly relevant to what we are doing as scaled agile” – VP</td>
</tr>
<tr>
<td></td>
<td>✓ Few other positive remarks acknowledged that difficulty in coping up with the strict regulations posed by various other maturity models in going from one stage to other. And this model being loosely coupled between stages and easy to communicate to the respective stakeholders.</td>
</tr>
<tr>
<td></td>
<td>✓ All the respondents acknowledged the ambidextrous nature of scaled agile systems and artifacts designed around those systems should give space for emergent behaviors.</td>
</tr>
<tr>
<td></td>
<td>✓ Next to that experts collaboratively agreed on the assessment criteria being included in each and every stage, which helps in more of the prescriptive analysis rather than a descriptive analysis.</td>
</tr>
<tr>
<td></td>
<td>“I like how it comes with a complete package, rather than a descriptive model” – Chief Scrum master</td>
</tr>
<tr>
<td></td>
<td>✓ One suggestion for improvement was the name, which says “maturity model” might give a similar feel like CMMI and SPICE which has been criticized by the software community for being very waterfall.</td>
</tr>
<tr>
<td>Consistency</td>
<td>✓ Many of the respondents acknowledged the use of agile and scaled agile Principles/practices/core values in developing a model, rather than</td>
</tr>
</tbody>
</table>
concentrating more on the processes for each stage. In this sense, they believe that agile factor sustained at the higher agile stages, which is a common challenge being acknowledged in the agile community.

✓ Another positive remark was that the model takes agility into account in creating stages.

> “Many models in the market forget agility” – Transition manager

✓ Some of the respondents felt that the model tends to be too broad and in depth extensive review is bit difficult at this stage, since it has both technical components and process component: and few respondents are not an expert in both dimensions.

✗ Some of the respondents felt that the practices for each of the stages were filled from left to right (taking core value as a start), rather than being filled from top to bottom, through which the maturation path of each practice or at least for most of the practices could be instantly visualized.

> “There should be an interplay between filling up the model along the column and rows” – General Manager.

✗ One of the respondents claimed that the stages 1 and 2 should explicitly take only team level practices into account, since scaling agile to program level can happen only when the team level practices are in place.

> “Scaling up can happen only if the teams are agile”

<table>
<thead>
<tr>
<th>General Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Some of the respondents claimed that Indicators should include a qualitative analysis and not a quantitative analysis, since quantitatively representing a team or a business unit with a number seems less appealing to improve, for a team which got lower scores.</td>
</tr>
</tbody>
</table>
“People usually don’t like them being called with a number and comparing the results merely based on numerical data” – Quality control and process manager.

✓ One of the respondents claimed that the design should also include a process improvement framework for few of the essential practices enlisted.

All of the respondents came up with suggestions for moving the practices across different stages, as described below.

✓ Automated testing in stage 2 should be in stage 3 and vice versa. Since automated testing is not possible without unit tests at place.
✓ The practice of DevOps in customer and collaboration column should be moved to technical expertise.
✓ Managing distributed teams should be in stage 2, in the first column and scrum of scrums should be moved to stage 4, since scrum of scrum is more scaled agile than managing distributed teams.
✓ Few practices such as backlog, adaptive and reflective tuning needs more clarity.
✓ Continuous integration and continuous delivery should be exchanged in their places.

✗ Few of the respondents claimed that the model tend to assume agile principles more than the scaled agile principles.

“At first look of the model, I get a feeling that it takes more agile principles than scaled agile principles” - VP

“I think scaled agile principles should be explicitly seen in a scaled agile model” – Senior product owner

✗ Few of the respondents claimed that, involving one core value at each stage of the model simply assumes that the core values explained in one stage is not essential when another stage is a point of discussion.
To summarize this ex-ante evaluation helped in providing inputs and guidelines in improving the initial design of the model. The adaptation of the model from the systematic inputs from expert evaluation is described in the following section.

5.4 Adaptation of the scaled agile maturity model

Based on the opportunities of improvement and areas of adaptation specified during the expert evaluation phase, the design of the model is developed into a final design as presented in figure below.

5.4.1 Structural adaptations

As a first step towards structural changes (design pattern of the model), the principles are extended with explicit scaled agile principles, adding to other columns: Alignment and transparency. This included in consideration with all the expert interviewees to create an explicit appeal on scaled agile factors. Alignment and transparency are preferred by the experts as explicit scaled agile principles, since the scaled agile methodologies focus on alignment across team, program and portfolio level. This is the main factor which differentiates agile with scaled agile methodologies. Scaled agile methodologies also stress on transparency among the team, program and portfolio levels at a greater extent. Hence, these principles are chosen explicitly to elucidate the scaled agile characteristics of the model. Many of the respondents acknowledged the least contribution of the column on Customer and stakeholder collaboration, due to the fact that many of the agile and scaled agile practices by default requires customer and stakeholder collaboration and showed least preference to extend it as a column. Hence it is removed from the model. Since many of the scaled agile practices are adopted along with tools that can create an effective scaled agile environment, few of the respondents were towards including “tools” as a separate column corresponding to the stages. It can also provide a separate discussion session on what kind of tools the team can adapt to adapt such a practice. Hence the suggestion was included in the model.

One of the major suggestions acknowledged by all the respondents is the naming of stages. Rethinking on the notion of including core values as stages of maturity in this research, presented a huge challenge in adapting to the expert evaluation. The initial design was based on a design pattern in which the core values of scaled agile represents the rows and scaled agile principles represent the column, as discussed in 4.3 Design pattern of the model. But the expert’s suggestion on including explicit scaled agile factor in the columns might re-consider the whole design. However, we acknowledged the suggestions from
the experts and renamed the stages from numbering to team, program and portfolio levels. This is
due to the fact that, scaled agile systems more than the core values, it hugely represents the three levels
in all approaches. Hence by including the levels as stages in the model will create a specific targeted
discussion on these three levels rather than random discussion on practices. This forms the rationale
behind restructuring the stages as team, program and portfolio. Further the stages on these three levels
are discerned into streamlined and mature. Streamlined represent the initial stage at which the
team/program/Portfolio look for practices to adopt to scaled agile methodologies. And mature
represent the stage at which the practices are further progressed from initial stage to a progressive
stage. As discussed earlier, by no means this research acknowledges a perfect matured stage for a
practice, as it depends on the team which employs the practice. However it provides a guide on how
the practices can evolve from initial streamlined stage to a progressive stage.

Table 22 Suggestion for improvement

<table>
<thead>
<tr>
<th>Suggestions on structural adaptations</th>
<th>Acknowledged in the model: Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit introduction of scaled agile principles in column</td>
<td>Yes, corresponding changes are made by introducing columns such as Transparency and Alignment. Inclusion of tools as a separate row.</td>
</tr>
<tr>
<td>Explicit introduction of scaled agile factors in rows</td>
<td>Yes, corresponding changes are made to include team, program and portfolio levels in stages</td>
</tr>
<tr>
<td>Elimination of customer and stakeholder collaboration column</td>
<td>Yes, acknowledged in the final design.</td>
</tr>
<tr>
<td>Change the name of model to Roadmap.</td>
<td>No, A Roadmap is a game plan to achieve to achieve the goals. It starts with the desired result of a seamless and delightful goals and takes into account all of the research required to achieve it. Meanwhile our research aims at helping or supporting an organization to a desired result. It doesn’t provide a game plan, but rather provides a specific result. Hence we claim to use maturity model as the research headline.</td>
</tr>
<tr>
<td>Naming of practices</td>
<td>Yes, The practices which are named as S1, S2 etc are changed to S1P1, S1P8 etc which gives a meaningful representation of the practices.</td>
</tr>
</tbody>
</table>

5.4.2 Model adaptations
This section deals with the analysis of changes included in the model on the whole. Since the columns
are extended with alignment and transparency, practices are shuffled across the columns to represent
a meaningful model. In this section, we discuss the adaptations made in the initial design via increasing
order of stages.
**Stage 1 Team level streamlined**

Pair programming which was introduced in technical expertise in the initial design is moved to people and interactions in the final design as suggested by the experts due to its nature of increasing the interactive nature of the agile teams than the technical expertise. DEV-UX is a new practice included in the final design which explicitly takes the relationship of developers and User Experience team in an organization. This is suggested by few of the experts to be a challenging aspect in a large scale organization which has a dedicated UX team to support the developers. The relationship between them is deemed important to help in supporting the goals of the project on time. Few of the respondents acknowledged the introduction of new practice on knowing the clear responsibilities of the role, since the conflicts on responsibilities deem to reduce the interactive nature of the people in a team.

Clear definition of done is introduced as a new practice in the column technical expertise as suggested by one of the respondents, this being the major factor in decreasing agility across the team levels. Scaled agile team being a self-organizing team involve self-assigning of tasks. Each and every member in the team might therefore have different notions on defining the tasks as done. Entailing a common definition of done thus plays a major role in improving the technical expertise of the team. In addition code refactoring is also added as a new practice in this column which represents the practice of continuous process of restructuring the existing codebase of the team to involve new changes. Experts also suggested to include the practices such as test – first development and exploratory testing in the team level. Test first development is a process in which the team members start the development process by writing the test cases first before writing the code. This forms the basis of exploratory testing in which the team involve themselves in exploring the new features developed via code. The practices introduced in this stage correspond to the evaluation on consistency criteria in which the experts acknowledges the lack of present state of art practices in the initial design. This helped in including the relevant practices as established in the IT community.

The practice on smaller and more frequent cycles introduced in stage 2 of initial design is changed to Stage 1 in the final design and renamed to “Shorter iterations”. This is in accordance with the expert suggestion. The task boards used in technical expertise column is moved to column on transparency due to the fact that the task boards helps in transparency of tasks assigned to different team members. The practices such as sprint backlog and scrum ceremonies are included as new practices as suggested by the experts. Sprint backlog corresponds to the tasks assigned to different team members, which has to be finished at the end of the sprint. Scrum ceremonies include daily stand up, sprint planning etc, which promotes transparency among the team members as they collaboratively speak on the tasks and challenged involved in completing the goal.

**Stage 2 Team level-Matured**

The practice on managing distributed teams is moved from stage 4 in initial design to Stage 2 in Final design due to the nature of adaptations in the stages of the model. In addition five different practices are acknowledged by the experts as state of art practices involved in improving the technical expertise
of the scaled agile team. These are included in the model to acknowledge the consistency of the model and openness of the model to state of art IT practices.

**Table 23 Adaptation of practices in Stage 2**

<table>
<thead>
<tr>
<th>New practices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S2P3 Test driven development</strong></td>
<td>Test driven development plays a major role in increasing the testing nature of the agile teams, in which the test is written prior to coding.</td>
</tr>
<tr>
<td><strong>S2P4 Acceptance driven development</strong></td>
<td>This type of development is prevalent in the IT community in which the development involves a feedback loop from the internal customers (small population) and business people before the software is released to public. Many of the respondents acknowledged the newness of this approach and playing a major role in an agile/scaled agile environment.</td>
</tr>
<tr>
<td><strong>S2P5 Behavior driven development</strong></td>
<td>Behavior-driven development is an extension of test-driven development development that makes use of simple test cases. The result is a closer relationship to acceptance criteria for a given function and the tests used to validate that functionality.</td>
</tr>
<tr>
<td><strong>S2P6 Continuous integration</strong></td>
<td>CI) is a development practice that requires developers to integrate code into a shared repository several times a day. This is also acknowledged by the respondents as a state of art approach in increasing the technical expertise of the team.</td>
</tr>
<tr>
<td><strong>S2P7 System integration</strong></td>
<td>Since the software development process has many different sub-systems working together, it is the team’s responsibility to check the working of several sub-systems on a regular basis. This practice is achieved to increase the technical expertise of the team.</td>
</tr>
</tbody>
</table>

Tracking iteration progress is moved from column on technical expertise to incremental builds in the final design. This is in accordance with the change suggested by the experts in which the metrics and tracking the progress activities are more often involved in continuous flow of software. The chief scrum master, who looks on the impediments and quality control manager who helps in increasing the quality of the software acknowledged the use of scaled agile metrics and software quality characteristics to increase the incremental and continuous flow of software. The practice on stakeholder alignment is introduced in stage 2 which helps in increased alignment between the internal and external stakeholders. The practices such as Inspect and adapt workshop are moved from stage 5 in the initial design to stage 2 in the final design. This is due to the fact that the practices concentrates more on the team level and cannot be introduced in the stages further than team level.

**Stage 3: Program level – streamlined**
The practice on clear definition of roles and responsibilities is repeated in the program level too, to denote the multitude of roles and responsibilities involved in the program level. Devops is included in technical expertise in accordance with the all of the experts, who claimed that the nature of Devops increases the technical expertise rather than the customer and stakeholder collaboration. Team level increments on functional and non-functional requirements is introduced in this stage to increase the
notion on incremental building taking into account both functional and nonfunctional requirements. Few of the respondents agreed to the fact that the organization should also look at the non-functional requirements at the program level to help supporting the incremental flow of software. Since stakeholder alignment is revisited again in this stage, the notation is given as S4P0 in contrast to the other stages where the practice starts from 1. This is due to the repetition of practices within the model.

**Stage 4: Program level – matured**
While the practices on scrum of scrums seen in stage 3 of the initial design, it is moved to stage 4 in the final design to emphasize more on the program level culture in the scaled agile environment. The practice on PI execution demo is introduced in this stage which helps in introducing a separate time span to discuss the feedbacks and reflection on the previous agile release train aspects. This is considered as an explicit scaled agile practice in the program level by few of the respondents.

**Stage 5 and 6 Portfolio level – streamlined and matured**
This stage represents couple of new practices which is explained in the table below:

<table>
<thead>
<tr>
<th>New practices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S5P1 Strategic themes estimation</strong></td>
<td>Strategic theme is a practice which helps in connecting the evolving enterprise strategy to portfolio level. The theme on new features and new business expansion are collected as strategic themes. This practice is confirmed by experts as state of art method to increase the stakeholder collaboration across the 3 levels.</td>
</tr>
<tr>
<td><strong>S6P1 Program portfolio management</strong></td>
<td>It is a practice of employing a team for managing the portfolio level. Which represents the people for fiduciary decision making to set initiatives in a business unit.</td>
</tr>
<tr>
<td><strong>S6P2 Scaled agile center of excellence</strong></td>
<td>This practice of constructing a center across the organization to discuss on the technical excellence and processes followed by the team and program levels has been acknowledged as an effective practice by the experts. This is the rationale behind choosing this practice in the technical expertise column.</td>
</tr>
<tr>
<td><strong>S6P3 Estimating funding for agile release trains</strong></td>
<td>The responsibility of the program level falls on the portfolio level. Hence this practice on estimating funds for the agile release trains across the program level helps in achieving incremental building.</td>
</tr>
</tbody>
</table>

**5.4.3 Assessment criteria**
As discussed earlier, we claim that in order for the discussion to be more effective, the model should include couple of factors in addition to practices. The model should also entail the progressive outcomes of adopting to such a practice (in case if the practice is being explored) or the indication of effective adaption (in case if the practices is adopted and the team is looking for improvement measures). Hence we claim that the model should entail the concept of indicators, which will provide intersubjective criteria for level of achievement on the practices included in the model. Introduction of indicators as means of assessment were considered effective by the experts in engaging a discussion for exploration of outcomes (Appendix ). The assessment criteria are included as a separate section
in the Appendix. However quantitative means of assessment is rejected due to the potential tendency of team members to compare the teams on the scores achieved by different teams/program/portfolio unit. Hence qualitative assessment is considered as a intersubjective criteria in assessing the practices enlisted in the model. The criteria chosen are “Fully achieved”, “Partially achieved”, “Not achieved”, “Not applicable” and “Reason”. As already discussed, indicators provide a means of discussion in an organization to kick start the scaled agile development process or means of discussion for areas of improvement. With this line of argumentation a column on reason is included so as to put forward the reason (if any), they consider as mean of improvement in evaluating the practices across the qualitative criteria. In addition, the indicators act as a tool in operationalizing the model across large scale organizations. This entails the point of discussion in focus groups to provide feedback on scaled agile practices in general or what factors the group perceives need further improvement in the team/portfolio/program level. With the adaptations from the expert evaluation, the final design of the model is presented below in table 24.
<table>
<thead>
<tr>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
</table>
| **Stage 1 – Team level streamlined** | S1P1 Pair programming  
S1P2 DEV-UX  
S1P3 Clear definition on scrum roles and responsibilities  
S1P4 Self-organized teams | S1P5 Clear definition of done  
S1P6 Coding standards  
S1P7 Code refactoring  
S1P8 Defect analysis  
S1P9 Exploratory testing  
S1P10 Test first development | S1P11 Short iterations  
S1P12 Task boards  
S1P13 Sprint Backlog  
S1P14 Scrum ceremonies | S1P15 Automated Status reporting  
S1P16 Common language |
| Tools of stage 1                | Common tool of communication for all stakeholders. | Code review tools | Issue tracking tools, productivity tools | Issue tracking tools |
| **Stage 2 – Team level matured** | S2P1 Managing distributed teams  
S2P2 Increased collaboration | S2P3 Test driven development  
S2P4 Acceptance driven development  
S2P5 Behavior driven development  
S2P6 Continuous integration  
S2P7 System integration | S2P8 Frequent builds  
S2P9 Measuring software quality characteristics  
S2P10 Tracking iteration progress | S2P11 Stakeholder alignment  
S2P12 Adaptation sessions – Inspect and adapt workshops team level  
S2P13 HIP Sprint – team level |
| Tools of stage 2                | Wikis and blogs  
Online project management tools  
Shared mailing list and folder | Automated deployment process tools | Agile metric estimation tools | |
| **Stage 3 – Program level streamlined** | S3P1 Clear definition of program stakeholder roles and responsibilities. | S3P2 DevOps | S3P3 Team level increments for functional and non-functional requirements. | S3P4 Program Backlog  
S3P5 Release planning  
S3P6 Inspect and adapt workshop PI level  
S3P7 HIP sprint – program level |
| Tools of stage 3                | Automated building and deployment tools | | Issue tracking tools | |
| **Stage 4 – Program level matured** | S4P0 Stakeholder alignment  
Scrum of scrums | S4P1 Clear architectural runway | S4P2 ART (Agile Release train)  
S4P3 PI Execution System demo | |
| Tools of stage 4                | | | Issue tracking and productivity tracking tools | |
| **Stage 5 – Portfolio level Streamlined** | S5P1 Strategic themes estimation | S5P2 Measuring business objectives | S5P3 Portfolio backlog | |
| Tools of stage 5                | | | Portfolio Kanban tools | |
| **Stage 6 – Portfolio level matured** | S6P1 Program portfolio management | S6P2 Scaled agile center of excellence | S6P3 Estimating funding for agile release trains | S6P4 Budget estimation |
| Tools of stage 6                | | | Portfolio Kanban tools | |

Table 24 Scaled agile maturity model design after expert evaluation
Chapter 6 Conclusions

6.1 Introduction
In this research project, we developed a scaled agile maturity model to help organizations for successful adoption of scaled agile practices in stages. In order to design the model, different research strategies were applied, including: investigating the broad range of scientific publications, participatory observation of scaled agile practices in an organization, conducting semi-structured interviews for evaluating the model across experts in the field of scaled agile.

This last chapter summarizes the significant findings of the research and is structured as follows: In the following section, an overview of the master thesis project will be given, by revisiting and answering the research questions proposed in the previous chapters. Secondly, the main contributions of the work in terms of scientific relevance will be given. In addition general discussion on the model and criticisms are discussed. Thirdly, the main limitations that hampered the extensive research on this study is provided. In the fourth section, recommendations and potential future research ideas will be given. In the last part, a reflection study is presented which will represent both my personal consideration and relevance of this research through the lens of my SEPAM Master programme.

6.2 Main findings
Focused on designing a scaled agile maturity model, the main research objective of this master thesis project was:

“How does the scaled agile maturity model look like?”

Following from this objective, the following main research question was determined:

“RQ1. What does the process of scaled agile development look like?”

This main research question was divided into four sub-research questions that were constructed for the purpose of providing step-by-step inputs to the main research question. The first sub-research question is formulated as,

To answer this question, a systematic literature review was conducted. First, we identified the software development process in general. This also included an analysis on traditional waterfall based approaches in software development. Thereafter, we started analyzing the need of agile software development and the main benefits of using it, rather than the traditional waterfall based approaches. In order to know more about the agile software development process, we investigated the practices followed by an agile methodology (scrum), to encourage agility in team level, for developing software.
With this data at hand, we continued investigating the need for scaled agile software development and its benefits over agile software development approaches. The research revealed two main dimensions of scaling, through which an organization can scale the agile practices across the whole enterprise from team level (Time dimension and Team dimension). Team dimensions of scale relates to multiple teams working towards the end goal, but with different tasks. In contrast, time dimension of scaling relates to duration of the development and lifecycle of the software development process on the whole. Over time, technology and the system may undergo changes in various ways. In response to the changes, the system needs to rhythmically scale up taking lifecycle budgeting, planning and milestone planning into account. (Ozkaya et al., 2013). It was also revealed that team dimension scaling is more popular and beneficial for large scale organization, which is looking for scaled agile practices to scale agility across the enterprise. In order to know more about the scaled agile software development, we investigated the process of Scaled Agile Framework (SAFe) and its practices in scaling agile from team level to portfolio level and program level. This research provided insights into the general practices followed by the scaled agile methodologies, with emphasize not only on the team level (as proposed in agile methodologies), but also on the program level (which forms the synchronization level of all the teams which needs to work together) and also on the portfolio level (which forms the alignment level of all the program levels).

The second research question is focused on investigating more on the requirements for the design of the agile maturity model and reads as follows:

**RQ2. What are the basic requirements which have to be considered in designing a scaled agile maturity model?**

This question was answered by investigating various other agile maturity models in literature. Since the scaled agile methodologies takes it roots from agile methodologies, agile maturity models were given a thorough review, exploring the merits and demerits of the design strategy employed in it. For reviewing the articles, we employed selection criteria for selecting the publications. We investigated five maturity models in depth and its design strategy in constructing the maturity model. As a start, the term maturity is carefully analyzed with respect to both linguistic and model perspectives. Model perspective was studied even more broadly, in which two perspectives emerged: Life cycle perspective and potential performance perspective. Life cycle perspective measures the state of the growth of the process areas, taking the evolution of an organization into account in evaluating the maturity. This perspective names the final stage as “Fully mature” and represents the ideal stage of maturity. This perspective aims at fulfilling certain characteristics at a particular stage of maturity and can evolve to next stage only if the characteristics are fully performed. Whereas, potential performance perspective defines the maturity stages through means of a development path which focusses more on the potential improvements which might occur by moving along the model. This perspective gives user the final decision to decide which stage of maturity is the “level of completeness” according to their situation. Considering the scope and objective of this research, we chose to go for potential
performance perspective in designing the model, which gives users the final decision to decide which stage of maturity is the “level of completeness” according to their situation. 

We claim to use potential perspective by taking the ambidextrous nature of the teams into account. It turned out the existing agile maturity models in literature are criticized for their waterfall approaches and defining a process to maturity. We claimed to have initiated a different notion on maturity model by including the ambidextrous nature of the team in exploiting and exploring the agile practices that is suitable for the specific problem and challenge faced by the team or organization. It is in our view there is an essential need to shift the focus from prescribing the predefined processes in defining maturity to ambidextrous nature of taking a broader subjective view, in which people play an essential role instead of predefined processes. This notion on maturity model gave us an insight of using maturity model as a discussion tool for the team level or program level or portfolio level stakeholders to discuss on the prospective practices which can be adopted and potential improvements that can be made for the adopted practices. We also claim that there is no unique solution to a problem hence defining the level of maturity for a specific practice can be achieved through discussion of the stakeholders working towards the practice. Hence this research doesn’t base on one-size fits all approach and claim that needs of one organization differs from the other.

The third research question is focused on investigating more on the initial design of the maturity model. 

**RQ3: How does the initial design of scaled agile maturity model look like?**

Based on the insights derived from literature, three requirements were considered essential for the design.

**Table 25 Requirements of the design**

<table>
<thead>
<tr>
<th>Design guideline</th>
<th>Requirements</th>
<th>Translation into model</th>
</tr>
</thead>
</table>
| 1. Maturing of agile systems cannot be based on waterfall based strict regulations in maturing from one stage to another.  
2. Maturing in agility is too context dependent and models should act as a guide leaving space to emergency of behaviors. | Model will act as a discussion tool in stressing the ambidextrous nature of practices. Though the model would not consider the strict process based regulations, it should provide an idea on the possible stages of the scaled agile practices to act as a discussion tool per se. | Each and every stage of the model, should provide improvement measures with clear level of granularity for the teams to discuss on the improvement measures. |
| 3. Stages/levels of the model should be backed up with an agile factor.            | • Agile values  
• Scaled agile values  
• Agile principles  
• Scaled agile principles  
• Agile practices  
• Scaled agile practices | Definition of levels should incorporate the core values or principles of the addressed methodology. |
These requirements are translated into model through practical implications enlisted in the table above.

With the requirements of the model at place, we started investigating the design pattern of the model. We again took the inspiration from the design pattern of the agile methodologies. Each practice is developed by taking the agile principles and core values into account. We apply same kind of logic in the design in which a scaled agile/agile practice will take the scaled agile/agile principle or value into account, thus forming a AxA matrix of principles and values, as shown in the figure below.

Figure 30 Design pattern

The relationship between these agile factors, combined with participatory observation has resulted in the initial design as depicted in Chapter 4. Subsequently the idea on indicators is proposed, which helps the user in determining the level of completeness/achievement of a particular scaled agile practice. Qualitative assessment on the level of completeness across the organization is advised to find the areas of improvements and also on an idea on exploring practices which has not been followed.

In addition to the literature review on practices, principles and values, we conducted participatory observation to investigate more on the Scaled agile processes and practices. The chosen enterprise is a large scale organization, which has recently adapted to one of the scaled agile methodology (SAFe). After the populating the model with practices (both from literature review and participatory observation), backed up with arguments on why the practice is chosen for the particular stage, we ended with the initial design of scaled agile maturity model, as shown below. The colored practices in brown are drawn explicitly from participatory observation.
<table>
<thead>
<tr>
<th>Stage</th>
<th>People and interactions</th>
<th>Technical expertise</th>
<th>Incremental building</th>
<th>Customer and stakeholder collaboration</th>
</tr>
</thead>
</table>
| Stage 1 | P1 Self-organizing teams  
P2 Empowered and motivated teams | P3 Pair programming  
P4 Knowledge sharing  
P5 Coding standards  
P6 Task volunteering  
P7 Task boards | P8 Collaborative planning  
P9 Epic Kanban systems  
P10 Status reporting  
P11 Adaptive and reflective tuning  
P12 Backlog grooming session | P13 Customer commitment to work with the development team |
| Stage 2 | P1 Daily stand up  
P2 Sprint review | P3 Automated testing  
P4 Acceptance testing  
P5 Tracking iteration progress  
P6 Progressive requirement discovery | P7 Smaller and more frequent releases  
P8 Requirements engineering  
P9 Release planning  
P10 Continuous delivery  
P11 Sprint retrospective | P12 Community inputs |
| Stage 3 | P1 Scrum of Scrum | P2 Continuous integration  
P3 Unit tests  
P4 Refactoring and continuous improvements | P5 Agile release train  
P6 Roadmap  
P7 Product backlog  
P8 Potentially Shippable Increment | P9 DevOps  
P10 Minimal viable product |
| Stage 4 | P1 Managing highly distributed teams | P2 Building Architecture runway (intentional architecture) | P3 Adaptive planning  
P4 Measuring business performance | P5 Close collaboration with the customers |
| Stage 5 | P1 Low process ceremony | P2 Scaled agile estimation | P3 Product and solution management  
P4 System and solution engineering  
P5 Inspect and adapt workshop  
P6 PI Planning | P7 Process management  
P8 Innovation and Planning |
Although the initial design proposed is to a greater extent based on literature and participatory observation, we decided to fetch systematic inputs from experts in the business world before realizing the final version through ex-ante evaluation. This would also help in reducing the number of iterations required before implementing the model in real life context. This exercise answered the fourth research question:

**RQ4. What is the value of the scaled agile maturity model in a real life context?**

The evaluation was realized with experts at TomTom, pertaining to the technical and process knowledge domain of scaled agile methodologies. This method of evaluation was chosen over other evaluation strategies, as it provides a quick and systematic way to fetch inputs from people with multitude levels of expertise. Expert evaluation was conducted by means of a semi-structured interview with three high points as a base for the questions: Overall relevance, consistency and the design constructs of the framework. Based on these high points several opportunities for improvement and adaptation of the framework were suggested: Firstly, experts suggested that the maturity model should include more scaled agile principles and core values along the columns. Secondly, the bidirectional approach should be stressed out in populating the model with practices (there should be interplay between the row-wise bottom-up approach and the column-wise top-down approach). Next to that, maturity flows of a couple of essential practices could be added in order to help the organization to get a high level abstract view of maturity for the practices. In addition the design pattern of the model is also adapted by taking team, program and portfolio level along the stages of the model. The assessment criteria

Following the input received via the evaluation interviews, the central research question was answered by delivering the final scaled agile maturity model.

*How would a scaled agile maturity model for assisting the large scaled IT organization in adopting to scaled agile practice, look like?*

The scaled agile maturity model consists of five scaled agile principles as columns and six stages as rows. The stages represent the scaled agile levels used in the scaled agile methodology: team, program and portfolio. There are four main conclusions on the model: 1. Maturing of agile systems cannot be based on strict predefined process based model. 2. Scaled agile maturity model will act as a discussion tool among the team members 3. It is in our view there is an essential need to shift the focus from prescribing the practices in the name of maturity model to take a subjective view in which people play an essential role instead. 4. Agile and scaled agile maturity models should take the ambidextrous nature of scaled agile systems into account. With these main conclusions in hand, we provided a scaled agile maturity model along with the assessment criteria. With these conclusions we stress on the fact that people plays a major role in scaled agile software development maturing process. Thus when people play the major role, it is explicit that processes come in a secondary place in our view. Majority of maturity models in software engineering field, are based on process definition and strict
regulations (Maier, Moultrie, & Clarkson, 2012). This give rise to criticisms on number of practical challenges such as lack of feedback loops, Lack of theoretical and empirical base on the design guidelines and limited focus on the aspects involved in the design components (Maheshwari & Janssen, 2013). Though these challenges and criticisms are partially addressed in literature, the criticism in defining process constructs in measuring the agility of the team is often neglected in the design of the maturity model. We took this perspective into account in our design guidelines and propose fostering ambidextrous nature of the teams by providing practices with clear expected outcomes, but leaving space for exploitation of practices With this we added the perspective of developing ambidextrous ability to mature in scaled agile systems, which is a challenge due to fact that there is no unique recipe to ambidexterity concerning a particular organization. With this line of argumentation we propose that the scaled agile maturity in this research doesn’t provide a one-size fit all approach, rather gives space for tailoring the maturity model based on challenges faced by an organization.

Though there are quite a number of adaptations between the initial and final design of the model, we still believe there is always a road for improvement. Hence we would like to reflect on the existing criticisms of the maturity model on our scaled agile maturity model.

Table 27 Criticisms on the model

<table>
<thead>
<tr>
<th>Criticism</th>
<th>Authors</th>
<th>Relevance to this research</th>
</tr>
</thead>
<tbody>
<tr>
<td>One size fits all approach</td>
<td>(Maheshwari &amp; Janssen, 2013)</td>
<td>Model can be tailored according to the company needs and the model claims that one –size fits all approach is inappropriate for scaled agile or agile methods.</td>
</tr>
<tr>
<td>High process definition levels</td>
<td>(Fontana et al., 2015)</td>
<td>Model rejects the idea of defined process definitions for the scaled agile team. It claims that teams cannot sustain agility at higher process definition levels.</td>
</tr>
<tr>
<td>The emergence of the practices “is not simply a random process, but something that occurs to achieve an intended vision where the detail of that designed future is not fully understood at the time of the action”</td>
<td>(Allison &amp; Merali, 2007)</td>
<td>The idea on providing an intended vision for emergence of a practice is acknowledged in this research. Hence the model leaves space for the team to explore how the practice can be adopted and whether the practice should be adopted or not specific to the challenge faced by the organization.</td>
</tr>
<tr>
<td>Objective assessment criteria</td>
<td>(Sidky &amp; Arthur, 2007)</td>
<td>The process of improvement needs to account for reactive, reflective changes if the processes are to be improved not just extemporized. We claim through our model</td>
</tr>
</tbody>
</table>
that there is a need to promote sustainable development of the processes by integrating the experiences of the developers, their learning through action, and sharing that learning and not just mere adaption of practices.

| Least attention to process metamorphosis | (Allison & Merali, 2007) | Though the research aims at emergent nature of the practices and maturity, the dynamics of emergence is underexplored. The emergent properties of practices mean that they cannot be known a priori, but it does not mean that there is no intended design for an action. Hence it will be proposed as a future study as how the practices enlisted in the model intend to emerge. |
| Change in Software process improvement is not linear through time, nor is it uniform across all actors or all tasks | (Chikhale & Mansouri, 2015) | We claimed to have discussed on the first point that the software process improvement is not linear through time, but an adaptive process which changes in reflection to the external environment. However the uniformity of the change across all the actors is unexplored in this research. There are multitude of stakeholders who contribute to the scaled agile process. Emergence of a practice creates different impacts on the actors. Though there is no unique solution on how the actor might pursue the notion on emergence, exploration on that notion might create an additional view on emergence. |

In addition to these criticisms we would like to add on quite a few reasons on how the model would be effective as a discussion tool. One might wonder, as why they need such a model as a discussion tool and a discussion can quite happen in an organization without such a guide. No way have we promoted that the model has to be involved in all the discussions related to scaled agile methodologies. However, we question on how a company is supposed to mature in scaled agile if they don’t have a blueprint on how the scaled agile practices and processes evolve across the team, program and portfolio practices. The model would act as a starting point for the discussion and act as a blueprint (not a roadmap) to the organizations trying to adapt scaled agile methodology. Likewise an agile or scaled agile methodology, the model should be continually evolving as the team or organizations learns...
and grow. While we don’t support the notion of one model which fits for all the scaled agile challenges faced by an organization, we aim at kick starting the process of how an organization can look forward in the future in terms of scaled agile approach. Regardless of which practice the team choose to adopt, we strongly suggest that this model be used in the spirit of collaboration. Hence this model would prove effective as a beginner’s guide on scaled agile approaches and as a discussion tool in evolving towards a scaled agility.

In addition we would also like to revisit the design guidelines proposed in the Chapter section 3.5 Requirements of scaled agile maturity model and analyze the final design taking the design guidelines into account.

![Figure 31 Revisiting design guidelines](image)

Though we aimed to achieve the design guidelines in eliminating the waterfall based strict regulations and context dependent design, incorporating the agile factor along the stages is not fully achieved. This is due the adaptations made in the final stage of the research, with inputs from expert evaluation. However we claim that the team, program and portfolio levels used as stages incorporates the scaled agile factor into the model.
<table>
<thead>
<tr>
<th>Stage 1 – Team level streamlined</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1P1</td>
<td>Pair programming</td>
<td>S1P5 Clear definition of done tasks</td>
<td>S1P13 Short iterations</td>
<td>S1P12 Task boards</td>
<td>S1P15 Automated Status reporting</td>
</tr>
<tr>
<td>S1P2 DEV-UX</td>
<td></td>
<td>S1P6 Coding standards</td>
<td></td>
<td>S1P13 Sprint Backlog</td>
<td></td>
</tr>
<tr>
<td>S1P3 Clear definition on scrum</td>
<td></td>
<td>S1P7 Code refactoring</td>
<td></td>
<td>S1P14 Scrum ceremonies</td>
<td></td>
</tr>
<tr>
<td>roles and responsibilities</td>
<td></td>
<td>S1P8 Defect analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P4 Self-organized teams</td>
<td></td>
<td>S1P9 Exploratory testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S1P10 Test first development</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

- Common tool of communication for all stakeholders.
- Code review tools
- Issue tracking tools, productivity tools
- Issue tracking tools

<table>
<thead>
<tr>
<th>Stage 2 – Team level matured</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2P1 Managing distributed teams</td>
<td></td>
<td>S2P3 Test driven development</td>
<td>S2P11 Stakeholder alignment</td>
<td>S2P12 Adaptation sessions – Inspect and adapt workshops team level</td>
<td></td>
</tr>
<tr>
<td>S2P2 Increased collaboration</td>
<td></td>
<td>S2P4 Acceptance driven development</td>
<td></td>
<td>S2P13 HIP Sprint – team level</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P5 Behavior driven development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P6 Continuous integration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P7 System integration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P8 Frequent builds</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P9 Measuring software quality characteristics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S2P10 Tracking iteration progress</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

- Wikis and blogs
- Online project management tools
- Shared mailing list and folder
- Automated deployment process tools
- Agile metric estimation tools
- Issue tracking tools, productivity tools
- Issue tracking tools

<table>
<thead>
<tr>
<th>Stage 3 – Program level streamlined</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3P1 Clear definition of program stakeholder roles and responsibilities.</td>
<td></td>
<td>S3P2 DevOps</td>
<td>S3P13 Program Backlog</td>
<td>S3P16 Inspect and adapt workshop PI level</td>
<td></td>
</tr>
<tr>
<td>S3P2 Clear architectural runway</td>
<td></td>
<td>S3P3 Team level increments for functional and non-functional requirements.</td>
<td>S3P14 Release planning</td>
<td>S3P17 HIP sprint – program level</td>
<td></td>
</tr>
<tr>
<td>S3P3 DevOps</td>
<td></td>
<td>S3P4 Estimating funding for agile release trains</td>
<td>S3P15 Portfolio backlog</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

- Automated building and deployment tools
- Issue tracking tools

<table>
<thead>
<tr>
<th>Stage 4 – Program level matured</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S4P1 Stakeholder alignment</td>
<td></td>
<td>S4P1 Clear architectural runway</td>
<td></td>
<td>S4P2 ART (Agile Release train)</td>
<td>S4P3 PI Execution System demo</td>
</tr>
<tr>
<td>S4P2 ART (Agile Release train)</td>
<td></td>
<td>S4P3 PI Execution System demo</td>
<td></td>
<td>S4P15 Portfolio backlog</td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

- Issue tracking and productivity tracking tools

<table>
<thead>
<tr>
<th>Stage 5 – Portfolio level Streamlined</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S5P1 Strategic themes estimation</td>
<td></td>
<td>S5P2 Measuring business objectives</td>
<td></td>
<td>S5P15 Portfolio backlog</td>
<td></td>
</tr>
<tr>
<td>S5P2 Measuring business objectives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5P3 Portfolio backlog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

<table>
<thead>
<tr>
<th>Stage 6 – Portfolio level matured</th>
<th>People and interactions</th>
<th>Technical expertise – Built in quality</th>
<th>Incremental building and continuous flow</th>
<th>Alignment</th>
<th>Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>S6P1 Program portfolio management</td>
<td></td>
<td>S6P2 Estimating funding for agile release trains</td>
<td>S6P16 Common language</td>
<td>S6P14 Scrum ceremonies</td>
<td></td>
</tr>
<tr>
<td>S6P2 Estimating funding for agile release trains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6P3 Budget estimation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tools**

- Portfolio Kanban tools
- Portfolio Kanban tools
6.5 Scientific relevance

As introduced in Chapter 1 Introduction, this thesis contributes to the creation of a new body of knowledge in the area of scaled agile, by investigating the scaled agile practices used in implementing the scaled agile methodology. We claim that as the main contribution to the academic world, with its uniqueness it brings in helping the organizations in adopting to scaled agile methodologies with respect to finding the areas of improvement in terms of scaled agile practices.

First of all, we determined in the introductory chapter that there was a lack of academic body of knowledge for the implementation of scaled agile software development in a large scale organization. Subsequently, we filled part of this gap by designing a scaled agile maturity model, along with the range of indicators to be considered for the implementation process.

We place the design of scaled agile maturity model in the domain of Socio-technical systems design (STSD), in which the design considers human, social, organizational and technical factors. We argue that there is a need for a pragmatic socio-technic approach to software development methods, focusing on both social and technical factors that influence the environment surrounding the agile systems. With this line of argumentation, we consider scaled agile software development system as a socio-technical system which mainly focuses on improving the efficiency through increased attention to human factor technology and external environment. This aspect of socio-technical design can be clearly seen in the design of scaled agile maturity model, where both the social factors (people & interactions) and technical factors (Incremental building and technical expertise) are placed as main pillars of the design. This research thus contributes to the new body of theory on socio technical design for scaled agile software development.

We also identified from the literature that the maturity models existing in literature, have strict regulations to move from one stage to another, which resembles the waterfall based approach. In order to be more agile not only in the methods the model preaches, but also on the design constructs, we introduced the concept of exploration of practices before moving on to the proper implementation of the practices. Moreover, the practices introduced in the model can be tailored by the user of the model for level of completeness, which highlights the guidance perspective of the model, instead of a strict rule based approach (with the help of assessment criteria as seen in appendix 3).

The model includes six stages as rows and five scaled agile principles in columns such as People and interactions, incremental and continuous flow, alignment, transparency and technical expertise, nevertheless there is always a scope for improvement. Therefore this thesis provides not only descriptive conclusions in the form of scaled agile maturity model, but also future recommendations for further research (section 6.8). We hope that this research will help the future researchers in contributing to the field of scaled agile methodologies.
6.6 Societal relevance
This thesis represents a design-oriented research, given the fact that the end goal represents a design of a scaled agile maturity model. Organizations which are planning to adopt scaled agile methodologies or organizations which have adopted scaled agile and are in need to find areas of improvements will be benefitted by careful analysis of the assessment criteria. So clearly the research aims in improving the quality of adoption and maintenance of scaled agile methodologies in a large scale IT organization. Stakeholders such as scrum master, release managers, developers, product owners and internal stakeholders across program, portfolio and team levels will greatly benefit from the model in finding areas of improvement in adopting to scaled agile approach. The societal relevance of the artifact is that, provided the scaled agile maturity model is used as a discussion tool for collaborative decision making, the model might contribute to the continuous improvement measures regarding software processes in the IT community. In addition, the research takes the productive interaction of societal stakeholders through means of expert ex-ante evaluation, which is considered vital in evaluating the societal impact of the maturity model that we developed. Furthermore societal stakeholders are greatly involved in the evaluation of the research, by involving them in the research committee, their inputs are clearly interwoven along in each phase of the design. This way the knowledge has been disseminated to the societal stakeholders, pertaining to the context of research.

6.7 Limitations
In order to complement the robust findings of the scaled agile maturity model, we would also like to acknowledge the limitations that hampered the research. The limitations described in this section have been extracted in each and every phase of the research journey, while the adaptation decisions to scope and de-scope the design were channeled through the voice of experts in the evaluation phase.

One of the limitations is the way the research design cycle was applied. In this research, we used an adapted version of, “Design Science Research Methodology” proposed by Pffers et.al(2005). While the steps such as “Problem identification”, “Definition of objectives”, “Design and development” are performed in the study, the step on demonstration is neglected in this study and is proposed as future research (section 6.5). Nevertheless we also consider the scholars point of view in deviating from the sequential order of steps, in relevance to the problem being addressed and the nature of the problem.

Another limitation is given by the type of sources that was used a research design tool, to construct this report. Due to newness of the topic we used non-academic references to derive the scaled agile process and practices in Chapter 2 (i.e articles published in internet by the SAFe founder Lean Deffingwell and case studies published on the SAFe’s public website). Though we tried to provide an extensive overview of the scaled agile process with the means of the sources we had, some of the elements might have been left out due to the unavailability of (more) sources.

With regards to the data gathered form the expert interviews, the respondents are mainly from one particular organization which adapted to SAFe as a scaled agile methodology. Firstly, the range of scaled
agile practices adopted, differs among organizations. This is considered as a main limitation in populating the model only with the SAFe practices, since the language and naming of practices might create confusion among the respondents if the practices seem unfamiliar for them to evaluate. Secondly, the experts selected are from one particular industry (automotive), in the sense the response might have been biased to a specific industry. Although we have started to approach the experts for expert evaluation phase in the early stage of the design (late May 2016), due to non-availability of the persons, it was not possible to conduct interviews with a larger group and with respondents from several organizations with another industry domain as a background. Given the fact that we had interviews with members from a specific industry, we have a small representation of the targeted audience. This might hinder the generalizability in various ways. First concerning the users of the scaled agile maturity model, the internal governance, process, and addressed problems are company specific. While there are high similarities between companies within the same industry, roles and responsibilities might differ across companies. As already indicated there might be a discrepancy on how the model is perceived across the companies due to the specific name given to the scaled agile practices and responsibilities enlisted for the stakeholders. We claim that the set of considerations (principles and scaled agile practices) are not fully organization-specific and to a greater extent, can be orchestrated to other organizations as well. They are meant to provide a baseline from which any large scale IT organization interested in scaled agile methodologies, can adjust the model to its own baseline of practices. This will therefore be a topic of starting a meaningful discussion in finding areas of improvements in adapting to scaled agile practices.

Also the completeness of the framework is a major limitation. The outcome of the research, the scaled agile maturity model contains five core principles, from which the practices are populated. However not all of the five principles were investigated in full depth. This is due to various reasons: availability of information, thesis scope and lack of previous publications on scaled agile. Based on these factors, a lot of attention was dedicated to scaled agile practices. Furthermore, focus of the thesis was more on the process knowledge of the scaled agile development and less on the technical aspects. Most of the technical changes were to a smaller extent mentioned in the model. However this aspect has been tackled in chapter 3, in which we performed an in-depth review on both technical and process aspects of various maturity models in literature.

Apart from the limitations raised in this section, we claim that with our model we made an essential first step towards a structured approach for large scale IT organizations in adapting towards scaled agile approaches, by letting them have a faster means of assessment in assessing the practices and process changes.

6.8 **Recommendations for future research**

This current research has provided a structured approach to advocate adoption of scaled agile methodology and an adaptive model to find areas of improvements for further initiatives. Furthermore several indicators have been identified, which forms the exhaustive list of assessment criteria for the model.
However there is always scope for improvement and space for further research while considering the limitations of the research. This is elaborated in this section as follows:

1. Firstly, we recommend investigating characteristics of the firm in coming up with scaled agile practices. As discussed in chapter 2, organizations differ in scaling their practices either through product level or platform level. Investigating the model based on these 2 dimensions might open different elements for study.

2. Another area of research could be diving deeper into the scaled agile principles in addition to those proposed in this thesis. For this analysis, we came with mapping the principles based on their characteristics and came up with columns on principles. This aspect provides the first indication on one of the strategies to be followed: de-mapping the principles and in-depth of each principle on its own.

3. Another possibility would be to explore each of the practices in detail and recording the outputs. As discussed in one of the design guidelines, agile team is an adaptive system in which the team adjusts itself for new practices and outcomes has to be studied before implementing the practice in real time. This exploration phase of the model will provide organizations with additional detail on the practices.

4. Furthermore, implementing a range of practices differs with respect to decisions on the management level. A subsequent area of research would be to engage stakeholder analysis for the implementation and adoption of all the practices involved in the model. As already discussed in Section found. Further research on dynamics of emergent practices is needed. As the practices emerge in an organization, the stakeholders would require a discussion guide on how the practices dynamically evolve for the next few years. Though the emergent properties of practices cannot be known a priori, future research on intended design of such an emergent property would create additional discussion domains.

5. As a continuation to the four research questions proposed in this study, a fifth research question would be to do an ex-post evaluation by implementing the model in an organization and record for validation outputs.

6.6 Reflection

As part of last aspect for consideration in this research, I would like to stress some important reflections with respect to System Engineering Policy Analysis – Information Architecture (SEPAM –IA) curriculum.

From a SEPAM - IA angle, this research is highly relevant to the curricula that are structured around the main themes: large scale and complex problems related to both the public and private sector, designing technical/ management solutions for socio-technical systems and process management strategies. These themes are of great importance in this research, as the business sector (software community) is placed as a central actor and involves designing a solution for a problem being addressed by the actors.

With regards to curriculum this research includes various concepts and knowledge gained from the lectures and project work from the courses as follows: Software Architecture [IN4315], Service
systems engineering [SPM5430], Thesis project definition [SPM5905], IA design project [SPM5920IA], ICT Management [SPM9640], ICT Design and valorization and mobile applications [SPM 9631].

“Software Architectures” discipline gave an idea on underlying high level structure of software systems and the software elements which compose those systems. This gave us a start in analyzing the software systems for new ideas and research opportunities in the field. Many of the agile practices are studied in depth during the course of study, through a real-time open source project, which helped in gaining a proper understanding of agile software development in the business world.

“Service System Engineering” provided a fundamental direction on the design cycles and a couple of the lean-agile practices such as user stories, acceptance criteria and design of a public platform. The design cycles employed in the course of study gave a complete understanding on the stages such as “problem identification” and “objectives formulation” in the digital ecosystem.

“Thesis Project Definition”, one of the intensive courses for the whole master program and I consider this course as an excellent source of inspiration in kick-starting the master thesis. It helped me to understand the complexity of the master thesis project and prepared me in facing the journey of master thesis with good hopes. By analyzing various articles, conducting a literature review, having meaningful discussion and constructive feedback with/from the fellow students, peer reviewing of other student's ideas and inspirational feedback from the mentors, I gained a holistic understanding on writing a project proposal, formulating the research scope. This helped me a lot in employing a meaningful research design in the thesis. Although, this research is not a 100% continuation of the “Master thesis project definition” course, I consider that the course kick-started my trajectory towards the research.

“IA design project” and “ICT Management” courses provided fundamental direction on topics such as software development systems and digital arena which helped in identifying the digital ecosystem in the business world. “ICT design, valorization and mobile applications” course provided the design steps and logical pillars in designing an IT artefact. The project work for this course helped me in getting an idea of good and bad design choices in designing an IT artefact, which is considered as a main building block in this research. Six-step design cycle approach proposed by Verschuren et.al (2010) was introduced in this course, which helped as a starting point for this research in considering the design cycle approach in designing the model.

From a personal perspective, I acknowledge that writing a master thesis is a remarkable experience and an exceptional arena, where I can showcase all of my knowledge obtained during the master program to the community. To begin with, during the journey of master thesis I have enhanced my cognitive and analytic skills required to put pieces of puzzling information in a logical way. It helped me in various phases of the project such as conducting a literature survey and Expert evaluation, where a lot of information had to be structured, in order to develop an initial design of the model.
In addition I would say, a critical way of thinking is required during the data analysis phase, where an objective analysis and evaluation of the dialogues (from the interview) is of paramount importance in arriving at a judgement on what a respondent tries to propose through his/her arguments. Without being said, master thesis requires a lot of dedication and discipline, in achieving the results at right time along the timeline proposed in the kick-off session.

Furthermore, my professional skills such as presentation skills, organizational skills, team building skills and communication skills improved throughout the course of the thesis. I have to give credits to TomTom and my external supervisors, who involved me in several discussions and participatory observation of various day-to-day meets in TomTom. This helped me in preparing myself for the professional world in advance, which would be my next step after graduation.

Nevertheless, I have to acknowledge that this remarkable journey would have not been possible without the constant support from my TU Delft staffs, TU Deft supervisors, TomTom mentors and employees of TomTom.


e=pdf


Appendix
**Appendix 1**

**Interview structure for expert evaluation**
- Short introduction on my background
- Description on the project

**Table 28 Expert evaluation interview probes**

<table>
<thead>
<tr>
<th>Theme</th>
<th>Topics</th>
<th>Personal notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction to scaled agile concept</strong></td>
<td>Talk about their professional background, roles and responsibilities in the current position etc.</td>
<td>Check if they are familiar with the scaled agile methodologies. Introduce the concept of scaled agile framework and why it is needed in the first place.</td>
</tr>
<tr>
<td><strong>General discussion</strong></td>
<td>What was in their view the challenge the organization was trying to overcome by moving to scaled agile practices?</td>
<td>Now that, scaled agile practices, do they consider those challenges have been overcome? What challenges they faced in their team in particular before adapting to scaled agile practices? How could they foresee the risks and gains of adapting a scaled agile practice? Was there any maturity model used prior?</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td>Ask whether there are any ambiguities in the initial design pattern</td>
<td>If the reply to the above question is positive, ask whether they can exemplify any strategy to remove the ambiguities.</td>
</tr>
<tr>
<td><strong>Practices</strong></td>
<td>How do they consider the placement of practices in different stages and principles?</td>
<td>What are the scaled agile practices they consider has been not used in the model?</td>
</tr>
<tr>
<td><strong>Indicators</strong></td>
<td>What are their views on the concept of using indicators for all the practices?</td>
<td>How would they see the indicators operationalized in practice?</td>
</tr>
<tr>
<td><strong>Open questions</strong></td>
<td>Ask if there are some topics were not engaged during discussion? Ask for contacts (further references) if they know of who wishes to talk in this topic.</td>
<td></td>
</tr>
<tr>
<td><strong>Summarizing interview</strong></td>
<td>Give them the contact information on how they could contact me for further discussions.</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix 2

### Table 29 Indicators for the assessment criteria

<table>
<thead>
<tr>
<th>Practice</th>
<th>Indicators</th>
<th>Not achieved</th>
<th>Partially achieved</th>
<th>Fully achieved</th>
<th>NA</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP1</td>
<td>The organization follows pair programming concept of working as one of the agile practices</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Each member of the team value the results of pair programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP2</td>
<td>Development team and UX have a transparent agile board with great visibility on the team’s backlog (both dev and UX)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dev and UX holistically plan the sprint and follow the sprint ceremonies.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indusion of UX criteria in definition of done UX iterates one sprint before the dev team</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP3</td>
<td>Each team is well aware of their roles and responsibilities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP4</td>
<td>Team members are empowered to evolve their own practices and environments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team members solve their own problems and make considerable improvements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP5</td>
<td>An upfront common understanding of “done” is established for the tasks involved in the team.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The level of DoD is detailed out to artefacts such as user story, sprint, and releases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP6</td>
<td>The team has a common and accepted style guide for coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coding standards documents are updated on a regular basis.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP7</td>
<td>Refactoring the code base is included as an enabler story in the team backlog and in line with story estimated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP8</td>
<td>Defects of the code base is included as an enabler story in the team backlog and in line with the story estimated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP9</td>
<td>Acknowledgement of the team that the automation testing is not sufficient to ensure that the system does everything we expect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP10</td>
<td>Team employs the act of producing automated unit tests for production code, before writing the production code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For every piece of code written, team builds and runs tests that helps in evaluating the code.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP11</td>
<td>Team builds the solution incrementally in a series of short-time boxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Each time-box results in an increment of a working system that can be evaluated by the system builder and the customer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subsequent time-boxes build upon the previous increments and the solution evolves until it is released.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Teams follow routine synchronization to help assure that the evolving solution addresses the real and current business needs, as opposed to the assumptions that were established at the beginning.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P12</td>
<td>Team employs a scrum task board to make the sprint backlog visible to the team members</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P13</td>
<td>Team identifies a list of tasks to be completed during each sprint and organizes a backlog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P14</td>
<td>Team employs the scrum ceremonies on a regular basis for each sprint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs daily stand up of all the team members on a day to day basis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>All team members collaboratively participate for all the scrum ceremonies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs sprint review session after every sprint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs sprint demo session after every sprint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs sprint retrospective after every sprint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P15</td>
<td>Team employs tracking tools to track the status of tasks involved in the sprint backlog, transparent to all team members</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1P16</td>
<td>Team employs a transparent common language within the team to denote the artefacts.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P1</td>
<td>Distributed teams effectively participates in all the scrum ceremonies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs an effective tool chain for requirement repositories, management, build and deployment, and project management. (Technology alignment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs remote pairing of team member to offer knowledge sharing and to foster collaboration among the team members.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P3</td>
<td>As a next step after TFD, team employs regular refactoring the code to clean up the growing code base.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team employs the mantra “Red/green/refactor” to ensure effective test driven development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P4</td>
<td>Team employs acceptance tests as a part of overall testing strategy, in which customer intent is specified in the tests.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P5</td>
<td>Team employs behavior tests as a part of overall testing strategy, in which the tests are specified in terms of desired behavior of the unit.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P6/S2P8</td>
<td>Software is always in a releasable state</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release time box is well defined and equal to or less than the business need.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comprehensive automated tests suites are created through TDD/BDD/ADD and maintained by developers and testers working together.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No work is considered done until it has passed unit acceptance tests associated with it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All the new requirements describe how the value of this feature will be measured.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P7 Systems are architected with continuous deployments in mind.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team employs frequent system and solution level integration.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team employs regular system and solution demos.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team employs incident management process, in which the production incidents over time and recurrence are measured. During the process the team defines the root cause and implements corrective measures which enables continuous improvements and prevents recurrence.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team captures the user sentiments by measuring how they feel when they interact with the application.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect trends are analyzed before each production release (Number of open defects, Defect cycle time and Defect spill over).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team employs techniques to track the progress of iteration.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Planned velocity – Tracking the total number of story points the team estimates to complete.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Sprint burn down.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Release burn down.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Scope creep – Total number of hours worked on a task versus the total number of hours estimated in the planning for a task.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team employs the exercise of evaluating the progress in one of the scrum ceremonies (sprint review/Sprint planning).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P11 Team employs all the stakeholders involved in the process for planning activities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2P12 Team employs Inspect and adapts sessions with all the relevant stakeholders, in which the lessons learned are incorporated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3P13 Team employs a hardening sprint in which the team stops focusing on building new features or architecture and instead spends their time in stabilizing the system to get ready for the release.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3P1 Roles of all the stakeholders are clearly communicated and defined.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product management</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop and communicate the program vision and roadmap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develops feature acceptance criteria that can be used to establish the “definition of done” for the feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaborate with Product owners on the team level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work with epic owners to develop the business case for epics that are related to the program increment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**System Architect**

- Work with customers and stakeholders to establish high-level solution intent
- Supervise and foster built in quality
- Establish non-functional requirements at the solution level

**Release train engineer**

- Facilitates Program increment planning
- Coach leaders, teams and scrum masters at team level for scaled agile practices
- Facilitate periodic synchronization of meetings involved in Agile release trains
- Escalate and track impediments of the agile release train

---

**S3P2**

The organization employs DevOps as a scaled agile practice under which the development teams and the operations teams collaborate on a continuous basis.

The deployment operations or production team members are active on the agile release train and fully engaged in the process. (The operations team is continuously engaged with the development team throughout the life cycle of solution development)

---

**S3P3**

The organization employs a metric system to measure the efficiency of DevOps

- Release date adherence percentage
- Time taken for release to production
- Defects attributed to platform requirements

Ops team is continuously involved in the sprint review and involve in continuous engagement and collaboration with the development team to predict the accuracy of the potential release dates.

---

**S3P4**

Product management together with solution management team identifies and prioritizes the backlog before the program increment planning session.
<p>| <strong>S3P6</strong> | All the teams involved in the agile release train involve in inspect and adapt sessions in which the lessons learned are incorporated for the previous program increment. |
| <strong>S3P7</strong> | Team employs a hardening sprint in which the team stops focusing on building new features or architecture and instead spends their time in stabilizing the system. |
| <strong>S4P0</strong> | Status reporting is automated and based on objectives of the working systems. Teams and programs have visibility into business and architecture systems. Portfolio managers and other stakeholders have a clear understanding on the PI goals and solution. |
| <strong>S4P1</strong> | The organization employs emergent design in place of Big upfront design. The architectural design evolves hand in hand with the business functionality, through constant testing and refactoring. The organization employs modeling constructs such as domain modeling or use case modeling in modeling the problem to gain a better understanding of the potential problems. The models are recorded as a solution intent, which is discussed in the PI or release planning events. The organization employs a collaborative effort in innovating the architecture, involving all the stakeholders such as agile teams, architects and engineers. The UX team tracks the artifacts of the architectural runway and get ready with the design guidelines and prototypes for the new program increment functionality. The UX team continuously validates the user experience/architectural designs via user experience testing, (via creating user personas. |
| <strong>S4P2</strong> | <strong>Planning readiness</strong>  Program backlog is ready to be discussed  Team backlog is ready to be discussed  <strong>Release planning event</strong>  Program stakeholders, business owners and relevant stakeholders are present  Risks are discusses and addresses  All teams are agreed with the SMART objectives  Confidence vote is exercised for the PI objectives  <strong>Portfolio alignment</strong>  Program vision and roadmap are aligned with portfolio vision. |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S4P3</strong></td>
<td>Product management effectively works with epic owners to split the epics and prioritize them into program backlog</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scrum of scrum meets routinely and effectively</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scope is managed effectively</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Teams proactively interact outside of scrum of scrums to address dependencies, impediments and resolve issues</td>
<td></td>
</tr>
<tr>
<td><strong>S5P1</strong></td>
<td>Strategic themes are established which helps in connecting the evolving enterprise strategy to portfolio level</td>
<td></td>
</tr>
<tr>
<td><strong>S5P2</strong></td>
<td>Early indicators are established to measure the progress of the strategic themes</td>
<td></td>
</tr>
<tr>
<td><strong>S5P3</strong></td>
<td>Strategic themes are converted into epics, forming the portfolio backlog</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inputs to the portfolio backlog is accompanied by business outcomes and return on investment</td>
<td></td>
</tr>
<tr>
<td><strong>S6P1</strong></td>
<td>Organization employs a team for program portfolio management which represents the people for fiduciary decision making to set initiatives in a business unit</td>
<td></td>
</tr>
<tr>
<td><strong>S6P2</strong></td>
<td>Program portfolio management measures and allocates funding to strategic themes and agile release trains</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Program portfolio management assists, drives and support program execution</td>
<td></td>
</tr>
<tr>
<td><strong>S6P3</strong></td>
<td>Involves lean-agile budgeting, in which budgets are allocated for each epic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Budget estimations are constantly reviewed for budget creeps</td>
<td></td>
</tr>
</tbody>
</table>
Appendix 3: Scientific article

Stages versus people: Using maturity model as a discussion tool for identifying maturity in scaled agile software development systems
Revathi M Chandrasekaran (4385276)
Faculty of Technology Policy and Management
Delft university of Technology
R.MangalrevuChandrasekaran@student.tudelft.nl

ABSTRACT
Researchers have proposed many agile software development maturity models to provide guidelines for agile process improvements. Most of them describe various maturity levels and processes the team should adopt to achieve certain goals in a level. These models differ in their underlying structure describing different possible paths to maturity in agile software development systems. The team then uses the model to assess its current level and identify the practices that would guide them to maturity. However the struggle for the agile team to follow the prescribed practices and processes has always been neglected in constructing the model. This is due to the fact that the agile practices are highly customized for specific contexts for a team and as a result the team doesn’t always rely on standard processes. This research is aimed at addressing this knowledge gap by exploring how agile or scaled agile teams evolve to maturity which takes the nature of agile teams into account, rather than stressing on predefined maturity levels. To answer this, we investigated the former maturity models available in literature and through participatory observation in an agile-centered organization. The result is shown as a vision for the maturity model in which people play a central role in agile and scaled agile software development systems, with the teams given a scope to explore the agile or scaled agile practices, and the maturity model identifying the progressive outcomes of adapting to a particular practice by means of discussion with the identified stakeholders. With the team given the exploitation and exploration choices on practices, the future research directions would be to investigate the assessment approaches in such an ambidextrous environment.

Keywords: Scaled agile, Maturity model, Agile software development, Maturity
1 Introduction
Information technology (IT) support of business processes is becoming increasingly important for organizations to help them in their development and to improve competitive advantage on newest developments. Hence organizations look for innovative practices and trends to continuously improve IT performance and thereby the economic efficiency (Henderson & Venkatraman, 1999). As such agile and scaled agile software development methodologies came into existence, which support organizations in building more adaptable software development system by increasing their responsiveness in business change. Organizations started using maturity models to assess its own situation in agile and scaled agile systems and find guidance to improve in a specific focus area (Fontana, Meyer, Reinehr, & Malucelli, 2015). A number agile maturity models has been proposed in the last years taking into account the guidelines proposed by Capability Maturity Model Integration for Development (CMMI-DEV) and International standard ISO/IEC 15504, which strictly focus on process improvements that are previously planned and designed. They further stress on continuous improvement only through institutionalization and quantitative management of these processes (CMMI Product Team, 2010). This is in contrast to the agile principles which stresses on interactions and people over processes. Therefore, a number of agile maturity models proposed in literature are built over agile values and the roadmap they suggest consider agility at the higher maturity levels than defining quantitative process improvements (Ozcan-top & Demirörs, 2013).

However, a couple of issues linger: First is that, though the maturity models consider agile values into account, they prescribe practices corresponding to hierarchical maturity levels. This is in contrast to the highly context specific nature of the agile or scaled agile teams, where certain practices cannot be forced to be accomplished before moving on to the next level. Second is that, the assessment criteria for the prescribed practices strictly deal with how the practices has to be implemented in an ideal agile/scaled agile setting, without being able to discuss on the suitability of the practices in an organization.

These issues thus, motivated this research. The goal of this paper is to develop a vision for the maturity model that overcome the issues lingering in the current maturity models, with specific focus to scaled agile environment. With this goal in mind, we formed the primary research question for this study: What characteristics should a maturity model possess to support the agile teams towards maturing in scaled agile environment?

To answer this question, we first investigate the maturity models in general. We then review the literature on agile maturity models in Section 2. In section 3, we explore the issues faced by the current maturity models towards the scaled agile environment. We provide the empirical evidence on the characteristics of the scaled agile environment in Section 4. Section 5 discusses on the research questions by presenting the characteristics and vision of the proposed maturity model that helps the agile teams in maturing towards scaled agile environment. In section 6, we draw conclusions and recommendations for future research.

2. Related work
2.1 concepts of maturity model
To get a better understanding on what maturity models are and the benefits of using such a model, the following section discusses the basic concepts of maturity models.

Glimko et.al (2001) gives a description on maturity models as “Maturity models describe the development of an entity over time. This entity can be anything of interest: a human being, an organizational function, etc.” (G.Klimko, 2001). Willem et.al (2007) defines maturity model as a “Structured
collection of elements that describe the characteristics of effective processes at different stages of development. It also suggests points of demarcation between stages and methods of transitioning from one stage to another” (Willem.P, 2007). In addition to that, many publications refer to the description provided by CMMI as a basic definition in defining a maturity model, which is described as “A framework representing a path of improvements recommended for software organizations that want to increase their software process capability”

Though these definitions give us the bottom line on the usage of maturity models, we need a clear understanding on elements of such a model and meaning of maturity in general. The term maturity is defined as “the state of being mature: fullness or perfection of development or growth” in oxford English dictionary. Further in the field of Information technology, maturity is normally referred to capabilities, in which capability is described as “the power or ability in general to fulfill specified tasks and goals” (Simpson & Weiner, 1989). Hence from a linguistic perspective maturity models will serve to ensure the conditions of growth when a specific element reaches the perfect state in their intended purposes(Wendler, 2012).

Next characteristic which has to be investigated when such a fullness of growth/maturity is reached is the perspective on the end goal of designing such a maturity model. Tom et.al (2009) defines two perspectives in defining the purpose of maturity models: Life cycle perspective and Potential performance perspective (Tom McBride, 2009). Life cycle perspective measures the state of the growth of the process areas, taking the evolution of an organization into account in evaluating the maturity. This perspective names the final stage as “Fully mature” and represents the ideal stage of maturity. This perspective aims at fulfilling certain characteristics at a particular stage of maturity and can evolve to next stage only if the characteristics are fully performed. Whereas, potential performance perspective defines the maturity stages through means of a development path which focusses more on the potential improvements which might occur by moving along the model. This perspective gives user the final decision to decide which stage of maturity is the “level of completeness” according to their situation(Wendler, 2012).

To summarize, maturity models define and explain the state of perfection or completeness of certain elements or objects and ensures a desired evolution or performance path along clear discrete stages. Though life cycle perspective provides a hard rule in evolving towards a maturity stage, we claim that the user should have final decision to decide the completeness of the element. This is in line with the argument that, every organizations face multitude of challenges and there can’t be a unique solution to all kind if challenges faced by an organization. Hence we claim taking the “Potential performance” perspective into account as a desired characteristic in designing a maturity model for agile teams.

The preceding section deals with maturity models with specific focus to agile systems.

2.2 Agile maturity models

Agile software development is an iterative and incremental approach to software development which takes the changing needs of stakeholders into account rather than the traditional software development methodologies which involves big upfront development without any increments or iterative loops(Palmquist, Lapham, Miller, Chick, & Ozkaya, 2013). Though agile approaches helped in overcoming the issues faced by traditional software development approaches, it has been widely acknowledged that agile approaches was only originally developed only for smaller teams(Tourtoglou & Virvou, 2012). Problems arise when there are dependencies between several agile teams working on a similar project, in a large scale organization. These dependencies can be either the dependencies between workflow in the development process, dependencies due to different software artefacts, or dependencies
due to tasks assigned to team members of different teams (Sekitoleko, Evbota, Knauss, Sandberg, & Ab, 2014). This led to the invention of scaled agile approaches in which the agile approaches are scaled from team level to program and portfolio level, where program level represents the configuration of all the teams working on a project and portfolio level represents the configuration of all the portfolio working on a similar strategic theme (Leffingwell, 2011).

Figure 1: Scaled agile environment

As agile values act as a core for both agile and scaled agile systems, we now investigate the agile maturity models available in literature. In the field of agile maturity models, two paths of research have been studied: First focuses on developing maturity models by adapting agile practices to the already existing software maturity models such as Capability Maturity Model Integration (CMMI). The second focuses on developing maturity stages by using agile values. The first group of studies started with discussion of Mark Paulk in investigating the complementary nature of combining an agile methodology like Extreme programming (XP) with CMMI (Paulk, 2001). Since then, number of studies has been explored on how agile methodologies can be combined with the CMMI (Al-Tarawneh, Abdullah, & Ali, 2011) McCaffery, Pikkarainen, & Richardson, 2008) Spoelstra, Iacob, & van Sinderen, 2011) Jakobsen & Johnson, 2008) (Tuan & Thang, 2013)

Second group of studies focused on agile values and gave a greater importance to improving agility at higher levels. For instance, the model proposed by Sidky et.al (2007) take agile values into account to define the five stages of maturity. Each level consists of set of practices corresponding to the agile values, with the last level focusing on sustaining the agile environment (Sidky, Arthur, & Bohner, 2007). Patel and Ramachandran (2009) propose a maturity model based on CMMI structure populated with agile practices. They define five maturity levels and the maturity gain is being measured through increasing process definitions based on the agile practices being adopted from stage one to five (Patel & Ramachandran, 2009). Benfield defines a maturity model based on seven agile practices or dimensions that collaboratively helps in increasing agility in an organization. These dimensions are represented through five maturity levels, with fifth level denoting practices which helps in sustaining highly productive teams and quick delivery of products (Benefield, 2010). Yin et.al (2011) proposes a maturity model for agile by explicitly taking only scrum practices into account. It comprises of five maturity levels to help organizations adopt to scrum practices in an incremental manner via suggested practices (Yin, Figueiredo, & Silva, 2011). Packlick (2007) defines an agile maturity model through goal-oriented approach, in which five goals enlisted in the model takes the self-serving acronym AGILE. With A for acceptance criteria, G for green bar test and builds, I for Iterative planning, L for learning and adapting and E for Engineering excellence. These goals are then mapped to five maturity levels. Instead of forcing on practices to be implemented, the authors define certain goals to be accomplished for certain stages. In this way agile team members understand the rationale behind the practice much better (Packlick, 2007).
3 Issues of current maturity models
We studied different agile maturity models in literature and investigated the characteristics of those models. In this section we discuss the issues of former maturity models, in order to arrive at goal of this study.

As can be seen from the review, most of the agile maturity models focus on agile practices in general and defines four to five maturity levels influenced by the structured process definitions defined in CMMI. Though the second group of studies pursues agile values in the design, they still focus on strict adherence to a structured process in defining the maturity stages, except for one publication Packlick (2007). The author considers the primary role of self-organization in designing the maturity model and explicitly stresses on the goal level approach instead of process oriented improvements (Packlick, 2007). This method of increasing self-organizing behavior in agile systems has been continuously researched upon by various scholars and proven to be effective in encouraging agility instead of forced process definitions (Hoda, Noble, & Marshall, 2013). The model’s key goal focused in overcoming prejudices against specific practices, by focusing more on goals. However the model lacked showing a holistic view on the practices from which the team can start exploring in working towards the goal approach. This is required in order to avoid inconsistencies in a scaled agile environment in which several teams work on a common goal. As can be noticed the characteristics on highest maturity differs in all the models. For instance, Sidky et.al (2007) stresses on sustaining agility at the last stage defined in the model, after all goals focused on the previous stages has been established. Whereas Patel and Ramachandran (2009) defines highest maturity through focusing on satisfaction of customers and stakeholders. However we claim that the models do not explicitly address the maturity concepts and there is a lack of consensus among the publications regarding the characteristics of highest maturity in a model.

Both our literature study and studies by different authors((Ozcan-top & Demirörs, 2013), (Fontana et al., 2015)) have identified the lack of consistent underlying structure of a maturity model.

4 Empirical study
We found the characteristics and issues of the former maturity models in literature that might influence the design and development of the maturity model. To help answer the research question in this study, we employed participatory observation in a scaled agile environment in a large scale organization. The goal of employing this research tool is to investigate the nature of scaled agile teams towards maturity. Participatory observation is used as a research tool because the project is exploratory in nature and in a relatively newer domain. This is particularly useful because of the nature of agile teams to be autonomous in nature and were described to be a learning systems that expand their decision space in response to every day learning (Hoda et al, 2013). In order to closely involved in the day to day activities of the team and observe the nature of scaled agile environment, participatory observation is chosen as a research tool in this research. Further, the study being a qualitative research, we presume that there will be multitude of perspectives within any given community (agile teams in this research). It is therefore essential to know the diverse perspectives and clear understanding of the interplay among them(Mack, Woodsong, MacQueen, Guest, & Namey, 2005). However there is couple of disadvantages to participatory observation which has to be acknowledged: Participant observation tends to be inherently subjective and requires a conscious effort to objectivity. The activity on reporting what is being observed rather than the exercise on interpreting what is seen helped in mitigating this issue and thereby reduces subjectivity in the research.
4.1 Participatory observation

In this research, the author acted as a participatory observant in a large scale organization, which adapted to Scaled agile practices in the year 2014. Author observed daily meetings, scrum ceremonies, planning meetings and release meetings, to understand and analyze the nature of practices and scaled agile teams in the organization for a period of four months. The research is restricted to participatory observation to one team due to limited time and scope restrictions. The organization followed an ad-hoc software development process, with a top-down approach before adapting to scaled agile practices. New roles and responsibilities emerged along with the adoption of scaled agile practices. The evolvement of agile teams towards a scaled agile environment shows that, the team started using scaled agile methods by book. They focused on scaled agile learning through workshops and training programs. Though the scaled agile learning was established through workshops, the team didn’t follow all the practices enlisted as ideal practices for a scaled agile environment. The act of sense-making the practices is accomplished only after the need for such a practice is established. For instance, the practice on requirements analysis by getting inputs from the customer is considered as an essential practice in agile and scaled agile software development approaches (Highsmith & Cockburn, 2001). However the team directly didn’t involve in customer requirement analysis on a regular basis. But when there was a need to allow themselves to work in closer collaboration with the customers, the team dedicated a specific time slot to go through the inputs and feedbacks given by the customers in a local community forum, where customers chat about their products and its features. The practice therefore was named as “Community input analysis” and was regularly scheduled every other week for team discussion with all the involved stakeholders. But even before the adaptation of such a practice, the team involved in an exploration phase to experiment the benefits of such a practice scheduled on a regular basis. Only when the practices bring efficiency to the team, it is considered for further discussion for the exploitation. This is in line with the argumentation on Ambidexterity, where organizations involve in a balance between exploitative and exploratory nature of practices (Turner, Swart, & Maylor, 2013). It can also be noted from this characteristic, that the scaled agile teams evolve towards agility through applying ambidexterity behavior, in the sense it improves self-organizing behavior of the teams in responding to business changes, which is the core value of agile and scaled agile systems. It also helps the agile teams evolve towards maturity from agile learning towards ambidextrous behavior, in responsive or adaptive to complex situations.

Figure 2: Agile teams evolving towards maturity

5 Characteristics of scaled agile maturity model

The analysis of scaled agile and agile systems resulted in four characteristics which have to be taken into account while designing a scaled agile maturity model: Potential performance perspective, ambidextrous nature, and underlying notion on maturity and progressive outcomes. In the next section, each of the characteristics is described in detail.

5.1 Potential performance perspective

In the first scan of the agile maturity models available in literature, quite a large number of model take life cycle perspective into account in defining the stages of the maturity model. Life cycle perspective helps to calculate the organizational maturity through stages of improvement. Only when the goals are accomplished for a particular stage, the preceding stage is unlocked. This in term is contrast to the agile environment which places
people over processes, and focuses on adaptive environment instead of a predefined system. Therefore, taking potential performance perspective into account which defines the maturity stage as a development path with focus on the potential improvements or outcomes can help in supporting agility factor throughout the model. In addition, this perspective gives the choice of final decision to the user, stressing on the fact that there can't be a unique solution to all kinds of challenges faced by the organization.

5.2 Ambidextrous nature
As discussed, agile and scaled agile environments are complex adaptive systems which adapts to changes introduced in the environment instead of predefined processes at the start. This is one of the essential features, which differentiates agile software development approaches to traditional software development methodologies. In line with this argumentation, if the agile and scaled agile teams aim at being mature in this perspective of adapting to external changes through a maturity model, the model should explicitly consider exploration and exploitation of practices into account in adapting to the system. It is in our view there is an essential need to shift the focus from prescribing predefined processes in defining to maturity to ambidextrous nature of taking a broader subjective view, in which people play an essential role instead of predefined processes.

5.3 Underlying notion on maturity
As seen from the literature review, maturity models lack in defining the concept of agile or scaled agile maturity in designing the model. We discern three characteristics that have been used as a rationale in describing “What makes the organizational capabilities mature?”

5.3.1 Adherence to a process
A number of agile maturity models in literature base the maturity levels on a defined process improvement approach in which the maturity is defined as the degree to which the process is institutionalized and effective (Maier, Moultrie, & Clarkson, 2012). Thus the maturity models encourage the use of existing well-known methods and practices to progress along the maturity scale.

5.3.2 Emphasis on people
Few of the other maturity models conceive people as an essential rationale for evolution towards agile methodologies. The maturity grids focus on evolution of cooperative and self-organizing behavior of agile teams, with maturity focusing on the intervention in the social behavior of agile teams in contrast to the structural process adherence as discussed earlier.

5.3.3 Emphasis on project
Since agile software development focuses more on the project management methodologies, maturity models involved the project management methodologies in the design. This was also reflected in the assumption of maturity, which stressed on process transparency and operational efficiency of the practices.

The aforementioned analysis shows how the notion on maturity has been conceptualized in different maturity models. We claim in using the characteristic on process maturity in scaled agile software development, where standardized scaled agile practices play an essential role in evolving towards scaled agile environment. However, we claim that the practices should be involved in an ambidextrous nature where exploration and exploitation are balanced out.

6 Conclusion
This study started with the research question: What characteristics should a maturity model possess to support the agile teams towards maturing in scaled agile environment? From our analysis three characteristics were identified that help in supporting the agile teams in maturing towards a scaled agile environment. Taking the desired
characteristics into account, we now propose the vision of the maturity model by translating the characteristics into requirements for the model. As a first stance, we propose process maturity as the underlying notion on maturity taking the scaled agile practices into account. As the scaled agile environment takes three levels of agility into account, the stages enlisted in the model should focus on the team, portfolio and program level on the whole. Taking life cycle perspective into account, which focusses on potential outcomes of the practices, we stress on including assessment criteria for the scaled agile practices as the function of potential outcomes. This might act as a discussion tool in exploring and exploitation of scaled agile practices as a team, in increasing the ambidextrous nature. In this study, we added the perspective of developing the ambidextrous ability of the teams to mature in scaled agile software development, which is a challenge, given that there is not such a unique recipe for such ambidexterity for every unique problems faced by the organizations. Our proposal is thus fostering alignment (exploitation) with prospective outcomes but leaving space for the emergence of variety (exploration) by not prescribing the practices the team should implement, however giving an idea on how the practices can be implemented in stages. Maturity model therefore will act as a guide to the team, in getting to know about the scaled agile practices and its potential outcomes better. In this way, the model supports the discussion for the agile team in exploring the practices. We also claim by this research that the notion on maturity model focusing on establishing specific process areas should change in which teams should focus on adapting practices which remain supportive to their working environment rather than predefined processes and practices established as stages.

Limitation of the study is that the conclusions are based on participatory observation and literature review. Although the researchers in qualitative analysis cannot pursue statistical generalization, our conclusions are specific to the contexts and agile teams included in the study. Considering the generalizability limits, we have proposed the desired characteristics for scaled agile maturity. The main implication of this study is inclusion of ambidexterity or subjective assessment of the scaled agile teams and favors such an environment through maturity model acting as a discussion tool. In comparison to the other maturity models, we claim to have introduced the desired characteristics in a novel approach which allows the emergence of context-specific practices instead of predefined processes. Though we have claimed to have done a qualitative analysis, we further stress on the future research questions that could be addressed:

- Which are the assessment approaches that can help the team in increasing the ambidextrous nature of the practices?
- With the characteristics of the scaled agile maturity model at hand, future research should consider analyzing more cases to replicate and update the findings.
- Given that, there is not a unique recipe for ambidexterity how can it be investigated further in the field of scaled agile environment?

References


110


