Requirements Prioritization in Scaled Agile Distributed Software Development

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Supervisors

Assoc. Prof. Tony Clear
Dr. Ramesh Lal

2022

By

Priyanka Antil
School of Engineering, Computer and Mathematical Sciences

Abstract

Context: The adoption of Agile methods has altered how Requirements Engineering (RE) is understood in development. Agile RE, for example, is a continuous and incremental process. There is a substantial amount of work being conducted with regards to Agile RE (Dikert et al., 2016; Inayat et al., 2015). The adoption of Agile RE practices in large-distributed software development is a recent trend. However, study on RE in Scaled Agile Distributed Software Development (SADSD) is growing, but little is known about requirements prioritization in SADSD, a decision-making process which plays a significant role in the successful development of software products. The motivation and gap emerged via Multi-vocal Literature Review (MLR) study (discussed in Chapter 3), that was conducted to understand the current state of art of RE in SADSD.

Aim: The overall aim of this research study was to provide an in-depth understanding of how requirements prioritization may be done in scaled agile distributed software development.

Research Methodology: To attain the aim of this research study, qualitative interpretive exploratory case study research methodology was adopted. Two large-distributed organizations (one from New Zealand, another was from Australia), employing Agile practices at the enterprise level, were studied. A corpus of data was collected via semi-structured interviews with practitioners (who were dispersed by location, function, and hierarchy), recordings of key decision-making events, requirements artefacts. An evolving strategy for the analysis of case study data was undertaken, primarily grounded within the data (i.e., grounded theoretic analysis) and complemented via deductive analysis. Thus, rich in-depth understanding of phenomena of interest is achieved.

Findings: The obtained results show that requirements prioritization occurs at various levels in scaled agile distributed software development. The findings indicated that, there are typically three key decision-making levels where prioritization of requirements is performed. These typically occurred in the form of inter-iteration and intra-iteration prioritization:

(i) portfolio level where business goals that connect with an organization's overall business strategy are decided and prioritized as well as ***High-level Requirements (HLRs)*** that are needed to achieve them are formally signed-off and prioritized.

(ii) domain level, two-fold role where in the first part, initial evaluation of HLRs is performed and submitted to portfolio level for formal decision-making. On the other hand, in the second part, ***Intermediate-level Requirements (ILRs)*** are decomposed from HLRs and priorities across teams are decided to implement them via short development cycle (e.g., 2 weeks).

(iii) team level where ***Low-level Requirements*** are decomposed from ILRs wherever needed, and priorities are decided for the individual delivery teams. At all these decision-making levels, required prioritization activities occurred that are not clearly separated. Instead, activities are highly integrated and occurred with varying details, for example levels of requirements abstraction, decision-making events, roles and responsibilities of decision-makers.

Another key finding of this research study is boundary spanning mechanisms that typically include *boundary spanning event(s), boundary spanning requirement artefact(s), boundary spanner role(s),* which primarily occurred when requirement prioritization decisions, are made at inter-iteration (i.e., portfolio level, second part of domain level). However, boundary spanning mechanisms partially occurred during intra-iteration prioritization as well, that typically yields knowledge acquisition, negotiation, consensus building, and conflict resolution, which are the key activities typically needed while making decisions on priority requirements.

Conclusions: This research study provides implications for both practice and theory by providing an in-depth understanding of the requirements prioritization process in SADSD. As an outcome, a conceptual framework has been developed that has the potential to be applicable in the contexts similar to the organizations that were studied in this research study. Recommendations for the researchers have been given to replicate and/or enhance the findings of this research study by conducting further research in a similar context and/or different context.

Table of Contents

Abstract i
List of Figures viii
List of Tables ix
List of Appendices xi
Attestation of Authorship xii
Acknowledgements xiv
Ethics Approval xv
Authors' Contribution xvi
1 Introduction 1
1.1 Motivation 1
1.2 Research Aim and Objectives 2
1.3 Research Questions 2
1.4 Research design 3
1.5 Contributions and implications for practitioners and researchers 3
1.6 Thesis outline 4
2 Foundations 6
2.1 Requirements engineering in software development 6
2.2 Requirements engineering and Agile software development 7
2.3 Large-scale Agile software development and Requirements Engineering 9
2.3.1 Large-scale Agile definition 9
2.3.2 RE in Large-scale Agile Software Development 9
2.4 Motivation to conduct secondary study on RE in SADSD 13
3 Chapter – Preliminary study to perform gap analysis 16
3.1 Study design and implementation: 16
3.1.1 Study design 16
3.1.2 Implementation 18
3.2 Summary of Findings 19
3.2.1 Over-scoping of requirements across scaling agile levels 19
3.2.2 Negligence of QRs in the early stages of requirements 20
3.2.3 Difficulty in prioritizing the requirements across scaling agile levels 21
3.3 Gap analysis and motivation 23
3.4 Research questions 23
3.5 Summary 24
4 Research design of the study 25
4.1 Research paradigm 25
4.1.1 Positivist paradigm 25
4.1.2 Interpretive paradigm 26
4.1.3 Critical paradigm 26
4.1.4 Interpretive research paradigm adopted for this research study 27

4.2 Qualitative Research Methodology 28
4.2.1 Exploratory case study method 28
4.2.2 Alternative Methods 30
4.2.3 Case study selection and context 30
4.3 Research Methods 32
4.3.1 Research questions 32
4.3.2 Ethical considerations 33
4.3.3 Data collection and organization 34
4.4 Data analysis 42
4.4.1 Pre-processing 42
4.4.2 Within the case analysis: Grounded theoretical analysis 43
4.4.3 Deductive analysis 50
4.4.4 Cross-case analysis 54
4.5 Threats to validity 54
4.6 Summary 57
5 Analysis and findings- MEL organization 58
5.1 Case overview 58
5.2 Overview of decision-making levels and associated requirements prioritization activities 60
5.2.1 Requirements discovery and understanding 60
5.2.2 Requirements organization and classification 60
5.2.3 Requirements prioritization and negotiation 65
5.2.4 Requirements artefacts 65
5.3 Portfolio level 66
5.3.1 Requirements discovery and understanding 67
5.3.2 Requirements organization and classification 68
5.3.3 Requirements prioritization and negotiation 69
5.3.4 Requirements artefact 71
5.4 First part of Segment level 73
5.4.1 Requirements discovery and understanding 74
5.4.2 Requirements organization and classification 79
5.4.3 Requirements prioritization and negotiation 79
5.4.4 Requirements artefacts 86
5.5 2nd part of Segment level 88
5.5.1 Requirements discovery and understanding 89
5.5.2 Requirements organization and classification 89
5.5.3 Requirements prioritization and negotiation 90
5.5.4 Requirements artefacts 93
5.6 Team level 94
5.6.1 Requirements discovery and understanding 96
5.6.2 Requirements classification and organization 98
5.6.3 Requirements prioritization and negotiation 99
5.6.4 Requirements artefacts 102

5.7 Incorporate distributed members of decision-making team 103
5.7.1 Mode of communication at the portfolio level:..... 103
5.7.2 Mode of communication at the Segment level..... 104
5.7.3 Mode of communication at the team level..... 104
5.7.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection) 104
5.7.5 Collaborative technologies..... 105
5.8 Challenges and overcoming strategies 106
5.8.1 Challenges specifically related to global distribution 106
5.8.2 Challenges regardless of global distribution..... 108
5.9 Summary 114
6 Analysis and Findings- AKL organization 115
6.1 Case overview 115
6.2 Overview of decision-making levels and associated requirements prioritization activities 116
6.2.1 Requirements discovery and understanding..... 116
6.2.2 Requirements organization and classification..... 117
6.2.3 Requirements prioritization and negotiation..... 121
6.2.4 Requirement artefacts..... 122
6.3 Portfolio level 122
6.3.1 Requirements discovery and understanding..... 123
6.3.2 Requirements organization and classification..... 124
6.3.3 Requirements prioritization and negotiation..... 125
6.3.4 Requirements artefacts 127
6.4 First Part of tribe level..... 129
6.4.1 Requirements discovery and understanding..... 130
6.4.2 Requirements organization and classification..... 134
6.4.3 Requirements prioritization and negotiation..... 134
6.4.4 Requirements artefacts 140
6.5 Second part of tribe level..... 141
6.5.1 Requirements discovery and understanding..... 142
6.5.2 Requirements classification and organization..... 142
6.5.3 Requirements prioritization and negotiation..... 143
6.5.4 Requirements artefacts 145
6.6 Team level 146
6.6.1 Requirements discovery and understanding..... 147
6.6.2 Requirements classification and organization..... 149
6.6.3 Requirements prioritization and negotiation..... 150
6.6.4 Requirements artefacts 152
6.7 Incorporate distributed members of decision-making team 153
6.7.1 Mode of communication at the portfolio level..... 153
6.7.2 Mode of communication at the tribe level 154
6.7.3 Mode of communication at the team level..... 154

6.7.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection) 154
6.7.5 Collaborative technologies 155
6.8 Challenges and potential strategies 156
6.8.1 Tension between product and engineering 156
6.8.2 Accumulating technical debt 156
6.8.3 Quality of requirements 157
6.8.4 Bias during decision-making 157
6.8.5 Difficult to measure business value 158
6.8.6 Dependencies across product 158
6.8.7 Dealing with quality requirements (QRs) 160
6.8.8 Difficulty in estimating effort 160
6.8.9 Inadequate requirements analysis 160
6.9 Summary 161
7 Cross-case analysis 162
7.1 Context comparison of the studied case organizations 163
7.2 Requirements discovery and understanding across scaling agile decision-making levels 164
7.2.1 Sources of requirements at the portfolio level 165
7.2.2 Sources of requirements at the domain level 165
7.2.3 Sources of requirements at the team level 168
7.3 Requirements classification and organization across scaling agile decision-making levels 168
7.4 Requirements prioritization and negotiation across scaling agile decision-making levels 173
7.4.1 Portfolio level 174
7.4.2 First part of domain level 176
7.4.3 Second part of domain level 180
7.4.4 Team level 184
7.5 Requirements artefacts across scaling agile decision-making levels 188
7.5.1 Requirements artefacts at the portfolio level 188
7.5.2 Requirements artefacts at the first part of domain level 189
7.5.3 Requirements artefacts at the second part of domain level 189
7.5.4 Requirements artefacts at team level 190
7.6 Incorporate distributed members of decision-making teams 190
7.6.1 Mode of communication at the portfolio level 190
7.6.2 Mode of communication at the domain level 191
7.6.3 Mode of communication at the team level 191
7.6.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection) 191
7.6.5 Collaborative technologies 191
7.7 Challenges and potential strategies 192
7.8 Salient categories emerged as an outcome of cross-case analysis 194
7.9 Summary 196

8 Discussion 197
8.1 Conceptual Framework overview 198
8.2 Portfolio level 202
8.2.1 Requirements discovery and understanding 202
8.2.2 Decision making team 203
8.2.3 Boundary spanning requirement artefact 204
8.2.4 Boundary spanning events 204
8.2.5 Decision making practices (decision-making factors and techniques) 205
8.3 Domain level 207
8.3.1 Intra-iteration prioritization: first part of Domain level 208
8.3.2 Inter-iteration prioritization: second part of domain level 214
8.4 Intra-iteration prioritization: Team level 220
8.4.1 Requirements discovery and understanding 220
8.4.2 Decision making team 221
8.4.3 Decision-making events 222
8.4.4 Requirements artefact 223
8.4.5 Decision making practices (decision-making factors and techniques) 223
8.5 Collaborative technologies and their role in requirements prioritization activities 225
8.6 Challenges and strategies 232
8.6.1 Challenges specifically related to distributed members of decision-making team 232
8.6.2 Challenges occurred regardless of distribution 233
8.7 Summary 239
9 Conclusion 240
9.1 Summary 240
9.2 Main findings related to research questions: 240
9.2.1 Findings as an outcome of RQ1 241
9.2.2 Findings as an outcome of RQ2 and RQ3 245
9.3 Research contributions 247
9.4 Limitations 248
9.5 Future directions 250
9.6 Final statement 251
References 252
Glossary 268
Appendices 272

List of Figures

Figure 2.1 Agile values and principles, Source: (Beck et al., 2001) 7
Figure 3.1 Legend for the MLR process 17
Figure 3.2 MLR process (Garousi et al., 2019; Kitchenham & Charters, 2007) 17
Figure 3.3 WL screening process 18
Figure 3.4 GL screening process 19
Figure 4.1 Research design (Creswell, 2009) 28
Figure 4.2 Case study designs (Yin, 2014) 32
Figure 4.3 A sample of the emerging concept 'External Sources' 48
Figure 4.4 A sample of the emerging concept 'Internal Sources' 48
Figure 4.5 A sample of emerging concept 'Sources of requirements' 49
Figure 4.6 Legend used in the conceptual framework 52
Figure 5.1 Requirements hierarchy model and scaling agile decision-making levels 61
Figure 5.2 Requirements classification 64
Figure 5.3 Snapshot of activities discussed at the portfolio level 66
Figure 5.4 Snapshot of activities discussed at the 1st part of segment level 74
Figure 5.5 Feature decision making stages 80
Figure 5.6 Snapshot of activities discussed at the 2nd part of segment level 89
Figure 5.7 Snapshot of activities discussed at the team level 96
Figure 6.1 Requirements hierarchy model and scaling agile decision-making levels 117
Figure 6.2 Snapshot of activities discussed at the portfolio level 123
Figure 6.3 Snapshot of activities discussed at the first part of tribe level 130
Figure 6.4 Epic decision-making stages 134
Figure 6.5 Snapshot of activities discussed at the second part of tribe level 142
Figure 6.6 Snapshot of activities discussed at the team level 147
Figure 7.1 Decision-making stages in the 1st part of domain level 177
Figure 8.1 Conceptual framework of requirements prioritization in SADSD 201
Figure 9.1 Conceptual framework of requirements prioritization in SADSD 242

List of Tables

Table 3.1 RE Challenges 21
Table 3.2 RE Strategies 22
Table 4.1 AKL interview details 38
Table 4.2 AKL participants' demographic details 39
Table 4.3 MEL participants' demographic details 40
Table 4.4 MEL interview details 41
Table 4.5 Sample of data organization via Spreadsheet 43
Table 4.6 Sample list of small portions of coded data from MEL_P5 45
Table 4.7 Sample list of small portions of coded data from MEL_P3 46
Table 4.8 Sample list of small portions of coded data from MEL_P1 46
Table 4.9 An example of aggregating codes via constant comparison method 47
Table 4.10 Memo for the category 'Sources of requirements' 50
Table 4.11 Initial conceptual framework 53
Table 5.1 Context description of MEL case organization 59
Table 5.2 Sample of strategic theme from MEL organization 72
Table 6.1 Context description of AKL case organization (Petersen & Wohlin, 2009) 116
Table 6.2 Sample of strategic theme from AKL organization 128
Table 7.1 Context comparison of studied case organizations, Adapted from (Petersen & Wohlin, 2009) 164
Table 7.2 Techniques to collate potential requirements from internal sources at the domain level 166
Table 7.3 Techniques to collate potential requirements from external sources at the domain level 167
Table 7.4 Techniques to collate potential requirements from internal sources at the team level 168
Table 7.5 Requirements classification across scaling agile decision-making levels 171
Table 7.6 Requirements hierarchy model employed by the case organizations 172
Table 7.7 Decision making at the portfolio level in the studied case organizations 175
Table 7.9 Decision making practices employed by the studied case organizations at the second part of domain level 182
Table 7.10 Decision making practices employed by the studied case organizations at the team level 186
Table 7.11 Challenges faced by studied case organizations 192
Table 7.12 Potential strategies to overcome challenges emerged from the studied case organizations 193
Table 8.1 Process for decision-making at the portfolio level 206

Table 8.2: Process specific practices at the portfolio level 207
Table 8.3 Process for decision-making on HLRs at the domain level 212
Table 8.4 Process specific practices at the portfolio level 213
Table 8.5 Process for decision-making on ILRs at the domain level..... 217
Table 8.6: Process specific practices for ILRs 219
Table 8.7 Process specific practices at the team level 224
Table 8.8 Process for decision-making on ILRs at the team level..... 225
Table 8.9 Collaborative Technologies 228
Table 9.1 Challenges that occur during prioritization of requirements in SADSD and potential strategies..... 246

List of Appendices

Appendix A: Permission to Access Sheet (Employer) 272
Appendix B: Participant Information Sheet 279
Appendix C: Consent Form 285
Appendix D: Interview protocol 287
Appendix E: Ethics Approval letter 291
Appendix F: High-level overview of research study for the interviewees 293
Appendix G: List of white literature (WL) 294
Appendix H: List of grey literature (GL) 300
Appendix I: MLR study 302

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

_____ 22/12/2022

Signature Date

This thesis is dedicated to:

The loving memory of my father, Sh. Randhir Singh who was my role model.

Gone but not forgotten!

Acknowledgements

First of all, I would like to express my deepest appreciation to my supervision team Associate Professor Tony Clear for his insightful discussions, guidance, and continuous support throughout my Ph.D. I'm deeply indebted to my secondary supervisor Dr. Ramesh Lal for his advice and guidance as well as making all possible resources available to support this research study.

In addition to my supervisor team, I would like to extend my deepest gratitude to Jim Buchan who expertly guided and supported me throughout this research journey. I would also like to sincerely thank Prof Stephen MacDonell for his support and feedback throughout this research study.

I am thankful to the Graduate Research School for conducting useful workshops, trainings and conferences throughout the research period which helped building strong research and interpersonal skills. I would also like to thank my colleagues from the University for advising and encouraging me throughout the research journey.

I must acknowledge the contribution of the participants of case organizations who were interviewed and provided me all possible resources to achieve the objective of this research study. Without their precious support, cooperation and time it would not have been possible to conduct this research.

I am sincerely grateful to my mother for her infinite sustenance and love, inspiration and prayers. I would like to say a great thanks to my husband, my son who supported and encouraged me throughout the research degree.

Ethics Approval

This research has been approved by the Auckland University of Technology Ethics Committee (AUTEC) on 16 March 2020 AUTEC Reference Number 19/448 “Requirements prioritization in global scaled agile software development”.

Authors' Contribution

The publications listed below has a primary author contribution of 80%, a secondary author contribution of 10%, and a third author contribution of 10%.

Publication 1: Antil, P., Clear, T., & Lal, R. (2021). Requirements Engineering in Global Scaled Agile Software Development: A Multi-Vocal Literature Review [Submitted for Review]. Journal of Systems and Software, TBD.

The multi-vocal literature review (MLR) study was planned by the authors of the publication in collaboration. The primary author wrote the first draft of the article, which was later improved with the help of the second and third authors' suggestions.

Revising the MLR after reviewer feedback, was put on hold as the thesis took priority, but it did help with developing and focusing my thinking.

Publication 2: Antil, P. (2020). Requirements engineering in global scaled agile software development environment: a multi-vocal literature review protocol. arXiv preprint arXiv:2004.12647

The MLR study protocol was planned by the authors of the publication in collaboration. The article's first draft was written by the primary author, and it was later enhanced with the help of the second and third authors' recommendations.

1 Introduction

This chapter introduces the phenomenon of interest (i.e., requirements prioritization in scaled agile distributed software development), which is investigated via this research study. In Section 1.1, the motivation behind conducting this research study is discussed. The research aim and research questions are presented in Section 1.2 and Section 1.3 respectively. The research design overview is provided in Section 1.4. In Section 1.5, contributions are discussed. An outline of thesis structure is provided in Section 1.6.

1.1 Motivation

The motivation for this research study is grounded in the current trends for developing software, indicating the need to scale development processes to suit global market needs, with distributed and Agile development being more prevalent (Lal & Clear, 2018). Although Agile methods are initially employed by the small organizations for the single team project(s), their proven benefits, that include reducing time to market and cost, enhancing the software quality, and managing the changing requirements, made them attractive in different contexts (Digital.ai, 2021; VersionOne, 2017). Thus, large organizations (Amazon, IBM, Google etc. (Dybå & Dingsøyr, 2008)) are increasingly adopting Agile practices for scaled projects involving distributing teams despite the fact that Agile methods are challenging to adapt at scale (Dingsøyr & Moe, 2014; Moe & Dingsøyr, 2017). This transition requires changes in the practices for Requirements Engineering (RE) and other software development processes.

RE, which is a branch of Software Engineering (SE), deals with discovering, specifying, analysing, and documenting the requirements of a system. The RE process in software development organizations employing Agile practices, has now become a continuous and incremental process instead of a closed phase implemented in the beginning, such as in traditional methods (e.g., Waterfall) and typically spans a whole development cycle. The current Agile RE practices (e.g., direct face to face communication, and user stories), are typically project based only. The RE in Agile Software development (ASD) community, requires enterprise wide RE practices to scale up ASD (Heikkilä et al., 2017), to suit global market trends (detailed explanation of Agile RE and their need to scale at the enterprise level is provided in Chapter 2 and Chapter 3). The Multi-vocal Literature (MLR) to understand the RE practices in Scaled Agile Distributed Software Development (SADSD) revealed that empirical studies on RE in SADSD are growing, but little is

known about requirements prioritization in SADSD. An MLR study (Appendix I) revealed a single empirical study conducted by Daneva et al. (2013), but the study's main focus was reprioritization of requirements in projects. There appears to be a lack of empirical investigation on the life cycle of prioritization of requirements (i.e., from an idea to operational) in SADSD. Another motivational factor was to conduct research on requirements prioritization in SADSD, which was my interest and included previous research experience in RE (master's research in Agile RE practices). This studied how RE was impacted by the relatively novel phenomenon of adoption of Agile methods in scaled distributed software development. The in-depth explanation of the motivation behind conducting this research study, is discussed further in Chapter 2 and Chapter 3.

1.2 Research Aim and Objectives

Taking the above motivation and limited prior work into consideration, the aim of this research study was to provide an in-depth understanding of requirements prioritization process (from an idea to operation), in SADSD.

The following were the research objectives to attain the aim of this research study:

- To investigate artefacts, processes, practices, and roles that are needed to perform requirements prioritization in SADSD.
- To propose a conceptual framework that could guide practitioners who are looking to scale Agile practices to perform prioritization of requirements in large scale distributed software development and further contributes to the Agile RE literature.
- To investigate the challenges faced by the practitioners while making decisions on the priority of requirements in SADSD.
- To investigate the potential strategies which could surmount the challenges occurred during prioritization of requirements in SADSD.

1.3 Research Questions

The following are the research questions (RQs) that are set to achieve the aim and objectives of this research study.

RQ1: How is prioritization of requirements done (i.e., from an idea to operation) in scaled agile distributed software development?

RQ1.1: Who are the decision makers and what are they responsible for?

RQ1.2: What prioritization criteria is employed to make decisions on the priority of requirements?

RQ1.3: What are the practices and artefacts employed for decision making on the priority of requirements?

RQ2: What are the challenges related to the requirements prioritization processes?

RQ3: What are the strategies to mitigate the impact of reported challenges?

1.4 Research design

This research study employed an interpretive research paradigm which is a good fit as suggested by Fitzgerald and Howcroft (1998), when the knowledge or reality is socially constructed (e.g., personnel involved in decision-making on the priority of requirements).

A multiple exploratory case study approach was undertaken to investigate the phenomena of interest that further helps in enhancing the generalizability, robustness, and applicability the findings of this research study (Yin, 2003). Data was collected primarily via semi-structured interviews which was supported via other sources (e.g., audio/video recording of decision-making events, requirements artefacts), as well to get multiple insights and strengthen the findings that emerged from the research (Lethbridge et al., 2005).

Data analysis was an iterative process primarily analysed via employing a grounded theoretic analysis technique (i.e., inductive analysis), which was complemented via deductive analysis (literature support) and cross-case analysis to enhance the generalizability, robustness, and applicability of findings.

1.5 Contributions and implications for practitioners and researchers

This research study has made several contributions which are categorized according to the domains (substantive, conceptual, and methodological) as suggested by McGrath and Brinberg (1983). This research study was able to traverse two of the domains that are discussed below:

Substantive domain:

- This study has been added to the limited case study research in the area of requirements prioritization in SADSD.

- Challenges faced by the practitioners while making decisions on the priority of requirements in SADSD and the potential strategies to surmount them, are identified.
- This study investigated structure (i.e., functional set-up, and specific role) needed to perform requirements prioritization in SADSD.
- This study identified the decision-making events, decision-making practices, requirements artefacts, Collaborative Technologies (CTs), boundary spanning mechanisms (described in Chapter 8) that are typically needed to make decision on the priority of requirements in SADSD.
- This study has profiled fine-grained in-depth multi-case studies which explored the life cycle activities of priority of requirements (from an idea to operation) in SADSD.

Conceptual domain:

- This study provides an Agile conceptual framework for requirements prioritization. The conceptual framework captures the life cycle activities of prioritization of requirements (from an idea to operation), synthesized from the two scaled agile distributed organizations. The conceptual framework creates knowledge on the functional set-up and units, artefacts, processes, practices, techniques, and roles that are needed to make decisions on the priority of requirements in SADSD. Detailed explanations of conceptual framework is provided in Chapter 8. The conceptual framework with its current level of abstraction has the potential to be applicable in the context similar to the organizations that were studied in this research study.
- This research study is useful for the researchers who are looking to further explore the phenomenon of interest and test the developed conceptual framework in different contexts.

1.6 Thesis outline

Chapter 1 presents the motivation behind conducting this research study, research aim and objectives, contributions, an overview of research design, and thesis structure.

Chapter 2 provides the background in broader context to provide understanding of RE process in software development (i.e., RE in traditional software development methods and RE in Agile methods).

Chapter 3 provides a summary of preliminary investigation which was conducted via MLR study, to identify the gaps about RE process in SADSD which in turned into a foundation for further research.

Chapter 4 explains the research design that includes justification of interpretive research paradigm, explaining multiple case study research approach, multiple sources of data, procedure to analyse the data, and the approaches to tackle the threats to validity.

Chapter 5 and Chapter 6 describe the findings that emerged with-in the case analysis. Findings of MEL organization is presented in Chapter 5. On the hand, Chapter 6 describes the findings that emerged from the AKL organization.

Chapter 7 presents the cross-case analysis that involves identifying commonalties, contrasting, or aggregating the findings of studied case organizations, in order to improve the robustness, generalizability, and applicability of findings.

Chapter 8 presents the conceptual framework that depicts decision-making levels, timeframe, and scope of requirements at each of decision-making levels, roles involved in decision-making, requirements artefacts, decision-making practices, boundary spanning mechanisms, and CTs that are needed to make decisions on the priority of requirements in SADSD.

Chapter 9 concludes the research work that is reported in this research study and future research recommendations are provided.

2 Foundations

This chapter describes the RE process and Agile in large-scale software development. First, the RE process in software development, in which the traditional RE process and their key activities are discussed. After that, RE in Agile Software Development (ASD) is discussed while making a comparison with traditional methods. Agile in scaled software development is discussed thereafter followed by a discussion on RE in scaled ASD.

2.1 Requirements engineering in software development

Requirements engineering is a process of identifying, validating, and managing the system requirements. According to International Requirements Engineering Board (IREB), RE is defined as a systematic and disciplined approach (Glinz, 2017) with the following goals:

- identify, create shared understanding and manage requirements systematically
- understand stakeholders' needs
- minimize stakeholder requirements expectation of risks (Glinz, 2017)

There are various RE models that includes the incremental model (Kotonya & Sommerville, 1998), linear model (Macaulay, 1996), spiral model (Boehm, 1988). However, there is no universal RE process that can be made to fit for all organizations (Sommerville, 2005). Regardless of the RE process, the fundamental RE activities are: requirements elicitation and analysis, requirements validation and management, with requirements specification, requirements prioritization and negotiation supporting requirements elicitation and analysis by identifying the most valuable requirements (Aurum & Wohlin, 2005; Sommerville, 2005; Zowghi & Damian, 2003). *Requirements elicitation* deals with eliciting the potential requirements from all possible sources (e.g., stakeholders, operational environment, organizational environment) (Paetsch et al., 2003; Sommerville, 2005; Zowghi & Damian, 2003). *Requirements analysis* evaluates the consistency, completeness, necessity, and viability of requirements, thus yielding an understanding of requirements (Sommerville, 2005; Zowghi & Damian, 2003). *Requirement specification* deals with documenting the agreed requirements in such a way that can be understood by the delivery teams and stakeholders. *In requirements validation*, requirements are confirmed with stakeholders in order to get shared understanding and avoid problems that may occur. *Requirements management* manages

the unavoidable changes to the requirements (Paetsch et al., 2003; Sommerville, 2005; Zowghi & Damian, 2003).

2.2 Requirements engineering and Agile software development

Requirements are typically emerging in nature instead of being pre-specifiable (Boehm, 2006). Therefore, adaptive methods such as Agile methods have emerged, with the aim to address the need to define a new way to develop software in a context (e.g., changing requirements), where classic software development approaches such as Waterfall were no longer effective (Schwaber, 2004). As stated by Beck et al. (2001), ASD is a set of practices based on 12 principles (as shown in Figure 2.1) and four values depicted in the Agile manifesto. Unlike traditional methods driven via documentation and upfront design, Agile methods (e.g., Scrum, Extreme Programming), ought to be customer-driven, encourage team collaboration, and have short development cycles (e.g., one to four weeks) (Heikkilä et al., 2015).

Figure 2.1

Agile values and principles, Source: (Beck et al., 2001)

Agile values:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

Agile Principles:

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

RE in Agile, the principal difference between Agile RE and traditional software development RE is not whether to do RE, but when to do it. RE in traditional development (e.g., Waterfall) is typically performed at the beginning of development (Boehm, 2006; De Lucia & Qusef, 2010), whereas RE in Agile development (e.g., Scrum) is a continuous and incremental process but occurring in small sessions in comparison with traditional development (De Lucia & Qusef, 2010; Paetsch et al., 2003). Commonly identified Agile RE practices include:

- regular communication using either face to face communication or virtual conferences
- continuous prioritization of requirements
- active customer or customer representative involvement to ensure requirements are properly defined, refined and prioritized
- review meetings, typically to validate requirements
- user stories, typically for requirements specification
- cross-functional teams, a yield shared understanding of priority of requirements
- iterative requirements, gradual detailing of requirements
- managing requirements change (Inayat et al., 2015; Ramesh et al., 2010)

The adoption of Agile RE practices vary widely depending on the organization, application domain, people involved, and the Agile methods chosen for development (Paetsch et al., 2003). For instance, in Scrum with regards to RE, requirements are broken down into sub-requirements, to implement them via short development cycles such as , two weeks, and deliver business value immediately (Beck et al., 2001; Schwaber, 2004). Requirements are generally specified in the form of user stories, sprint review meeting to get feedback from the customers, sprint planning meeting for managing the requirements, backlog grooming to prioritize the requirements. The product owner is typically act as a customer representative who generally owns the responsibility of prioritization of requirements and formulating the product backlog (product backlog contains priority list of requirements) (Schwaber, 2004).

2.3 Large-scale Agile software development and Requirements Engineering

The fundamental assumptions suggest that Agile practices are best suited for small and collocated development teams (Boehm, 2006; Schwaber, 2004). Their proven benefits include reducing time to market and cost, enhancing the software quality, and managing the changing requirements that make them attractive in different contexts (Dikert et al., 2016; Dingsøyr et al., 2018; VersionOne, 2017). Thus, large organizations (Amazon, IBM, Google etc. (Dybå & Dingsøyr, 2008)) are increasingly applying Agile practices in scaled software development involving distributed teams (Dingsøyr et al., 2018; Dybå & Dingsøyr, 2008; Moe & Dingsøyr, 2017).

2.3.1 Large-scale Agile definition

In the field of applying Agile at large scale, a precise definition of "Large-scale Agile" is lacking in the literature (Batra & Dinesh, 2020; Conboy & Carroll, 2019; Dikert et al., 2016; Dingsøyr & Moe, 2014; Dumitriu et al., 2019; Kalenda et al., 2018). Participants of the XP workshops 2013-2016 on Large-scale Agile suggested that "scaling is about more than the size (number of people and teams), but also about scaling the organization and the distribution (number of sites)" (Moe & Dingsøyr, 2017). For example, Large-scale agile software development might involve having 50 or more people or at least six teams (Dikert et al., 2016) or more than three sites (Dingsøyr & Moe, 2014). Therefore, building on the definition of "Large-Scale Agile" proposed by Dikert et al. (2016), as *"software development organizations with 50 or more people or at least six teams",* and recognising the nature of case study sites investigated in this research study, I include the additional stipulation to incorporate a distributed focus, that *"these people or teams must work across sites located in at least two different locations"*. When the organization has people or teams distributed globally across country boundaries, it is termed as "Globally Distributed Large-scale Agile", and when the people or teams are nationally distributed, it is termed as "Distributed Large-scale Agile".

2.3.2 RE in Large-scale Agile Software Development

As discussed in Section 2.2, Agile's approach to handle requirements is fundamentally different, for example, continuous and iterative ways to perform RE activities. There is a substantial amount of work conducted in regards to Agile RE (Dikert et al., 2016; Inayat et al., 2015). The existing research on Agile RE explored the RE practices such as face to face communication, continuous prioritization, and user stories (discussed in Section 2.2) (Ramesh et al., 2010; Schön et al., 2017). Some researchers focused on identifying

challenges, that include minimal documentation, negligence of quality requirements of RE in agile development (Elghariani & Kama, 2016; Heikkilä et al., 2015; Inayat et al., 2015; Ramesh et al., 2010). Some researchers compared the RE process of Agile with a traditional RE process while discussing how Agile RE practices mitigated the traditional RE challenges (Inayat et al., 2015; Ramesh et al., 2010). In addition to this, some studies focused on how Agile RE can get benefit from traditional RE practices in order to manage Agile RE specific challenges (Medeiros et al., 2015; Zamudio et al., 2017).

The recent trends of adopting Agile practices in scaled distributed software development involving distributed teams requiring enterprise wide RE practices to scale up ASD. Various scaling agile frameworks have been developed by practitioners and consultants to support large organizations, to support enterprise-wide agility and scale agile practices (Alqudah & Razali, 2016; VersionOne, 2017). Alqudah and Razali (2016) reported 20 scaling agile frameworks, of which Disciplined Agile Delivery (DAD), Scrum @scale, Scaled Agile Framework (SAFe), were some of the popular frameworks at the time of writing this dissertation (Digital.ai, 2021). The empirical evaluation of these frameworks and the RE practices that are defined in these frameworks to scale up ASD, are weak (Dikert et al., 2016; Moe & Dingsøyr, 2017; Putta et al., 2018). Therefore, investigation needs to be undertaken to understand the applicability of these framework RE practices in real-life contexts. The studied case organizations of this research study adopted DAD and Scrum @ scale, therefore RE practices that are defined in these frameworks to scale up ASD are discussed in detail in the rest of this Section.

2.3.2.1 Disciplined Agile Delivery

DAD encompasses a set of roles, phases, and practices (Ambler & Lines, 2012). The full details of DAD are available on its official website https://www.pmi.org/disciplined-agile/process/introduction-to-dad, as well as in its official textbook "Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise" (Ambler & Lines, 2012), where authors of the framework, state that DAD assists organizations to adopt enterprise wide Agility via providing light weight guidelines. Regardless of the size of organization, DAD describes its framework as a toolkit that enables the development of an organization's specific scaling agile practices. DAD suggests approaches from already implementing practices and offers guidance on when and how to combine them. For example, Agile methods such as Scrum typically served as process bricks, while DAD framework provides the mortar to fit the bricks together (Ambler & Lines, 2012).

RE practices in DAD

Portfolio level: Portfolio management, which is central to DAD, is claimed to yield the ability to make first time right decisions, removing bias during decision making on the potential requirements (e.g., products, projects, or exploratory work). Portfolio management is typically derived from development and operations intelligence, including recommendations from the other functional units (e.g., enterprise architecture for technology perspectives, product management for business perspectives). As a result of this, portfolio management make first time right decisions in terms of resources and budget (Ambler & Lines, 2012).

Program level: DAD defines various roles that are specific to the program management and typically involved in the RE activities such as requirements elicitation, requirements prioritization, are performed at the program level.

- Chief Product Owner (CPO): within a program, CPO typically acts as a team lead for the Product Owner (PO) team, negotiating the functional dependencies with PO team and provides guidance to them wherever needed. CPO work closely with coordinator/program manager to deliver the solution produced and may play the role of PO in one or more delivery team(s) (Ambler & Lines, 2012).
- Chief Architecture Owner (CAO): within a program, CAO is typically a team lead for the Architecture Owner (AO) team, negotiating the technical dependencies with AO team and providing guidance to them wherever needed. CAO work closely with the enterprise architecture team and may play a role of AO in one or more delivery team(s) (Ambler & Lines, 2012).
- Program manager/Coordinator: responsible for coordinating activities within the program and works closely with CPO, CAO in order to guide the overall work of delivery team(s) (Ambler & Lines, 2012).

Project/team level: Potential requirements that are imposed on products, are typically delivered via multiple delivery life cycles: Agile (Scrum based that have '*workitem list'* instead of product backlog), lean (Kanban based), exploratory (Lean start-up), continuous delivery (Agile based, Lean based) (Ambler & Lines, 2012). DAD outlines a set of primary and secondary roles and their responsibilities, including technical and leadership roles, and it promotes the growth of "T-skilled engineers/T-shaped engineers" (Ambler & Lines, 2012). T-shaped engineers typically have great competencies in one area but also have knowledge of other areas as well (Ambler & Lines, 2012). A delivery team's

primary role includes product owner (representing customers' needs), architecture owner (represents architectural needs), team lead (facilitates communication and removes impediments if any), team members (typically engineers/developers who produce actual solutions), stakeholders (who consume the outcome of solution), these are all typically involved in making decisions on the requirements. The secondary role that includes specialist, subject matter expert, integrator, technical expert, independent tester (Ambler & Lines, 2012), may also be involved in the requirements engineering activities (e.g., requirements prioritization) wherever needed.

Inception phase: In DAD, an inception phase is typically held before starting the implementation of a new project or feature, that yields requirements envisioning, shared understanding of requirements among team members, identifies solution alternatives, risks identification and generally takes few days to few weeks (Ambler & Lines, 2012). In core Agile methods such as in Scrum and XP, Inception phase is typically called Sprint zero and planning game respectively.

Iteration planning: In DAD, typically in Agile based cycle (such as Scrum), requirements are split up into numerous time-boxed iterations (typically one to four weeks). Iteration (IR) planning is performed where requirements are decided and prioritized for the upcoming IR (Ambler & Lines, 2012).

Look ahead meeting: which is typically held before organizing the IR planning to identify requirement dependencies, shared understanding of requirements among delivery teams (Ambler & Lines, 2012).

2.3.2.2 Scrum @ Scale

Scrum @ scale is a lightweight scaling framework where a group of teams via employing scrum guide consistently produce products of the highest potential value (Sutherland, 2006). These products could be a complex integrated system, physical, and digital services. Scrum @ scale employs the idea of scaling up starting with a team and moving on to a team of teams, a network of teams, and so forth. The full details of Scrum @ scale are available on its official website https://www.scrumatscale.com/scrum-at-scale-guide-online/. The framework consists of two components: Product Owner cycle that constitutes "what" and the Scrum Master cycle that constitutes "how". As Scrum @ Scale framework is based on Scrum, the Scrum Master and PO roles are used and have the same competencies as those listed in the Scrum Guide (Sutherland, 2006).

With the teams of teams approach, the following are the new roles introduced, who typically are involved in the RE activities, requirements discovery, requirements prioritization.

- Chief Product Owner (CPO): CPO works with PO team to set the priorities of requirements, negotiates the functional dependencies and provides guidance to them wherever needed. CPO may play a role of PO in one or more delivery team(s). The "Executive Meta Scrum (EMS)" is a hub of PO cycle that represents PO responsibilities for the entire organization. The EMS team and CPO works collaboratively to decide the organizational vision and strategic priorities (Sutherland, 2006).
- Scrum of Scrums Master (SoSM): SoSM holds the similar responsibility as a Scrum Master (SM) but at scale. The SoSM may take the role SM in one or more delivery team(s). SoSM works closely with the CPO to produce potential consumable solutions at the end of each sprint, facilitate cross teams' coordination and communication of requirements, remove impediments if any. The "Executive Action Team" is a hub of SM cycle that represents that SM responsibilities for the entire organization. The Executive Action Team's role is to coordinate with multiple SoS (broader network) and also interface with any non-agile parts of the organisation. Like any Scrum team, "Executive Action Team" also have a PO and SM and a backlog (Sutherland, 2006).

2.4 Motivation to conduct secondary study on RE in SADSD

As discussed in Section 2.3, over the past few years, Agile in large-scale development where teams are distributed has received significant attention from the research community as well as from the practitioners, with many stating successful adoption although there are some challenges, for example difficulty in creating the priority list of requirements, managing the requirements. Systematic literature review (SLR) as well as mapping studies are also gaining attention among researchers to understand the current state of art related to Agile, in large-scale organizations.

The following secondary studies related to Agile in large organizations were found before commencing the secondary study to understand the RE practices in SADSD.

In 2016, Dikert et al. conducted an SLR study on the scaled Agile transformation. That study identified 35 challenges and 29 success factors classified into 9 and 11 categories,

respectively. The noticeable success factors were adapting the Agile model, mind-set and alignment, management support, and education and training. The common challenges identified, were change resistance, integrating non-development functions, Agile being difficult to implement, and RE challenges such as, refining the requirements, managing the requirements, difficulty in estimating the requirements.

Khalid et al. (2015) performed an SLR study on Agile scalability and adoptability. They described and identified the limitations of Agile practices for scaled projects. Documentation of requirements, time-period, human resources related problems, coordination and communication with multiple and distributed teams, were identified as limitations for scaling using Agile practices in projects.

Gustavsson (2017) investigated the literature on roles required for team coordination in the scaled agile software development. He discovered that additional roles are required for coordination. His investigation also found that only a few organizations have adopted the required coordination responsibility within the teams, based on the scaled agile frameworks (e.g., SAFe, LeSS). In addition, the organizations which had coordination roles, mainly focused on vertical coordination. This resulted in managers developing plans and accordingly allocating work by themselves. However, in a scaled agile setup the self- organizing principle ought to be a key practice whereby the entire team are responsible for planning and task allocation (horizontal coordination).

Razavi and Ahmad (2014) in their SLR study, concluded that an effective set of tools are needed for coordination and communication (e.g., communicating and coordinating the requirements with distributed teams) in distributed scaled agile development environment. They also claimed that the next severe challenge will be to produce quality products.

Alsaqaf et al. (2017) investigated the literature on non- functional requirements, focusing on – Quality Requirements (QRs) in scaled agile distributed projects. Alsaqaf et al. (2017) discovered 12 QRs related challenges which were mostly as a result of the adopted Agile practices. These QRs related challenges were mostly as a result of inability of the product owners to handle QRs. They also identified strategies to overcome the challenges relating to QRs at a proposal level, while pointing out more research is required in order to understand the QRs in SADSD.

Putta et al. (2018) conducted an MLR study on adopting scaled agile framework (SAFe), identifying benefits and challenges. The most common benefits include alignment, productivity, quality, collaboration, and time to market. Challenges that received most attention include adapting Agile practices, global distributed challenges, Agile release train challenges, inadequate requirements prioritization, requirements management challenges, change resistance, and staffing product owner challenges. In addition, the authors suggested that more primary research is required on transformation steps, implementation, and challenges of the SAFe framework (Putta et al., 2018).

To sum up, the existing secondary studies demonstrated that SADSD is a growing area of interest to researchers and practitioners, but little is known about RE in SADSD. Only one secondary study was found conducted by Alsaqaf et al. (2017). However, the main focus of the Alsaqaf et al. (2017) study was to identify the challenges relating to QRs in SADSD and strategies to mitigate them. Other RE activities such as requirements prioritization were missing. Furthermore, the Alsaqaf et al. (2017) study focused on the white literature (e.g., journal paper) studies only, despite scaled agile mostly being driven by the practitioners and consultants in the field. An MLR study must be conducted as a preliminary study in Software Engineering (SE) as practitioners produce grey literature to a large extent, as the most common way of sharing knowledge, advice, and experiences. Otherwise, the researcher could miss out crucial current information on the real-world phenomena of interest (Garousi et al., 2016). Hence, an MLR study was needed to provide information and understanding on RE practices in SADSD, as well as to identify research gaps that need addressing. This motivated me to undertake an MLR study on RE in SADSD.

Next Chapter 3 describes the current state of art related to the RE in SADSD which was understood via an MLR study.

3 Chapter – Preliminary study to perform gap analysis

The chapter presents a summary of the MLR study which was conducted on RE in scaled agile distributed software development. An MLR refers to the process to identify, analyse and interpret the phenomena of interest via assessing all possible literature. This includes formally published academic research (e.g., journals and conference papers) termed as White Literature (WL) as well as unpublished and practitioners' literature (e.g., white articles, methods' creator website) termed as Grey Literature (GL) (Garousi et al., 2019). The main purpose of the MLR study was to identify the gaps about the RE process in SADSD which turned into a foundation for further research. This was motivated by an interest in RE, as impacted by the relatively novel phenomenon of adoption of Agile methods in large-scale distributed software development (Dikert et al., 2016; Lal & Clear, 2017). The detailed MLR study is presented in Appendix I.

There are two research questions (RQs) that were set to achieve the objectives of the MLR study:

RQ 1: What are the challenges for requirements engineering in scaled agile distributed software development?

RQ 2: What are the strategies for surmounting challenges in requirements engineering activities in scaled agile distributed software development?

The summary of the study design and implementation process are presented in Section 3.1. The summary of findings is presented in Section 3.2. Gap analysis and motivation for further research is discussed in Section 3.3.

3.1 Study design and implementation:

This Section summarizes the study design and implementation that is employed to conduct MLR.

3.1.1 Study design

The Kitchenham and Charters (2007), and Garousi et al. (2019) guidelines were followed to conduct MLR study. A visual representation of the MLR process that consists of planning phase, pilot study, conducting the review, and reporting the review was adopted by adapting the Kitchenham & Charters (2007), and Garousi et al. (2019) guidelines as

shown in Figure 3.2. Legend that was used for a visual representation of MLR process is shown in Figure 3.1.

Figure 3.1

Legend for the MLR process

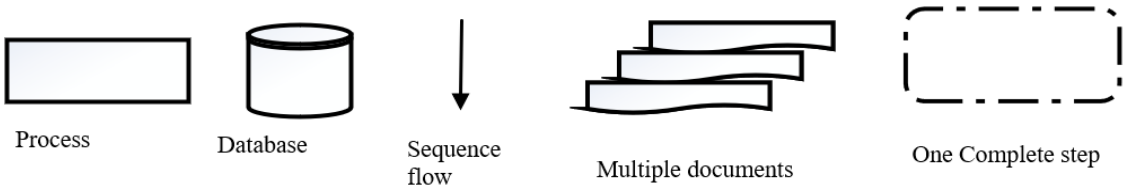
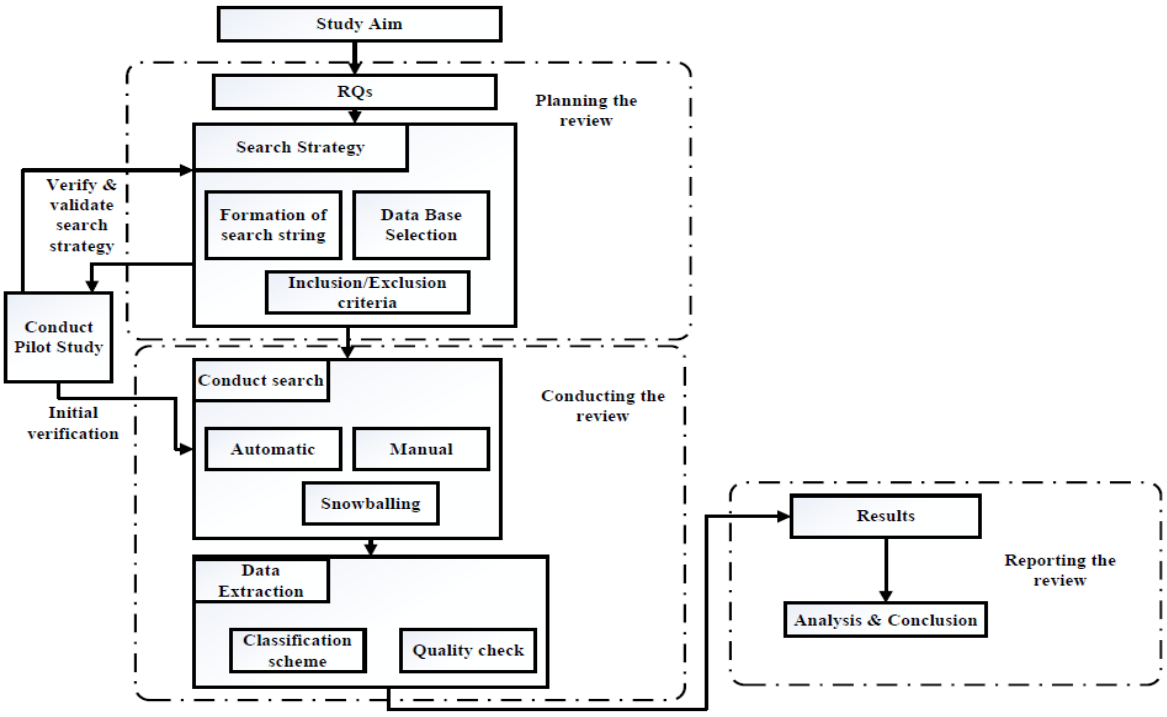


Figure 3.2

MLR process (Garousi et al., 2019; Kitchenham & Charters, 2007)

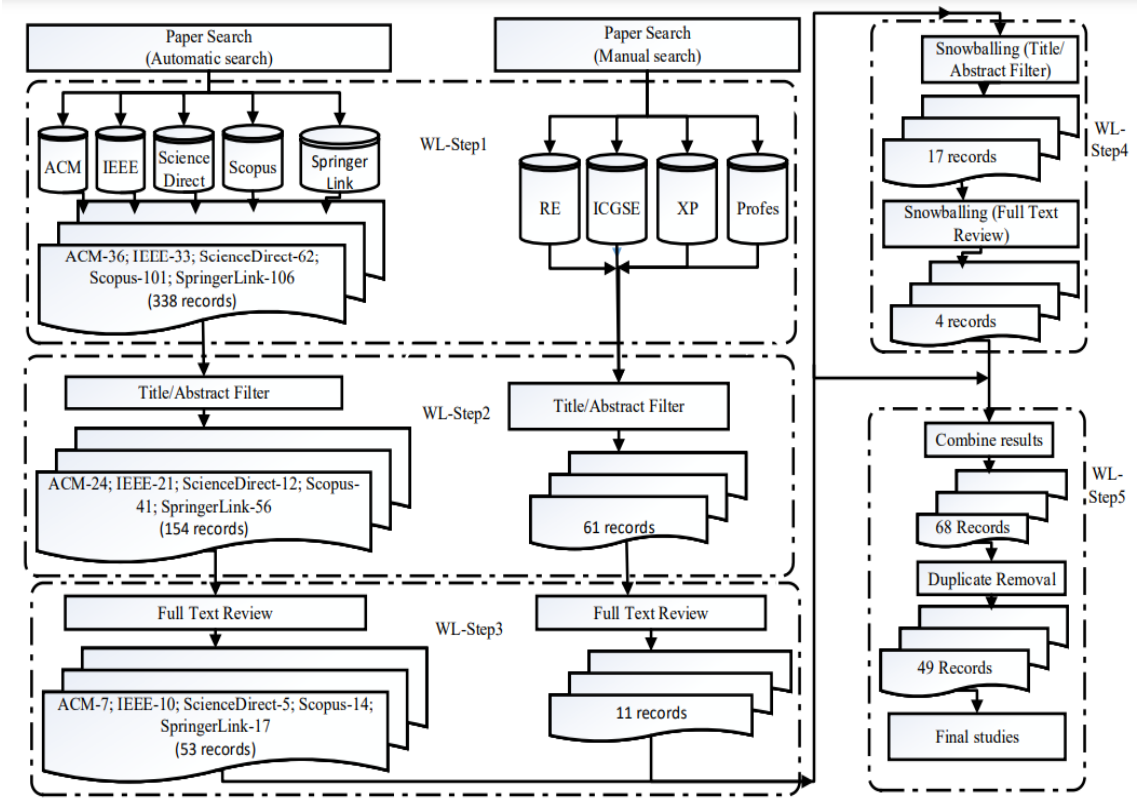


3.1.2 Implementation

In order to retrieve relevant primary studies from the WL and GL, an automatic and manual search were performed on digital databases, online web search, key conference proceedings, and methods' creator websites. The WL and GL search process is shown in Figure 3.3, and Figure 3.4 respectively.

Figure 3.3

WL screening process



The relevant WL studies were retrieved via automatic and manual search (as shown in Figure 3.3). Candidate WL studies were primarily filtered, based on the Title/Abstract which were further refined via full text review. Snowballing was also performed on studies that were retrieved via automatic and manual search to retrieve all relevant WL studies. As an outcome, WL studies selection ended up with 49 studies that contain seven studies from ACM, 10 from IEEE, five from ScienceDirect, six from Scopus, 17 from SpringerLink, and four from Snowballing.

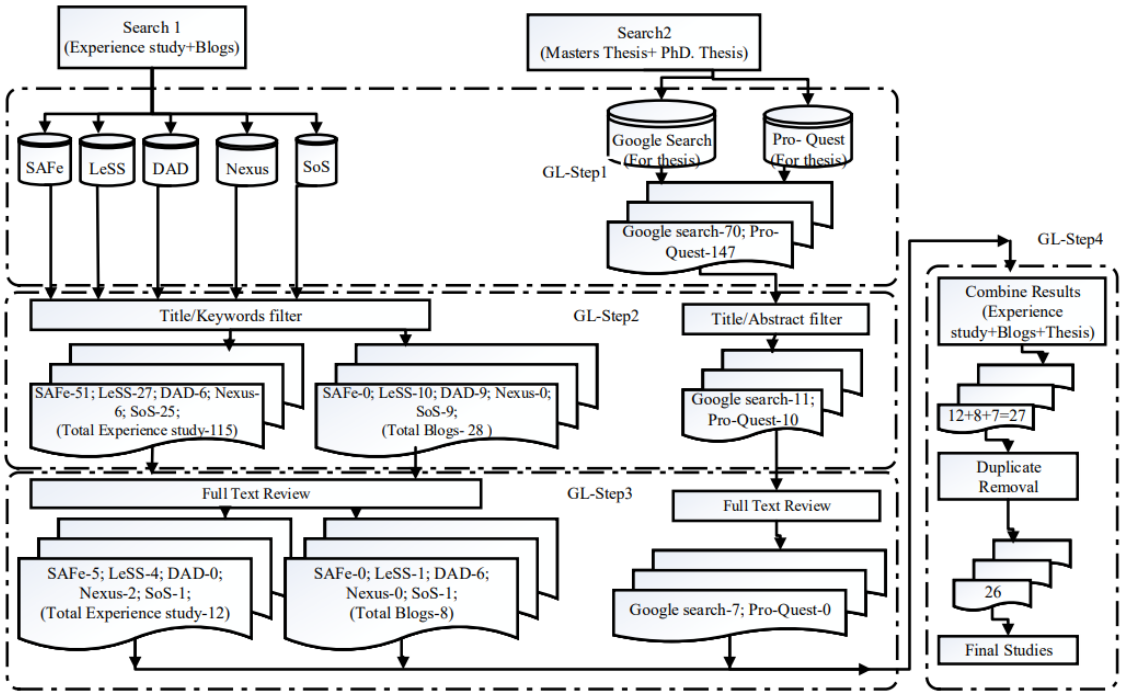
Relevant GL studies were also retrieved via automatic and manual search (as shown in Figure 3.4) which were primarily filtered based on Title/Keywords/Abstract and further refined via reading full text. A total of 26 GL primary studies were retrieved that

contained 12 experience studies, eight blogs from the methods' creator websites, and seven masters' thesis that were retrieved from the Google search.

After removing the duplicate studies from the candidate WL and GL, the search process yielded a total of 71 primary studies that contained 49 WL studies and 22 GL studies.

Figure 3.4

GL screening process



3.2 Summary of Findings

The analysis of the literature discovered that implementing Agile methods in large-scale distributed software development organizations, is posing various challenges related to RE. Identified RE challenges in SADSD were classified into four categories and seven categories of potential RE strategies were found, that can be employed to overcome those challenges as shown in Table 3.1, Table 3.2 respectively.

3.2.1 Over-scoping of requirements across scaling agile levels

In SADSD, requirements that are estimated and allocated to the engineering teams for implementation, are often beyond their capacity (Sekitoleko et al., 2014; Usman et al., 2018). Factors that emerged from the literature results show over-scoping of requirements, including complexity of requirements (Evbota et al., 2016; Usman et al., 2018), lack of expertise for developing achievable plans (Evbota et al., 2016), lack of awareness of team maturity (Badampudi et al., 2013; Tripathi et al., 2015), business

pressure (Heikkilä et al., 2017), requirements changes (Hannay & Benestad, 2010; Roman, 2016), inadequate breakdown of requirements (Bjarnason et al., 2011; Paasivaara et al., 2018).

Potential strategies:

- Provide management support (Usman et al., 2018; Wildt & Prikladnicki, 2010)
- Ensure training and education (Guyot, n.d.; Kim & Cho, 2018; Rautiainen et al., 2011)
- Appropriate engineering practices should be adopted (e.g., classify requirements, hierarchical breakdown of requirements) (Bass, 2016; Daneva et al., 2013)

3.2.2 Negligence of QRs in the early stages of requirements

Negligence of the QRs (e.g., usability, performance), which is a known challenge of basic Agile methods (e.g., Scrum, XP) (Heikkilä et al., 2017; Inayat et al., 2015; Ramesh et al., 2010) exist in SADSD as well. These should be considered in the early stages of requirements analysis, such as a requirements definition. According to MLR findings, QRs not determined during decision-making events, often were revealed in production (Alsaqaf et al., 2018). Common factors that result in negligence of QRs, include lack of clear conceptual definition of QRs (Alsaqaf et al., 2017), lack of stakeholders' viewpoints in QRs (Roopa et al., 2017; Van der Heijden et al., 2018), lack of QRs visibility (Alsaqaf et al., 2018), lack of appropriate QRs specification approaches (Alsaqaf et al., 2018), lack of collaborative technologies (Alsaqaf et al., 2018). As a consequence of this, discovering QRs during or after IR implementation, particularly in development and testing, makes it complex to implement them (Alsaqaf et al., 2018). Moreover, software architecture may become inappropriate due to lack of upfront analysis of QRs (Alsaqaf et al., 2017).

Potential strategies:

- governance adopted (e.g., cross-functional decision-making team) (Ambler, 2013; Daneva et al., 2013; Gat, 2006)
- appropriate engineering practices (e.g., establish QRs specialist team, capacity allotment for requirements) (Alsaqaf et al., 2017; Roopa et al., 2017)
- ensure training and education (Paasivaara et al., 2018; Shrivastava & Rathod, 2017)

Table 3.1

RE Challenges[1]

RE Challenges	WL studies	GL studies
Difficulties in coordinating and communicating the requirements across teams	WL5, WL7, WL11, WL17, WL18, WL24, WL26, WL28, WL34, WL35, WL36, WL38, WL48, WL49	GL1, GL12, GL5, GL13, GL15, GL16, GL20
Over-scoping of requirements across scaling agile levels	WL18, WL16, WL24, WL26, WL28, WL29, WL30, WL31, WL36, WL38, WL39, WL46, WL44, WL48, WL49	GL3, GL12, GL13, GL15, GL18, GL2
Negligence of QRs in the early stages of requirements	WL1, WL3, WL10, WL27, WL31, WL34, WL36, WL45, WL47, WL48, WL17, WL33, WL31, WL2, WL18, WL16, WL41, WL30	GL8, GL10, GL20, GL14, GL21, GL2, GL22
Difficulties in the prioritization of requirements across scaling agile levels	WL1, WL3, WL4, WL7, WL8, WL11, WL13, WL14, WL19, WL20, WL23, WL24, WL25, WL28, WL30, WL32, WL37, WL43, WL46, WL33, WL42, WL29, WL22, WL30, WL48	GL12, GL13, GL14, GL15, GL5, GL1, GL6, GL19

3.2.3 Difficulty in prioritizing the requirements across scaling agile levels

Scaled agile distributed software development has an inherent problem with prioritizing and organizing the requirements, due to various organizational levels and structures involved with product planning and development (Rolland, 2015). Factors that emerged from the literature posing challenges, included bias during decision-making on the priority of requirements (e.g., strategic requirements vs internal improvement related requirements) (Bass, 2015; Vlietland & Van Vliet, 2015), requirements dependencies due to inadequate decomposition of requirements, that is the greater the requirements dependencies, the lower the degree of freedom (Pavlichenko, 2018; Scheerer et al., 2015), volatile requirements, insufficiency of basic Agile practices (e.g., user story) to specify and/or represent requirements at scaling levels (Daneva et al., 2013; Reinikainen, n.d.). Finally, lack of shared understanding of requirements across scaling agile levels (Evbota et al., 2016; Paasivaara et al., 2018), and retention level of decision making (Tripathi et al., 2015).

Potential strategies:

- governance adopted (e.g., skill set for product owners, cross-functional decision-making team) (Ambler, 2013; Gat, 2006; Wildt & Prikladnicki, 2010)

[1] In MLR study, primary white literature (WL) studies were represented as WL1, WL2, ——, WL49 and grey literature (GL) studies were represented as GL1, GL2, —-, GL22. The list of WL and GL studies are presented in Appendix G, Appendix H.

- use of strategic decision-making strategies (e.g., debiasing judgement, taking an outside view) (Bass, 2015; Ktata & Lévesque, 2009)
- appropriate engineering practices (e.g., employ decision making criteria, hierarchical breakdown of requirements) (Ambler, 2014; Jha et al., 2016; Pavlichenko, 2018; Scheerer et al., 2015)
- adopt appropriate collaborative technologies (e.g., requirements management tool) (Kalenda et al., 2018; Schnitter & Mackert, 2010)
- encourage communication, collaboration, and transparency (Scheerer et al., 2015)
- provide management support (Paasivaara et al., 2018)
- ensure training and education (Evbota et al., 2016)

Table 3.2

RE Strategies[2]

RE Strategies	**WL studies**	**GL studies**
Encourage communication, collaboration, and transparency	[WL48] [WL38] [WL32]	[GL5] [GL12] [GL20]
Adopt appropriate collaborative technologies	[WL46] [WL22] [WL18] [WL41]	[GL1] [GL11] [GL15] [GL22]
Provide management support	[WL36] [WL44] [WL24]	
Ensure training and education	[WL12] [WL30] [WL48]	[GL3] [GL4] [GL19]
Appropriate engineering practices (e.g., classify requirements, hierarchical breakdown of requirements)	[WL21] [WL48] [WL44] [WL49]	[GL8] [GL9] [GL6]
Governance adopted (e.g., cross-functional decision-making team)	[WL44] [WL39] [WL49] [WL24]	[GL7] [GL11] [GL17]
Use of strategic decision-making strategies (e.g., debiasing judgement, taking an outside view)	[WL4] [WL7]	

[2] Only a few studies references are indicated as a sample in the Table 3.2. Full details of the references are provided in the MLR study which is placed in Appendix I.

3.3 Gap analysis and motivation

The analysis of literature revealed that the main context of the selected studies that discussed RE challenges in SADSD, were implementing Agile methods in distributed large organizations and discussed challenges as a part of their implementation. There were 11 studies out of 71 primarily devoted to RE in SADSD. The analysis of primary studies (total 71 primary studies - 49 WL and 22 GL), indicated that there is a diverse set of RE challenges in SADSD, but the requirements prioritization across scaling agile levels was the commonly reported challenge in WL (25 studies out of 49) as well as in GL (8 studies out of 22). Only one empirical study conducted by Daneva et al. (2013) was found which, was specifically dedicated for requirements prioritization in SADSD but the study was mainly devoted to reprioritization of requirements across teams.

Therefore, I could conclude that empirical investigation was lacking on the complete life cycle of prioritization of requirements (i.e., from an idea to operation) in SADSD at the time of conducting this MLR study. In addition to this, strategies that could be employed to mitigate challenges were mainly on the proposal level. This finding about the lack of empirically-based studies, concurred with the findings of secondary studies conducted by Inayat et al. (2015) and Heikkilä et al. (2015). Therefore, empirical research was needed to understand the prioritization of requirements process in the SADSD environment. Another motivating factor to conduct research on requirements prioritization in SADSD, was my interest and previous research experience in RE (Masters research in Agile RE practices), as impacted by the relatively novel phenomenon of adoption of Agile methods in large-scale distributed software development.

3.4 Research questions

Requirements prioritization is essentially a decision-making process (Aurum & Wohlin, 2005; Berander & Andrews, 2005) that typically implies consideration of roles involved and their responsibilities, prioritization criteria, practices, and artefacts being adapted and needing to be investigated as a part of conducting research on understanding the prioritization process (Alenljung & Persson, 2008; Herrmann & Daneva, 2008). Therefore, following are the research questions (RQs) that are set to achieve the aim this research study:

RQ1: How is prioritization of requirements done (i.e., from an idea to operation) in scaled agile distributed software development?

RQ1.1: Who are the decision makers and what are they responsible for?

RQ1.2: What prioritization criteria is employed to make decision on the priority of requirements?

RQ1.3: What practices and artefacts are employed for decision making on the priority of requirements?

RQ2: What are the challenges related to the requirements prioritization processes?

RQ3: What are the strategies to mitigate the impact of reported challenges?

3.5 Summary

Summary of the MLR study which was conducted to understand the RE practices in SADSD, is presented in this chapter which helped me to narrow down the research topic that was initially decided during the commencement of this research study (RE in Scaled Agile Distributed Software Development). As an outcome of MLR study, the aim of the research study (i.e., understanding of requirements prioritization process in SADSD) and the research questions are defined.

In the next Chapter 4, the research design that was employed to achieve the aim of this research study is discussed.

4 Research design of the study

This chapter explains and justifies the research approach which was adopted to investigate the requirements prioritization process in SADSD, and has the following structure:

- Research paradigm
- Research methodology
- Research methods
- Threats to validity
- Summary

4.1 Research paradigm

Beliefs about knowledge can be divided into two categories: epistemological, methodological (Chua, 1986). Epistemology assumptions indicate researcher's perspectives about knowledge and how knowledge can be obtained (Myers, 2009), or the criteria in order to construct and evaluate valid knowledge about the phenomena of interest (Orlikowski & Baroudi, 1991). Methodological assumptions point to adopting an appropriate research method to gather valid evidence.

Following Orlikowski and Baroudi (1991), and Chua (1986), epistemology assumptions are divided into three distinct paradigms: positivist, interpretive, and critical.

4.1.1 Positivist paradigm

Positivist studies typically aim to test theories in order to enhance predictive understanding of the phenomena of interest (e.g., test existing requirements prioritization model in similar and/or different context) (Myers, 2009). In the positivist paradigm, study phenomena are generally single, fragmentable and quantifiable and cause/affect relationships often exist that are being identified and tested via hypothetic-deductive logic (Creswell, 2009). Positivists break the research phenomena into simpler components and corresponding knowledge is developed incrementally from verifiable observations (Creswell, 2009).

In the positivist paradigm, the researcher plays a passive role, that is., does not interfere in the phenomena of interest, and a one to one correspondence exists between the researcher's construct and the phenomena of interest (Myers, 2009). Positivist researchers prefer methods that start with precise theory by which hypothesis can be extracted and

tested in isolation (Easterbrook et al., 2008; Hussey & Hussey, 1997). Controlled experiments, surveys, and case studies are commonly associated methods with a positivist stance (Hussey & Hussey, 1997). Case study method is associated with other research paradigms (e.g., interpretive) as well (Creswell, 2009; Myers, 2009).

4.1.2 Interpretive paradigm

The positivist paradigm is often critiqued due to its incapability to handle richness of social interactions (Klein & Myers, 1999). On the other hand, the interpretive paradigm asserts that knowledge and reality cannot be constructed without the involvement of social actors. The knowledge is constructed through language, shared meaning, and other artefacts (Hoda, 2011; Klein & Myers, 1999). The epistemological belief underpinning interpretive paradigm, asserts that social knowledge such as perspectives of decision making team in requirements prioritization, cannot be constructed through hypothetical deductions (Fitzgerald & Howcroft, 1998; Orlikowski & Baroudi, 1991). Instead, social phenomena can only be understood by looking at them from the inside.

Interpretive studies do not prove or disapprove a hypothesis (Easterbrook et al., 2008). Instead, interpretive studies typically aim to produce an in-depth understanding of the phenomena in a sample context that then can be used to inform other contexts (Lázaro & Marcos, 2006). Interpretivists prefer methods through which theories can be built (Creswell, 2009). Ethnography, exploratory case study, and survey research are commonly associated methods with an interpretive stance (Creswell, 2009). Dawson et al. (2004) noted the importance of positivist and interpretive paradigms in SE and Information System (IS) research. Initially, the positivist stance was the dominant form of research in these fields. However, after realizing the incapability of positivist stance in accommodating the human context, as in decision making events, adoption of the interpretive stance gradually increased in the SE and IS fields (Dawson et al., 2004; Klein & Myers, 1999).

4.1.3 Critical paradigm

The typical aims of positivism and an interpretive approach, is to predict and explain the social phenomena respectively (Creswell, 2009; Orlikowski & Baroudi, 1991). On the other hand, critical researchers decide research based on whom it helps. Typically, participatory methods are preferred by the critical researchers so that the group of people who will get benefit, can participate actively (Chua, 1986). Critical researchers often employ case study research to investigate the phenomenon of interest (Creswell, 2009).

However, action research is the approach that reflects closely the philosophy of critical researchers (Myers, 2009). In the SE field, the critical stance has received less attention as compared to the positivism and interpretive philosophy (Clear & MacDonell, 2011; Glass et al., 2002).

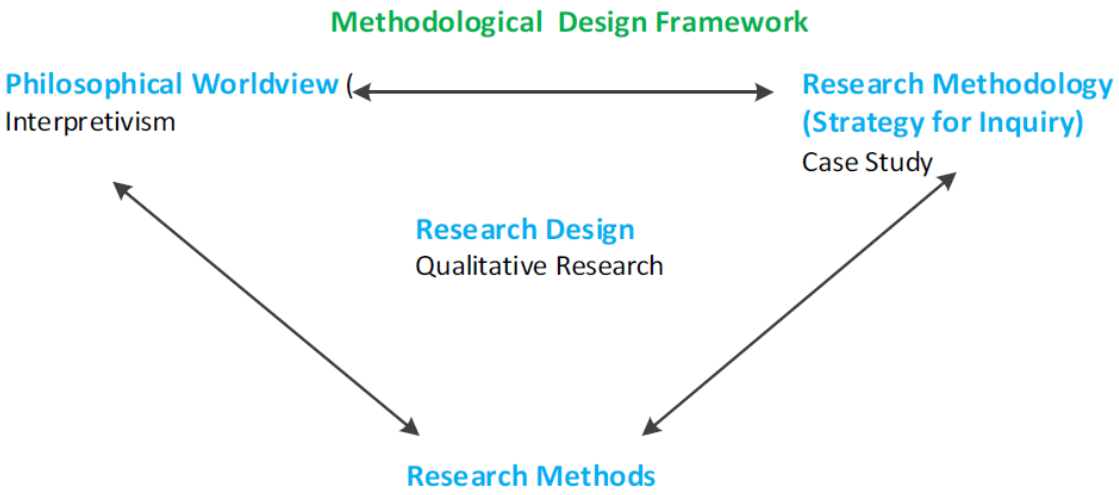
4.1.4 Interpretive research paradigm adopted for this research study

Research paradigms that are discussed in the previous section can provide instinctive perceptions. However, a researcher needs to ensure the chosen approach is compatible with his/her research phenomena and predispositions. Each of the paradigms (discussed in Above Section) highlight a particular perspective; however, none is superior to another or provide an encompassing framework (Myers, 2009).

In SE research, the positivist research paradigm is a dominant paradigm, but this method might be suboptimal when addressing the cultural or social dimensions that are encountered in a distributed team setting (e.g., distributed members of decision making team) (Clear & MacDonell, 2011). It is similarly arguable that a positivist stance would be an appropriate philosophy to understand requirements of the prioritization process in SADSD, given that phenomena are relying on social aspects. Fitzgerald and Howcroft (1998) assert that an interpretive approach best describes the research phenomena when the knowledge or reality is socially constructed (roles involved in decision making on the priority of requirements). Interpretivist researchers argue that relationship between technology, organization, and people are not static (e.g., Ad-hoc requirements prioritization, adapting CTs as per organizations). Therefore, interpretivists intend to understand the moving phenomena (Rosen, 1991). Further, organization/system/groups are typically dependent on humans (e.g., decision-making team). Therefore, they cannot be measured objectively (Rosen, 1991). These considerations make for an interpretive paradigm that was suited to conducting research on investigating the requirement prioritization process in SADSD. Figure 4.1 shows the research design framework that was employed to perform this research (adapted from Creswell (2009)).

Figure 4.1

Research design (Creswell, 2009)



4.2 Qualitative Research Methodology

Qualitative research methodology was employed that fits with the research aim of this research study, understanding of requirements prioritization in SADSD. Qualitative research methodologies are available in various flavours (such as Wolcott, 2001, identified 19 strategies, cited by Creswell (2009), page 12) that can be employed to conduct qualitative study. However, case study, phenomenological research, grounded theory, narrative research, and ethnography are commonly employed research methods (Creswell, 2009).

For this research study, an exploratory multiple case study was employed. The requirements prioritization process was selected as the unit of analysis in studied case organizations. Relevant data was primarily collected via semi-structured interviews. Grounded theoretic analysis was performed to analyse the collected data. The rest of this Section describes the research methodology and the rationale behind choosing the specific research method, in detail.

4.2.1 Exploratory case study method

The phenomena of interest (i.e., requirements prioritization process in SADSD) that the study aimed to investigate, typically involves human behaviour, their languages, perceptions, and interactions in a real-life context (Easterbrook et al., 2008; Fitzgerald & Howcroft, 1998; Klein & Myers, 1999; Orlikowski & Baroudi, 1991). Therefore, an

interpretive approach, was the most appropriate approach as it is encouraged to be employed in qualitative research.

Case studies can be classified into four types: exploratory, explanatory, descriptive, or evaluatory (Yin, 2003). An exploratory case study's aim is to generate new insights and ideas for new research (Yin, 2003). Case study can also be used for explanatory purposes in which existing theories are tested (Creswell, 2009). Descriptive studies aim to portray a phenomenon, whereas evaluatory case studies' purposes are to evaluate tools, methods (Yin, 2003).

An exploratory case study was employed, as this research study's aim was to explore the phenomena of interest in a real-life context (e.g., case study), to generate ideas and insights for new research (Creswell, 2009). The scarcity of research on the life cycle activities of requirements prioritization in SADSD (Dikert et al., 2016; Heikkilä et al., 2015, 2017; Inayat et al., 2015; Petersen & Wohlin, 2010; Rolland, 2015), influenced me in adopting an exploratory case study approach to investigate the requirements prioritization process in SADSD (Runeson & Höst, 2008; Yin, 2003).

The qualitative interpretive exploratory case study approach for the research was influenced by these factors: my past research experience with case study method (Master's research), good fit with research aim (exploring the requirements prioritization process in SADSD), philosophical beliefs (personnel involved in requirements prioritization), time constraints (officially three years recommended for PhD. by AUT), nature of the phenomena of interest being investigated (lack of empirical study on requirements prioritization in SADSD) (Yin, 2003). In addition to this, research questions that were posed to investigate the phenomena of interest, were exploratory in nature, there was limited control over actual events, and the focus on a contemporary event (e.g., ad hoc decision-making events) further influenced me to employ an exploratory case study approach (Yin, 2009, p.8).

Research questions that focus on "What" questions (e.g., what are the practices used for prioritizing the requirements in SADSD?) and "How" questions (e.g., how is the prioritization of requirements done in SADSD?), encourages conducting an exploratory study (Yin, 2009, p.9). In contrast, research questions that focus on "How much" or "How many" gave a rationale to adopt quantitative approach (Yin, 2009, p.9).

An exploratory study has been repeatedly adopted by the researchers to conduct research in the distributed software development field (Alsaqaf et al., 2017; Daneva et al., 2013; Paasivaara et al., 2017). More specifically, RE in distributed setting have also been scrutinized by employing a case study approach (Heikkilä et al., 2010; Paasivaara et al., 2014, 2017, 2018).

4.2.2 Alternative Methods

Research methods provide complementary functions where a particular phenomenon might be examined better by one method than the other (Green et al., 2012). For example, in this research study, a survey approach was not an appropriate method as the aim of this research was not to discover the 'cause' and 'effect' relationship based on generalizable statistical data (Gable, 1994). Surveys were appropriate for the research study that poses "How much" and "How many" type of research questions which are against the research questions that were posed in this research study (Yin, 2009). Furthermore, surveys typically offer a "snap-shot" of the situation at a certain point in time, but yield little information on the phenomena of interest under investigation. (Gable, 1994) Therefore, an in-depth understanding of the phenomena of interest might not be achieved via surveys.

Similarly, experiments (such as laboratory experiments) were not applicable, due to their suitability for inferring casual relationships (Creswell, 2009). Furthermore, experiments remove the phenomena from their context in order to control variables, which was a requirement of this research study (Creswell, 2009). An exploratory nature and epistemological assumption of this research study entailed flexibility and lesser control over events (e.g., decision-making events) that made experiments inappropriate for this research study (Creswell, 2009).

Limited time-period and resources also made ethnography an inappropriate method for this research study (Creswell, 2009).

4.2.3 Case study selection and context

The Multiple case study approach was undertaken as a single case study is typically employed when the case is revelatory (Yin, 2009). But adoption of Agile methods in large distributed organizations is growing (Bass, 2015; Bjarnason et al., 2011; Gat, 2006; Heikkilä et al., 2010; Tripathi et al., 2015) that made selection of multiple cases possible for the research. The multiple case study approach enables cross-site comparison without

necessarily compromising the understanding within each individual site that results in the findings being more compelling (Yin, 2009).

Figure 4.2 shows possible case study designs as suggested by Yin (2009) that could be adapted for single case and multiple case designs. As this research followed multiple case designs, single case designs were crossed out as shown in Figure 4.2. Among the available multiple case designs, this study fell under multiple case study designs (Type 3 which is highlighted with yellow colour). There were two cases with the same context and unit of analysis which were investigated; therefore, remaining two were crossed out as shown in Figure 4.2.

Scaled Agile distributed development was seen as the context and the *requirements prioritization process* (derived from the main study's research question see RQ1 in Section 4.3) that typically involves artefacts, practices, roles, technology employed, constituted the unit of analysis.

Two software development organizations were selected for this research study, one from Australia (Pseudonym: MEL organization) and another one was from New Zealand (Pseudonym: AKL organization). These two organizations were selected as they both were distributed large organizations and were implementing Agile methods at the enterprise level, which was the context of this research study. In order to do this, DAD framework, which is scaling agile framework, was adopted by MEL organization. On the other hand, Scrum @ Scale was employed by the AKL organization. However, the case study organizations were selected without any particular scaling agile framework in mind in order to generalize the findings of this research study. I used my supervisors' and Software Engineering Research Laboratory (SERL)[3] researchers' professional networks, in order to identify the suitable case organizations for this research study.

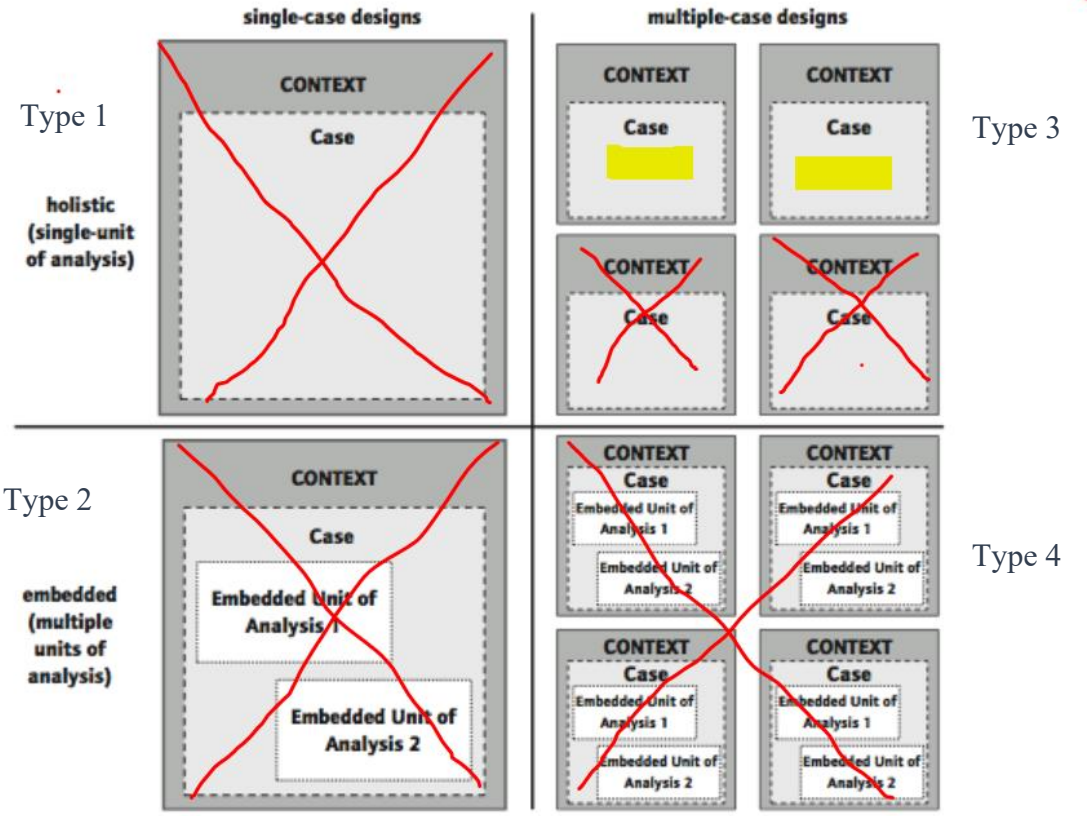
In November 2019, an email was sent to the managers (tribe lead- primary contact for me in the AKL organization, director of engineering in MEL organization)[4], of participating organizations regarding the objectives of this research study and received general commitment from them. In the year 2020 (AKL contacted in March 2020, MEL in August 2020), formal invitations were sent to the participated organizations. The studied

[3] https://serc.aut.ac.nz/
[4] See Glossary (page no. 268) for the description of Tribe lead and director of engineering role.

organizations' anonymity is maintained in the findings and pseudonyms are used to represent them throughout this study.

Figure 4.2

Case study designs (Yin, 2014)



4.3 Research Methods

This section describes the research methods that constitute research questions, data collection, data analysis, and validation.

4.3.1 Research questions

The aim of the research study was to develop an in-depth understanding of the prioritization of requirements in scaled agile distributed software development.

Following are the research questions (RQ) that were set to attain the aim of this research study.

RQ1: How is prioritization of requirements done (i.e., from an idea to operation) in scaled agile distributed software development?

RQ1.1: Who are the decision makers and what are they responsible for?

RQ1.2: What prioritization criteria are employed to make decisions on the priority of requirements?

RQ1.3: What practices and artefacts are employed for decision making on the priority of requirements?

RQ2: What are the challenges related to the requirements prioritization processes?

RQ3: What are the strategies to mitigate the impact of reported challenges?

4.3.2 Ethical considerations

This research study needed human participation and their personal and organizational knowledge to understand the phenomena of interest. Therefore, ethics approval was mandatory before making the formal contact with the case organizations. In order to do this, an ethics application form was filled, signed and then submitted to the Auckland University of Technology Ethics Committee (AUTEC) who then approved the ethics application (a copy of ethics approval letter is available in Appendix E). During the time of ethics application, the title of this research study was "Requirements Prioritization Process in Global Scaled Agile Software Development". Later, a slight change in the title of research topic was made to capture the locally distributed as well as globally distributed focus and the research topic was updated as "Requirements Prioritization in Scaled Agile Distributed Software Development". The ethics application form had the following key characteristics which were then followed throughout this research study.

- demonstration of integrity
- data handling- confidentiality and privacy
- avoiding plagiarism
- conflict of interest

The following are the other forms that were prepared and submitted to the AUTEC as a part ethics application.

- Organization information sheet (a copy is available in Appendix A) describes the objectives of research study, and potential participants that were required also mentioned.
- Participant information sheet (a copy is available in Appendix B) was prepared to describe the research objectives, benefits of research study to the individual participants of the participating case organizations.

- Consent form (a copy is available in Appendix C) to get an agreement for data collection from the participant.
- Interview protocol (a copy is available in Appendix D) to set the pre-defined questions towards achieving the objective of research study.

These documents were shared with the participants, before scheduling an interview with them in order to provide an understanding of this research study. All the participants were informed about the confidentiality, privacy and ability to withdraw their participation at any stage of research project.

4.3.3 Data collection and organization

Data collection categorizations proposed by Lethbridge et al. (2005) for the software engineering field were adapted for this research study, that includes (i) direct method-collected directly via study's participants through semi-structured interviews, which was a primary source of data employed for this research study (Lethbridge et al., 2005). In addition to this, (ii) indirect method - raw data which was collected via minimal interaction with participants, such as recorded video/audio of decision-making events (Lethbridge et al., 2005), (iii) independent techniques - already available data such as requirements artefacts (e.g., business-case) were also employed wherever possible (Lethbridge et al., 2005) to triangulate the data intended to minimize the impacts of a single-source interpretations and to accomplish the objectives of this research study (Creswell, 2009; Maxwell, 2012; Yin, 2009).

4.3.3.1 Semi-structured interviews

Semi-structured interviews were employed as a primary source of data to understand the phenomenon of interest (i.e., requirements prioritization process in SADSD) and to gain further insights (Creswell, 2009; Fossey et al., 2016; Yin, 2003). The semi-structured interviews enabled me to reword the interview questions in order to adjust the language level. Semi-structured interviews also helped me to answer the questions asked by the interviewees or clarify their doubts as and when necessary. During the interviews, questions were asked by the interviewees according to the flow of conversation rather than following the strict order that was defined in the interview protocol (Larsson, 2009). An adequate time was given to the interviewees to express their views on the questions intended to achieve the objectives of this research study. This method enabled me to get in-depth understanding of the interviewees' experiences as well as helped me in building the relationship with them. As a result, I was able to probe for more information on

particular points and requested additional artefacts wherever needed from the interviewees (sample evidence in the below quote, when I requested more information from the Participant MEL_P2).

> *Interviewer: I'm still not clear on how the actual ranking is decided. I mean what particular factors do you use to make decision.*
>
> *MEL_P2: Like so I said there's certain vectors they use to rank. I'm not really sure. There are three categories they look into. So that determines the score. I can share them. Probably ping them to you later, 'cause I can't remember them on top of my head.*
>
> *Interviewer: no problem, please share the document with me, it would be a great help.*

4.3.3.1.1 Designing the interview questions

Open ended interview questions (a copy of interview protocol is available in Appendix D), were designed via studying the literature related to the requirements prioritization and scaled agile distributed development. The interview protocol was reviewed by the supervision team, and the researchers of SERL with experience in scaling agile practices and RE activities in order to ensure clarity. Based on their feedback, interview questions were modified wherever needed.

The interview protocol had three main sections: participant briefing, background information, and requirements prioritization process

Participant briefing, generally aimed to describe the aim of research to the interviewees. The aim of the research study was described via a diagram[5] and shared with the interviewees during the interview in order to provide them a high-level overview of this research study.

- Background section included questions related to the interviewees' details: e.g., age, education, experience, ethnicity, designation, and studied organizations' details: type of scaling agile methods, number of sites involved in software development.
- Requirements prioritization process included questions related to the sources of requirements: factors consideration, practices, decision making events, decision makers, collaborative technologies, distributed team members, challenges, potential strategies.

[5] Diagram is shown in Appendix F.

Pilot interviews were conducted with the managers of both participating organizations (i.e., director of engineering in MEL organization, tribe lead in AKL organization), (Yin, 2009) to fine tune the interview questions. These key participants (purposive sampling) (Barbour, 2001) were selected due their ability to provide information on the requirements prioritization process as they were the senior and key members of decision-making team(s) who make decisions on the priority of requirements (Barbour, 2001; Flyvbjerg, 2006). The pilot interview confirmed interview questions were relevant in terms of answering the main research questions that were posed to achieve the aim and objectives of this research study.

4.3.3.1.2 Participants' selection

Kick off meeting occurred with AKL organization in March 2020, and with MEL organization in September 2020. They were held with the managers of both participating organizations, that is the director of engineering in MEL organization, and tribe lead in AKL organization, to discuss the aim and objectives of research and to understand the development process of the participating organizations. In addition to this, Non-disclosure Agreements (NDA) were also received and signed during the kick-off meeting between the participating organizations and the researcher. This was to protect the sensitive commercial information of the organization and to enable them to open joint discussions around organizational confidentiality agreements. These key participants were chosen because they were my primary contact in both studied case organizations. In addition to this, they were the senior members of the decision-making teams who make decisions on priority requirements.

Purposive sampling (Barbour, 2001) was employed to select the potential participants for this research study. First, I prepared an initial list of potential participants. These included: product owner, product manager, team leader, chief product owner, chief technology officer, chief customer officer, enterprise architecture, user experience/user interface (UX/UI), developers, testers, architecture owner via studying the literature and some of the popular scaling agile frameworks (e.g., DAD, Scrum @ Scale), who are typically involved in decision-making on the priority of requirements. This list was then discussed with the primary contact of the participating organizations and revised as per their feedback.

An initial interview request was sent to the potential participants by the primary contacts. However, their participation was completely voluntary. After that I was introduced by the

primary contact via email to the potential participants to arrange an interview with them. Before commencing the interview with the potential participants, participant information sheet and consent form were sent to the interviewees in order to provide an overview of the research study and receive their approval before proceeding to the interview stage.

4.3.3.1.3 Interview details

Semi-structured face to face interviews were initially planned to collect relevant data for this research study (Yin, 2009). However, due to the COVID-19 pandemic restrictions the majority of the interviews were conducted virtually via MSTeams[6]. In addition to this, the time-period for data collection also got extended from six months to one year due to COVID-19 pandemic. Before starting the interview session, signed consent form and an agreement on the audio/video recording of the interview were taken from the interviewee as recording enhanced accuracy in data collection. However, notes as well as emerging questions were taken during the interview session.

Two rounds of interviews were organized, with the participating organizations selected for this research study. The first round of interviews was a formal round typically planned for 45 minutes to 60 minutes at both case organizations. However, with some participants it went beyond 60 minutes. In such cases, permission was obtained from the interviewees, to extend the interview duration. But the second round of interviews was typically short in duration mainly conducted to confirm the findings that were made via analysing the data that were collected in the first round. The second round of interview was typically informal and conducted with the key participants to provide in-depth information on the requirements of the prioritization process. In some cases, a follow up email was sent to the participants to seek clarity on the requirements prioritization process as and when needed.

AKL organization's interview details: In the AKL organization, data collection started in March 2020. As AKL was a locally distributed organization with two sites (Auckland, Hamilton), participants from both sites were interviewed to get viewpoints of the distributed members of the decision-making team on the requirements prioritization process. However, the majority of the participants were from Auckland, because most of the teams were Auckland based. Interviews were mainly conducted virtually over MSTeams due to COVID-19 pandemic restrictions. However, some of the interviews were conducted face to face wherever possible. A total of ten interviews were conducted

[6] Information MSTeams can assessed via https://www.microsoft.com/.

in the first round and two informal interviews were conducted in the second round of interviews. In addition to this, six follow-up emails were sent to the participants to get further clarification on the requirements prioritization practices from them. After conducting the ten interviews in the first round of interview, I stopped data collection in the AKL organization as no new insights were emerging from the interviews. Participants' interview details and demographic information are shown in Table 4.1 and Table 4.2 respectively. Participants' anonymity is maintained, and pseudonyms are used to represent the participants in this research study.

Table 4.1

AKL interview details

Participant pseudonym	**First round interviews (Formal interviews)**	**Second round interviews (Informal interviews)**	**Email follow-up**	**Venue**	**Site**
AKL_P1	0 hr 51 min 43 sec	0 hr 25 min 08 sec	3	Company meeting room	Auckland
AKL_P2	1 hr 03 min 01 sec		1	MSTeams	Auckland
AKL_P3	0 hr 46 min 24 sec			MSTeams	Auckland
AKL_P4	0 hr 59 min 23 sec			MSTeams	Auckland
AKL_P5	0 hr 36 min 04 sec			MSTeams	Hamilton
AKL_P6	0 hr 53 min 05 sec			MSTeams	Hamilton
AKL_P7	0 hr 48 min 45 sec	0 hr 21 min 11 sec	2	Company meeting room	Auckland
AKL_P8	0 hr 58 min 26 sec			MSTeams	Auckland
AKL_P9	0 hr 37 min 04 sec			MSTeams	Auckland
AKL_P10	0 hr 31 min 48 sec			MSTeams	Auckland

Table 4.2

AKL participants' demographic details

Role	Gender	Experience	Qualification	Site
Tribe lead	Male	10 years	Masters	Auckland
Senior Product Owner	Male	13 years	Masters	Auckland
General Manager	Male	15 years	Masters	Auckland
Tribe lead	Male	12 years	Bachelor	Auckland
Senior Business Analyst	Female	15 years	Diploma	Hamilton
Business Analyst	Female	15 years	Masters	Hamilton
Scrum master	Male	3 years	Diploma	Auckland
Scrum master	Female	7 years	Masters	Auckland
Senior developer	Male	12 years	Masters	Auckland
Tester	Male	10 years	Bachelor	Auckland

MEL organization's interview details: In MEL organization, data collection was started in September 2020. As MEL organization was a globally distributed organization which had six sites, but only participants from the two sites (USA, Australia) were interviewed. The company's head office was in USA and the majority of the members of the decision-making team specifically the portfolio decision making team, were located in USA. On the other hand, majority of the engineering teams were located at the Australian site. Therefore, participants from these two sites were selected to be investigated on the requirements prioritization process in the participating organization. All the interviews were conducted virtually over MSTeams in the MEL organization due to COVID-19 pandemic travel restrictions. A total of 18 interviews were conducted in the first round and three informal interviews were conducted in the second round of interviews. In addition to this, 11 follow-up emails were sent to the participants to get further clarification on the requirements prioritization practices from them. Participants' demographic information and interview details are shown in Table 4.3 and Table 4.4 respectively.

Table 4.3

MEL participants' demographic details

Role	**Gender**	**Experience**	**Qualification**	**Site**
Product manager	Male	15 years	Bachelor	Australia
Product Owner	Female	10 years	Masters	Australia
Product Manager	Female	11 years	Masters	USA
Senior manager UX	Male	18 years	Bachelor	USA
Director of engineering	Male	25 years	Bachelor	Australia
Program Architecture Owner	Male	20 years	Bachelor	Australia
Senior Director Product management	Male	23 years	Bachelor	USA
Program manager	Female	21 years	Bachelor	USA
Product manager	Male	20 years	Masters	Australia
Team lead	Male	14 years	Masters	Australia
Vice president of engineering	Male	23 years	Bachelors	USA
Team lead	Male	16 years	Masters	Australia
Architecture Owner	Male	25 years	Bachelor	Australia
Developer	Female	9 years	Bachelor	Australia
Team lead	Male	15 years	Masters	Australia
Senior vice president of architecture	Male	24 years	Masters	USA
Senior vice president of UX	Female	20 years	Masters	USA
Program manager	Male	23 years	Bachelor	USA

Table 4.4

MEL interview details

Participant pseudonym	First round interviews (Formal interviews)	Second round interviews (Informal interviews)	Email follow-up	Venue	Site
MEL_P1	1 hr 01 min 58 sec	0 hr 20 min 35 sec	2	MSTeams	Australia
MEL_P2	0 hr 43 min 31 sec		1	MSTeams	Australia
MEL_P3	0 hr 44 min 07 sec			MSTeams	USA
MEL_P4	0 hr 55 min 24 sec			MSTeams	USA
MEL_P5	1 hr 04 min 31 sec	0 hr 37 min 25 sec 0 hr 46 min 16 sec	5	MSTeams	Australia
MEL_P6	1 hr 06 min 30 sec		1	MSTeams	USA
MEL_P7	0 hr 51 min 44 sec			MSTeams	USA
MEL_P8	1 hr 21 min 45 sec			MSTeams	USA
MEL_P9	1 hr 18 min 18 sec		1	MSTeams	Australia
MEL_P10	1 hr 23 min 23 sec		1	MSTeams	Australia
MEL_P11	0 hr 47 min 50 sec			MSTeams	USA
MEL_P12	1 hr 09 min 08 sec			MSTeams	Australia
MEL_P13	1 hr 08 min 04 sec			MSTeams	Australia
MEL_P14	0 hr 27 min 23 sec			MSTeams	Australia
MEL_P15	0 hr 52 min 23 sec			MSTeams	Australia
MEL_P16	0 hr 56 min 51 sec			MSTeams	USA
MEL_P17	0 hr 38 min 55 sec			MSTeams	USA
MEL_P18	1 hr 01 min 47 sec			MSTeams	USA

4.3.3.2 Artefacts collection

The second source of data collection was requirements artefacts (such as decision-making criteria, business-case, requirements hierarchy structure) and were collected from the studied organizations wherever possible to validate the data that were collected through semi-structured interviews.

4.3.3.3 Access of decision-making events

In the AKL organization, I attended some of their decision-making events, such as the bigroom planning- a quarterly planning meeting, as an observer and in MEL organization I managed to get the recordings of their events in order to understand and to validate the requirements prioritization activities and practices that I understood, via semi-structured interviews which was the primary data collection method.

4.4 Data analysis

Data analysis was employed within the case analysis and cross-case analysis. In the first approach, each studied organization's data was analysed separately without any particular theoretical framework in mind. On the other hand, cross-case analysis aids commonalities and differences between the studied case organizations, and further helped in building the conceptual framework of requirements prioritization process in SADSD.

4.4.1 Pre-processing

For each studied organization, an individual electronic folder was maintained to organize the collected data that include transcribed interviews, follow-up data, recordings of interviews, recordings of decision-making events, notes. Manual verbatim transcription instead of automatic transcription services of all the interviews was done in order to analyse the data. The manual verbatim transcription aided closeness with data that further helped me in probing the questions that were typically clarified in the upcoming participants' interview and follow-up email. Another reason behind the selection of manual transcription was the intellectual property that could have been breached if automatic transcription software was adopted. In the beginning (with the first three interviews), an interview was transcribed before scheduling the next interview. As transcription was a time-consuming process, potential questions that emerged and/or were unanswered during the interview, were noted down via listening to the recording of interviews before scheduling the next interview. This enabled me to cover them via follow-up or in the future interviews.

An appropriate tool that could be adopted for data analysis was also explored during the pre-processing phase. Initially, I decided to use Spreadsheet for data analysis, but I was influenced through the learning of NVivo a popular qualitative data analysis tool as well (Hutchisona et al., 2009). As I was new to the NVivo tool, firstly, I learned the basics about NVivo tool and then transported two of the interview transcripts from the MEL organization in the NVivo tool for a trial purpose. During that trial period, I realized I was not comfortable with NVivo for data analysis because its structural framework was limiting the way that data could be organized. As a result, I switched back to the Spreadsheet for data analysis that enabled better visualization of the data. This observation coincides with Hoda (2011), who also experienced the structural framework of the tool posing limitations on data organization.

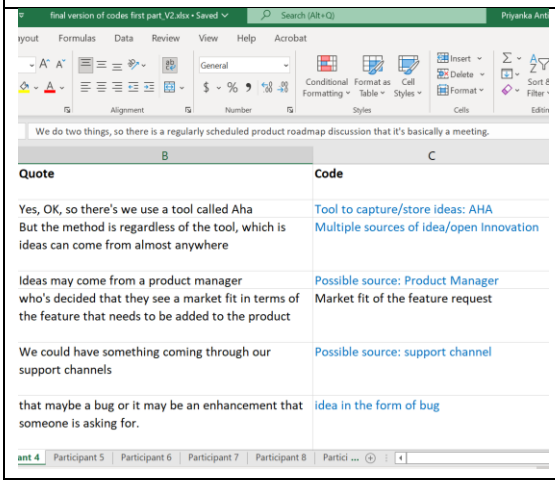
4.4.2 Within the case analysis: Grounded theoretical analysis

Each of the studied case organizations was treated as an individual unit. Individual case study data was analysed without any particular theoretical framework in mind. In order to analyse the collected data, individual spreadsheets were maintained for each of the studied case organizations. Two levels of spreadsheet were maintained, at the first level, codes were assigned to the indicators and in the second level of spreadsheet, emerging concepts and categories were defined (sample is shown in Table 4.5). *Indicators* are the real data that were discovered from the data (such as events described in the interviews, documents) collected from the studied case organizations (Adolph et al., 2011). A word, phrase, sentence, or a paragraph in the data typically served as an indicator (Adolph et al., 2011).

Table 4.5

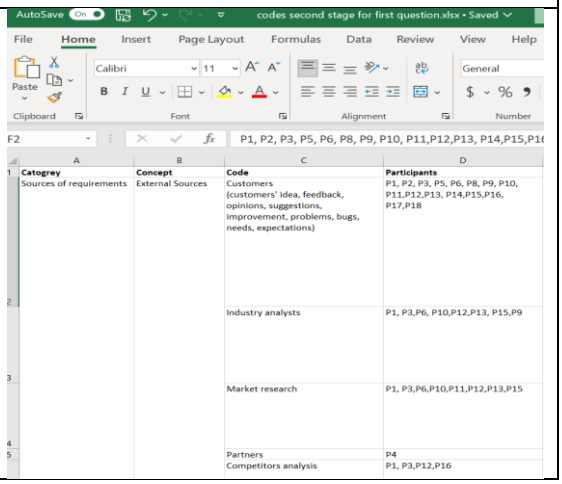
Sample of data organization via Spreadsheet

First stage



Quote	Code
Yes, OK, so there's we use a tool called Aha But the method is regardless of the tool, which is ideas can come from almost anywhere	Tool to capture/store ideas: AHA Multiple sources of idea/open Innovation
Ideas may come from a product manager who's decided that they see a market fit in terms of the feature that needs to be added to the product	Possible source: Product Manager Market fit of the feature request
We could have something coming through our support channels	Possible source: support channel
that maybe a bug or it may be an enhancement that someone is asking for.	idea in the form of bug

Second Stage



Catogrey	Concept	Code	Participants
Sources of requirements	External Sources	Customers (customers' idea, feedback, opinions, suggestions, improvement, problems, bugs, needs, expectations)	P1, P2, P3, P5, P6, P8, P9, P10, P11,P12,P13, P14,P15,P16, P17,P18
		Industry analysts	P1, P3,P6, P10,P12,P13, P15,P9
		Market research	P1, P3,P6,P10,P11,P12,P13,P15
		Partners	P4
		Competitors analysis	P1, P3,P12,P16

4.4.2.1 Data analysis technique: Grounded theoretical analysis

Grounded theory is a commonly adapted method when the research study is qualitative and exploratory in nature and the theoretical development of the phenomena of interest is in the early stages (Glaser & Strauss, 1967). This research fulfils the condition posed by Alberto Espinosa et al. (2007) as during the MLR study that was conducted to understand the current state of RE in SADSD, I found research on requirements prioritization in SADSD were under theorised. Therefore, an empirical grounded theoretical analysis technique was adopted in order to analyse the collected data.

Inductive analysis was performed via following the steps that include *coding, emerging concepts, emerging categories* (Glaser & Strauss, 1967). *Constant comparison* which is

an important component of grounded theory (GT), was adapted at each step of data analysis that helped in defining and re-defining the emerging codes, concepts, and categories wherever needed. In parallel, memo notes which is another key component of GT, were also maintained that helped in describing and identifying the relationship among codes, concepts and categories (Glaser & Strauss, 1967).

Emerging Codes: each transcribed interview and all the relevant documents (e.g., requirements artefacts) that were collected, were read several times to ensure none of the information was left uncovered (Seaman, 1999). A very simple process suggested by (Adolph et al., 2011) was followed for open coding where a code can emerge from a single word or a single line or multiple lines, and more than one code can emerge from a single indicator (Adolph et al., 2011). Two types of coding were performed, that include in-vivo coding (words used by the interviewees) and descriptive coding (summarizing the indicator in a word or phrase) (Glaser & Strauss, 1967). In parallel, constant comparison analysis was also performed to compare the emerging codes against the new or existing codes that were found within the interview transcript and across the transcripts (Glaser & Strauss, 1967). This comparison yielded data reduction as well as data abstraction. Table 4.6, Table 4.7, and Table 4.8 shows a small portion of data, and the descriptive and in-vivo codes that emerged from the participants MEL_P5, MEL_P3, MEL_P1 respectively.

Table 4.6

Sample list of small portions of coded data from MEL_P5

Collected Data	**Codes**
Yes, it's a good question. I mean we try and come out of a principle of that anyone can have a good idea or that ideas can come from anywhere. But equally, you know, we make sure that anyone within [..] can contribute to the ideations or ideas for our products.	various sources of ideas,
Like we have half a day reserve for the engineers for innovative ideas. […] Its fortnightly.	internal sources- engineers innovation morning
but predominantly they come from your customers	Dominant source of idea: Customers
but we use a tool called Aha for that and it's a web-based tool for capturing ideas,	Tool to capture ideas: AHA (web-based tool)
So, customers have the ability to submit ideas themselves.	self-submission of customers' idea
Right now, that's generally done through our support channels. You know if there are user of our system then they can raise the idea through support and ask it to be logged.	Support channel: get customers' ideas through support channel
It could come through other mechanisms; they have customer success managers and that type of thing.	Customer success manager to get ideas from customers
What we are in the process of doing this will be done before the end of the year is having the ability for customers to submit ideas directly via into Aha via the web so any customer would be able to submit ideas.	Collaborative tool AHA, self-submission of customers' idea- duplicate code
At a customer level, you know we do quarterly reviews with them, particularly the larger customers to understand, you know they have it getting value from our solution and working with them that way	Quarterly reviews with biggest customers
there are some user groups where customers actually get together and also includes […] folks in more of a group setting to bring ideas together and collaborate together on ideas.	User groups (i.e., customers and participants from the organization) to collaborate on ideas
So, there's multiple ways that we interact with our customers	various techniques to collaborate with customers
and our customers interact with each other as well. There's a system we have, a community. It's again another essentially a web application where customers can come together and help each other, share ideas and share ways of using our solutions.	customer community (web portal): to perform collaboration among customers to share ideas, help each other etc.
So yeah, so some of the groups that the user groups that we've got, I think meet monthly	User groups meet monthly
And then we do these quarterly business reviews with our biggest customers. So that's you know, four times a year as well as they are, the more formal processes that we have in place.	Quarterly business review with biggest customers (duplicate code)

Table 4.7

Sample list of small portions of coded data from MEL_P3

Collected Data	**Open Codes**
They really come from all over the place, right? Like people love to have ideas and they like to have their ideas heard, so they definitely come from everywhere	Multiple sources of idea
they come from product, they come from engineers, they come from senior leadership team.	Product managers, Engineers, Senior leadership team
but the most important place they come from customers right	Dominant source of idea: Customers
it could be that our competitors have something that we don't have	competitor analysis
We also have what we call customer community and it's a place where customers could go and submit like bug tickets and they can have enhancement requests there as well.	self-submission of customer ideas (bug, enhancement request) customer community

Table 4.8

Sample list of small portions of coded data from MEL_P1

Collected Data	**Open Codes**
address new area that they [customers] think is important. That's one of the ways that we find out and our customers tell us	Customer recommendation
the different channels we have for them	Multiple channels for customer feedback
Then there's market research.	Market research
So we as product managers, continuously do product research for our product areas as well	Continuous product research
So, we as product managers, continuously do product research for our product areas as well find out what is changing	Product research via Product managers, track trends
if there's anything that we need to do to also differentiate ourselves from our competitors	Product differentiation, competitor analysis
It can be industry analysts telling us companies like Gartner and Forrester, which put out reports in our respective areas every year	Industry analyst (Gartner; Forrester); yearly Industry analysts' report
this can come through various channels	various sources for ideas
We have a product management tool called Aha we capture everything in and not just ideas	PM tool (AHA)
the hackathons are not as frequently, but hackathons I think we have at least one or two every year major events. so everybody who's got an idea gets to fly down to the US, and it doesn't have to be just product, it's product engineering.	Occasionally run hackathons, Internal sources- product, engineering

Constant comparison: Table 4.9 shows how codes were aggregated via constant comparison method within the interview transcript and across the transcript. From the participant MEL_P5, initially 14 codes emerged (shown in Table 4.6), that were later aggregated into 12 open codes via constant comparison method. On the other hand, one new code emerged from the participant MEL_P3 and five new codes emerged from the participant MEL_P1 shown in Table 4.9.

Table 4.9

An example of aggregating codes via constant comparison method

Open Codes	**Source**
Multiple sources of idea	MEL_P5, MEL_P3, MEL_P1
Internal sources	MEL_P5
Product managers	MEL_P3, MEL_P1
Engineers	MEL_P5, MEL_P3, MEL_P1
Senior leadership team	MEL_P3
Occasionally run hackathons	MEL_P1
Innovation morning fortnightly	MEL_P5
Customers' ideas	MEL_P5, MEL_P3, MEL_P1
Various techniques to collate customers ideas	MEL_P5, MEL_P1
Collaborative tool Aha for capturing the customers ideas	MEL_P5, MEL_P3, MEL_P1
self-submission of customers' idea via AHA	MEL_P5, MEL_P3
Customers' ideas via Support channel	MEL_P5
Customers' ideas via customer success managers	MEL_P5
Quarterly reviews with biggest customers	MEL_P5
User groups (Monthly meeting)	MEL_P5
competitor analysis	MEL_P3, MEL_P1
Market research (track market trends)	MEL_P1
Continuous idea generation	MEL_P1
Customers' ideas via Product managers	MEL_P1
Industry analysts	MEL_P1

Emerging Concepts: As the analysis progressed, some of the open codes turned into concepts and concepts turned into categories. As shown in Table 4.9, codes that were emerged indicating internal sources (e.g., product managers, engineers) and external (e.g., customers, industry analysts) are typically the sources of potential requirements. Thus, using constant comparison method, I grouped the codes into emerging concepts 'External sources and internal sources' shown in Figure 4.3 and Figure 4.4 which describes various practices that are employed to collate potential requirements.

Figure 4.3

A sample of the emerging concept 'External Sources'

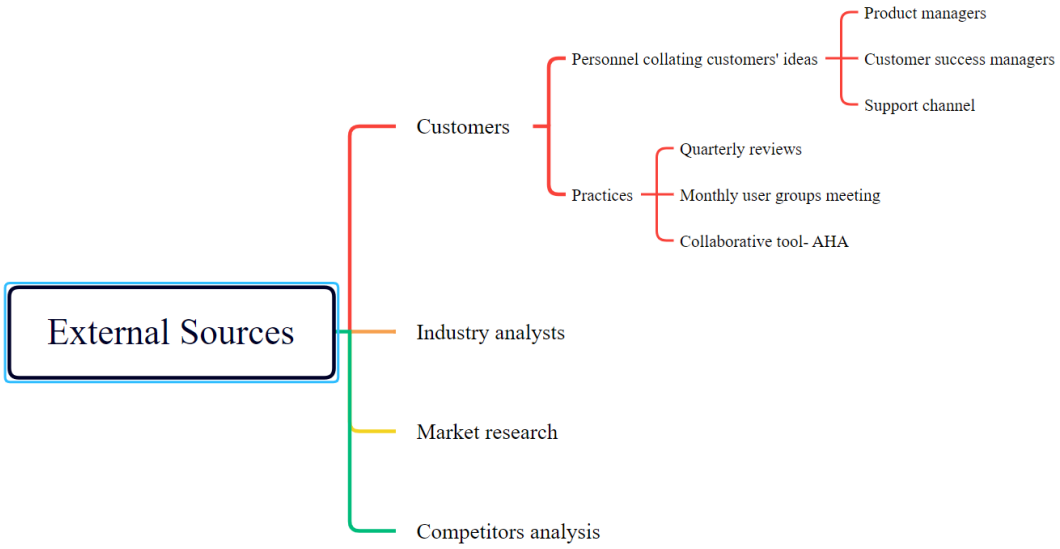
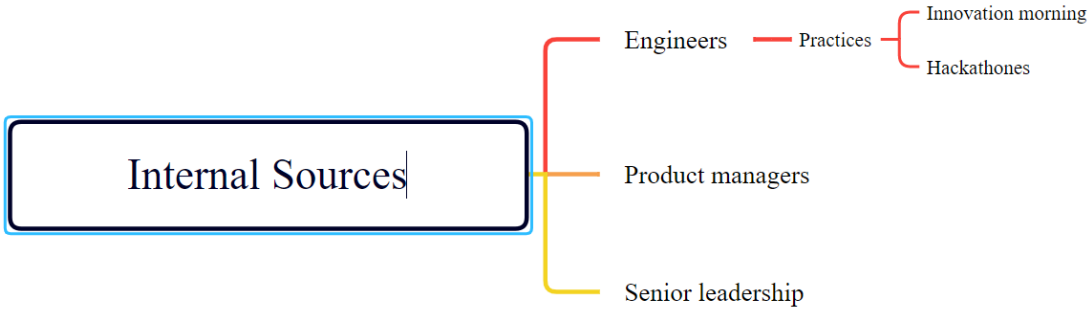


Figure 4.4

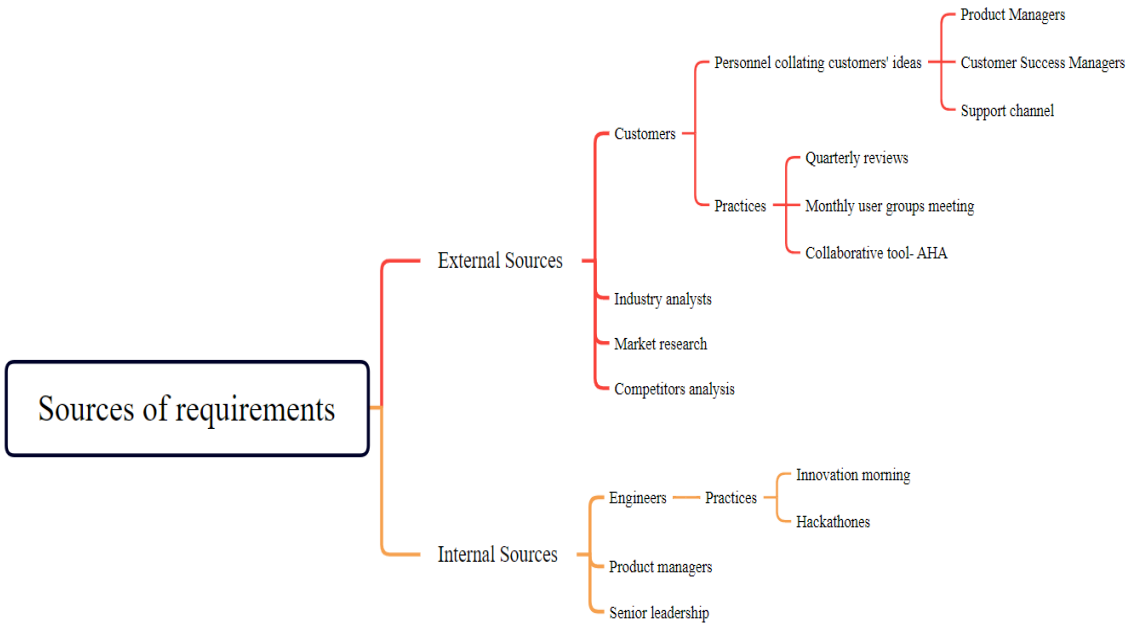
A sample of the emerging concept 'Internal Sources'



Emerging categories: As shown in Figure 4.3, Figure 4.4, external and internal sources emerged as concepts indicating that potential requirements are collated from the various sources. Using constant comparison analysis, an emerging category 'sources of requirements' constituted (shown in Figure 4.5) from the concepts '*internal sources*' and '*external sources'* as well as open codes (i.e., a common code 'various sources of ideas' which emerged from all three participants' data).

Figure 4.5

A sample of emerging concept 'Sources of requirements'[7]



Memos: Memos were constantly maintained to record the relationship among codes, concepts and categories (Glaser & Strauss, 1967). Memos helped me to record my thoughts and opinions with regards to the codes, concepts, and categories. An example of memo of concept 'external sources' is shown in Table 4.10

[7] Note: this is just a sample of small portions of coding (Analysis of small portion of three participants' data).

Table 4.10

Memo for the category ‘Sources of requirements’

Memo
Collecting the requirements is a continuous process. Main sources of requirements include internal and external sources. In case of internal sources, Product management is the dominant source. On the other hand, customers are the dominant external source. This is commonly stated by the participants of the studied case organizations. As per my observation, an open innovation approach is employed for collecting the potential requirements which was validated with MEL_P5 and MEL_P1 via follow-up. I can support this finding with Tony’s and Danila’s paper as well during writing.

The following are the 14 salient categories that emerged by analysing data of studied case organizations. However, some of the categories were derived directly from the research questions such as decision-making team, decision making practices, requirements artefacts, challenges, mitigation strategies.

1. Sources of requirements
2. Portfolio level
3. Segment level (Segment level in MEL organization), Tribe level (Tribe level in AKL organization)
4. Team level
5. Requirements hierarchy
6. Requirements classification
7. Requirements artefacts
8. Decision making practices
9. Decision making events
10. Decision making team
11. Collaborative technologies
12. Incorporate distributed members of decision-making team(s)
13. Challenges
14. Mitigation strategies

4.4.3 Deductive analysis

Salient categories that were identified as an outcome of inductive analysis (as discussed above in Section 4.4.2), were refined and redefined wherever needed via deductive analysis to structure the findings of this study. Some of the salient categories which had potential to become a core category include: portfolio level, segment level/tribe level, team level. As a result, relevant literature was searched, relating to the initial findings that emerged in the form of ‘salient categories’ via inductive analysis.

The remainder of this Section describes the literature that were studied in relation to the categories that emerged from the inductive analysis and provided guidance on building the initial conceptual framework as shown in Table 4.11.

4.4.3.1 Key theory elements adopted from the literature to structure the findings

Requirements prioritization which is the focus of this research study is essentially a decision-making process through which the requirements are sequenced and decided for implementation (Alenljung & Persson, 2008; Aurum & Wohlin, 2005; Herrmann & Daneva, 2008; Ngo-The & Ruhe, 2005). According to the traditional RE process as suggested by Sommerville (2009), requirements prioritization is generally a part of requirements elicitation and analysis. Sommerville (2009) defined requirements elicitation and analysis as an iterative process that typically consists of requirements discovery and understanding, requirements classification, requirements prioritization and negotiation, requirements artefacts where activities typically occur in the form of cycles and are adapted by the organizations as per their need. Berander and Andrews (2005), Lehtola et al. (2004) and Aurum and Wohlin (2005) also stated that prioritization of requirements is not a onetime activity. Instead, requirements prioritization is a continuous decision-making process which is typically performed iteratively at different abstraction levels and with different information in different phases of software development process. Typical decision-making levels as suggested by Aurum and Wohlin (2005) to prioritize the requirements include organizational level, product level, and project level

- Organizational level where requirements are typically defined and analysed in the form of organization's strategic objectives that generally influence the products that an organization ought to develop (Aurum & Wohlin, 2005)
- Product level where the requirements that are imposed on product(s) typically in the form of project(s) are defined and prioritized (Aurum & Wohlin, 2005).
- Project level where requirements within the project(s) are defined and prioritized (Aurum & Wohlin, 2005)

As can be seen in the Section 4.4.2 the decision-making levels (i.e., portfolio level, segment level/tribe level, team level) that emerged from the inductive analysis of each of the studied case organizations, coincides with the studied literature. The decision-making levels suggested by Aurum and Wohlin (2005) and the requirements prioritization activities as a part of requirements elicitation and analysis as suggested by Sommerville (2009) guided me to develop an initial conceptual framework and structure the findings

of the studied case organizations. Also, the salient categories that had emerged from the inductive analysis were refined wherever needed.

The deductive analysis guided me to settle on three core categories in terms of decision-making levels, that include: portfolio level, segment level/tribe level, team level. The remaining categories were marked as sub-categories which were aggregated via adapting the Sommerville (2009) framework. The emerging category 'Sources of requirements' was refined, and a new name was given as 'requirements discovery and understanding'. Decision-making events, decision-making practices, decision-making teams were grouped under requirements prioritization and negotiation activities. Requirements classification and requirements hierarchy were also aggregated, based on their relevance with each other. No changes were made in some of the categories, that include requirements artefacts, collaborative technologies, incorporate distributed members of decision-making teams, challenges and strategies.

These core elements of the initial conceptual framework are depicted in Table 4.11. Pragmatic notations are employed to draw the conceptual framework instead of using formal notation such as process diagram, data flow diagram (DFD). The main reason to employ pragmatic notation is that I wanted to provide a pictorial representation of the high-level overview of prioritization process that could be difficult to achieve via formal pragmatic notation. However, the pragmatic notations are initially adapted from the DFD. Figure 4.6 shows the legend that is employed to draw an initial conceptual framework (shown in Table 4.11)

Figure 4.6

Legend used in the conceptual framework

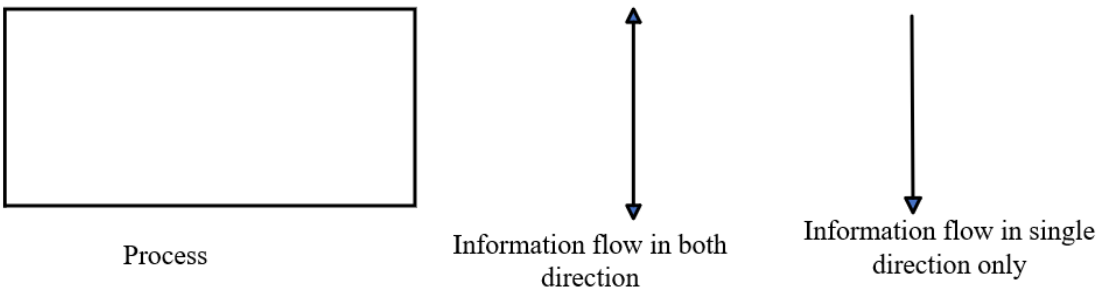
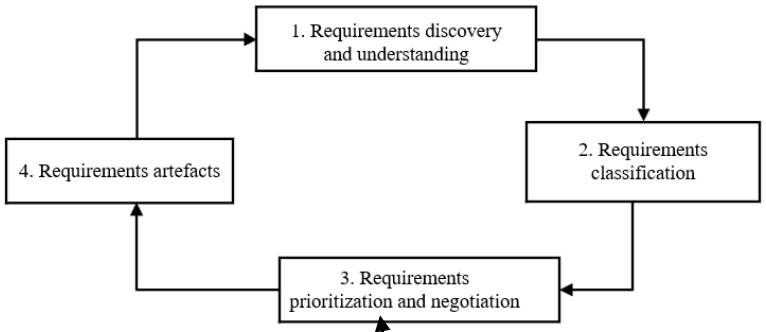
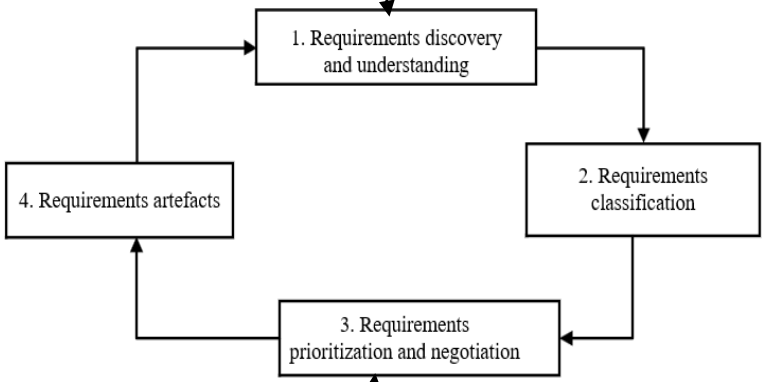
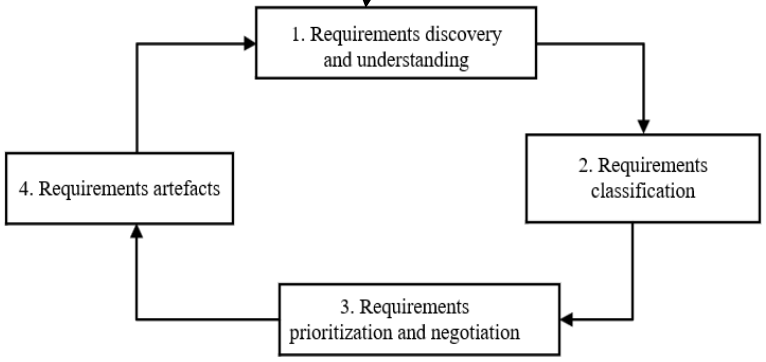


Table 4.11

Initial conceptual framework

Requirements prioritization decision-making scaling agile levels adapted from (Aurum & Wohlin, 2005)	Requirements prioritization as a part of elicitation and analysis adapted from (Sommerville, 2009)
Portfolio level	
Segment level/Tribe level	
Team level	

4.4.4 Cross-case analysis

Cross-case analysis typically involves identifying commonalties, contrasting, or aggregating the findings of studied case organizations in order to enhance the generalizability, robustness, and applicability of findings. The salient categories that emerged as an outcome of cross-case analysis (discussed in Chapter 7: Section 7.8) guided me in refining some of the existing categories that emerged from the individual case analysis. One of the significant changes, was made in the salient category middle decision-making level, which is termed as *segment level* in MEL organization, *tribe level* in AKL organization respectively. In the cross-case analysis, I adopted the term ***domain level*** to denote the middle decision-making level, due to the commonalities in terms of process followed on the priority of requirements in the studied case organizations (discussed in Chapter 7). The salient categories that emerged as an outcome of cross-case analysis, guided me in evolving the conceptual framework of requirements prioritization in SADSD.

Following are the salient categories that emerged as an outcome of cross-case analysis (explanation of these categories is provided in Chapter 7):

- Domain level
- High-level requirements
- Intermediate-level requirements
- Low-level requirements
- Boundary spanning mechanisms
- Inter-iteration prioritization
- Intra-iteration prioritization
- Integrated agile RE practices

4.5 Threats to validity

Validity deals with the 'truthfulness' of research results. The set of principles proposed by Klein and Myers (1999) were followed to ensure the validity of research.

- the principle of hermeneutic circle
- the principle of contextualization
- the principle of interaction between the researcher(s) and the subject
- the principle of abstraction and generalization
- the principle of dialogical reasoning

- the principle of multiple interpretations
- the principle of suspicion

The principle of hermeneutic circle: the phenomenon of interest (i.e., requirements prioritization process in SADSD) has been empirically investigated from the raw data within the case study (e.g., concepts, categories) and that further guided me in developing the conceptual framework of requirements prioritization in SADSD.

The understanding of the phenomenon of interest, improved by moving back and forth between phases of this research study. Data analysis was performed in various phases that includes within case analysis, where individual case organizations' data was analysed without any pre-existing framework in mind. Literature support was undertaken to structure the findings that emerged from the individual case analysis and an initial conceptual framework was developed into cross-case analysis, to aggregate the findings of studied case organizations, redefine the salient categories that emerged from the individual case analysis wherever needed, that further guided me in building the conceptual framework of requirements prioritization in SADSD.

The principle of contextualization: Thick rich description (e.g., historical background) of the studied organizations is provided by following the principle of contextualization in order to provide clarity to the audiences regarding how the current situation of the phenomena of interest under investigation, emerged. Petersen and Wohlin (2009) guidelines were followed to provide the thick rich description of the studied case organizations (MEL organization context description provided in Chapter 5, AKL organization context description is provided in Chapter 6) to ensure other researchers can replicate a context approximating this study.

The principle of interaction between the researcher and the subject: continuous interaction had been maintained with the study participant(s) and the interviewer throughout the research period. Some of the examples include the main round of interviews, follow up interviews, getting clarification on the findings from the participants via email wherever needed.

The principle of abstraction and generalization: conceptual framework of requirements prioritization in SADSD, has been built by following the principle of abstraction and generalization, by studying the two case organizations. Cross-case analysis was performed that involves identifying commonalties, contrasting or

aggregating the findings of studied case organizations in order to enhance the generalizability, robustness, and applicability of findings.

The principle of dialogical reasoning: the emerging conceptual framework of requirements prioritization in SADSD has been built via various cycles of revisions that involves individual case analysis, literature support (deductive analysis), and cross-case analysis. Constant comparison was adapted at each step of data analysis that helped me in defining and re-defining the emerging codes, concepts, categories, wherever needed. In addition to this, member checking with study participants was sought wherever needed. For example, this meant following up with participants either via email or informal interviews wherever possible to get further clarification on the findings from them. Expert opinions were also undertaken by having discussions and feedback from the supervisors, relevant researchers and adjunct professors of AUT.

The principle of multiple interpretations: data was collected from the diverse participants to attain the aim and strengthen the findings. Semi-structured interviews were conducted to investigate the phenomenon of interest from all possible roles, who are typically involved in decision-making on the priority of requirements in SADSD. In order to deal with this, kick off meetings, with AKL organization in March 2020, and with MEL organization in September 2020, were held. The managers of both participating organizations, the director of engineering in MEL organization, and tribe lead in AKL organization, discussed the aim and objectives of research and to understand the development process of the participating organizations. Purposive sampling (Barbour, 2001) was used to select the potential participants for this research study. First, I prepared an initial list of potential participants: product owner, product manager, team leader, chief product owner, chief technology officer, chief customer officer, enterprise architecture, UX/UI, developers, testers, architecture owner. Assisting, was studying the literature and using some of the popular scaling agile frameworks (e.g., DAD, Scrum @ Scale). This list was then discussed with the primary contacts of participating organizations and revised as per their feedback.

The principle of suspicion: multiple sources of evidence were employed intending to minimize the impacts of single-source interpretations and to strengthen the findings. For this research study, data collection categorizations proposed by Lethbridge et al. (2005) for the software engineering field, were adapted. These included (i) direct method-collected directly via study's participants through semi-structured interviews, which was

a primary source of data employed for this research study (Lethbridge et al., 2005). In addition to this, (ii) indirect method - raw data was collected via minimal interaction with participants such as recorded video/audio of decision-making events (Lethbridge et al., 2005) and (iii) independent techniques - already available data such as requirements artefacts and business-case), were also employed wherever possible to triangulate the data (Lethbridge et al., 2005).

4.6 Summary

This chapter has discussed the overall research process adopted for this research study and provided justification for the appropriateness of the interpretive multi-case study research method.

The next chapter describes the findings that emerged from the first (MEL organization) studied case organization.

5 Analysis and findings- MEL organization

This chapter presents the analysis and findings that emerged from the MEL case study. First, a company background is provided. Presentation of findings begin with their high-level overview followed by a detailed discussion on the decision-making levels and the requirements prioritization activities that emerged via analysing the MEL case study data.

5.1 Case overview

The context of the studied case organization is described via adapting the guidelines provided by Petersen and Wohlin (2009). The context facets include organization, product, process, people, and market which are described to provide the thick rich description of the studied case organization's context (as discussed below and summarized via Table 5.1).

- **Organization:** information that describes the studied case organizational structure, such as size (number of employees), hierarchical organizational model, organizational unit, distribution (Petersen & Wohlin, 2009).
- **Product:** information concerning the properties of software systems such as domain, product type, product size, maturity of product, and customization (Petersen & Wohlin, 2009).
- **Process:** describes any systematic approach or technology employed by the organization to develop software systems. Example: development method, distributed development, team size, method adoption (Petersen & Wohlin, 2009).
- **People:** this element describes roles of those who are involved and what are they responsible for in using the phenomena of interest. Example: personnel involved in the decision-making of priority of requirements (Petersen & Wohlin, 2009).
- **Market:** information concerning the current state of market and business in general. Example: setting, access to customers/end users (Petersen & Wohlin, 2009).

Table 5.1

Context description of MEL case organization

Context facet	**Context element**	**MEL case organization**
Product	Domain	IT asset management
	Product size	Large
	Maturity of product	Long-lived mature product
	Customization	Yes
Process	Characteristics of software development process	Scaling agile practices, Adapted DAD framework
	Start of method adoption	2015
	Distributed development	Globally distributed
	Team size	Up to 8 members in a team
Organization	Size	Large (1000+ employees)
	Number of development teams	25+ delivery teams
	Hierarchical organizational model	Yes
	Organizational unit involved in the study	Product and engineering
People	Roles and their responsibilities	Cross-functional decision-making team
Market	Setting	Business to Business
	Access to customers or end users	Business customers, key end users

MEL organization is one of the reputed international software vendors known for its asset management software products. The studied case organization has undergone a transition towards scaling Agile development via adopting the DAD framework over the period of 2015-2017. It was also continuously evolving scaling Agile practices at the time of conducting this research study as well. The MEL organization has 25+ engineering teams that are distributed over seven global sites that include Melbourne (Australia), Cheshire, Belfast (England), Itasca (USA), Bangalore (India), Hamburg (Germany), China. The studied case organization has its head office in Itasca (USA), and most of the engineering teams are in Melbourne (Australia). Therefore, participants were interviewed from the Melbourne and USA sites to understand the phenomena of interest (i.e., requirements prioritization process).

5.2 Overview of decision-making levels and associated requirements prioritization activities

This section provides a high-level overview of decision-making levels and the requirements prioritization activities that are performed at each of the scaling agile decision-making levels (structured via employing the initial conceptual framework discussed in Chapter 4: Section 4.4.3) in the studied case organization.

5.2.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021). In this section, ideas and requirements are used interchangeably to denote the potential requirements that could end up with big initiatives, or workstreams that generally get assigned to the teams or team of teams for the implementation. Sources of potential requirements are classified into two categories: internal sources and external sources (Damian et al., 2021).

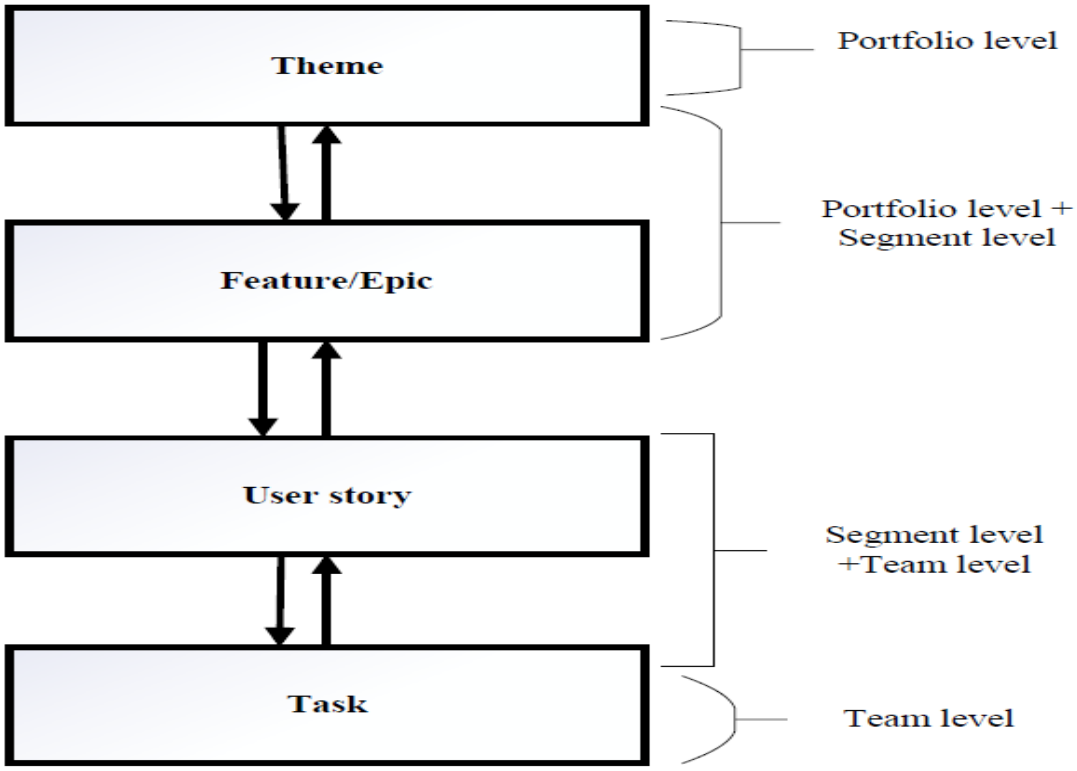
- Internal sources mean generating or collecting the potential requirements from/within the organization, such as product managers and engineers (Damian et al., 2021).
- External sources mean generating or collecting the potential requirements from outside of the organization, e.g., market, customers (Damian et al., 2021).

5.2.2 Requirements organization and classification

The case organization employed a hierarchy model to organize the requirements. Figure 5.1 shows the requirements hierarchy and their associated scaling agile decision-making levels.

Figure 5.1

Requirements hierarchy model and scaling agile decision-making levels



5.2.2.1 Requirements hierarchy model

The studied case organization had four levels of requirements hierarchy model which includes strategic themes, features/epics, user stories, tasks. At each level of the requirements hierarchy model, requirements vary in terms of requirement definition, timespan, decision makers. ***Strategic Theme*** represents the high-level business goals/strategies and resides at the top in the requirements hierarchy. The organization had six major themes that were set at the beginning of the year 2021. Themes are generally stable and have an influence on decision making at all the levels of requirements hierarchy. A theme is a collection of features that drive towards a common goal. The number of features that belong to each of those themes varies but there were in total 130 features that were proposed and distributed over 25 plus engineering teams to implement during the second quarter (Q2) of 2021.

The second level in the requirements hierarchy model is ***Feature***. A feature is something that provides concrete value to the customers and business and is generally independent of other features. A feature can be defined as a service that is requested by the stakeholders. A feature usually requires more than one engineering teams that can go up

to three engineering teams and can be implemented in a single release IR (i.e., three months).

At the third level, a Feature/Epic is decomposed into a ***user story***. A user story is the quickest sort of element deliverable through a thin pipeline that can give some valuable feedback and can be implemented in a single development IR (e.g., two weeks).

The fourth level is ***Task*** (e.g., create a new database table) that describes in technical terms what things are required to be done in order to claim a user story at the end of development IR.

5.2.2.2 Scaling agile decision-making levels

There are typically three scaling agile decision-making levels that emerged from the case organization's data that, includes Portfolio level, Segment level, and Team level.

Portfolio level: At the portfolio level, strategic themes that reside at the top in the requirements hierarchy model, are defined and the features that are required to realize a theme are formally signed off and/or prioritized by the portfolio management. As a result of this level, a portfolio backlog is constructed that contains themes and their associated features that are distributed over segments to implement them. The studied case organization has two product portfolios: on-premises product portfolio where packaged software is delivered to the customers; software-as-a-service product portfolio where services are offered to the customers via cloud. The software-as-a-service (SaaS) product portfolio was studied to investigate the research phenomena (i.e., prioritization of requirements across scaling agile levels).

Portfolio level and the requirements types that are associated at the portfolio level in terms of decision making in their priority are discussed in detail in Section 5.3.

Segment level: At the segment level, the product is divided into segment(s) that are generally termed as *program* in DAD framework (Ambler & Lines, 2012). The product that was selected for this research study had four segments. A segment is a kind of domain/solution area (e.g., cloud spend and migration segment) where teams are *generalizing specialist'* teams who typically have required skills and knowledge that are needed to solve a business problem that resides in a particular domain but they have general knowledge of the other domain and technologies as well (Ambler & Lines, 2012).

> *We have segments. A segment is like a technical product area. We have four segments. […] IT asset management, cloud spend and migration. I am working with IT asset management. [MEL_P5]*

Each segment had segment leaders from the product side as well as from an engineering side. Segment leaders generally act as a boundary spanner between the portfolio level and the segment level in terms of providing the shared understanding of potential workitems and/or requirements. As a part of portfolio level discussion, a segment leader from the product side provides business needs and a segment leader from an engineering side, provide technical needs that are emerging from their solution area/domain.

Role of Segment is twofold: First part contributed as a Product management

At the first part of segment level, features that are required to realize a theme and/or contribute to the theme(s) formation are defined, analysed, and submitted to the portfolio management via segment leaders for formal sign-off. Each segment had one to two Product Managers (PMs) who are typically responsible to define and analyse the *features* that are required to realize a *theme* and/or contribute to the *theme formation*.

Role of Segment: Second part contributed for engineering

At the second part of a segment level, features that are formally signed off via portfolio management are converted into Epics and Epics are broken down into user stories. Typically, the Product Manager (PM) is responsible for decomposing a Feature/Epic into user stories with an input from all possible stakeholders. As an outcome of this level, a *workitem list* (Ambler & Lines, 2012) is defined that contains user stories, spike, and tech-debt items. (Ambler & Lines, 2012). Segment level and the type of requirements that are associated at the segment level in terms of decision making in their priority, are discussed in detail in Section 5.4 and Section 5.5.

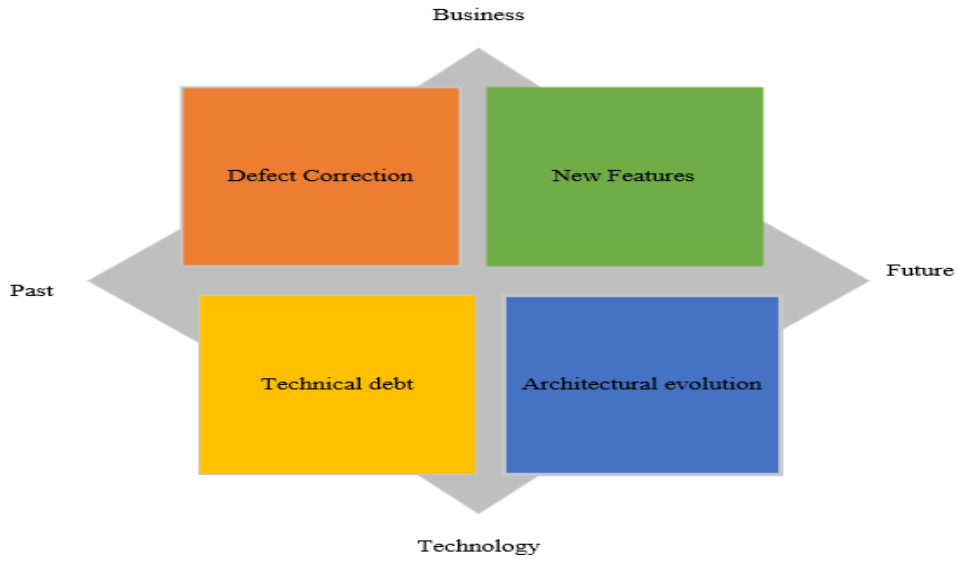
Team level: At the team level, user stories are broken down into further user stories if required, and task(s). Typically, the engineering team: Product owner (PO), Architecture Owner (AO), Team Leader (TL), Engineers, (Ambler & Lines, 2012) who are responsible for the implementation of a user story, decomposed the user story into further user stories and tasks wherever needed. Team level and the requirements' types that are associated at the team level in terms of decision making based on their priority, are discussed in detail in Section 5.6.

5.2.2.3 Requirements classification

Requirements are classified into four categories (i) new feature work (ii) defect correction work (iii) architecture evolution work (iv) technical debt work (as shown in Figure 5.2)

Figure 5.2

Requirements classification



New feature work and ***defect correction work*** represent the work that is performed for business, specifically for customers. On the other hand, ***architectural evolution*** and ***technical debt*** represents the work that is required to build to sustain the technology. ***Architectural evolution*** represents the work that is needed to be technically built into the product architecture in order to develop the new features that will be requested by customers in the future. However, architectural evolution work and new feature work are closely interdependent, which is generally a healthy interdependence that makes hard to distinguish them (e.g., customer platform application programming interface (API) - is that evolving an architecture or is that providing a new capability to the customers?).

Technical debt represents the gaps that are needed to be filled in order to have fully solid technical infrastructure. The right part of the Figure 5.2 represents the requirements- new feature work and architectural evolution, that are required to be built, to get new opportunities. The left side represents the requirements- defect correction, technical debt, that are needed to be done to address gaps relative to the work that has been attempted in the past.

The studied case organization typically in Research and Development (R&D), spend 40-60% on new features, architecture evolution 5-15%, defect correction 10-20%, tech-debt 25-30%. This is the recently adopted R&D spend allocation. In the past, R&D spend contributed highly towards new feature work (around 70-80% resources) and the remaining 20-30% resources allocated for others, tech-debt, architectural work.

5.2.3 Requirements prioritization and negotiation

During this activity, requirements are evaluated and the decisions on the priority of requirements, is made. Decision-making team, decision-making factors, trade-offs being made during decision-making, decision-making event(s), decision-making technique(s), are typically emerge at each of the scaling agile decision-making levels as a part of requirements prioritization and negotiation.

5.2.4 Requirements artefacts

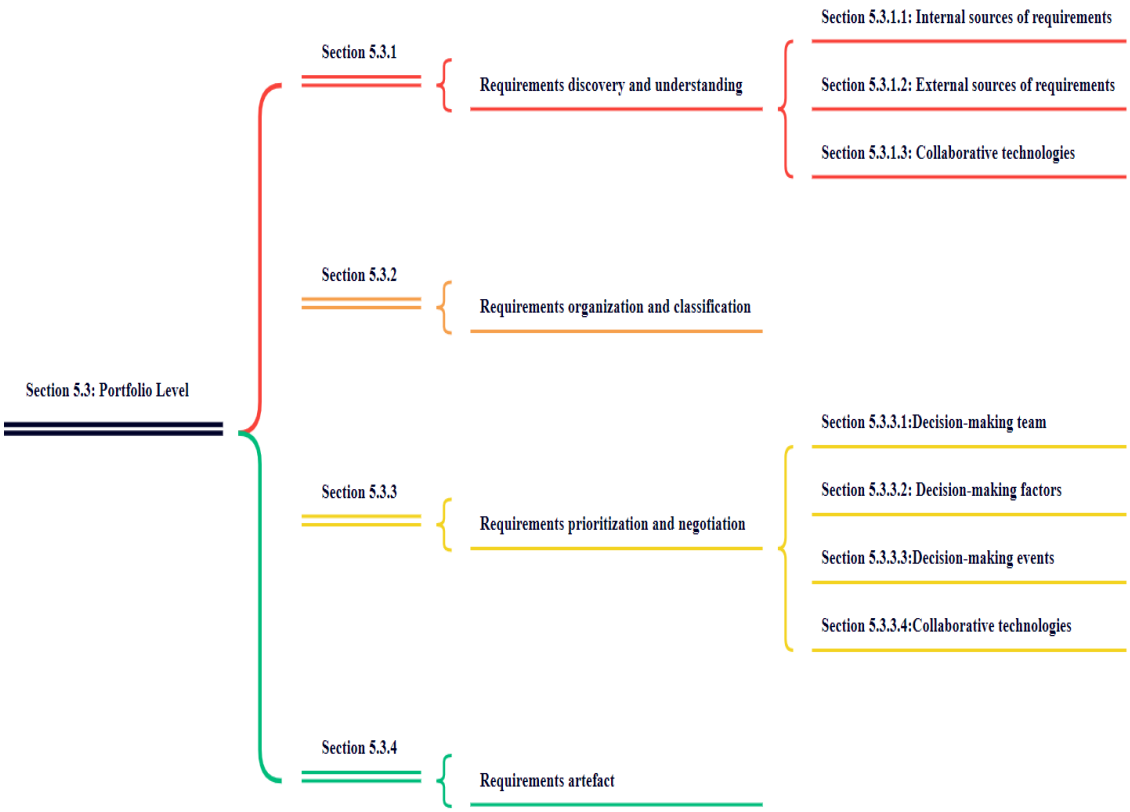
During this activity, requirement artefacts (e.g., business-case, storyboard) are produced to capture relevant information that typically serve as inputs during decision making on the priority of requirements and/or shared with delivery teams to provide them with required information for implementation.

5.3 Portfolio level

In the case organization, the portfolio level is the level where requirements are typically defined and analysed in the form of the organization's strategic objectives or themes, that generally influence the product that an organization ought to develop. In addition to this, requirements (i.e., features discussed in Section 5.4), that are posed on the product, are formally signed-off at the portfolio level by the portfolio management and ensure alignment of requirements (i.e., features) with the organization's strategic objectives.

Figure 5.3 shows the mechanisms that emerged at the portfolio level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 5.3
Snapshot of activities discussed at the portfolio level



5.3.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021).

The following are the sources of requirements.

5.3.1.1 Internal sources of requirements:

Within the portfolio level, typically the portfolio management is the main source of potential requirements. The portfolio management consists of senior leaders that include Senior Vice President (SVP) engineering, SVP product, SVP architecture, SVP UX, Segment leaders from the product side as well as from the engineering side. Where, SVP architecture brings the strategic technical need across the product and SVP engineering brings the resourcing perspective. SVP product provides overall business perspectives, the SVP UX brings the customer voice while Segment leaders (from product as well as engineering) provide emerging business facing and technical needs within their segment.

> *So, […] includes SVP product, [….], Segment leaders from the product as well as from an engineering, SVP UX, SVP architecture. SVP product provide overall product priorities, SVP architecture presents technical direction, Segment leaders present ideas that are emerging from their area.[MEL_P5]*

However, potential requirements also emerged from the members of other levels and/or teams such as engineers, support team, UX/UI team, customer success managers, product managers, sales team, and marketing team.

> *they definitely come from everywhere. they come from engineers, they come from product, they even come from senior leadership. [MEL_P3]*

5.3.1.2 External sources of requirements:

Generally, the market, comprising competitors, customers and partners, are the external sources of potential ideas/requirements (detailed discussion is in Section 5.4). At the portfolio level, SVP product and SVP UX typically collate the emerging business and customer needs respectively.

> *It's mainly SVP product and SVP UX who presents business and customer needs. [MEL_P9]*

Techniques that are employed to gather potential ideas/requirements from the external sources are not isolated to the portfolio level only. The same technique using., customer

interviews, can be employed at the other levels (e.g., segment level, team level) and/or teams (e.g., support team, UX/UI team, sales, and marketing team), to discover potential customers' ideas/requirements. Potential techniques that are adopted by the case organization to gather potential ideas/requirements from the external sources, are discussed in Section 5.4.

5.3.1.3 Collaborative technologies

Collaborative Technologies (CTs) - AHA[8] is a formal place to capture potential requirements that are emerged at the portfolio level (Daniels et al., 2015).

> *AHA, we use AHA tool at the portfolio level. [MEL_P2]*

5.3.2 Requirements organization and classification

As discussed in Section 5.2.2, the case organization employed a four level requirements hierarchy model.

Themes*→ *Features/Epics → User stories →Tasks

Primarily ***Strategic themes*/themes** belongs to the portfolio level in terms of decision making about their priorities. In addition to this, requirements (i.e., ***features)*** that are needed to accomplish the strategic themes are formally signed-off at the portfolio level.

As discussed in Section 5.2.2, the case organization has classified the requirements into four categories that includes new feature work, defect correction work, technical debt work, architectural work. Typically, the requirements, such as feature work, architectural work, defect correction work, and technical debt work that requires significant investment and/or effort (e.g., two-three months of work) required portfolio level discussion of decision making on their priorities, are generally termed as Features.

> *But the features are the ones because we are going to do huge investment. [MEL_P9]*

> *but there can be larger feature sets or strategic initiatives that need a higher-level decision. [MEL_P11]*

> *So, if it's something that's major or big for example 2-3 months work, which will require significant investment, you need to have portfolio discussion for it [MEL_P1]*

[8] Information about AHA can be accessed via this link: https://www.aha.io/

5.3.3 Requirements prioritization and negotiation

5.3.3.1 Decision-making team

Portfolio management, which is a cross-functional decision-making team, works collaboratively and continuously in deciding the strategic themes. Portfolio management consists of senior leaders that include SVP engineering, SVP product, SVP architecture, SVP UX. Also involved are Segment leaders (from the product side as well as from the engineering side) where SVP architecture provide strategic technical direction across product and SVP engineering provide resourcing perspective; SVP product provides overall business perspectives; SVP UX brings the customer voice; Segment leaders (from product as well as engineering) who provide emerging business facing and technical needs within their segment. Below are some of interview snippets that are used to support the above findings.

> *So, the formal process involves the delivery heads (SVP engineering) and the product management heads (SVP product), SVP UX so they would be part of the discussions. The product from the perspective of OK overall what our priorities and our overall direction and engineering from resourcing perspective and UX come up with customers' voice. [MEL_ P1]*

> *So, product management [i.e., SVP Product], user experience [SVP UX], engineering [SVP engineering] and their direct reports [i.e., Segment leaders] get together […]and go through each area of our portfolio [….] and we decide on the roadmap for that quarter of which are the ideas that will actually execute on [….] [MEL_P11]*

> *So, the theme formation includes SVP product, themes leaders, Segment leaders from the product as well as from an engineering, SVP UX, SVP architecture. SVP product provide overall product priorities, SVP architecture presents technical direction, Segment leaders present ideas that are emerging from their area. [MEL_P5]*

Each of the themes that are set by the portfolio management have theme leaders from the product side as well engineering side who are generally responsible for providing all the facts/data/information during decision-making. Themes' leadership role is basically a volunteer role by a person who wants to take the responsibility. Often the senior leaders, Segment leaders and SVPs, occupy the role of theme leader. In most of cases, there is a one-to-one relationship between the theme leader and the theme. However, if required, one person can be responsible for more than one theme.

> *So all these theme have theme leaders who will provide all the required information during decision making […] The role is purely voluntary but you know we have segment leaders, they are the one who generally occupy this role. [MEL_P5]*

5.3.3.2 Decision-making factors

In the studied case organization, business value is the main prioritization driver employed to make decisions on the strategic themes. Strategic theme(s) that provides the biggest opportunities to their business and their customers, are typically selected as a high priority theme(s).

> *I think this development strategy is a different level of prioritization. But it's same set of factors like business value we look for in terms deciding their priority. [MEL_P7]*

> *we need to understand what our market opportunity is […] look for biggest opportunity. [MEL_P2]*

5.3.3.3 Decision-making events

The following are the decision-making events that occurred at the portfolio level.

Strategic Portfolio sync-up: this event generally occurs on a quarterly basis, where portfolio management collaboratively decides strategic themes.

> *We have a strategic portfolio sync-up every quarter to set strategic themes. Normally it organized at least two weeks before the big room planning. [MEL_P5]*

Bigroom planning: This event also occurs on a quarterly basis, which is a five-day event, but the scope of this event is broader than the strategic-portfolio sync-up. During this event, strategic themes and the features that are needed to accomplish strategic themes, are formally signed-off by the portfolio management and communicated to the engineering team for implementation.

> *We do a roadmap review every quarter. [MEL_P4]*

> *So ideally, it's an ongoing process. In practice, we do have to sync them four times, right now it's been every quarter. [MEL_P16]*

However, a decision on the priority of requirements (i.e., features), is made ahead of the *bigroom planning event*. The planning for the upcoming quarterly bigroom planning, typically starts in the fourth development IR of current release IR. This event typically yields shared understanding of requirements (i.e., strategic themes, features) among product and engineering, visualizing and negotiating the priorities and dependencies across segments.

> *This quarterly planning is just communicating the plan. We call it bigroom planning. But we do adjustment as well wherever needed. It's like getting understanding of what we have to do in the coming quarter. […] this event is a good place that helps in highlighting the hidden dependencies as well. [MEL_P1]*

> *The planning is a continuous process but normally gets more focus towards the end of current quarter. […] normally it starts in the 4th iteration of current increment. [MEL_P12]*

Portfolio sync-up: is typically organized on a monthly basis for check-ins that generally provide the visibility on the progress of requirements (i.e., features), towards achieving business objectives/strategic themes, removing impediments, addressing dependencies and any change in the priorities of requirements.

> *And I think there are check-ins on a monthly basis […] they're working on it all throughout the quarter in terms of just moving things around in adjusting things based on, you know, things move pretty fast around here, right. So you might have to add a feature, take something off or, you know, sometimes we learn things are harder than we thought they were, and you have to make adjustments so those happen. [MEL_P4]*

5.3.3.4 Collaborative technology

At the portfolio level, AHA is employed as CT that provide visibility across the entire portfolio and enable tracking the progress of which stage a specific feature is in i.e., discovery/ready to develop/in development. The integration feature of AHA with other CTs such as Jira[9] and Confluence[10] enable traceability and shared understanding of requirements across all the scaling agile levels.

> *All the tools that we use to manage requirements like our product management tool AHA, engineering tool Jira, documentation tool - confluence are integrated. We can easily trace progress of a particular feature, we can check who is working on what and things like that. [MEL_P11]*

5.3.4 Requirements artefact

The following requirement artefacts are built at the portfolio level that are specified via CTs- AHA, Confluence, Spreadsheet, PowerPoint (Daniels et al., 2015).

***Theme* (an electronic template configured via CTs- *AHA*):** a theme is a requirement artefact that is built at the portfolio level and connects the portfolio with the strategic territory. A theme is generally a business goal that an organization wants to achieve in a certain time-period, such as one year. A theme is generally stable that has an influence on decision making at all levels of the requirements hierarchy. A theme is generally defined in the form of a phrase, usually one or two lines, see Table 5.2.

An electronic template which is configured via CTs *AHA,* is followed to specify the themes. However, CTs- *PowerPoints* (Daniels et al., 2015) slides are also maintained

[9] Information on Jira tool can be accessed via https://www.atlassian.com/software/jira
[10] Information on Confluence tool can be accessed via https://www.atlassian.com/software/confluence

specifically to be used during decision-making events (e.g., bigroom planning) for knowledge sharing.

> *We use AHA template for themes. [MEL_P8]*

> *Themes are like one year business goals and our portfolio level people decide them [….] so AHA is the tool that they use, its like a template in AHA which they need to fill like business value, some justification. But sometime they use PowerPoint slides as well specifically during our bigroom planning which is happened every quarter, its like a presentation through that they highlight what will be our focus in this whole year. [MEL_P10]*

Table 5.2

Sample of strategic theme from MEL organization

Strategic Theme Example[11]
Theme: Make SaaS Manager more competitive. The why's • Because it's the market that is growing the most • Because sales have no confidence in SaaS Management right now

Portfolio backlog (electronic artefact configured via CTs- *AHA*): portfolio backlog that contains a priority list of requirements (i.e., Themes and features) that is built at the portfolio level. CTs- *AHA* is a formal place employed by the studied case organization to build a portfolio backlog. However, the portfolio backlog is maintained in the form of a *Spreadsheet* as well which is specifically used during decision-making event (i.e., bigroom planning). The main reason behind the adoption of a *Spreadsheet* is that *Spreadsheet* enables synchronous editing during decision-making event that is specifically helpful when the members of decision-making team are distributed. But the main drawback is that the decision-making team need to maintain two artefacts which results in duplication of effort.

> *We have portfolio backlog that get refined on quarterly basis. We call that event is bigroom planning. [MEL_P10]*

> *we started using a tool called Aha. And it lets us create, you know, feature cards and backlog [portfolio backlog] and things like that. [MEL_P3]*

> *we have spreadsheet as well that contains all the features that we are going to implement in the coming quarter. But this spreadsheet is not a formal backlog, in AHA we maintain our backlog. […] downside of it [maintain spreadsheet] is, it creates extra work for us. But we still use it specifically during our quarterly planning because*

[11] This example is taken as a snapshot from the recording of decision-making event

I think you are already aware about that we are globally distributed team, editing is easy for them during that time, you understand what I am trying to say. [MEL_P10]

Confluence: CTs- *confluence* is employed to maintain documentation of requirements (e.g., for a detailed discussions and/or analysis on requirements such as recordings of discussion with customers), that helps in building the shared understanding of requirements. The integration feature of CTs *confluence* with *Jira* and *AHA* provide traceability of requirements.

The outcomes are recorded in confluence page as well. [MEL_P1]

All the tools that we use they all are integrated. [MEL_P4]

5.4 First part of Segment level

Introduction: In the studied case organization, the product is typically classified into segments that are generally termed as programs, in Disciplined agile delivery (DAD) framework (Ambler & Lines, 2012). A Segment is a kind of domain (e.g., cloud spend and migration segment), where teams are generalizing specialists' teams in a particular domain. The main reason behind this adaptation was to manage dependencies across segments.

We have segments. A segment is like a technical product area. We have four segments. […] IT asset management, cloud spend and migration. I am working with IT asset management. [MEL_P5]

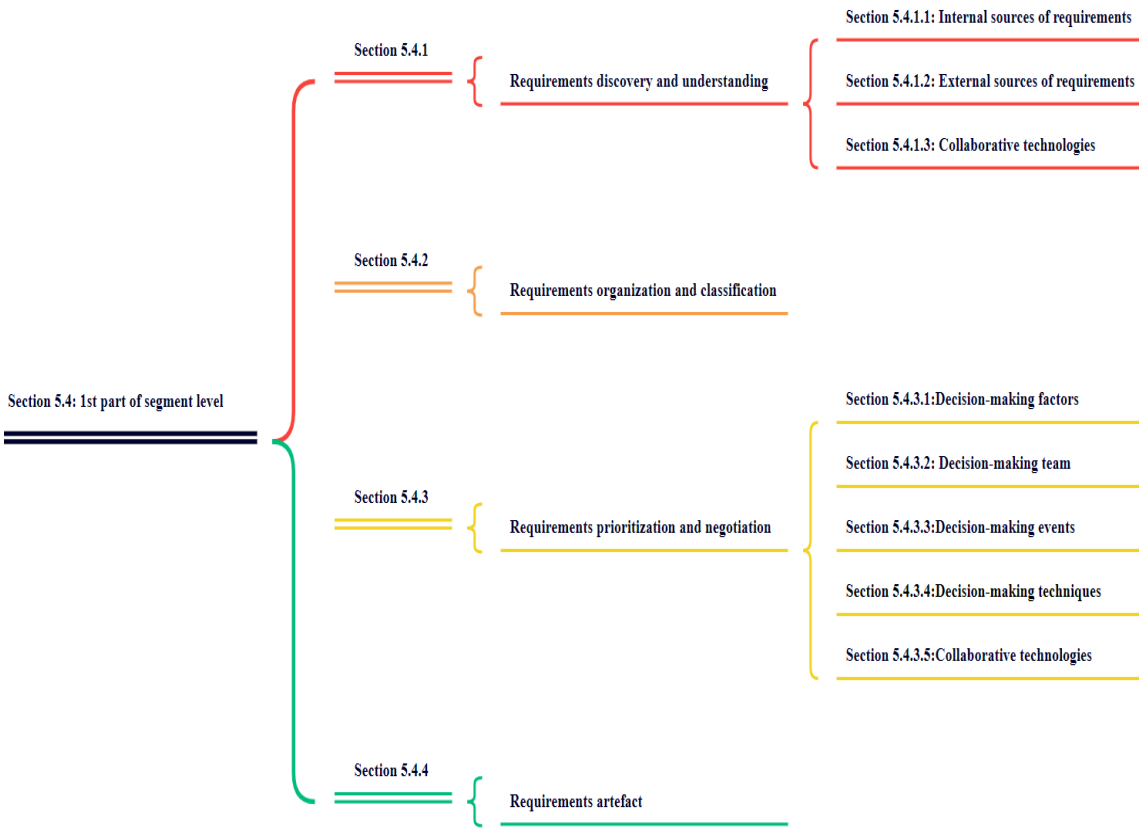
Role of segment is twofold, in the first part of segment, potential requirements (i.e., features) are defined and analysed, and submitted to the portfolio management via segment leaders for formal sign-off. Segment leaders act as boundary spanners in terms of communicating the requirements between the portfolio level and segment level. Segment leaders are the one who typically communicate any change in the decided priorities if it occurred on behalf of portfolio management and vice versa. Each segment typically has one segment leader from the product side as well from the engineering side.

Figure 5.4 shows the mechanisms that are emerged at the first part of segment level in terms of decision-making on the priority of requirements that are structured via

employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 5.4

Snapshot of activities discussed at the 1st part of segment level



5.4.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021).

The Following are the sources of requirements.

5.4.1.1 Internal sources of requirements

Typically, the PMs, program architecture owner, director of engineering, director of UX/UI wherever needed, generally work collaboratively to collate and/or understand the potential ideas/requirements that are imposed on the product, but PMs are the dominant internal source of ideas/requirements. In addition to this, potential requirements/ideas are collated from the other levels and/or teams, that includes portfolio level, team level, support team, sales, and marketing team.

> *Predominantly ideas come from our product management team. And they are the ones who feed us our what we call the roadmap right. [MEL_P8]*

> *We obviously talk to internal stakeholders as well, engineering/architecture, product managers, and UX. those are the three main source of ideas. [MEL_P17]*

> *they definitely come from everywhere. they come from engineers, they come from product, they even come from senior leadership. [MEL_P3]*

Practices to collate requirements/ideas from the internal sources: Various practices are employed by the case organization to collate potential ideas/requirements, as explained below. Participant MEL_P3 reports that practices that are used to collect ideas from internal stakeholders are least formal.

> *Probably that's the least formal ideas that get captured. [MEL_P3]*

Run hackathons: the case study data revealed that occasionally hackathons are run to collect ideas from engineers. Hackathons are typically conducted on an annual basis that are either conducted at the head office, or Melbourne. PMs and engineers are the main actors who generally participate in hackathons.

> *the hackathons are not as frequently, but hackathons I think we have at least one every year major events.[…] typically conducted in at our headquarters in the US, or here in Melbourne.[…] its mainly product management and engineers [MEL_P1]*

Engineering and support summit: The company has an annual engineering summit involving around 50 to 60 engineers, (a couple from each team), that is usually held for three days to collect engineering ideas.

> *we had actually first sort of Engineering Summit […] into this three-day conference summit at least once a year. […] I think its one or two members from each team [MEL_P6]*

Reward incentives and recognition: The company encourage staff to bring innovative ideas. Staff get rewards and recognition for bringing innovative ideas. The company has a quarterly reward system.

> *company encourages that so people are empowered to think about it. and they get rewarded. So, we have a quarterly reward. [MEL_P9]*

Run workshops: Subject matter experts (SMEs) are typically collaborating via ad-hoc workshops to generate innovative ideas. However, participant MEL_P16 reports that they are in-progress to conduct workshops more frequently.

> *And then there's also things we do internally where we're going to do, we did a few workshops last year and we're going to do more next year where they*

[SMEs]brainstorm around, you know what could we do to innovate in this area that would be different. [MEL_P16]

Casual conversation with staff: Participant MEL_P3 mentioned that sometime ideas are generated via casual conversation with staff (e.g., engineers).

You know sometimes it's just the conversation. [MEL_P3]

Business strategy: Participants MEL_P9 and MEL_P17 report that business strategy is another major source for the segment level that can yield ideas/requirements.

internally we based on that we have our own product strategy and so where we want to move it. Ideas come from there as well. [MEL_P19]

so, ideas are really depending upon our strategy as well. [MEL_P17]

5.4.1.2 External sources of requirements:

As indicated in Section 5.3.1, market (e.g., competitors), customers, partners are typically the external sources of potential ideas/requirements.

Practices to collate requirements/ideas from the external sources:

Market research: The case organization employs various techniques to do market research/product research. The case study data indicated that typically product management group does product research/market research. PMs collaborate with industry analysts to do competitor analysis (analyse five main competitors) and track change trends/track market trends. PMs typically conduct *interviews* with industry analysts (e.g., Gartner, Forrester) and the yearly report that is produced by industry analysts in their respective areas is also analysed to identify gaps. They track market trends, competitors' product offerings, to capture emerging needs of potential customers.

It can be industry analysts telling us companies like Gartner and Forrester, which put out reports in our respective areas every year. [MEL_P1]

So we as product managers, continuously do product research for our product areas as well. find out what is changing. [MEL_P1]

They [Product management] are also involved with you know people like Gartner to sort of work out where the gaps in relation to our market segment, our competitors, that kind of thing. So the ideas come from that also. [MEL_P6]

So all those requirements are captured as well during those interviews with the Gartner specialists. [MEL_P12]

Partners: Participant MEL_P5 asserted that potential ideas/requirements are generated from the partners, such as development partners, with whom product components are developed as well.

> *We have partners as well. So some of the ideas come from them as well. [MEL_P5]*

Customers: Customers are the dominant source of ideas/requirements. Getting ideas/requirements from the customers is a continuous process. Customers' ideas/requirements are typically communicated via the customer services team, sales and marketing team, support team, UX/UI team, PM group. However, PM group is typically responsible to understand and/or collect customers' needs. PM group generally spend 40% to 50% of their time to understand and evolve customers' needs.

> *So it really can come from anywhere but you know predominantly, they come from our customers. [MEL_P7]*
>
> *our product management group have been told to spend 40 to 50% of their time talking to customers. [MEL_P6]*
>
> *that comes in either via their like customer service manager. Or like rep sales person or UX group or anyone who is the part of organization. [MEL_P8]*
>
> *I guess it is closer to the customer that there's the product management team whose job is really to talk to customers about these things about what their needs are. [MEL_P13]*
>
> *I mean they [PM] coordinate with customers on a regular basis to try to get an idea of, you know, what they need, what they want. [MEL_P17]*

The following are the practices employed to collate customers' ideas/requirements:

- *Run focus group:* Frequent focus group sessions are organized with customers to collect their ideas/feedback. The studied case organization data indicated that generally the people (e.g., customer service team, product management group) who are closer to the customers organize focus group session to collect their ideas/feedback.

> *I think marketing might have done some [focus group] with customers. [MEL_P3]*
>
> *sometimes it's product managers. sometimes it's sales and marketing. it could be customer managers conduct those focus groups. [MEL_P1]*

- *Customers Advisory board:* Customers advisory board is another mechanism where typically the PMs collaborate with a group of customers in order to get and/or understand their needs.

  There are issues that come up at Customer Advisory Board. [MEL_P11]

- *Customer portal:* The customer portal is a recently adopted approach which is an online customer community operated via CTs a product management tool AHA, to get customers' feedback/ideas. The customer portal is a mechanism through which customers can submit their ideas/feedback by themselves. Participant MEL_P4 reports that customer portal helps in evaluating the potential ideas/requirements (i.e., measuring the impact on customers). This further helps in filtering the most valuable ideas, that receive highest voting, from the weak ideas (i.e., ideas that require critical discussion).

  there's issues that come up in our community, where people can post and our new tooling for product management which used to be product board before and now the tool we use is called Aha. It has an area for idea submission. [MEL_P11]

  which is things that we think need public input. We will surface those on customer portal and customers can actually vote right And so items that float to the top are ones that we would take a more serious consideration of and items that get no reception or ones that we will need to make a more critical eye on decide like is there any business value here or any other value. [MEL_P4]

- *Business review:* A business review is a formal place to capture ideas from customers. A business review is typically conducted on a quarterly basis with the biggest customers, to understand their need/feedback/suggestions.

  So that's you know, four times a year [Business review] as well as they are, the more formal processes that we have in place. [MEL_P5]

- *Run customer interviews:* Frequent customer interviews, probably once a week; 10-15 customers in a period of two or three weeks, are conducted to understand their needs. However, frequency of interviews usually depends on the potential emerging requirements.

  From what I have come across so far through various interactive discussions I think they are more sort of need basis. [MEL_P12]

  but primary source of ideas are usually customer interviews and that's what I rely on for the most part to get ideas. [MEL_P12]

- *UX research studies:* some of the techniques include co-creation workshops, usually held on a quarterly basis, usability testing usually weekly, validation

testing usually weekly, discovery research quarterly, contextual inquiry observing customers' behaviour in their natural setting - usually involves spending one full day, and design workshops to understand and/or collect customer needs/ideas/feedback/suggestions.

> *I would say it on UX research studies. So that includes Co-creation workshops, usability testing, discovery research. [MEL_P17]*

5.4.1.3 Collaborative technologies

At the first part of segment level, AHA is a formal place to capture potential requirements/ideas.

> *we've currently adopted Aha as a version two of our like having a formal product management tool. [MEL_P7]*

5.4.2 Requirements organization and classification

As discussed in Section 5.2.2, four levels of a requirements hierarchy model is adopted by the case organization.

Theme(s) → ***Feature(s)***/Epic(s) → User story(s) → Task(s)

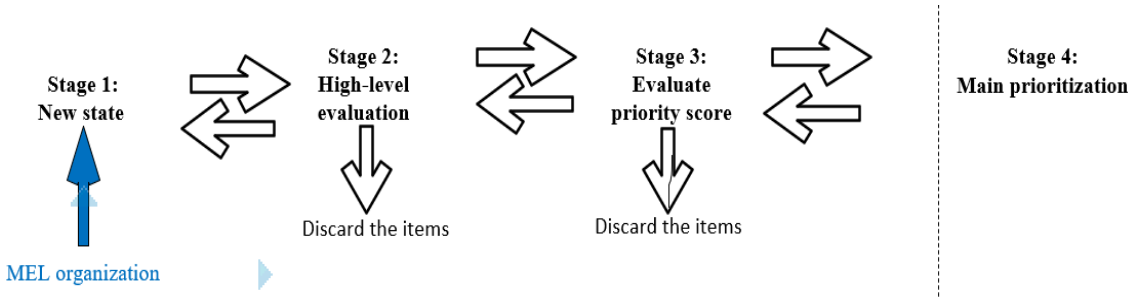
In the studied case organization, the potential requirements (e.g., feature work, architectural work, defect correction work, tech-debt work) that requires significant effort and/or investment (e.g., two-three months of work) are typically termed as features. In the first part of segment level, ***features*** are defined, analysed, and submitted to the portfolio management via segment leaders for formal signoff on their priority.

5.4.3 Requirements prioritization and negotiation

As mentioned in the above Section, initial decision making on the candidate requirements or features, that emerge from each of the segments and/or contribute to themes formation, is typically made in the first part of segment level. This initial decision-making goes through various stages, as shown in Figure 5.5 before submitting them to the portfolio management for formal decision making. At each of the decision-making stages new potential requirements can emerge that are initially pushed into Stage 1. In addition to this, the potential requirements can be pushed back to the previous stage for further analysis if the provided decision-making information is not sufficient.

Figure 5.5

Feature decision making stages



Stage 1: All the potential requirements that are collated via requirements discovery practices (described in Section 5.4.1), are typically at brand new state during Stage 1.

Stage 2: The potential ideas/requirements are evaluated at the high level typically against organizational strategy (i.e., business goals/themes). As an outcome of this stage, either the candidate requirements will be discarded, or will be moved into Stage 3 for further evaluation. However, a priority list of requirements will not be produced in the Stage 2 but the candidate requirements that are valuable for the organization will be filtered out typically based on their alignment with the organizational strategy i.e., themes.

> *we've really become very intentional about what is our mission and strategy, and so the good news is, at least from my standpoint, is as new ideas are coming in We are able to at least quickly say no if it's really something that there, it just does not align with what we're doing. We can quickly say no, and that helps clear out some of the noise. [MEL_P8]*

Stage 3: Candidate requirements that are filtered as a result of Stage 2 are evaluated, based on the whole set of decision-making factors, that include business value, opportunity and risk reduction, urgency, effort which are described in Section 5.4.3.1. At this stage, a formal name is assigned to the candidate requirement(s), either simply labelled as a feature (feature means it is ready for implementation) or discovery feature (means further evaluation is required to make a decision on the candidate feature).

> *Once the potential idea pass high-level evaluation, we apply whole bunch of factors to evaluate the score. These factors are like what the business value is, what are the risks, effort, what the opportunity is., you know stuffs like that. This score again helps us to clear noise. [MEL_P18]*

> *In the backlog, we just not only have features. We have discovery features as well. You can differentiate them like implementation features and discovery features. Discovery feature means we need to explore that feature more, you know sometime we are not 100% sure, should we proceed with the feature or not. In that case, we normally create a discovery feature to investigate that. [MEL_P13]*

As an outcome of Stage 3, either the candidate feature will be accepted, or it will be discarded. In the latter case, either the accepted feature will stay in the backlog till the priority is raised or the accepted candidate feature moves into Stage 4 for formal signoff from the portfolio management, either as a discovery feature or implementation feature.

Discovery phase (Discovery feature): Discovery phase is a recently adopted approach to get an in-depth understanding of a candidate feature and evaluate all the possible risks that might be associated with a potential feature. Possible risks that are evaluated as a part of discovery phase, include whether a feature is usable (i.e., feature idea is user centric), valuable (i.e., the feature idea delivers value), feasible, and technically viable. The case study data indicated that discovery phase is an optional phase. Not all the candidate features go through a discovery phase. The discovery phase is required for the candidate features that have a lot of uncertainty, e.g., solution alternatives are not present for a candidate feature. The main purpose of a discovery phase is to evaluate the viability of a candidate feature.

> *discovery phase is all about the understanding the problem, talking to customers, and understanding the scope. [MEL_P9]*

> *actually, the ideas that have huge uncertainty go through a formal discovery process. Discovery process is to explore the idea and confirm that it's valuable that the customers are multiple customers that are going to get value from that idea, and any risks that are associated with the idea, can we actually fulfil the idea so we go through a discovery process. It's really a recent change. [MEL_P5]*

> *so four kinds of risks that we look at. like value, technical feasibility, works with the business and then usability. [MEL_P16]*

As an outcome of discovery phase, either a candidate feature will be accepted or discarded. In the latter case, either candidate feature will stay in the backlog till the time priority is raised or, if accepted, move into Stage 4 for formal signoff from the portfolio management.

Stage 4: This is the stage which is typically performed at the portfolio level where formal decision-making happens on the candidate features that are emerging from all the segments and are allocated to the engineering for implementation. During this stage, Portfolio management typically validates the candidate features against the organizational strategy and may make decisions where there are constraints, such as deciding between two important features where they only have capacity to do one. But ultimately recommendations and priorities come from the engineering teams.

We do a roadmap review every quarter where features are formally signed-off. [MEL_P5]

This quarterly planning event what we call as bigroom planning, its like high-level validation of features the portfolio management basically do. But you know, we [engineering teams] are the one who actually give recommendation. [MEL_P11]

5.4.3.1 Decision making factors

The following are the factors that are considered for evaluating the priority score of requirements or features that require significant effort and/or investment:

Business value: Factors that are considered to evaluate the business value, include *importance for customers*- the value it is going to provide the customers, *importance for the business*- the alignment with product strategic direction, return on investment, and *brand reputation.* Potential features are rated between one to ten, where one represents the lowest value item and the highest number represents the most valuable item.

So is that's really a good idea- Is something customers want to pay us for and give us money for. [MEL_P16]

So the product managers need to have a sense of the value of whatever we're going to do to both the customer and the business. That could be a monetary value. And It could be just some other intangible value. [MEL_P4]

How does it help the business as a whole. The business values also goes back to the competitive analysis, right. [MEL_P3]

Participants emphasized that business value plays an important role during decision making, especially when more than one feature has the same importance, and the delivery team have a capacity to do only one. In such cases, a feature holding higher business value and customer impact, is generally given more importance than another.

And then we have another call around is aligned with the strategy, so you know for next year we have a strategy that's really around integration and building our sales platform. So, if this is an idea that's going to support that, then it's going to be a higher score […] I mean there's always going to be something that you know whether one is going to be more aligned with the strategic direction. [MEL_P16]

So, business value really helps in a situation when we have more than one items holding same importance. [MEL_P5]

The biggest thing is you know alignment, alignment with strategy. [MEL_P11]

Urgency: Urgency defines the 'shelf life' of a candidate feature. Urgency is driven by customer demand and adoption, competitive advantage urgency, dependencies where the feature delivery is needed to support upcoming capabilities. Similar to the business value,

urgency of potential feature(s) rated between one to ten where one represents the lowest time criticality and highest number represents the most urgent.

> *When it should be done. If not done up to certain time, then may be no point of doing it later. Basically, the customer demand and adoption drive urgency. [MEL_P9]*

> *Urgency is, you know, you could think of that as we need it in order to meet some sort of market review or we have a data privacy problem and we need to address it quickly or something like that. [MEL_P4]*

Risk reduction and opportunity: Risk can include *security risk* such as security vulnerability in one of their products, and *technical risk* or, technical debt impacting user experience that is significantly affecting renewals, sales and reputation. Also, technical debt significantly impacting development teams' ability to deliver future product enhancements, *product risk* (e.g., significant chance of losing market position).

> *Risk reduction (So maybe it's a technical debt item, product wants to allocate resources to You know, maybe refactoring something or maybe its security risk or maybe its if will not get implement our reputation will be affect) [MEL_P4]*

> *And also risk factor. risk can be it maybe security risk, tech-debt risk, product risk. [MEL_P2]*

On the other hand, consideration is critical in the case of assessing the opportunity of a candidate feature including new business opportunities, differentiator in the market, alignment with competitors in the market, potential revenue from new bookings, significant projected new customer growth, revenue retention for existing customer.

> *opportunity (so opportunity is going to be related to is just going to get us more sales). [MEL_P4]*

> *So is it a current customer? What's the renewal cost? Are they close to renewal? Is this somebody in the sales process You know, if knowing that they're going to have this will put them over the edge in the sales process. What's the cost of not implementing this idea right, Does it put us behind competitively. [MEL_P3]*

The process is to assign one value from one to ten taking into account both the risk reduction and the opportunity. Where one represents the lowest opportunity and/or risk and higher number represents the highest opportunity and/or risk.

Effort: Effort determines the complexity of feature(s). The effort scoring metric adopted as a part of the portfolio is a very rough T-shirt size *(*i.e., XL=8-9 IRs, L=6-7 IRs, M= 4-5 IRs, S=2-3 IRs, XS=1-2 IRs) estimate of the amount of time it could take for an entire team to implement the feature. An estimate as part of the portfolio, is treated as indicative and not the commitment from the engineering team.

I think it is T-shirt sizing at the product management level. it's XL, L, M, S, XS depending upon the piece of work. [MEL_P12]

T-shirt sizing. So in in just so we're on the same page it's we don't spend a lot of time on this. [MEL_P8]

then there's an effort score, which is how hard is this going to be. [MEL_P4]

5.4.3.2 Decision-making team

The decision-making team is a cross-functional team that makes initial decisions on the potential features. Typically, segment triad (PM, program architecture owner, director of engineering) and director of UX/UI, wherever required works collaboratively to evaluate the priority score of potential features where PM brings business perspectives, UX/UI brings the customers voice, Architecture and director of engineering together evaluate the solution alternatives and resource perspectives.

And they [PM, architecture, director of engineering and UX resource in certain cases] will decide which things are accepted, which things need discovery, and which things are, you know, an immediate decline. Like we're not going to do this. […] actually responsibility wise product manager usually come up with business need, UX/UI are the one who come up customers need, Architect and our engineering manager represents technology requirements, what are resources we required etc. etc.[MEL_P4]

5.4.3.3 Decision-making events

The decision-making team- segment triad, generally meet *fortnightly* to plan current features, future items, and take ideas through discovery. A *weekly sync up call* is also set up for check-ins about feature progress, any blockers, changes in decided priority, dependencies. However, ad-hoc discussion can also happen wherever needed. These decision-making events are attended by the segment leaders as well wherever needed to communicate any change in the decided priority on the behalf of portfolio management.

Yeah, at the segment level we meet [PM, architecture, director of engineering and UX resource in certain cases] on fortnightly basis in a sort of like a portfolio meeting where we go through current ideas and take them through that discovery process. [MEL_P5]

Generally, we meet fortnightly. But we have weekly check-ins as well. Sometimes we meet on ad-hoc basis, rarely it happens. but if required we do meet on an ad-hoc basis as well. [MEL_P7]

If any change happens from the portfolio management side, we have segment leaders who communicate that change to us. They do attend our meetings time to time, you know to see how things are progressing, any blockers and stuff like that. [MEL_P1]

5.4.3.4 Decision-making techniques

Determine relative priority: The priority score of features can be adjusted, especially when more than one item holds the same priority score and/or has the same importance. In that case, generally the benefits, the return on investment and customer impact of one over another, are compared.

> *And then, of course, there's relative priority that we need to look at, so it's more of a consultative discussion where they may ask questions- OK, how much more important do you think this is compared to something else that we're working on or looking to work on. [MEL_P1]*

Perform additional research if required: in the experience of the participants, sometimes consensus requires lots of negotiation, which is generally a healthy discussion during decision making on feature priorities. In such cases, typically additional research is performed to collect all the relevant facts/data/information in order to convince and/or satisfy other stakeholders. In addition to this, in such cases, if needed, generally product management takes an ultimate decision on feature priority.

> *If there's any gaps in based on the questions that get asked, you go back and do the additional research or get those data points (the missing data points) and bring it back to the team. [MEL_P1]*

> *but, you know occasionally there will be someone will have to make a call. Senior persons generally the SVP Product ultimately responsible for deciding based on what's been raised if people can't reach a consensus here. [MEL_P13]*

Weighted Shortest Job First Approach (WSJF): *WSJF* a key technique that is adopted by the case organization to evaluate the priority score of potential features. WSJF approach produces the priority score of potential features, based on the decision-making factors that include business value, opportunity and risk reduction, urgency, effort (an automated matrix configured via *CTs- AHA*). However, it emerges from the data, that the priority of potential features, is not totally based on quantitative weights (i.e., WSJF score). Participants asserted that qualitative weights (i.e., human judgement) also plays an important role during decision making on the features' priority.

> *but the idea is that we have a weighting system and then it's managed by people, not just by math. [MEL_P4]*

> *weighted shortest job first model. It's honestly a bit loose. [MEL_P7]*

> *It is not as objective as I would have liked it to be. [MEL_P11]*

> *The priority matrix is automated matrix. Its in our product management tool AHA. [MEL_P17]*

5.4.3.5 Collaborative technologies

In the first part of segment level, CTs- *AHA* an online software-as-service (SaaS) tool is adopted to track requirements or features priorities and progress which stage a specific feature, is in the Discovery/Ready to develop/In development. The integration feature of AHA with Jira and Confluence enable traceability and shared understanding of requirements.

> *they [Product management] are using AHA so ends up as feature cards in AHA. […] Yeah, there're different dashboards for different product. […] so it's all being tracked in AHA. [MEL_P7]*

> *All the tools that we use to manage requirements like our product management tool AHA, engineering tool Jira, documentation tool- confluence are integrated. We can easily trace progress of a particular feature, we can who is working on what and things like that. [MEL_P11]*

5.4.4 Requirements artefacts

The following requirement artefacts are built in the first part of segment levels which are specified via CTs- AHA, Confluence (Daniels et al., 2015)

Feature (an electronic template configured via CTs- *AHA*): A feature(s) is a requirement artefact which is defined to realize a theme and/or contribute to the formation of theme. A feature is something that typically provides concrete value to the customers and business.

Business-case (an electronic template configured via CTs- *AHA*): A high-level rough *business-case* is built to justify the potential feature, which is typically employed by the portfolio management during decision making. A business-case template is followed, that contains the following information: type of problem solving, benefits, return on investment, impact, successful criteria, minimal successful criteria, broad scope of the problem, out of scope, non-functional requirements, definition of minimum viable product/minimum marketable product, cost of delay, impact due to delay, have possibility to delay, identification of stakeholders, risks, cost, outcome, business opportunity, revenue retention for existing customers, effort, dependencies with other products.

> *Our business-case template is very comprehensive[…] While we could quickly make a decision or what we often call lean business case[…]so business-case would typically contain what kinds of benefits you can get, what is the return on investment, just generally, the definition of what the minimum viable or minimum marketable product is, stakeholders have to be identified. [MEL_P1]*

> *There'll be business cases for each of the features. [MEL_P13]*

We use a template and this template is configured in our product management tool AHA. So basically, what's the goal of the thing we're trying to achieve? What's the impact will be? What products will be, what benefit, any risks, outcomes. so basically, you need to get stakeholders for that Business-case. MEL_[P2]

Like why is it required, And what sort of impact it's going to have, etc. etc. [MEL_P10]

We have a template for the features in AHA. [MEL_P2]

However, the participants asserted that all the information that is mentioned above is not always present in the business-case. Facts/data that can sufficiently justify the potential feature need to be present in the business-case or 75% to 80% of the information is captured. The main purpose of business-case is the identification of key stakeholders who will own the feature.

So it's I mean, we never are able to get 100% ready business case. If we can get 75 or 80% of the information in, that's good enough to make a decision. [MEL_P1]

so basically, you need to get stakeholders for that Business-case. [MEL_P2]

so make sure all the stakeholders are listed as well. [MEL_P1]

Confluence: CTs - *confluence* is employed to maintain documentation of requirements (e.g., for a detailed discussions and/or analysis on requirements such as recordings of discussion with customers), that helps in building the shared understanding of requirements. The integration feature of *confluence* with *Jira* and *AHA* provide traceability of requirements.

The outcomes are recorded in confluence page as well. [MEL_P1]

All the tools that we use they all are integrated. [MEL_P4]

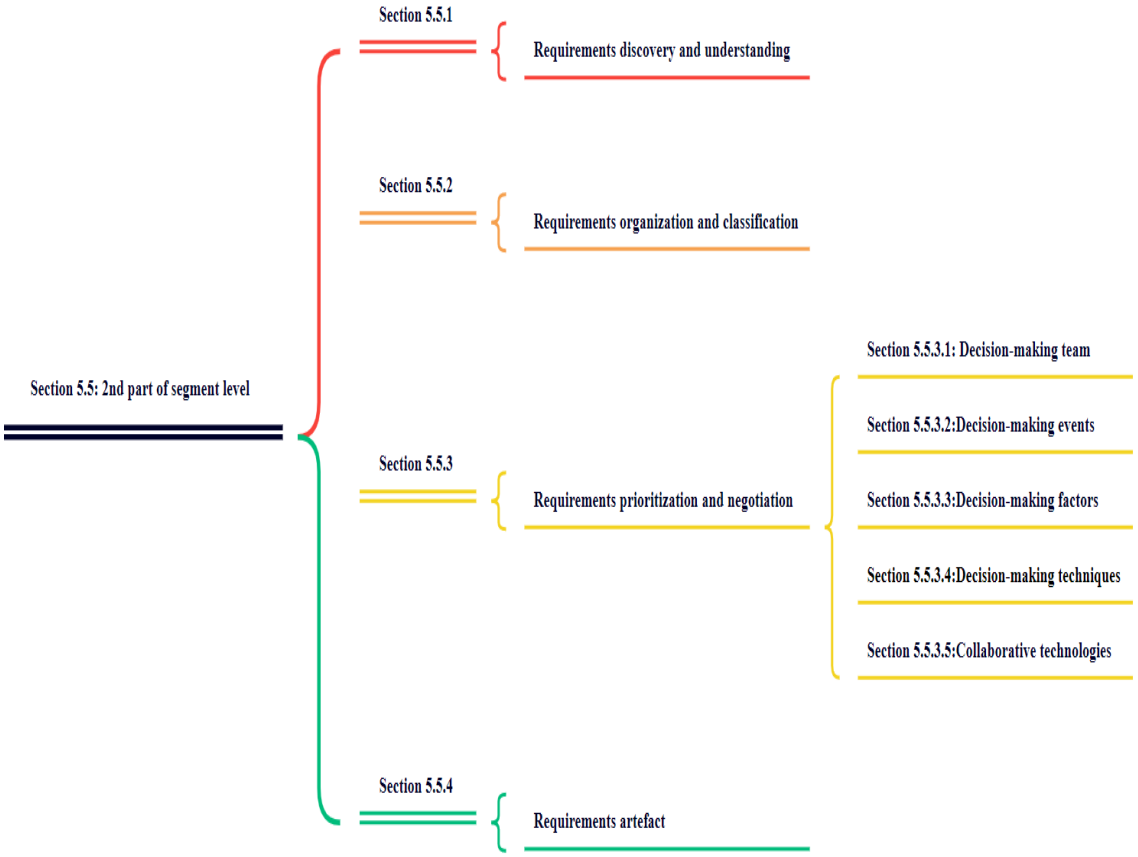
5.5 2nd part of Segment level

Introduction: In the second part of a segment level, a priority list of features that resides in the portfolio backlog, serves as an input for each of the segments. Features that are formally signed off by the portfolio management for implementation, are allocated to each of the segments based on their relevance to the segment during a ***bigroom planning*** *event*. The number of features that are allocated to each of the segments may vary. Each feature generally requires more than one engineering team for implementation. For each feature there is a dedicated PM who is generally responsible for providing all the relevant information (e.g., requirements clarity), to the engineering team, for implementation. A PM often occupies a role of a PO in one or more engineering teams, who are involved in a particular feature implementation. For example, Feature A may require three teams for implementation, in such a case PM can be a PO of any of the teams. Generally, a one-to-one relationship between the PO and engineering team is maintained. However, one PO can manage more than one team as well if required.

Figure 5.6 shows the mechanisms that are discussed at the second part of segment level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 5.6

Snapshot of activities discussed at the 2nd part of segment level



5.5.1 Requirements discovery and understanding

In the second part of a segment level, a feature description that is typically documented in the form of a business-case is generally employed to discover and understand potential requirements that can be posed to implement a particular feature.

> *So what we do, we have business-case that contains all the information regarding a particular feature. So we use that business-case to explore. I mean in terms of what we need to do to implement that feature. [MEL_P8]*

5.5.2 Requirements organization and classification

As discussed in Section 5.2.2, four level of requirements hierarchy model is adopted by the case organization.

Theme(s) → Feature(s)/***Epic(s)*** → ***User story(s)*** → Task(s)

In the second part of a segment level, a feature is turned into an ***Epic*** for the engineering teams for implementation. The case study data revealed that an epic is a reflection of a

feature. Participants report that, the main reason to convert a feature into an epic is the adoption of CTs- ***Jira*** that is employed by the case organization for the engineering teams.

> *you know, AHA feature card is equivalent to an epic in Jira. [MEL_P4]*

> *So, then we were talking about Epic, every feature sort of in JIRA become an epic. In a Jira we create epic. [MEL_P9]*

> *within in AHA its called a feature but within Jira, its called an Epic. [MEL_P7]*

In addition to this, an Epic is broken down into *user stories*. A user story is the quickest sort of element deliverable through a thin pipeline (i.e., end-to end functionality) that can give some valuable feedback and can be implemented in a single development IR (i.e., 2 weeks). *User story* typically spans the second part of segment level and the team level in terms of decision making on their priority. Each Epic/Feature has a dedicated PM who works closely with the engineering for decomposing an epic into user stories.

> *then we have stories for that epic. [MEL_P9]*

> *Product managers write user stories with an input from engineering. [MEL_P3]*

5.5.3 Requirements prioritization and negotiation

5.5.3.1 Decision making team:

Teams' triad (team of team meeting) that includes **PO**, **AO**, **TL** is embedded in each engineering team (Ambler & Lines, 2012), where PO provides business perspectives, AO provides technical needs, and TL ensures delivery works collaboratively to make decisions on the priority of requirements. This includes user story, technical debt work, defect work, that are posed to achieve an objective of a particular feature/epic.

> *All the teams' triad they come together and breakdown a feature into user stories. Triad is like POs, TLs, AOs. As I mentioned POs come up with business need, technical needs represented by AOs, and resources by TLs. [MEL_P11]*

5.5.3.2 Decision-making events

At the beginning of the development of a new feature/epic, a discussion is held with the teams' triad who are involved in a particular feature/epic implementation for decision making on the requirement. For example, Feature A requires two teams, and in that case both teams' triad will be involved in decision making on the requirements. During decision making, each teams' triad provides their perspectives and collectively makes the decision on the requirements, the user story. As an outcome of this event, a ***work item list***, that contains strategic work typically in the form of user stories, technical debt work,

defect correction work (Ambler & Lines, 2012), is created, where a particular section of the ***workitems list*** is reserved for the teams. That section will act as a ***team backlog*** for a particular team and is allocated to each of the teams who are involved in a particular feature implementation.

> *So what happens is every time when the new epic starts, we triads [all teams triad] meet and go through all the user stories, enhancement work etc etc. and together create workitem list belongs to that particular feature. […] So it's not like everything is mixed up, so there's certain sections and the sections are for every team, so I know what this section is for. Like one of my team's name is Teapot, so this is what is belongs to teapots. [MEL_P2]*

The Teams' triad meeting (team of team meeting) is generally held on a ***fortnightly*** basis, usually an hour-long meeting where requirements' progress, issues, blockers, dependencies with other teams, any change in priority are discussed. However, they have ***weekly check-ins*** as well.

> *Generally, we meet bi-weekly but we catch up once every week. […] where the Triad [PO, AO, TL] will catch up from all the teams who are involved in that feature implementation. And they will provide updates, what we're doing. if they need anything that's higher priority for them for us to pick up and stuff like that. [MEL_P2]*

5.5.3.3 Decision-making factors

The case study data indicated that decision making factors vary with requirements classification. The following are the decision-making factors that are adopted to make decision on the priority of requirements that are decomposed from a feature.

Decision-making factors for user story: *Business value* (i.e., importance for customers) and *urgency* (i.e., customer demand and adoption), are the decision-making factors that are employed to set priority on the user stories.

Decision-making factors for technical debt/defect correction work/enhancement work: *Business value* is the main decision-making factor that constitutes *scope* and *severity.*

Numerical scale: Each of the included factors is evaluated on a numerical scale one to five on a 2D board where x-axis represents business value and y-axis represents urgency. Each of the factors is rated between one to five, where lowest number represents less valuable and higher number represents most valuable.

> *so it can go from 1 is like minimal work and 5 is like there will be risks and uncertainties with the work we're doing and the customer is factors one is low customer impact and five is like customer will gain a lot of value if we do that feature. [MEL_P2]*

but if its tech-debt related work or if its bug, or we can say if its internal improvement work- business value is calculated differently. In such cases scope and severity are the two main vectors that get evaluated. [MEL_P9]

business value, and urgency are the factors to evaluate the score of user story. [MEL_P13]

5.5.3.4 Decision making techniques

- Priority on user stories is typically driven by the feature/epic priority
- Relative priority is considered
- Team capacity and ability is considered
- Dependencies among user stories are considered.
- MoSCoW (Must have, Should have, Could have, Would have) is the main prioritization technique that is employed to set the priorities on user stories where "Must" should be implemented, and implementation of "Should" is based on the availability of time.

We mainly use MoSCoW to prioritize the user stories. [MEL_P1]

Along with the numerical value, various other stuff are considered. Like team ability and capacity, relative priority. [MEL_P3]

Priority of user story basically depends on the epic priority. [MEL_P16]

We use the relative story points size in technique. […] We basically pick one OK. Call this 5. And have a look at another story. So effectively, do you think this is more risky? Do you think this will consume more efforts. […] it's like we draw on a white board at 2 dimensional. [MEL_P15]

But you know that's where we will be putting like based on the capacity and you know the velocity of the team- we will be assigning those stories to the iterations. [MEL_P10]

Is there any specific dependency regarding the orders. [..] we will be asking ourselves questions. OK, this seems to be quite risky. It's big. It has to happen first. [MEL_P15]

5.5.3.5 Collaborative technologies

CTs- Jira is employed in the second part of segment level to manage requirements (e.g., priority list of requirements via an electronic team backlog). An individual user story or any other requirements' progress can be tracked through Jira. There is an AHA tool integration with Jira to push status back from an Epic in Jira up into AHA.

Engineering uses Jira heavily so they drive their reporting through Jira. [MEL_P1]

we use I mean within the teams all of their work is within Jira as the tool used for capturing those work items. [MEL_P5]

Jira, so we obviously maintain all our epics, stories, tasks, issues as a bugs, incidents everything in Jira. [MEL_P12]

All the tools that we use to manage requirements like our product management tool AHA, engineering tool Jira, documentation tool- confluence are integrated. We can easily trace progress of a particular feature, we can who is working on what and things like that. [MEL_P11]

5.5.4 Requirements artefacts

The following requirement artefacts are built in the second part of segment level which are specified via CTs- Jira, Confluence (Daniels et al., 2015).

Storyboard (an electronic artefact configured via CTs- *Jira*): For each feature/epic, a ***storyboard*** is built that contains user stories and any other requirements, including solution alternatives, that typically provide detailed information for the engineering for implementation. Participants report that the number of user stories that are contained in a storyboard depends on the size of the feature. The ***PM/PO*** with an input from ***engineering,*** is generally responsible for building a storyboard.

Depending on the feature, there could be a couple of stories that could be lot more. [MEL_P9]

So there will be a storyboard in fact for every feature. [MEL_P1]

so generally these are written by the PM/PO but they get input from engineering as well. [MEL_P10]

Workitem list (electronic artefact configured via CTs- *Jira*): a *workitem list* is a requirement artefact which is built in the second part of segment level that contains priority list requirements for the teams who will involve in feature(s) implementation.

So what happens is every time when the new epic starts, we triads [all teams triad] meet and go through all the user stories, enhancement work etc etc. and together create workitem list belongs to that particular feature. […] So it's not like everything is mixed up, so there's certain sections and the sections are for every team, so I know what this section is for. Like one of my team's name is Teapot, so this is what is belongs to teapots. [MEL_P2]

Confluence: CTs- *confluence* is employed to maintain documentation of requirements, such as recordings of discussion with customers, that helps in building the shared understanding of requirements. The integration feature of *confluence* with *Jira* and *A*HA provide traceability of requirements.

The outcomes are recorded in confluence page as well. [MEL_P1]

All the tools that we use they all are integrated. [MEL_P4]

5.6 Team level

Introduction: Methods adopted: At the team level, engineering team(s) is flexible to adopt any of the Agile methods for implementation. Agile methods that are commonly used by the engineering team includes *Scrum* (Schwaber, 2004) and *Lean* (Ambler & Lines, 2012). At present (during data collection), *Scrum* is the dominant method that is employed by the engineering team for implementation (Schwaber, 2004). However, the Lean way of working is getting increasing attention among engineering teams, over Scrum. Participants mentioned that they are in-progress to generalize the lean way of working at the team level. The main reason behind this is that they consider lean supports continuous prioritization, yielding the result that changes are easy to make in the decided priority if required (e.g., unplanned work). On the other hand, the Scrum method typically supports a time-boxed event (e.g., 2 weeks iteration), that makes it difficult to make changes to in-progress work. Moreover, the Lean way of working is considered to require less planning. On the other hand, Scrum have dedicated planning events (e.g., retrospective meeting at the end of iteration) (Schwaber, 2004).

> *Right now it varies across [..]. Some teams are working in more of a lean Lifecycle where it's more of a Kanban Model. You're just grabbing the next thing off the queue and working it through the process. At the moment that teams I'm working with them follow basic scrum style lifecycle where they plan 2 weeks' worth of work at a time. [MEL_P5]*

> *we have some teams using the lean process. So, I think we want to generalize it, but a lot of the teams are still on the scrum model and they use two weeks sprints. [MEL_P16]*

> *With scrum process ideally, you do not interpret team for the entire duration of the sprint. I think if we could add up to that lean process more broadly, they'll be useful. [MEL_P16]*

> *I think the issue that people have with sprints is that there tends to be very hard lines. [MEL_P3]*

Team structure: Each team generally has six to eight members that includes one PO, one AO, one TL, and three to five engineers (Schwaber, 2004). However, some teams are smaller, especially the teams that are built for a specific purpose, for example, a customer focused engineering team to work on customers' issues. Most of the teams follow pair programming where two engineers (usually novice-expert) work together on one problem. However, practices vary in terms of way of working. Teams are flexible to adapt any of the practices that work best for them, but the process is the same. The main

benefit of pair programming is that change (e.g., change in the decided priority) is easy to manage in a team. For instance, when a team member occupied by the new high priority requirement, in-progress requirements can be continued by another member as he/she is already aware of the item in terms of what it needs to do. Other benefits include no need for code reviews, good for engineers to learn to new skills, and mentoring less experienced team members to enhance their skills.

> *So we have generally like all the teams will be like 2 pizza team kind of thing. So 6 to 8 people generally we have. […] There are few teams smaller than that which are for a dedicated mission kind of a thing. [MEL_P10]*

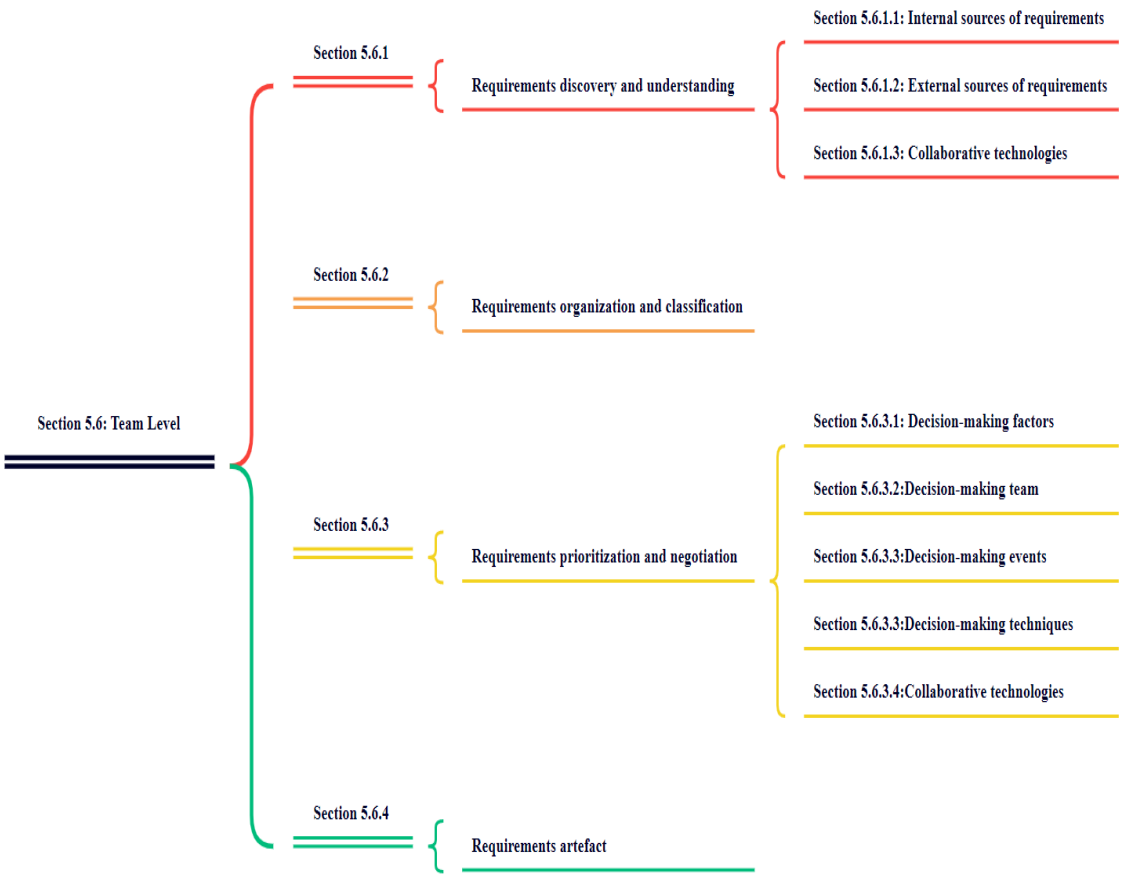
> *If someone is good at particular thing what we try to do is like pair them with someone who doesn't know and that's how we are you know, balancing that learning and enabling sort of things as well so. [MEL_P10]*

> *I actually encourage my team to work on it together because of the fact that for any reason, for example, something new comes and person is already occupied with other work or something like that. We have another person who knows. […] Code reviews we don't have to then do because they are already involved into end-to-end development. [MEL_P10]*

Figure 5.7 shows the mechanisms that are discussed at the team level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 5.7

Snapshot of activities discussed at the team level



5.6.1 Requirements discovery and understanding

At the team level, ***Business case*** and the ***storyboard*** are the two main guiding requirement artefacts that act as ***boundary spanning artefacts*** that are shared with the engineering team in order to explore a particular feature/epic.

> *From execution point of view team actually, you know, PM actually share the business use case, storyboard with the team. [MEL_P9]*

> *there's the business case [provide feature description] and the storyboard which contains the user stories, that we provide to engineering. [MEL_P1]*

5.6.1.1 Internal sources of requirements

At the team level, the engineering team that includes PO, AO, TL, and engineers are generally the internal sources of potential requirements.

Practices to collate potential requirements from the engineering team members (internal sources).

Innovation morning: An innovation morning is technique that is used to generate potential new ideas/requirements specifically from the engineering team. Participants asserted that fortnightly half a day is generally reserved for engineers for innovative work.

> *So we do get some time here in there few hours where we can just work on this idea. [MEL_P14]*

> *but there is the idea that you have certain amount of time like maybe half a day every fortnight within the engineering team to innovate. [MEL_P13]*

Inception phase: At the beginning of the development of a new feature/epic, the *Inception phase* is conducted by engineering team (i.e., PO, AO, TL, Engineers) in order to explore the scope and/or requirements being envisioned (Ambler & Lines, 2012). All possible information such as spikes (e.g., solution alternatives), breakdown user story into further user stories if needed related to a particular feature implementation, is gathered as a part of inception.

> *So instead of jumping into the development right away, we spend some time before hand to understand you know what are the roadblocks? What other things, which we don't know. So we do some analysis on those things. […] The whole team usually gets involved. [MEL_P14]*

> *team break stories into further stories wherever possible. […] [MEL_P9]*

> *it's possible that you might split it out further into two stories, so that's how you can do. [MEL_P10]*

> *And is this feasable or is, you know, Or if there are two different approaches, which one would be more effective [MEL_P6]*

Inception of the upcoming feature, typically starts during the final stages (i.e., hardening) of a current feature. Duration of inception generally depends on the feature that may vary from ***one week to several weeks.*** Inception typically takes ***one development IR (i.e., two weeks)*** but it can go beyond one development IR specifically for complex features.

> *It'll be at least one sprints; 1 IR we call it like 2 weeks, but it could be longer depending on we just. [..] But if it's really complicated, it could take that long. [MEL_P13]*

> *It depends on the feature, but it's OK if it's take more than one iteration also. [MEL_P10]*

5.6.1.2 External sources of requirements:

The case study revealed that PMs, UX/UI team, support team, sales, and marketing team generally collate potential ideas/requirements from the external sources (e.g., customers). However, the engineering team typically the PO, can also collate potential ideas/requirements from the external sources (e.g., customers) wherever needed.

> *Collating customers needs, its product management who works with them on day-to-day basis. It's their job, you know, they spend most of their time with customers, you what they want, what their pain points are etc.etc. We can also communicate with them. but it's very rare. if needed, our PO generally communicate. [MEL_P7]*

5.6.1.3 Collaborative technology

At the team level, Jira is the main CT to capture potential requirements (e.g., enhancement work/bugs/defects etc.), from the engineering team members.

> *we use a system called Jira to track, you know it's more of an engineering level system for tracking Feature improvements like enhancements as well as any defects or bugs within the system. [MEL_P5]*

> *we can directly submit our request via Jira as well. [MEL_P13]*

5.6.2 Requirements classification and organization

As discussed, the four levels of requirements hierarchy model is adopted by the case organization.

Theme(s) → Feature(s)/Epic(s) → ***User story(s)*** → ***Task(s)***

Requirements that require small amount of effort (e.g., one or two weeks of works) generally sorted at the team level. The requirement type such as user story(s)- typically a breakdown of a feature, defect correction work, and technical debt work generally, does not require a lot of ceremony (i.e., portfolio level decision making). These requirements type typically require a small amount of effort and are sorted at the team level.

> *There can be a simple idea or put two more dropdowns on this particular web page or move this button around or something and they're easy. They are sometimes taken by the team there, sometimes taken by the engineering team along with the product manager. [MEL_P11]*

> *and if it's not like a major change in what we're doing like 1 or 2 weeks of work, we would, essentially they sit there and then you know, probably prioritize it in the next sprint or the sprint after. [MEL_P14]*

5.6.3 Requirements prioritization and negotiation

Decision making factors: same decision-making factors that are adopted to make decisions on the priority of requirements (e.g., user story, defect correction work, technical debt work, enhancement work) in the second part of segment level (discussed in Section 5.5.3.3), are adopted at the team level.

Decision making team*:* Team triad that is PO, AO, TL, UX/UI wherever it's required to work collaboratively to make decisions on the priority of requirements; user story, defect correction work, technical debt work.

Decision making event(s): Backlog refinement meeting, a look ahead meeting, and development IR planning meeting are the key decision-making events that are employed at the team level to make decisions on the priority of requirements.

Backlog refinement meeting: Each team has their own backlog that contains a priority list of requirements. Possible requirements that are contained in the team backlog include strategic work (i.e., typically in the form of user story), technical debt work, defect correction work, spikes, and slipping stories (i.e., left over work from the previous development IR).

> *I'll try to bring in everything like enhancement, bugs and whatever kind of work there is sits in the backlog based on the business value. [MEL_P10]*

During the development of a feature, each team has an on-going backlog refinement process owned by PO of the team and most teams typically do *weekly meetings* among team triads (PO, AO, TL) as well for alignment.

> *So, triad meet weekly to prioritize the stuffs within the team. Actually, it's a continuous process. Triad all the time working on that. That's their responsibility but PO is the one who generally own this [i.e., prioritization]. [MEL_P5]*

> *They will consider everything. They will look for the work that strategically we have to do, any unplanned work, slipping stories etc. [MEL_P2]*

> *Yeah, because that's something [i.e., prioritization] that continuously keeps happening in the background. [MEL_P2]*

Look ahead meeting: During the development of a feature, each team conducts a look ahead meeting (Ambler & Lines, 2012) that helps with *backlog refinement* and *IR planning*. The look ahead meeting is generally held in the second week of the current development IR.

> *So, I [PO] do look ahead with a triad. As I mentioned the AO or the TL. [MEL_P2]*

> *So, we do a look ahead before the Sprint planning. Which is in the existing Sprint, we will do a look ahead for the next Sprint. [MEL_P12]*

The look ahead meeting is typically held in two sessions:

Triad look ahead: This is typically a half an hour to one-hour session which is conducted between triads (TL, AO, PO and UX representation in certain cases). This is the place where what will be the next highest business value items that the team needs to tackle in the next two development IRs, are decided. Furthermore, the triad lookahead also ensures requirements that are considered for delivery, are still relevant from a business needs and priority perspective.

> *so, I do one look ahead with Triad obviously which is basically the prioritized backlog items for the next 2 iterations. [P2]*

> *We have like a session [i.e., Triad look ahead] generally a half hour session where we'll be looking at what items we have to work on in the next 2IRs and what their priorities are. And do we feel that there's any item that might slipping from current IR to the next IR. Those kinds of discussions generally happen in there. [MEL_P10]*

Team lookahead: This is another 30-45 minutes session where the PO presents the next development IR items to the team at high level. However, the AO can also be involved if required. During team lookahead, a team reviews the rough estimates and ensures requirements are clear. Any spikes or investigation that needs to happen in the current development IR to be successful in a future development IR, are identified. Moreover, any risks or unknowns which need to be handled or taken into consideration so execution will be smooth, are identified. The overall goal of the team lookahead is when the team goes to IR planning, they should have as much clarity as possible.

> *And if it's too technical, the AO steps in and explains this is what we need to do. [MEL_P2]*

> *Then I do a team look ahead with their last questions or if they have any concerns, they ask me that and then it's my job to address those concerns. [….] So, we do start asking them what the risks are. [MEL_P2]*

IR planning meeting: During the development of a feature, IR planning (Ambler & Lines, 2012) happens at the end of the current IR (after the retrospective of current IR), where user stories are broken down into *tasks* if needed and are assigned to the engineers.

> *Iteration planning is the most likely meeting that is set the priorities and allocate work. [MEL_P12]*

So, we breakdown them [i.e., user stories] into tasks if required. This is the place where […] user stories and tasks are assigned to us [i.e., team members]. [MEL_P15]

5.6.3.1 Decision making techniques

- Priority on user stories is typically driven by the feature/epic priority
- Relative priority is considered
- Team capacity and ability is considered
- Dependencies among user stories (generally technical dependencies at this stage) are considered.
- MoSCoW (Must have, Should have, Could have, Would have) is the main prioritization technique that is employed to set the priorities on user stories where "Must" should be implemented, and implementation of "Should" is based on the availability of time.

We mainly use MoSCoW to prioritize the user stories. [MEL_P1]

Along with the numerical value, various other stuffs are considered. Like team ability and capacity, relative priority. [MEL_P3]

Priority of user story basically depends on the epic priority. [MEL_P16]

We use the relative story points size in technique. […] We basically pick one OK. Call this 5. And have a look at another story. So effectively, do you think this is more risky? Do you think this will consume more efforts. […] it's like we draw on a white board at 2 dimensional. [MEL_P15]

But you know that's where we will be putting like based on the capacity and you know the velocity of the team- we will be assigning those stories to the iterations. [MEL_P10]

Is there any specific dependency regarding the orders. [..] we will be asking ourselves questions. OK, this seems to be quite risky. It's big. It has to happen first. [MEL_P15]

5.6.3.2 Collaborative technologies

CTs - Jira is employed at the team level to manage requirements. An individual user story or any other requirements' progress, can be tracked through Jira. There is an AHA tool integration with Jira to push the status back from an Epic in Jira up into Aha. The integration feature of Jira with AHA and Confluence enable traceability and shared understanding of requirements.

Engineering uses Jira heavily so they drive their reporting through Jira. [MEL_P1]

we use I mean within the teams all of their work is within Jira as the tool used for capturing those work items. [MEL_P5]

Jira, so we obviously maintain all our epics, stories, tasks, sub tasks, issues as a bugs, incidents everything in Jira. [MEL_P12]

5.6.4 Requirements artefacts

The following requirement artefacts are built at the team level that are specified via collaborative technologies - Jira, Confluence (Daniels et al., 2015).

User story(s), task(s), spike(s) (electronic template configured via Jira):

User story(s) is the quickest sort of element deliverable that can be implemented in a single iteration (i.e., two weeks).

There are task(s) that describe in technical terms what things are required to be done in order to claim a user story at the end of an IR.

Another requirement is spike(s) which is typically technical investigation that needs to happen in the current development IR to be successful in a future development IR, are identified.

Team backlog (configured via CTs- Jira): team backlog that contains priority list of requirements (e.g., user stories, spike etc.) is built at the team level.

so all the stories, spikes, there are some spikes […] we have a template for them [user stories] in Jira. [MEL_P2]

for documentation we use confluence. Its like detailed analysis we maintain in confluence [MEL_P3]

delivery team actually from execution point of view breakdown that story into tasks. [MEL_P9]

And then if we need deeper details, you know we're often linking out to confluence, linking upto Jira, linking attaching documents. [MEL_P7]

Confluence: CTs - *confluence* is employed to maintain documentation of requirements, for a detailed discussions and/or analysis on requirements such as recordings of discussion with customers, that helps in building the shared understanding of requirements. The integration feature of *confluence* with *Jira* and *AHA* provide traceability of requirements.

The outcomes are recorded in confluence page as well. [MEL_P1]

All the tools that we use they all are integrated. [MEL_P4]

5.7 Incorporate distributed members of decision-making team

In the studied organization, members of a decision-making team who make decisions on the priority of requirements across scaling agile levels, are globally distributed. This section describes the mode of communication at the scaling agile decision-making levels and the CTs that are employed by the studied case organization to be incorporated with the distributed members of a decision-making team.

5.7.1 Mode of communication at the portfolio level:

At the portfolio level, *bigroom planning* is a formal decision-making event that is organized on a Quarterly basis (Q1, Q2, Q3, Q4) where priorities on requirements (i.e., themes, features) are decided. It emerged from the data, that the whole decision-making team (i.e., portfolio management) is globally distributed. Decision makers are generally located at the head office (USA) and Melbourne sites. However, the majority of them are located at the head office. The case data revealed that sometimes the company has arranged travel for the decision makers, in order to attend decision-making event(s) (i.e., *bigroom planning*) in-person. This has yielded outcomes that it has been easy to reach a consensus, better understanding of requirements, and easy to collaborate during decision making. But due to the cost constraints, this does not happen frequently. Participants reported that portfolio management generally meets four times a year, where they meet in-person at least twice a year in an auditorium either at the head office or Melbourne site and virtually alternating every three months (via CTs as discussed in Section 5.7.5).

> *you know either face to face or virtually look at those strategic priorities. […] then twice a year, its face to face where people across the globe meet together in a bigroom and work through those strategic priorities together. [..] Yeah, an alternating face to face every six months. [MEL_P5]*

> *We're sort of a global organization. We've got teams around the Globe in six or eight locations. […] so getting people face to face to make these high level strategic priority decisions around ideas we find quite important. [MEL_P5]*

> *So it's [during face to face meeting] very easy to collaborate. you know it's [bigroom planning] usually a week long meeting so you get to spend the whole day in a room with the same number of people. It's very easy to reach a decision when you're locked in a room with the same person through the whole day so we can discuss and debate and argue and make your case. [MEL_P11]*

> *as a group physically, You know, I'd say at least twice a year. [MEL_P6]*

> *So, you know everybody would fly into Australia or fly into Chicago but you know it costs us so It generally happen one or two times a year. [MEL_P4]*

5.7.2 Mode of communication at the Segment level

At the segment level, where teams are generalizing specialist teams that are designed in such a way that all their engineering teams are typically at the same site.

Team of Teams meeting (Teams' triad meeting) is the main event that is typically held at the segment level on the fortnightly basis. The case study data showed that teams who are involved in a particular feature implementation, were either at the same site or globally distributed. The teams' triad either meet in-person or virtually. Participants reported that if the teams who are involved in a particular feature implementation are at the same site in that case, Teams' triads generally meet in-person otherwise they meet virtually via CTs (discussed in Section 5.7.5).

> *So the thing is, we need to coordinate between them [teams] and they are global teams. So one is in the UK [total 3 teams]. […] so like tomorrow we have a team level where the Triad will catch up from all the teams who are involved in that project [i.e., feature]. [MEL_P2]*

> *We are lucky, we are 2 team who are working on this feature, so what we do, once in two weeks we triads meets face to face in a company meeting room. [MEL_P3]*

5.7.3 Mode of communication at the team level

It emerged from the data that team members are generally at the same location at the team level. Decision making events (e.g., Look ahead meeting, IR planning, backlog grooming meeting) are typically held via face-to-face meetings.

> *So, we are all at the same site. So, we [i.e., team members] all get together when we have planning meeting [like IR planning]. [MEL_P14]*

> *But then as you get further down into the portfolios and the delivery teams that they might meet more frequently, more face to face in smaller groups that overcome the time zone challenges because they're all in the same time zone. [MEL_P5]*

5.7.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection)

During the time of data collection - in September 2020, due to COVID situation, team members were working remotely. Therefore, decision makers typically met virtually via CTs (web conferencing tools discussed in Section 5.7.5). For example, decision-making event at the portfolio level (during the time of data collection), bigroom planning event was completely virtual, which has its own benefits and drawbacks. One of the benefits is that it gives an opportunity for the participants who cannot attend, the prioritization discussion (e.g., engineers). However, the main drawback is too many meetings.

It's much, much tougher now that everybody is remote and everybody is on team, so it ends up being a series of teams meetings. [MEL_P11]

But yeah, this time around it's remote. […] The benefit though, is that you can involve more people. […] and you know those are people that we wouldn't be pretty fly all the way to Chicago to participate. [MEL_P16]

5.7.5 Collaborative technologies

The participant MEL_P5 emphasized that there is no alternative to face-to-face communication but using CTs effectively can reduce the distance between distributed stakeholders. The case organization has employed a rich infrastructure that enable to work with distributed members of decision-making team.

But none of that [technology- MSTeams] you know replaces face to face communication. [P5]

Web conferencing tool: the studied case organization employed web conferencing tool such as MSTeams[12] as their primary synchronous communication tool to communicate with cross-site members of decision-making teams. The application sharing feature of MSTeams as CT enabled cross-site members of decision-making team to allow synchronous editing of requirements artefacts if needed during decision-making event(s).

In addition to the formal synchronous communication, informal synchronous communication (e.g., instant messaging, instant calling, team(s) specific channel to communicate internally within the team) as well as asynchronous communication (e.g., offline chat, tag to notify people, recordings of decision-making events) were also organized via web conferencing tool (i.e., MSTeams) to communicate with distributed members wherever needed.

So at the very least, even if I'm not on a call for example, or even if product management isn't on a call, we end up with usually recording notes and people tagged appropriately so that they can go follow up and review it themselves after the session. [MEL_P7]

We use teams to comment or annotate on things. [MEL_P1]

we just exchange ideas inside teams. We have our own private channels and we have our public channels for different purposes. [MEL_P15]

I think we all join video conferencing calls and you know we have discussions one on one discussions. we have teams to do.[P1]

[12] Information on MSTeams can be accessed via https://www.microsoft.com/

so I can approach them anytime of the day. They will see whenever they available, they see the message and get back to me. [P2]

white boarding writing out things and discussing them using teams. [MEL_P1]

MSTeams is been invaluable in terms of communication.[MEL_P4]

Email: the studied case organization employed 'email' as CT to communicate with a decision-making team. According to the participants, email is a less frequently used asynchronous communication mechanisms in the studied case organization. The main reason behind this is email as CT introduces unnecessary delay in the requirements' communication.

In the studied case organization, email is specifically used to communicate with external customers and end users, as they do not have access to their MSTeams structure.

Email comes into play less frequently, but does happen, especially if there are customers involved. [MEL_P7]

I think email as well but it's not that much frequent. [MEL_P4]

Email we do use but you know sometime people busy with some other stuffs and in that situation, things gets easily missed. then my personal rules is, If it takes more than three balances an email, you should just call them. […] for calling as well, teams we use heavily. [MEL_P2]

5.8 Challenges and overcoming strategies

This section describes the challenges that occurred while making decisions on the priority of requirements across scaling agile decision-making levels. First, challenges that occurred specifically due to the distributed nature of decision-making teams are presented and the strategies that are employed to overcome them, are presented. Next, the challenges that have occurred regardless of distributed decision-making and the potential strategies are discussed.

5.8.1 Challenges specifically related to global distribution

Challenges are identified on the basis of three distance dimension (i.e., temporal distance, geographical distance, socio-cultural distance) which are discussed below (Zowghi & Damian, 2003).

5.8.1.1 Geographical distance:

Geographical distance (Zowghi & Damian, 2003) limits the face-to-face collaboration during decision-making events (e.g., Bigroom planning). Participants asserted that

geographical distance is manageable as CTs enables them to work remotely. A rich infrastructure (e.g., MSTeams) is employed by the case organization to meet virtually during decision-making events (as discussed in Section 5.7.5). Furthermore, participants asserted that they have developed a mind-set to work with distributed members and using tools that also helps in reducing the global distance.

> *I think we're all used to using virtual tools for sometime in our careers now. We don't really feel the distance as much. [MEL_P1]*

> *we again technologically we are enabled so we have all the tools we need to be able to work remotely, and work with teams in different locations. [MEL_P2]*

5.8.1.2 Temporal distance

One of the commonly stated challenges is temporal distance (Zowghi & Damian, 2003). As discussed in Section 5.7, decision-making team specifically portfolio management is highly globally distributed members that proves difficult for them to collaborate during decision-making events (e.g., bigroom planning) due to the time zone differences. Participants indicated that managing the time-zone is more difficult especially during day light saving.

The case study data indicated that temporal distance often results delay in communication and/or response has happened especially at the weekend, and week start (typically lose one to two days with some time-zones).

Participants emphasized that use of proper communication mechanisms and flexibility in routine are needed to manage temporal distance. Early morning meeting and late evening meeting are the remedy that are often used to manage temporal distance that yields a long working day, and work life balance problem. Some of the other practices include recordings of decision-making events, communicate via email, offline conversation, having proper documentation, ad-hoc meetings if needed, organizing travel (discussed in Section 5.7).

> *So I think what happens is you kind of work a flexible day. In my case I end up having early morning calls, late night calls and doing work in the middle of the day as well. [MEL_P1]*

> *Time zones are certainly challenging, It's the biggest one. [MEL_P10]*

> *Communication can sometimes be delayed. For instance, you are at Saturday for you so normally my co-worker, she's in Melbourne like if I haven't asked her for something by Thursday. She may reply on Sunday, but I won't get it till it's Tuesday for her, right. […] it's work life balance problem. […] when there's a confusion or something that needs some active decision making we set up another meeting. [MEL_P4]*

> *when these daylight saving happened we left Bangalore and then our UK Office and an also Hamburg office to collaborate with the task because they have a better overlapping with US. [MEL_P9]*

5.8.2 Challenges regardless of global distribution

This section discusses the challenges that confront the globally distributed decision-making teams and potential strategies that can be employed to overcome those challenges.

5.8.2.1 Tension between product and engineering

It emerged from the data, that there is always a tension (which is generally a healthy tension) between product and engineering in terms of decision making specifically on smaller impact requirements such as tech-debt work and/or internal improvement work, UX-debt. In the experiences of participants, generally the decision making on smaller impact requirements requires a lot of negotiation, as value is not clearly articulated in smaller impact requirements. On the other hand, bigger impact requirements like strategic work that provides direct value to customers and business, require less negotiation due to their direct alignment with business strategy. In addition to this, participants asserted that product management generally give importance to the requirements that bring high revenue, which in-case of smaller impact requirements is hard to articulate. In such cases, often additional research is needed to gather all required information/facts/data, in order to justify the potential requirements. These conflicting perspectives between product and engineering, often cause tension, breakdown in the ability to collaborate, and creates pressure on the team members.

Potential strategies: Participant MEL_P3 emphasized that in order to ease these tensions, the decision-making process should yield an optimal mix of requirements (i.e., prioritize smaller impact (e.g., tech-debt work) requirements along with bigger impact requirements (e.g., strategic work).

> *I mean it causes tension right. I think the impact to, at least in that particular case with the dashboard because I'm the one that had the dashboard designed. You know, everybody's kind of looking to me like not you got a push it through like I've got the lead of UX and she's like,OK, let's make it happen like we're not going to use that other tools so you have to make your use case and she's like you need to validate with seven more customers in the next 2 weeks and then present it out. [MEL_P3]*

> *I think the other part about prioritization is that It's very easy to get caught up in the bigger things that you can do because you feel that they'll have a bigger impact. […] I want to look at like maybe the two or three big things that we can do, and then what are the two or three small things that we can squeeze in so that you're balancing what you're actually providing to your customers? Otherwise, small things will never happen [MEL_P3]*

> *outliers- things that are not obvious things. But if built would have outsize returns. Those kind of ideas are not always picked up. [MEL_P11]*

> *for them [Product management] it's always about bringing more revenue. […] And sometimes will my team It's [enhancement work] about doing the right thing for the users. So there's always a fight. […] I'm not saying they're wrong because they have a business to run, yeah, but that's the conflict you see. [MEL_P17]*

5.8.2.2 Accumulating technical debt

In the experience of participants accumulating tech-debt is obvious. It is impossible to get completely rid of tech-debt. A possible reason to accumulate tech-debt includes sometimes teams struggle to find a time payoff for tech-debt, lack of clearly articulated value associated with tech-debt work, as a result of cut scope, urgency to deliver features within a specified timeframe and/or business need.

Participants asserted that not paying off tech-debt often results in increased turnover, but creates frustration among team members, quality issues start to creep in, and impacts delivery as well.

> *sometimes you need to build something just to allow us to support that architecture. I know it's a very tough sale, right because there's no hard dollar associated with that. [MEL_P16]*

> *things slowing down, I think that would be the only impact l can see. [MEL_P1]*

> *We need to clean something up [tech-debt or internal improvement]. It will make our future better and the fact is we don't have enough time to do that. So we sort of leave things behind. Right, and that can be frustrating for the team, so I would say that's the number one. [MEL_P8]*

However, participants asserted that managing the tech-debt was more challenging in the past (one whole quarter focused on hardening the product) than now. The following are some of the strategies that are employed by the studied case organization to manage tech-debt:

- implement technical debt together with strategic work (e.g., user story);
- encourage teams to bring-in tech-debt;
- establish a dedicated technical debt team;
- categorize the requirements (e.g., strategic work, technical debt work, maintenance work etc.) and allocate capacity allotment (e.g., 10 to 20% of engineering time to handle tech-debt) to each of those categories;
- payoff tech-debt as a part of each development IR (i.e., usually one or two tech-debt items in each IR); and

- plan development IR to pay off tech-debt when in-between the release IR.

> *A percentage of our time is for these internal improvements or tech-debt. [MEL_P8]*

> *And to basically make sure that they've got, you know 20-30% of their time kind of dedicated to those sorts of things, because there will always be technical debt. […] I am big fan of paying off tech-debt. What I normally do, bring 1 or 2 dech-debt in each IR and then plan one IR in between release. […] I always encourage my teams to bring tech-debt work. [MEL_P6]*

> *how do I make this fit so if it's a small thing it can fit in that 20 to 30% that I mentioned earlier then we'll do it. And if its more than that we need to repriortize the roadmap. [MEL_P16]*

5.8.2.3 Quality of requirements

Participants' reports that use cases, acceptance criteria (AC) and definition of done (DoD), in some cases, are ill defined or continue to focus on output rather than customer outcomes resulting in missed expectations.

> *Quality of information/use cases usually is poor that one of the challenges we are facing currently. [MEL_P9]*

> *Sometimes we struggle with clarity, I mean acceptance criteria and definition of done needs improvement. […] all this often impact delivery. [MEL_P6]*

Potential strategies: To overcome this challenge, feature AC and DOD need to support the lean principles of delivering working code which is fit for purpose in as short a time as possible. In addition to this, PMs and engineers need to agree on clear AC and DOD (including performance testing, documentation etc).

> *Consensus between product and engineering on acceptance criteria and definition of done is required to resolve this [i.e., requirements clarity] issue. [MEL_P6]*

> *I think lean way of working might solve this. It's not fully adopted; this strategy is in progress. [MEL_P9]*

5.8.2.4 Bias during decision making

It emerged from the data, that a matrix/scoring system is employed in order to make decisions on the priority of requirements across scaling agile decision-making levels. However, decision-making is not entirely based on the objective data. Loud voices still exist during decision making on the priority of requirements specifically on decision making on strategic work. Participants report that a command-and-control nature exists in terms of decision making on the priority of requirements. Priority decisions are generally biased by senior leadership team's viewpoints or goals. The following are some of the interview snippets that support the above findings.

you know, we've had cases where they [Product management] basically just ordered us to do something 'cause it's just requirements that don't exist. This happened last week and Engineering said, hang on, no, we need business requirements. We want to see them 'cause we want to understand what we're actually building. Otherwise we can't do a good job. [MEL_P6]

Sometimes we aren't in agreement that we should change the priority. But if somebody with more seniority we defer to that decision. So if somebody above us is saying, Yep, we think it should move. You know, we respect that right and say OK, [MEL_P8]

executives are just going to say look, I get that. But this is an immediate need. So that throws a monkey wrench in, so that would be the ultimate deciders when our CEO or one of the senior leadership members decides that. We're changing priorities and there's I don't know what the defense is for that. It just happens sometimes. [MEL_P4]

5.8.2.5 Difficult to measure Business Value

It emerged from the data that business value is the main decision-making factor in order to make decisions on the priority of requirements across scaling agile decision-making levels. Participants asserted that sometimes it is difficult to evaluate the business value, as everyone may have a different level of understanding, and sometimes scoring may not align with the importance of the requirements (e.g., bug/enhancement). In such cases, as recommended by participant MEL_P7, business education for the teams may help them to understand the business value of requirements.

I think that's [Business value] a much harder one to assess. [MEL_P3]

It [Business value] generally comes down to those sorts of metrics that sometimes hard to measure. [MEL_P5]

Sometimes you know its [Business value] hard to assess. We do our best to quantify the value, but you know its managed by people. Actually, at the end final value depends on the person knowledge who is evaluating that. [MEL_P9]

I mean I would love for the teams to have more business education. It would be more comfortable discussing the business value, the money aspects of why we do. I think that's a significant inhibitor. [MEL_P7]

5.8.2.6 Dealing with quality requirements (QRs)

Participant MEL_P6 reports that the case organization is facing challenges with QRs such as security and performance. He asserted that QRs (specifically performance), are not met, which is often revealed in production. Some of the reasons include QRs are ill defined, urgency to deliver features in a specified time frame. Participant MEL_P8 asserted that sometimes effort required for testing, bug fixing, is left out during estimation.

I don't think we do a very good job with the non-functional requirements. […] the teams are very much struggling with non-functional requirements and often we get

> *code in production that hasn't met some of those requirements, probably because they're not well defined as well, […] performance in particular is one. [MEL_P6]*

> *I noticed sometimes they don't reserve space for bug fixing, testing etc. etc. it happens especially when we have a pressure to deliver business features. [MEL_P8]*

The following are some of the strategies that are employed by the case organization to overcome the QRs problems:

- unambiguously specified the QRs
- capacity allotment to work on QRs
- automate QRs
- service level objectives dashboard (SLO) - response time, resource utilization monitored via SLO dashboard)
- in-Progress: security/to ensure zero attacks via API gateway
- security related requirements covered via Microservices style architecture (authorization layer and security)
- discovery phase to identify QRs (specifically performance requirements) upfront
- inception phase to identify QRs upfront

> *what we have done is we have tried to automate all the non-functional requirements. […] we have a couple of tools. […] SLO dashboard. [MEL_P11]*

> *so when it comes to performance for example, when we do that discovery, we do the technical feasibility. […] So the way to make request so the architecture that we are implementing now; is a cloud based micro-services. And in that architecture, the only way that the service can make a request with the service is by going through authorization layer and security. […] we also have something in terms of SLA is we do have a dashboard that captured SLAs. [MEL_P16]*

> *we cover as part of our inception activities determine what NFR we will require. [MEL_P12]*

5.8.2.7 Difficulty in estimating effort

Participants asserted that it is difficult to estimate the effort upfront. Sometimes effort that is required to implement a requirement(s) is underestimated.

> *the challenge with that is the effort part of it is not a It's not a science, so there is no such thing as software estimation. [MEL_P11]*

> *we sometimes get caught in underestimating. [MEL_P8]*

5.8.2.8 Inadequate requirements analysis

It emerged from the case study data, that insufficient understanding of the candidate requirements. often yields inadequate prioritization. As a consequence of this, often a feature is built where the business value wasn't there or there wasn't the appropriate priority

> *I think there's been some instances where perhaps, you know, we've gone too quickly into sort of delivery mode or you're moving on something without sufficient discovery or sufficient prioritization. And then we realize at the end that what we've delivered wasn't the priority. [MEL_P5]*

Therefore, the right amount of effort and collaboration is required in order to make an efficient decision on the priority of requirements. Participants asserted that they are working towards in-depth understanding of the potential requirements, before commencing them for implementation and provide a solution that is easy to consume by business users. One of the recently adopted approaches in this direction is introducing a ***discovery phase*** (for detailed discussion on discovery phase see in Section 5.4) where product and engineering work collaboratively to identify all possible unknowns and risks (i.e., usable, valuable, and feasible) that are associated with a business problem before it is committed for delivery.

> *you know we have to just dig in, get more information and figure out what's really going on in each of these cases.. [MEL_P7]*

> *I think that's something that we're really trying to work on […] sufficient effort and collaboration in that discovery process in the building the roadmap out. So that we are building the right thing, and trying to minimize the instances where the business value wasn't there or wasn't the appropriate priority. [MEL_P5]*

> *I think we still rushing things a bit. I mean we did better than we used to […] So yeah, so more discovery and better collaboration and getting ahead of things so that by the time we start working on something, those risks [usable, valuable, feasible] I mentioned earlier, are alleviated. [MEL_P16]*

5.8.2.9 Lack of requirements clarity

Participant MEL_P6 asserted that the team often lacks clarity, specifically on the requirements that are needed to implement to unblock the other team. As a consequence of this, the team often face difficulty in managing their own work vs work that is required to unblock the other team.

> *we often have a team that has to build something that they don't even fully understand what it's important because it's important for another team. […] you know that team is probably trying to juggle […] their domain versus other work. [MEL_P6]*

Therefore, a continuous communication and collaboration mechanism is required to manage this issue. Some of the strategies that are employed include:

- **Escalation matrix:** as an example - within the team, it should be the PO's responsibility to provide clarity to the teams.
- **Teams' catch-up:** Participant MEL_P2 explained that all the teams who are involved in a particular feature implementation, typically have a fortnightly common meeting that helps in building healthy relationship between teams. The healthy relationship between teams unblocks the communication barrier between them and enables them to approach each other wherever needed (e.g., clarity of requirements).

> *once a fortnight is OK for them to catch up and to build relationships with the other engineers so they can approach each other directly. [MEL_P2]*

> *PO responsibility to provide clarity to our engineering team. [MEL_P9]*

5.9 Summary

This chapter described the prioritization process and the process specific practices that are employed by MEL organization, to make decision on the priority of requirements across scaling agile levels. In addition to this, challenges that are faced by the studied case organization while making decisions on the priority of requirements and the potential strategies that can be employed to surmount those challenges, are discussed.

Findings that emerged from the studied AKL organization are discussed in Chapter 6.

6 Analysis and Findings- AKL organization

This chapter presents the analysis and findings that emerged from the AKL case study. First, company background is provided. Presentation of findings are started with a high-level overview, followed by detailed discussion on the decision-making levels and the requirements prioritization activities that emerged, via analysing the AKL case study data.

6.1 Case overview

The context of studied case organization is described via adapting the guidelines provided by Petersen and Wohlin (2009). The contextual facets include organization, product, process, people, and market are described, to provide the thick rich description of studied case organization's context, as discussed below and summarized via Table 6.1.

The AKL organization is one of the largest natural gas and electricity retailers in New Zealand. The studied case organization has undergone a transition towards scaling agile development via adapting Scrum @ Scale framework during 2017 and was continuously evolving scaling agile practices, at time of conducting this research study as well. The AKL organization has 15 engineering teams that are distributed over two nationally distributed sites that include Auckland and Hamilton. The studied case organization has its head office in Auckland, and most of the engineering teams are also in the Auckland site. Therefore, majority of the participants were interviewed from the Auckland site to understand the phenomena of interest (i.e., requirements prioritization process). However, participants from the Hamilton site were also interviewed to understand distributed aspects of the studied research phenomena.

Table 6.1

Context description of AKL case organization (Petersen & Wohlin, 2009)

Context facet	Context element	AKL case organization
Product	Domain	Utility
	Product size	Large
	Maturity of product	Long-lived mature product
	Customization	No (enterprise product)
Process	Characteristics of software development process	Scaling agile practices, Adapted scrum @ scale framework
	Start of method adoption	2017
	Distributed development	Nationally distributed
	Team size	Up to 13 members in a team
Organization	Size	Large (500+)
	Number of development teams	15+ delivery teams
	Hierarchical organizational model	Yes
	Organizational unit involved in the study	Product and engineering
People	Roles and their responsibilities	Cross-functional decision-making team
Market	Setting	Enterprise product (in-house development)
	Access to customers or end users	Internal customers (in-house development), key end users

6.2 Overview of decision-making levels and associated requirements prioritization activities

This Section provides a high-level overview of decision-making levels and the requirements prioritization activities that are performed at each of these scaling agile decision-making levels. This is achieved by employing the initial conceptual framework discussed in Chapter 4: Section 4.4.3.

6.2.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021). In this section, ideas and requirements are used interchangeably to denote the potential requirements that could end up with big initiatives or work streams, that generally get assigned to the teams or team of team for the implementation. Sources of potential requirements are classified into two categories: internal sources and external sources.

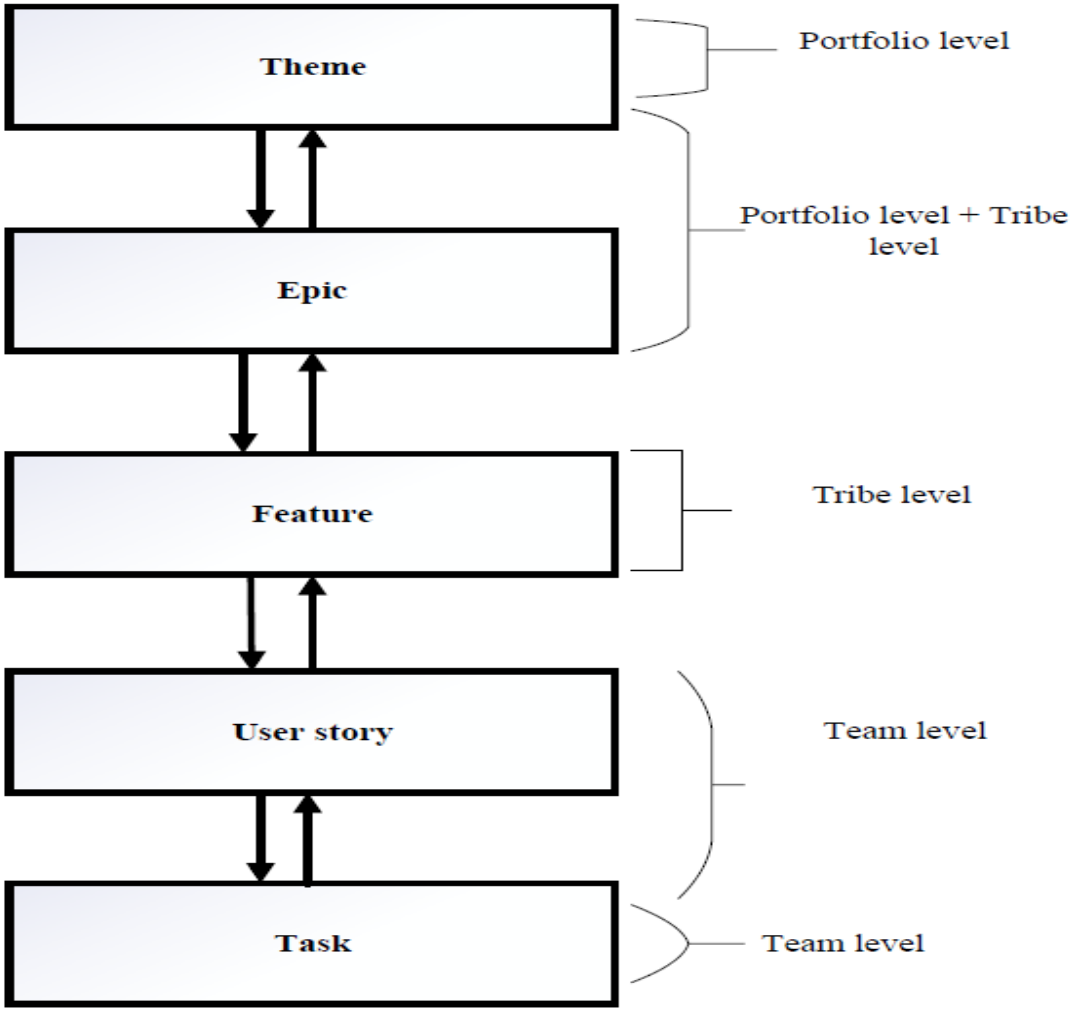
- Internal sources mean generating or collecting the potential requirements from/within the organization (e.g., senior product owners, engineers etc.) (Damian et al., 2021)
- External sources mean generating or collecting the potential requirements from the outside of the organization (e.g., market, customers etc.) (Damian et al., 2021)

6.2.2 Requirements organization and classification

The case organization employed a hierarchy model to organize the requirements. Figure 6.1 shows the requirements hierarchy and their associated scaling agile decision-making levels.

Figure 6.1

Requirements hierarchy model and scaling agile decision-making levels



6.2.2.1 Requirements hierarchy model

Theme(s) reside at the top in the requirements hierarchy. Theme(s) represent the high-level business goals that connect the portfolio with strategic territory, where investments are allocated. The organization had six major themes that were set at the beginning of the year 2021.

> *they have a number of categories. I think right now we have 6 categories [themes]. one is like energy management in customer in hands. Which is a strategic theme, when I joined and it's still a strategy. So, we are kind of giving control of energy management in customers hand. [AKL_P2]*

> *on their strategic side we have I think we have six categories now. [AKL_P3]*

Themes are generally stable and have an influence on the decision making at all the levels of requirements hierarchy. A theme is a collection of ***Epics*** that drive towards a common goal. The number of Epics that belong to each of the themes varies, but there were in total 150 Epics that were proposed and distributed over 15 engineering teams, to implement during the second quarter (Q2) of 2021.

> *so just to give you some idea in one in the main I suppose the core retail we have 150 prioritized epics. [AKL_P6]*

The second level in the requirements hierarchy is an ***Epic(s)***. An Epic is a large development initiative that realizes the value of strategic themes and can take longer than one release IR (i.e., three months). Typically, more than one engineering teams that can go up to five teams are involved in an epic implementation.

The third level in the requirements hierarchy is a ***Feature(s)***. A Feature is something that provides concrete value to the customers and business and is generally independent of other features. A feature can be defined as a service that is requested by the stakeholders. A feature usually requires one engineering team and can be implemented in a single sprint (i.e., two weeks). However, more than one engineering team can also be involved in a feature implementation if needed.

At the fourth level, a feature(s) is decomposed into ***User story(s)*** to be developed in a sprint. A user story is the quickest sort of element deliverable through a thin pipeline that can give some valuable feedback.

The fifth level is ***Task(s)***, that describes in technical terms what things are required to be done in order to claim a user story at the end of a sprint.

we've got an epic which is your highest form, so it's a large chunk of work that seem broken down into features, which is then broken down into user stories, which is then broken down into tasks. [AKL_P5]

It goes from an epic, then features, then user stories and tasks. [AKL_P8]

6.2.2.1 Scaling agile decision-making levels

The case organization has a three-layer model of agile enterprise. The three layers are: Portfolio level, Tribe level, and Team level.

Portfolio level: At the portfolio level, a ***Theme(s)*** that resides at the top in the requirements hierarchy, is defined that connects the portfolio with the strategic territory and the ***Epics*** that are required to realize a theme, are formally ranked and scored across the function of business by the ***Governance forum***. As a result of this level, a portfolio backlog is constructed that contains themes and their associated epics that are distributed over ***tribes*** to implement them. The case organization has one product portfolio: 'retail core' portfolio, where energy management products and services are built, that were studied to understand the research phenomena. The portfolio level and the type of requirements that are associated at the portfolio level are discussed in detail in Section 6.3.

Tribe level: the product portfolio (i.e., the retail portfolio) is divided into tribes. A tribe is a kind of domain/technical product area where each of the tribes has delivery teams who are generalizing specialist teams, in a particular domain.

We have sub-tribes underneath retail tribe. […] business tribe. [AKL_P8]

The retail portfolio that is studied for this research study, has five tribes (e.g., *digital new services tribe*, *residential tribe*). The tribe that is studied is *digital new services,* as most of the development happens in the digital tribe. Each of the tribes has a General Manager (GM) who is generally acts as a boundary spanner between the portfolio level and the tribe level. As a part of the portfolio level discussion, the GM typically presents the ideas/epics that are emerging from their tribe and/or contribute to theme(s) formation.

I suppose digital and technology space which is actually where a large proportion of the delivery actually happens from those squads and teams. [AKL_P6]

The Twofold Role of Tribe:

First part of tribe: At the first part of tribe level, ***Epics*** that are required to realize a theme and/or contribute to theme formation, are defined, analysed, and submitted to the ***governance forum*** via GM where epics are formally ranked and scored across the function of the business. Each tribe has a senior Product Owner (PO), who typically collates the potential ideas/epics, defines and validates the potential ideas/epics that are required to realize a theme and/or contribute to the theme(s) formation.

Second part of tribe: At the second part of tribe level, epics that are formally signed off via *Governance forum* and allocated to the engineering teams, are broken down into features. Typically, the senior PO is responsible for decomposing an epic into features with inputs from all possible stakeholders. As an outcome of this level, a *product backlog* (Schwaber, 2004) is constructed that contains a priority list of features.

Tribe level and the requirements that are associated at the tribe level in terms of decision making on their priority, are discussed in detail in Section 6.4 and Section 6.5.

Team level: At the team level, a feature(s) is broken down into ***user story(s)*** and ***task(s),*** if required. Typically, the engineering team (i.e., product owner, business analyst, scrum master, engineers), who own the feature implementation, break down a feature into user stories and tasks. The team level and the requirements that are associated at the team level in terms of decision-making on their priority, are discussed in detail in Section 6.6.

6.2.2.1 Requirements classification

Requirements are classified into five categories: (i) Compliance (ii) Essential Maintenance (iii) Growth and Efficiency (iv) Discretionary Maintenance (v) Transform

- ***Compliance*** represents the work that is considered as mandatory otherwise they face breaking the law; for example, price changes.
- ***Essential Maintenance*** represents the work that is considered as mandatory that results in stopping work if it's not resolved; for example, an issue with pricing-charging customers at the wrong rate.
- ***Growth and efficiency*** represent the work that is performed to deal with new products and features. E.g., energy saving tips.
- ***Discretionary Maintenance*** represents the work that may be considered as optional; for example, tech-debt work, internal improvement work.

- ***Transform*** represents the work that is needed to transform the business.

> *so we have a category. You know that essential maintenance. Is it something to do with transform? It's a new idea? Is it a regulatory or compliance so we have I think 5 different categories that we use. [AKL_P6]*

> *We also look at platforms because we actually need to make sure our platform to maintain and libraries up-to-date otherwise. Otherwise you're without doing sort of the technical debt part of it. Your stack falls a pretty quickly, so it's not all about the new stuff and we have some really big things. Will also put it in a benefit case in there. [AKL_P2]*

> *there is a bunch of stuffs we have to do, so regulatory that stuffs, plus price changes, we have to do those things. Plus anything that come out from a legal perspective etc. [AKL_P4]*

Participants asserted that compliance work and essential maintenance work, always gets higher priority than other requirements., Growth and efficiency, Discretionary Maintenance, Transform. In most of the cases though, compliance and essential maintenance work have negative net present value (NPV) associated with it. Despite this, the requirements' get higher priority than other categories, otherwise the organization might face penalty charges, or something will stop working in production.

> *So that the Governance forum does know this particular epic [compliance/regulatory work] doesn't give you any profit or benefit, but still they have to do it otherwise we will face breaking the law. [AKL_P2]*

> *essential or compliance, so that's something that business must do right. You get your must do's and then you got the stuff you would like to do is. Right, so if it, and that's why often, what might come in at the top, there's a new say we find a bug or something with our pricing, and we've been charging customers the wrong rate. Making this up, by the way, but let's just say that happened or there was change- the regulator change, make sure we're following compliant. [AKL_P3]*

6.2.3 Requirements prioritization and negotiation

During this activity, requirements are evaluated and the decision on the priority of requirements is made. The decision-making team, decision-making factors, trade-off while making decisions on the priority of requirements, decision-making event(s), decision-making technique(s) typically emerged at each of the scaling agile decision-making levels. These yield requirements prioritization and negotiation in the studied case organization.

6.2.4 Requirement artefacts

During this activity, requirement artefacts, such as a business-case, are produced to capture relevant information that typically serve as inputs during decision making on the priority of requirements and shared with delivery teams for implementation.

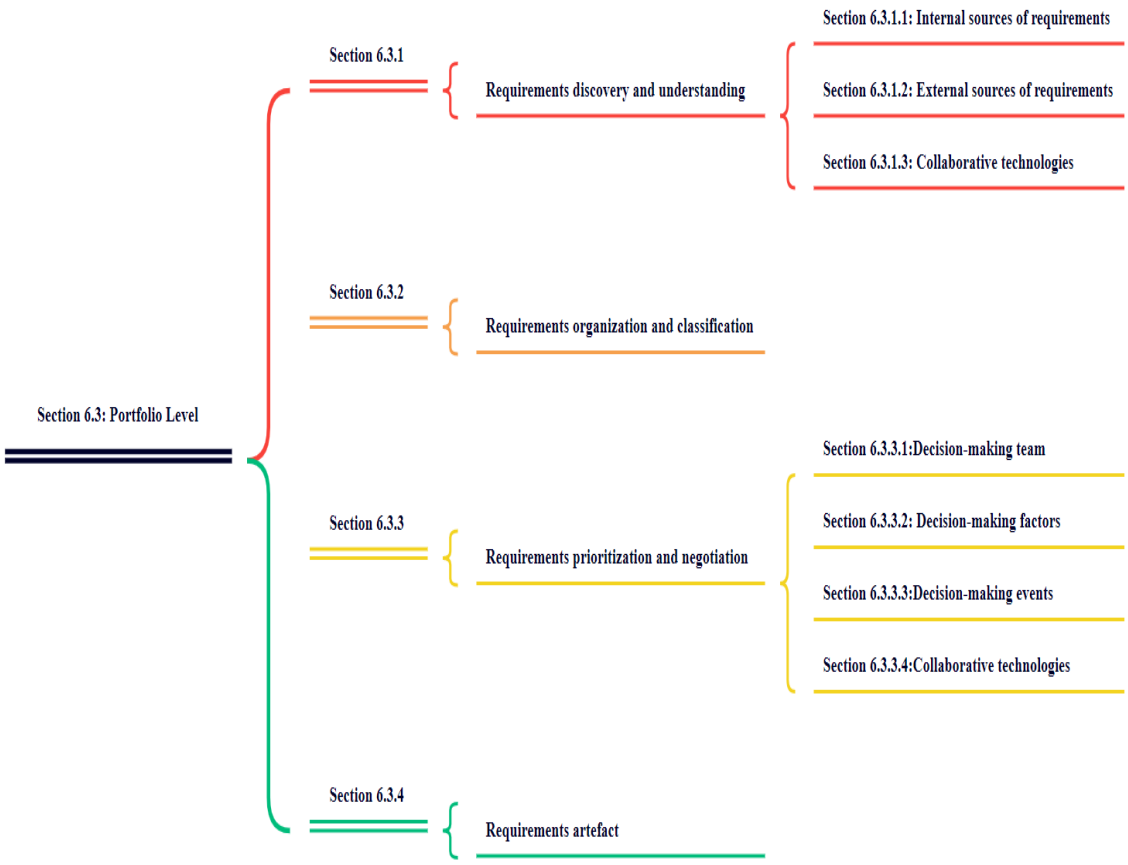
6.3 Portfolio level

In the case organization, portfolio level is the level where requirements are typically defined and analysed in the form of the organization's strategic objectives and/or themes, that generally influence the product(s) which an organization ought to develop. In addition to this, requirements such as epics that (discussed in Section 6.4) are posed on the product(s) are formally signed-off at the portfolio level, by the ***Governance forum.*** They ensure alignment of requirements (i.e., epics) with the organization's strategic objectives.

Figure 6.2 shows the mechanisms that are discussed at the portfolio level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 6.2

Snapshot of activities discussed at the portfolio level



6.3.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021). Following are the sources of requirements.

6.3.1.1 Internal sources of requirements

Within the portfolio level, typically the ***Governance forum*** is the main source of potential requirements.

Governance forum consists of the senior leadership team that includes Chief Digital Officer (CDO), Chief Customer Officer (CCO), General Managers (GMs) of all tribes where the CDO typically brings technical needs, CCO typically represents customer facing needs, GMs bring requirements that are emerging within their tribe.

> *We have governance forum who is basically continuously working on this direction. […] its basically our chief technical officer, chief customer officer. But its not like*

they are the only one in this forum. They work with GMs as well to form themes. [AKL_P3]

they [idea] come from like our executive leaders [i.e., CCO, CDO] often have ideas. [AKL_P8]

so obviously you get a fair amount of input from your leadership team our exec [Governance forum] are fairly invested in our […] products, which is both good and bad depending on how you look at it. So ideas will come from them as well. [AKL_P4]

However, potential requirements emerge from the members of other levels and/or team(s) such as engineers, support team, UX/UI team, customer success manager(s), product owners(s), sales, and marketing.

6.3.1.2 External sources of requirements

Generally, the market, including competitors, customers and partners, are the external sources of potential ideas/requirements (detailed discussion is in Section 6.4). At the portfolio level, CDO and CCO typically collate the emerging business and customer needs respectively.

But Ideas predominantly comes from the CDO and CCO. [AKL_P6]

Techniques that are employed to gather potential ideas/requirements from the external sources are not isolated to the portfolio level only. The same technique, customer interviews, can be employed at the other levels - tribe level, team level, and/or teams: support team, UX/UI team, sales and marketing team, to discover potential customers' ideas/requirements. Potential techniques that are adopted by the case organization to gather potential ideas/requirements from the external sources, are discussed in Section 6.4.

6.3.1.3 Collaborative technologies

Collaborative technologies (CTs) - ADO[13] (Daniels et al., 2015) is a formal place to capture potential requirements that are emerged at the portfolio level.

ADO is the tool that we use where all the potential ideas are formally placed. [AKL_P4]

6.3.2 Requirements organization and classification

As discussed in Section 6.2.2, the case organization employed a five level of requirements hierarchy model.

[13] Information on ADO can be accessed via https://azure.microsoft.com/

Strategic Theme(s)*→ *Epic(s) → Feature(s) → User story(s) →Task(s)

Primarily ***Strategic themes*/themes** belongs to the portfolio level in terms of decision making on their priorities. In addition to this, requirements or epics, that are typically defined and evaluated at the first part of tribe level in order to accomplish the strategic themes, are formally signed-off at the portfolio level.

As discussed in Section 6.2.2, the case organization has classified the five categories of requirements, that includes compliance work, essential maintenance work, Growth and efficiency work, discretionary maintenance work, and transform work. Typically, the requirements, such as compliance work, essential maintenance work, Growth and efficiency work, discretionary maintenance work, and transform work, that require significant investment and/or effort formally decided at portfolio level.

> *So anything, that looks like one or two plus months of work we have to put in a mini business case. And then it will go through the formal process. […] something as small as little, or basically I'll just fix that and no one will see it. […] something small enough that like I said [tech-debt work, enhancement work], it was one week or less, work will just do it and get that prioritized the next Sprint also. [AKL_P2]*

> *so we have a category. You know that essential maintenance. Is it something to do with transform? It's a new idea? Is it a regulatory or compliance so we have I think 5 different categories that we use. [AKL_P6]*

6.3.3 Requirements prioritization and negotiation

6.3.3.1 Decision-making team

The governance forum, which is a cross-functional decision-making team, works collaboratively and continuously for deciding the strategic themes. Governance forum consists of senior leadership, that include CDO, CCO, GMs where CDO typically brings technical and business needs, CCO typically represents customer facing needs, and GMs bring requirements that are emerging within their tribe.

> *We have governance forum who is basically continuously working on this direction. […] its basically our chief digital officer, chief customer officer. But its not like they are the only one in this forum. They work with GMs as well to form themes. […] CCO represents customer needs and CDO represents technical and business needs. [AKL_P3]*

6.3.3.2 Decision-making factors

Alignment with company's overall strategy and, opportunity (i.e., customer opportunity, market opportunity), are typically consider in order to make decision on strategic

theme(s). Generally, the strategic themes which are holding the biggest opportunity, are selected for further exploration.

> *we'll look at the size of the customer opportunity and the size of the market opportunity. And then we'll rank those strategic areas. [AKL_P1]*

> *we [governance forum] look at various things, you know, alignment with company's overall strategy, customer opportunity. [AKL_P3]*

6.3.3.3 Decision-making events

The following are the decision-making events that are occurred at the portfolio level.

Strategic Portfolio sync-up: this event generally occurs on a quarterly basis where portfolio management collaboratively decides and revises strategic themes.

> *we've historically reviewed them [themes] every 1-2 years but this varies depending on the market. [AKL_P1]*

> *basically, themes are like business goals. [AKL_P2]*

> *Strategic themes we don't revise frequently. But if needed we have strategic portfolio sync-up every quarter where our leadership team revise them. [AKL_P7]*

Bigroom planning: This event also occurs on a quarterly basis, which is a one-day event, but the scope of this event is broader than strategic portfolio sync-up. During this event, strategic themes and the epics that are needed to accomplish strategic themes, are formally decided by the Governance forum and communicated to the engineering teams for implementation.

However, decisions on the priority of requirements are made ahead of the *bigroom planning event*. The planning for the upcoming quarterly bigroom planning meeting, typically starts in the fourth sprint of current release IR. This event typically yields shared understanding of priority of requirements among product and engineering.

> *We have something called bigroom planning. So we just had that [bigroom planning] recently because we've well, sort of going into a new financial year. […] retail strategy refresh so the executives for retail did that. […] So it should be the whole company, everyone can watch, but it's also recorded so you can if you'd miss that timeslot, you can go and watch it afterwards [AKL_P10]*

> *Its [bigroom planning] more of a chance to get together and talk about dependencies and share the plan we've already created. Yes, it's more about sharing the plan rather than creating the plan. […] Five or six hours and they do basically move around? It's a bit of a social event as well. [AKL_P3]*

> *We do what's called bigroom planning. And that is 3 months of work, so we tend to have a three month increments of work. [AKL_P7]*

Planning starts normally in the 4th or 5th sprint of the current release IR. [AKL_P8]

Portfolio sync-up: This is typically organized on a fortnightly basis for check-ins that generally provide visibility on the progress of requirements (i.e., epics), towards achieving business objectives/strategic themes, revisit existing requirements, removing impediments, scoring and ranking of new requirements, addressing dependencies and any changes in the priorities of requirements.

> *every two weeks we refresh, new Epics come in- existing Epics might get rescored and things move up and down. We have a little particular saying up 3 -- 10 you know and you can see your backlog is alive thing; the squads will review this life backlog and make sure that their future next work item reflecting the highest priority that they can address. [AKL_P3]*

> *these guys [governance forum] meet every two weeks to see this prioritization; I mean they talk about other things as well. So yeah, they look after the budget for the retail tribe OK and And they're responsible for the outcomes that the retail tribe delivers the value. [AKL_P1]*

6.3.3.4 Collaborative technologies

At the portfolio level, CTs ADO (Daniels et al., 2015) are employed that provide visibility across the entire portfolio and enable tracking the progress of which stage a specific requirement is at (i.e., discovery/ready to develop/in development). The integration feature of ADO with other CTs, such as Confluence enable traceability and shared understanding of requirements is used across all the scaling agile levels.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

> *so the teams use Azure DevOps (ADO). Just say it to set up the sprints and to set up the I guess asks everyone just works off that. they also use confluence to documents information. so, a combination of those then I guess yeah, you individually the different people have their own tools like the developers and the designers have their own tools, but those are I guess the workflow ones. AKL_[P10]*

6.3.4 Requirements artefacts

The following requirements artefacts are built at the portfolio level, that are specified via CTs- ADO, Confluence, PowerPoint (Daniels et al., 2015).

Theme* (an electronic template configured via CTs- *ADO*):** Theme is the main requirement artefact that connects the portfolio with strategic territory, and which is built at the portfolio level. Themes are generally defined in the form of a phrase, one or two lines. Themes are generally stable and act as a ***boundary spanning requirement artefact that have an influence on decision making at all the levels of requirements hierarchy.

An electronic template which is configured via CTs *ADO,* is followed to specify the themes. However, CTs *PowerPoints* slides for themes (a sample of strategic theme is shown in Table 6.2), are also maintained and are specifically used during decision-making events like bigroom planning, for knowledge sharing.

Table 6.2

Sample of strategic theme from AKL organization

Strategic Theme Example[14]
Theme: Retail digital transformation: Invest in technology and data to create consistent and distinctive end to end experiences.

Every year typically around ten strategic areas are validated. However, one or two strategic areas are selected for further exploration such as in concept refinement- generate lots of ideas/epics for the selected strategic area.

> *it basically we have some strategic areas that we'd like to focus on. And so this is kind of where the team that I'm embedded in at the moment kind of plays. We look at the strategy and we go OK out of, these five big areas we'd like to work in. we would do market research and customer research on those board areas. And then in that strategic area, we might pick one or two that we think has got the biggest opportunity and then We'll explore that a bit more, […] We might be validating 10 ideas a year and maybe one or two of those will scale [AKL_P1]*

> *So we have this idea of strategic territory. We have themes within this strategic territory and then we have concepts. So that's kind of it's sort of connects what the thing you might do is a project or a change back to is it within the right sort of strategic filtering of your business. [AKL_P3]*

> *we've historically reviewed them [themes] every 1-2 years but this varies depending on the market. [AKL_P1]*

> *there is a template that we follow for themes. […] it's in ADO. [AKL_P8]*

[14] This example is taken as a snapshot from the PowerPoint slides that were used for knowledge sharing decision-making event (bigroom planning).

Portfolio backlog (electronic artefact configured via CTs- *ADO*): Portfolio backlog is another requirement artefact that contains a priority list of requirements (i.e., themes and epics) built at the portfolio level. CTs - *ADO* is a formal place employed by the studied case organization to build portfolio backlog.

> *we use the ado board as you know which is got the ranks list of epics on it. [AKL_P8]*

Confluence: CTs - *confluence* is employed to maintain documentation of requirements, for detailed discussions and/or analysis on requirements such as recordings of discussion with customers, that help in building the shared understanding of requirements. The integration feature of CTs *confluence* with *ADO* provide traceability of requirements.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

> *so the teams use Azure DevOps (ADO). Just say it to set up the sprints and to set up the I guess asks everyone just works off that. they also use confluence to documents information. so, a combination of those then I guess yeah, you individually the different people have their own tools like the developers and the designers have their own tools, but those are I guess the workflow ones. AKL_[P10]*

6.4 First Part of tribe level

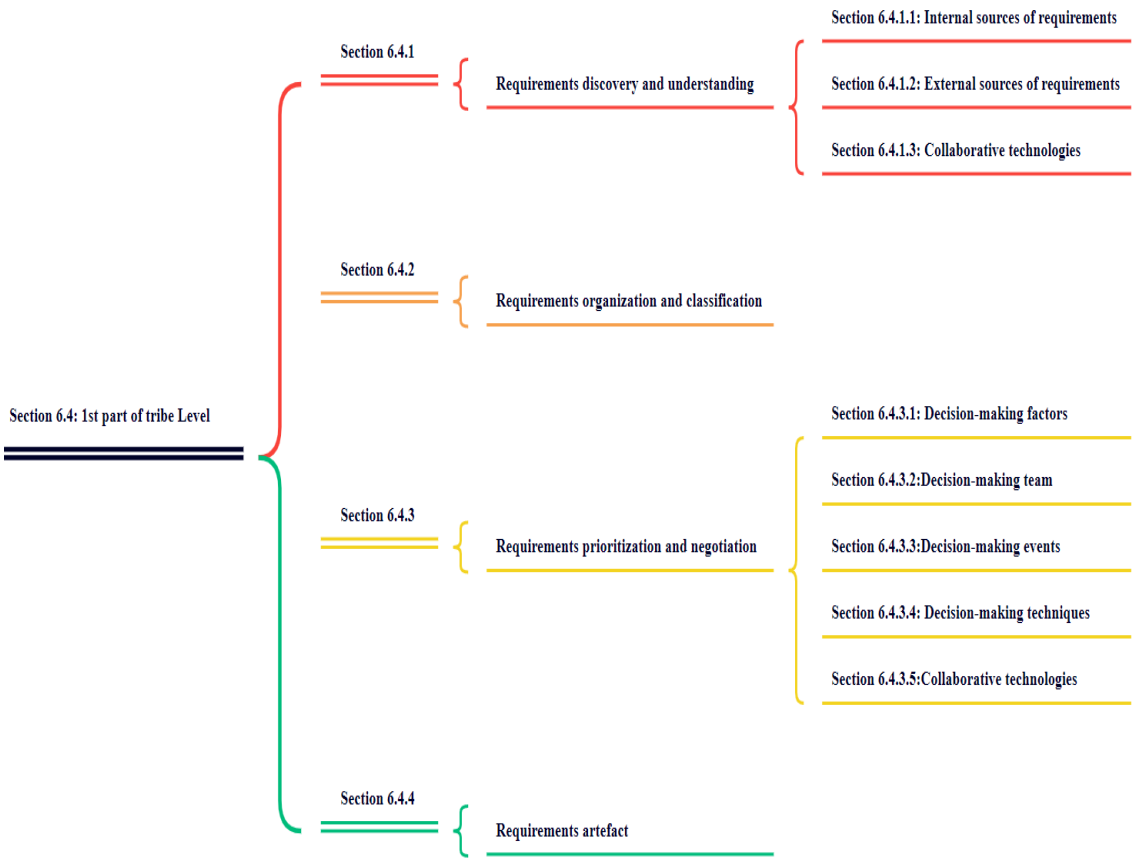
Introduction: In the studied case organization, product is typically classified into tribes. A tribe is a kind of technical product area area/domain, like the digital new services tribe, where teams are generalizing specialist teams in a particular technical product area/domain.

Role of tribe is twofold, in the first part of tribe, potential requirements (i.e., epics) are defined and analysed, and submitted to the *governance forum* via GMs for formal sign-off. GMs are act as a boundary spanner in terms of communicating the requirements between the portfolio level and tribe level. GMs are the one who typically communicate any change in the decided priorities if occurred on the behalf of governance forum and vice versa.

Figure 6.3 shows the mechanisms that are discussed at the first part of tribe level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 6.3

Snapshot of activities discussed at the first part of tribe level



6.4.1 Requirements discovery and understanding

An open innovation approach is employed by the case organization where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021). Following are the sources of requirements.

6.4.1.1 Internal sources of requirements

Typically, the senior PO, solution architect, tribe lead, senior UX/UI wherever needed, generally work collaboratively to collate and/or understand the potential ideas/requirements that are posed on product(s). But senior PO is the dominant internal source of ideas/requirements. In addition to this, potential requirements/ideas are collated

from the other levels and/or teams, that include portfolio level, team level, sales and marketing team, and support team.

> *Ideas come from various places like support team, marketing team, sales team, developers, testers etc. but senior POs are the main sources. [AKL_P4]*

> *Multiple sources we have who brings ideas. like product owners, engineers, support people. [AKL_P2]*

Practices to collate potential ideas/requirements from the internal sources:

Various practices are employed by the case organization to collate potential ideas/requirements, as explained below. Participant AKL_P1 reports that practices that are used to collect ideas from internal stakeholders are least formal.

> *We don't. I've seen in other companies they have like a, you know, a web form or something you can fill it. Yeah, yeah, here's my idea. We don't have that. [AKL_P1]*

Run hackathons: participants mentioned that occasionally, generally on a yearly basis, hackathons are run, where generally the POs and engineers collaborate on potential ideas/requirements.

> *So we run things like hackathons. [AKL_P1]*

> *they arrange hackathons as well. They started doing it once in a year. […] its POs and engineers. [AKL_P9]*

Business strategy: it is mentioned by the participants that potential ideas/requirements that are posed on the product, are generated from the business strategy as well.

> *We also look at high level strategy which says in general we want to be playing in these areas in the future, right? So that high level strategy gives us some ideas about really weird stuff, but it also helps us to direct our ideas. [AKL_P1]*

> *Part of it is where company strategy wants to go as well. [AKL_P2]*

Champions across the organization: Participants mentioned that delivery teams generally have champions who collate and represent their delivery team' members ideas. For example, delivery teams' ideas are communicated via their POs.

> *You know they [team] came up with an idea and so we feed it basically up the food chain so we talk to product owners in the room and we talked to our senior manager who then talk to the people that the other sort of general managers. [AKL_P8]*

> *we have kind of champions across our customer excellent centre who would then feedback to PO and GMs. [AKL_P4]*

6.4.1.2 External sources of requirements

As indicated in Section 6.3, market typically the competitors and customers, are the external sources of potential ideas/requirements.

Practices to collate requirements/ideas from the external sources:

Market: The case organization typically does competitors' analysis/market research to track current market trends and/or capture ideas and/or understand the emerging needs of potential customers.

> *We also do research from the point of view of people who want [our] customers. So we looking at the market. [AKL_P4]*

> *So they look at the competitors' product as well. [AKL_P9]*

Customers: Customers are the dominant source of potential ideas/requirements. Collating the potential ideas/requirements from the customers is a continuous process. Customers' potential ideas/requirements are typically communicated via customer services team, sales and marketing team, support team, UX team, and senior POs.

> *They [senior POs] go and test the waters a little bit so they come up with a bunch of ideas and those things could be Electric vehicles, it can be solar charging and they sort of go out and talk to customers and they figure out some idea about what customers actually want. [AKL_P2]*

> *so we have For example, the call centre people are called up- they Record that information. [AKL_P1]*

> *So if there is something that a customer sees and quite often that will come in throughout our contact center, then those ideas will bubble their way up as well. [AKL_P5]*

> *We have sales team, we have marketing team, support team who can also communicate with customers to collect their ideas. [AKL_P7]*

The following are the practices to collate customers' potential ideas/requirements:

Online survey: participants mentioned that an online survey both regular and ad-hoc surveys are conducted to get customers' feedback.

> *We had online surveys, we send both regular and ad-hoc surveys. [AKL_P1]*

Customer interviews: participants asserted that a single face-to-face interview may be conducted to get customers' feedback/ideas.

> *interviewing customers or talking with customers, or reviewing feedback or something that we've said well, is the problem here or an opportunity. [AKL_P3]*

we also do customer interviews. [AKL_P1]

UX research: participants indicated that they have an UX group for user research. As part of user research customer journey, mapping is conducted to identify customers' pain points and opportunities.

they do some user research on those ideas. […] so it's more of our basically UX designers who do the user research. [AKL_P9]

we have a UX lead who's joined. So she's helping with customer journey mapping to identify pain points and Opportunities. [AKL_P6]

Customer panel: participants mentioned they have a panel of customers, around 10 to 15K customers, who have agreed to give feedback on the company's product or the potential new ideas.

We have a panel of customers who are, some of whom are [company's] customers, but some of whom are other companies' customers. [AKL_P1]

We go out and speak to our panel. We are lucky enough I think, we have a panel of somewhere in the region of 10 to 15,000 of customers who have signed up and said yeas, we are happy to answer questions. So we talk to them. [AKL_P4]

Feedback via website: Participant AKL_P1 mentioned that there is a feedback pop-up on the company's website through which customers can give their feedback on the services that are provided by the organization.

we have feedback on our websites which pops up and says what do you think about this service We do better. [AKL_P1]

App store: The app store is another channel through which customers' feedback is collected.

we look at things like app stores, you know, get the feedback. [AKL_P1]

6.4.1.3 Collaborative technologies

At the first part of tribe level, CTs ADO (Daniels et al., 2015) is a formal place to capture potential requirements/ideas.

ADO is the tool that we use where all the potential ideas are formally placed. [AKL_P4]

6.4.2 Requirements organization and classification

As discussed in 6.2.2, a five level requirements hierarchy model is adopted by the case organization.

Theme(s) → ***Epic(s)*** → Feature(s) → User story(s) → Task(s)

In the studied case organization, potential requirements, like compliance work, discretionary work, maintenance work, that requires significant effort and/or investment of two-to-three months of work, are typically termed as epics. In the first part of tribe, ***Epic(s)*** are defined and analysed, and submitted to the *Governance forum* for formal signoff.

6.4.3 Requirements prioritization and negotiation

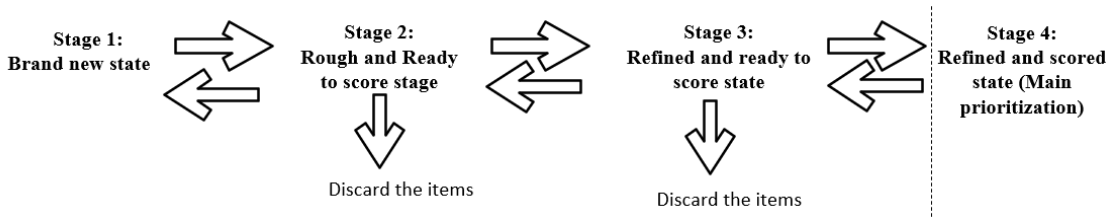
As mentioned above, initial decision making on the candidate requirements, or epics that emerge from each of the tribes and/or contribute to theme(s) formation, is typically made in the first part of tribe level. This initial decision-making goes through various stages as shown in Figure 6.4 before submitting them to the *Governance forum* for formal decision making. At each of the decision-making stages new potential requirements can emerge that are initially pushed into Stage 1. In addition to this, the potential requirements can be pushed back to the previous stage for further analysis if the provided decision-making information is not sufficient.

> *That epic can be in a number of phases of its life cycle. it can be new. So that's an idea that's very rough as being formed up. Then we have a rough and ready to score or refined and ready to score. So we have this idea that actually before any work actually happens or is actually scored to commence, it needs to be refined. [AKL_P3]*

> *we have several states that the epics go through. [AKL_P1]*

Figure 6.4

Epic decision-making stages



Stage 1: All the potential requirements that are collated via requirements discovery practices (described in Section 6.4.1) are typically at a brand-new state during Stage 1.

> *They'll [epics] go basically from a brand-new epic. [AKL_P1]*

Stage 2: Rough and Ready to Score: The potential ideas/requirements are evaluated at the high level, typically against organizational strategy, the business goals and/or themes. During the *rough and ready to score* state, the potential idea/requirement(s) generally has ***low confidence*** and has +/- ***50% to 100% margin of error*** associated with it. As an outcome of this stage, either the candidate requirement(s) will be discarded or will be moved into Stage 3 for further evaluation. However, a priority list of requirements will not be produced in the Stage 2 but the candidate requirements that are valuable for the organization, will be filtered out typically based on their alignment with the organizational strategy and themes.

> *then they'll go through basic or rough scoring. Some rough sizing from the teams really high level and you know it might take them half an hour or may be one hour to do that. Yep, there'll be some rough business case kind of benefits assigned to it. So the margin of error on the rough one, maybe a 50 to 100% yeah right. [AKL_P1]*

> *the rough and ready to score state its like evaluate them [epics] at high level. atleast they [epics] should have alignment with one or two strategic themes to reach to next level. [AKL_P5]*

Stage 3: Refined and ready to score: Candidate requirements that are filtered as a result of Stage 2, are refined and then detailed information is produced for decision-making. During the *refined and ready to score* state, the candidate ideas/requirements generally have high confidence and have +/- 20% margin of error associated with it. Priority score of potential ideas/requirements is evaluated by using whole set of decision-making factors, such as strategic fit, and NPV which are described in Section 6.4.3.1. This is an automated matrix configured via *CTs- ADO*.

> *margin of error on the refined state is maybe 20%. [AKL_P1]*

> *Its [score calculation of epic] in ADO, this whole calculation done automatically by that tool [ADO]. [AKL_P4]*

During this Stage 3, a formal name is assigned to the candidate requirement(s), either simply labelled as implementation epic, which means it is ready for implementation, or discovery epic, which means that further evaluation is required to make a decision.

As an outcome of Stage 3, either the candidate epic(s) will be discarded, or the candidate epic(s) will be accepted. In the latter case, either the accepted epic(s) will stay in the backlog till the priority is raised or the accepted candidate epic(s) will be moved into Stage 4 for formal signoff from the Governance forum, either as a discovery epic or implementation epic.

> *We come up with priority score of ideas in the refined and ready state. […] As I mentioned we look for NPV, strategic fit to rank ideas. [AKL_P8]*

> *And part of that [workitems]could just be discovery. Alright, so you could just have any Epic to do discovery because you just want to work. Is there any value in actually doing this piece so even there is from the criteria we workout? Yes is value to this and it's high priority and it prioritized. Or it could be that we need to piece of discovery to see actually what value we're going to get from us, 'cause we're not sure on the value on this yet. if after we've done a piece of discovery work, we find the piece of work is no value being, there's no point carrying on any further work on that piece of work. [AKL_P7]*

Discovery phase (Discovery epic): Discovery phase is a recently adopted approach to get an in-depth understanding of a candidate epic(s) and involves evaluating all the possible risks that might be associated with a potential epic(s).

The participants asserted that the discovery phase is an optional phase, and not all the candidate epics require a discovery phase. The discovery phase is required for the epics labelled as 'discovery' (e.g., test the pricing plan with real customers) when there are a lot of uncertainties that require the delivery teams' capacity for evaluation.

As an outcome of discovery phase, either the candidate epic will be discarded or accepted. In the latter case, either candidate feature will stay in the backlog till the time priority is raised or, if accepted, move into Stage 4 for formal signoff from the Governance forum.

> *If it's a brand new idea, you need to prove this out before we go through the formal process of giving it some money and that would be how do you plan to do that generally the discovery phase, so that DOR would be new idea, go away, test it with customers or test it with some product development teams or make some prototypes or whatever it is depending on which team it is land with and then tell us whether or not this is a good idea. [AKL_P4]*

> *And part of that [potential requirements]could just be discovery. Alright, so you could just have any Epic to do discovery because you just want to work. Is there any value in actually doing this piece so even there is from the criteria we workout? Yes is value to this and it's high priority and it prioritized. Or it could be that we need to piece of discovery to see actually what value we're going to get from us, 'cause we're not sure on the value on this yet. if after we've done a piece of discovery work, we find the piece of work is no value being, there's no point carrying on any further work on that piece of work. [AKL_P7]*

> *if its an idea that currently fits in with some of the development model we have got, say it's a change to plans or change to offer etc. that would probably skip a discovery phase because we know that its got a business legs. [AKL_P4]*

Stage 4: Scored and ranked stage*:** this is the stage where an epic is handed over to the ***governance forum for decision making. Epics that are coming from all the tribes (e.g., digital tribe), are formally ranked and scored at the portfolio level by the *Governance forum*. The governance forum may adjust the score of an epic wherever it is required and check the exceptions. But the recommendation for priority on an epic ultimately comes from the delivery teams.

Bigroom planning and ***portfolio sync call*** (as discussed in Section 6.3) are the decision-making events where the governance forum formally decides the priority on requirements, across the function of business.

> *so, we have the we have a common backlog [portfolio backlog] across the whole retail and those epics are all ranked. [AKL_P1]*

> *So if it makes sense through number one and three together, which is ahead of #2, they [squad] can propose that and we would agree first and then we feed that back to the Governance forum. [AKL_P3*

> *it's not just, you know, sort of dictatorial. And it will be . all the GM from the various functions that work within this bit of […] and the CCO again to arbitrate if any arbitration is needed. [AKL_P4]*

6.4.3.1 Decision-making factors

The following are the factors that are considered for evaluating the priority score of requirements that are posed on product(s):

Strategic fit: strategic fit defines the alignment of potential idea/requirements with business strategy or strategic themes. Each of the potential idea/requirements is evaluated against strategic themes that yields a strategic score of the potential idea. A numerical scale one to five is employed by the studied case organization to evaluate the strategic fit of the candidate ideas/requirements, where strategic score one indicates minimal impact on the strategic theme and the strategic score five indicates high impact.

> *if there's an idea [..] customers might love it. It's not our strategy. We don't do it 'cause you know, so it doesn't matter. Just because a customer likes, it doesn't mean it's the right thing to do, so we've obviously got strategic filters up here when it gets down to concept, which is kind of like an idea. […] we have basically this we call an epic. [AKL_P3]*

> *We score at one to five against the extent to which a support strategy. [AKL_P3]*

we also look at strategic fit. So whether or not it how closely it aligns to our strategic goals. [AKL_P5]

so within each of those ones [themes] will be kind of weighting associated with each of those ones, So what happen is, you end up getting a strategic score by adding that. [AKL_P4]

***Net present value (NPV)*:** NPV defines the commercial value of the potential idea/requirements. Cost like number of sprints, delivery teams required, any third party, hardware and benefits such as incremental adds up to five years, reduced cost to serve, reduced churn, are evaluated in order to allocate the NPV score of the potential idea/requirements.

They have a ranking system and so we work on a net return essentially or NPV over five years, so it's number is created around in NPV and the particularly NPV we're using at this point at the moment is time, so it's whether or not the return will be paid off within under a year or up to five years. [AKL_P5]

NPV, five years divided by the number of sprints. So that's the commercial. [AKL_P3]

6.4.3.2 Decision-making team

The decision-making team, which is a cross-functional team, makes the initial decision on the potential epics. Decision-making team typically consists of senior PO (who brings business perspectives), senior UX/UI (who brings the customers voice) wherever needed, solution architect (who brings solution perspectives), and tribe lead (who brings resource perspectives) works collaboratively to make initial decisions on the potential epics.

At the tribe level, senior product owners, tribe lead, solution architect and senior UX/UI member evaluate the ideas. [AKL_P2]

It's a collaborative process. Senior product owner who presents business stuffs, solution architect represent architectural stuffs, squads requirement represented by tribe lead, and there is senior member of UX/UI who also works with them. UX/UI represent customers' need. [AKL_P6]

6.4.3.3 Decision-making events

The decision-making team generally meet fortnightly to plan current epics, future items, and take ideas through discovery. A weekly sync up call is also set up for check-ins such as epic progress, any blockers, change in decided priority, dependencies. However, ad-hoc discussion can also happen wherever needed. These decision-making events are attended by the GMs as well wherever needed, to communicate any change in the decided priority on the behalf of Governance forum, discussion on potential epics with GMs to get their feedback.

At the tribe level, they meet fortnightly. But weekly for check-ins. Sometime its ad-hoc but that happens rarely.

the business case or the benefits case is again a collaborative work. They senior POs work with their general manager. [AKL_P3]

6.4.3.4 Decision-making techniques

Determine relative priority: The priority score of epics can be adjusted, especially when more than one epic holds the same priority score and/or has the same importance. In such-cases, often the benefits of one over another are compared. The epic that returns best value in the shortest time frame, gets a higher weighting. The customer impact is looked at, the category of epic also (e.g., compliance/regulatory, essential maintenance always gets higher priority than other categories).

In such cases [competing priority], the key is deciding which one of those is going to return the best value in the shortest time. [AKL_P1]

We also tend to use operate under a WSJF, so if you have a multiple things that have the same priority order, we then look from T-shirt sizing point of view, which we have hopefully done at the epic level or a feature that runs up to epic of finding things are of equal value, but which one gets delivered first/ well if one takes 3 months, one takes 9 months then we go for the shortest first to see value being driven more quickly. [AKL_P4]

we look at whether or not the work is required from a regulatory or government enforced. and of course those needs are prioritized higher than something that is most strategic fit and NPV. [AKL_P5]

Perform additional research if needed: Participants indicated that sometimes reaching an agreement on the priority of requirements is difficult. In such cases, often additional research is performed to collect all the required information to convince decision makers. Further the participants asserted that the CCO and CDO take an ultimate decision if any arbitration is required.

So we advised to do further research if the decision makers are not satisfied. [AKL_P10]

If there is conflict then CCO and CDO actually makes final decision. [AKL_P9]

6.4.3.5 Collaborative technologies

At the first part of tribe level, CTs ADO (Daniels et al., 2015) is employed that provide visibility across entire portfolio and enable tracking the progress of which stage a specific requirement is at (i.e., discovery/ready to develop/in development). The integration feature of ADO with other CTs such as Confluence, enable traceability and shared understanding of requirements across all the scaling agile levels.

ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]

Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]

so the teams use Azure DevOps (ADO). Just say it to set up the sprints and to set up the I guess asks everyone just works off that. they also use confluence to documents information. so, a combination of those then I guess yeah, you individually the different people have their own tools like the developers and the designers have their own tools, but those are I guess the workflow ones. AKL_[P10]

6.4.4 Requirements artefacts

Following requirements, artefacts are built in the first part of tribe level which are specified via CTs- ADO, Confluence (Daniels et al., 2015).

Epic(s) (an electronic template configured via CTs- ADO): An epic(s) is a requirement artefact which is defined to realize a theme and/or contribute to the formation of theme. An epic is a large development initiative that can take longer than one release IR (i.e., more than three months).

we have a template that we follow. And we have ADO to do that. [AKL_P10]

Epic usually take multiple releases to implement. [AKL_P7]

Business-case (an electronic template configured via CTs- ADO): A high-level rough business-case is built to justify the potential epic(s), which is typically employed by the governance forum during decision making.

A high-level rough business-case also called *benefit framework* consists of the following information: business outcome hypothesis, in scope, out of scope, time to deliver, stakeholders, teams and skills required, effort, benefits, costs, including all dependent squad's costs and hardware. Also net profit value and payback, quality requirements performance, security, usability are considered.

business case, which is as I was saying this is more to do with the financial benefits, so the benefits, NPV as well as the costs. [AKL_P6]

so when you write a business case or benefit case, you write the epic it at the same time and the two of them joined together and goes into the Governance forum. [AKL_P2]

So at that point, connecting at high level business case like what's the strategic value, what's the benefits, what's the cost? What's the NPV score. [AKL_P8]

> *Alongside that there would be light financials, so that would be a idea of you know if we're to scale this up roughly what would be the cost for teams perspectives and what would we except aa an ROI back from this. but that like touch points,, its not a full business case, its what we refer to as a benefit framework. [AKL_P4]*

Confluence: CTs- *confluence* is employed to maintain documentation of requirements, for detailed discussions and/or analysis on requirements such as recordings of discussion with customers, that help in building the shared understanding of requirements. The integration feature of *confluence* with *ADO* provides traceability of requirements.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

6.5 Second part of tribe level

Introduction: In the second part of tribe level, a priority list of epics that reside in the portfolio backlog serves as an input for each of the tribes. Epics that are formally signed off by the *governance forum* for implementation are allocated to each of the tribes, based on their relevance to the tribe during *bigroom planning event*. The number of epics that are allocated to each of the tribes may vary.

Each epic generally requires more than one engineering teams for implementation. For each epic there is a dedicated sponsoring PO, generally played by a senior PO, who generally acts as a ***boundary spanner*** responsible for providing all the relevant information. This includes requirements clarity, defined scope, for the delivery teams to implement. A sponsoring PO often occupies the role of a PO in one or more delivery teams who are involved in a particular epic implementation, For example, Epic A requires three teams for implementation, and in such a case, a sponsoring PO can be a product owner for any of the teams. Generally, a one-to-one relationship between the PO and delivery team is maintained. However, one PO can manage more than one delivery team, if required.

> *what actually happens is the senior product owners will sponsor an epic to that group [Governance forum]. [AKL_P1]*

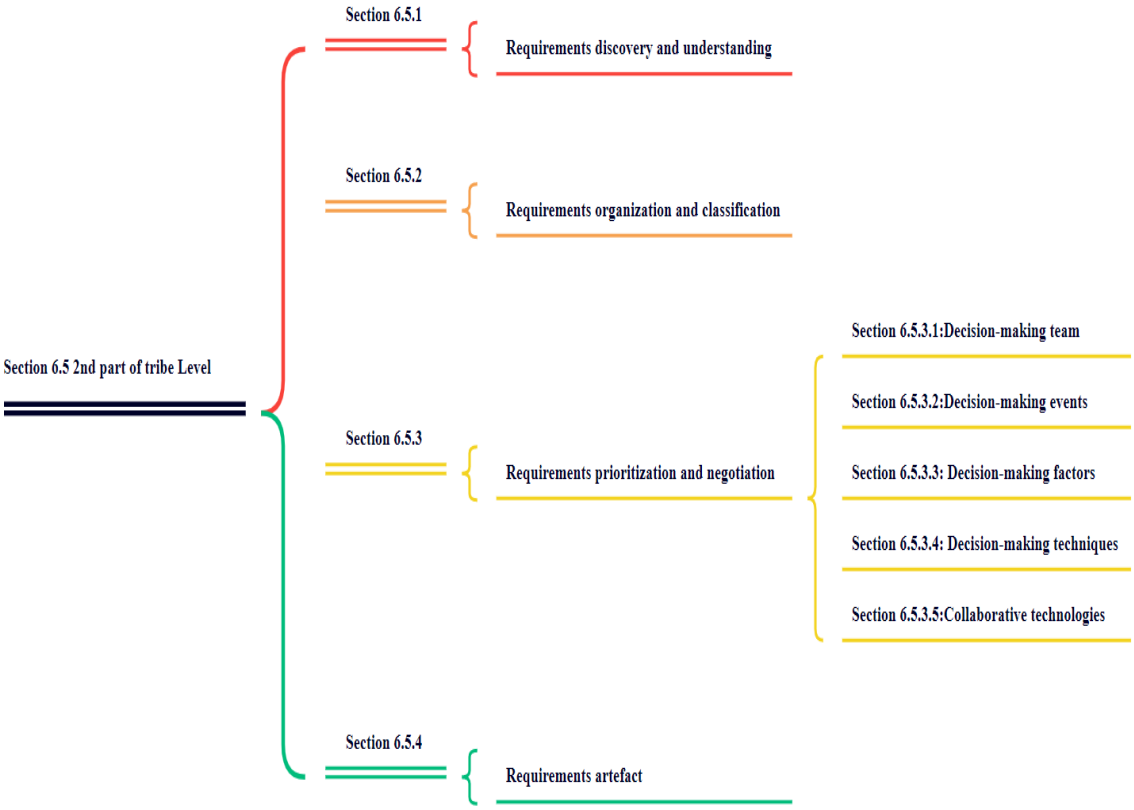
> *I think it [epic] need more than one team. On and average can go upto 5 teams. [AKL_P1]*

> *Each team have one PO but sometime one PO manage 2 teams. [AKL_P1]*

Figure 6.5 shows the mechanisms that are discussed at the second part of tribe level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 6.5

Snapshot of activities discussed at the second part of tribe level



6.5.1 Requirements discovery and understanding

In the second part of tribe level, an epic description that is typically documented in the form of a business-case is generally consumed by the decision-making team to discover and/or understand potential requirements that can be posed, to implement a particular epic.

Business-case is the artefact that we use to explore a particular epic. [AKL_P5]

6.5.2 Requirements classification and organization

As discussed in 6.2.2, the five levels of requirements hierarchy model, is adopted by the case organization.

Theme(s) → Epic(s) → ***Feature(s)*** → User story(s) → Task(s)

In the second part of tribe level, an epic is broken down into feature(s). A feature is something that provides value to the customers and business. A feature generally takes one sprint (i.e., two weeks), and typically is implemented by a single delivery team. Participants indicated that at present, the number of features that belong to a particular epic varies from epic to epic. However, Participant AKL_P1 asserted that they are on progress to clarify Epic sizes to make them more similar sizes.

> *Feature is like a one sprint of work. [number of features per epic] This varies a lot – from 5 to 50. We're currently trying to clarify Epic sizes to make them more similar sizes. [AKL_P1]*

> *We try and get a feature to be owned by one team. [AKL_P8]*

6.5.3 Requirements prioritization and negotiation

6.5.3.1 Decision-making team

Teams' triad, a cross-functional team that typically consists of POs who provide business perspectives, SMs providing resource perspectives, technical BAs providing technical perspectives and they work collaboratively in order to breakdown an epic(s) into features and set their priorities.

> *So our BAs, POs, scrum masters they are the one who breakdown them [epic] into features. [AKL_P6]*

> *POs with support from BAs, and scrum masters. [AKL_P4]*

6.5.3.2 Decision-making event

At the beginning of the development of a new epic, a discussion is held with the teams' triad, who are involved in a particular epic implementation for decision making on the requirements. For example, Epic A requires four teams, in that case all four teams' triad will be involved in decision making on the requirements. During decision making, each teams' triad provides their perspectives and collectively make decisions on the requirements or features. As an outcome of this event, a ***product backlog*** that contains a priority list of features (Schwaber, 2004) is created, where a particular section of ***product backlog*** is reserved for each of delivery teams**.** That section will act as a ***team backlog*** for a particular team and is allocated to each of the teams who are involved in a particular epic implementation.

> *So our BAs, POs, scrum masters they are the one who breakdown them [epic] into features. [AKL_P6]*

> *POs with support from BAs, and scrum masters. [AKL_P4]*

> *So all squads' triad meet fortnightly. They have a quick weekly catchup as well. But sometime they have ad-hoc call as well. [AKL_P9]*

> *We have a product backlog that contains a list of features. for each squad, they have their own portion in the product backlog. [AKL_P7]*

> *When a new epic is picked up, we all [teams' triad] comes together and decompose that epic into features. [AKL_P3]*

The Teams' triad meeting (i.e., team of team meeting) is generally held fortnightly where requirements or features- progress, issues, blockers, dependencies with other teams, any change in priority, are discussed. However, they have weekly check-ins as well.

6.5.3.3 Decision-making factors

Business value is the main decision-making factor which is typically evaluated via tacit knowledge on the priority of features.

> *features again are typically prioritized based on their business values. [AKL_P5]*

> *we don't have scoring system for features. [AKL_P1]*

6.5.3.4 Decision-making techniques

- Priority on features is typically driven by epic and strategic priority
- Relative priority is considered
- Team capacity and ability is considered
- Dependencies among features (e.g., functional dependencies) are considered

> *there's no real set criteria. it depends. One of my squads set the priority of features based on customer research. So they, Got a whole bunch of features and they asked customers you know, like what is? What would you put things you prefer to say so they ranked the features based on customer preference. But then the other thing that had to come into it is things like. Uhm, OK. In order to build this thing this. This is the number one thing the customer wants in order to be able to build that we might actually have to do some work on a different feature first. So then there's a bit of give and take. [AKL_P8]*

> *As I said for features, we don't use any scoring system. Its normally driven by epic priority and strategic themes. […] we look for team capacity and ability while deciding features' priority. [AKL_P1]*

6.5.3.5 Collaborative technologies

At the second part of tribe level, CTs ADO (Daniels et al., 2015) are employed that provide visibility across the entire portfolio and enable tracking the progress of which stage a specific requirement is in. This is whether it's in discovery/ready to develop/in

development stage. The integration feature of ADO with other CTs such as Confluence, enable traceability and shared understanding of requirements across all the scaling agile levels.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

> *so the teams use Azure DevOps (ADO). Just say it to set up the sprints and to set up the I guess asks everyone just works off that. they also use confluence to documents information. so, a combination of those then I guess yeah, you individually the different people have their own tools like the developers and the designers have their own tools, but those are I guess the workflow ones. AKL_[P10]*

6.5.4 Requirements artefacts

The following requirements artefacts are built in the second part of tribe level which are specified via CTs- ADO, Confluence (Daniels et al., 2015).

Feature(s) (an electronic template configured via CTs- ADO): A feature generally takes one sprint (i.e., two weeks), and typically is implemented by a single delivery team.

> *Feature is like a one sprint of work. [number of features per epic] This varies a lot – from 5 to 50. We're currently trying to clarify Epic sizes to make them more similar sizes. [AKL_P1]*

> *We try and get a feature to be owned by one team. [AKL_P8]*

Product backlog (electronic artefact configured via CTs- *ADO*): product backlog is a requirements artefact which is built in the second part of tribe level that contains priority list requirements for the teams who will be involved in epic(s) implementation.

> *We have a product backlog that contains a list of features. for each squad, they have their own portion in the product backlog. [AKL_P7]*

Confluence: CTs- *confluence* is employed to maintain documentation of requirements (e.g., for detailed discussions and/or analysis on requirements such as recordings of discussion with customers), that help in building the shared understanding of requirements. The integration feature of *confluence* with *ADO* provides traceability of requirements.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]

6.6 Team level

Introduction: Method adopted: at the team level, the Scrum method is adopted by the delivery teams for implementing the requirements, e.g., the user story (Schwaber, 2004). Delivery teams typically follow two weeks sprint to implement requirements and to get some valuable feedback from the stakeholders at the end of sprint.

we tend to have a three month increments of work but is broken down into two-week sprints. [AKL_P7]

scrum method that we follow for implementation. [AKL_P9]

Team structure: a delivery team is cross-functional that generally has 12-13 members. Each of the delivery team typically has one PO, one BA, one SM, two testers, three UX designers, four to five developers. Teams are self-organized. They are encouraged to adapt practices that work best for them.

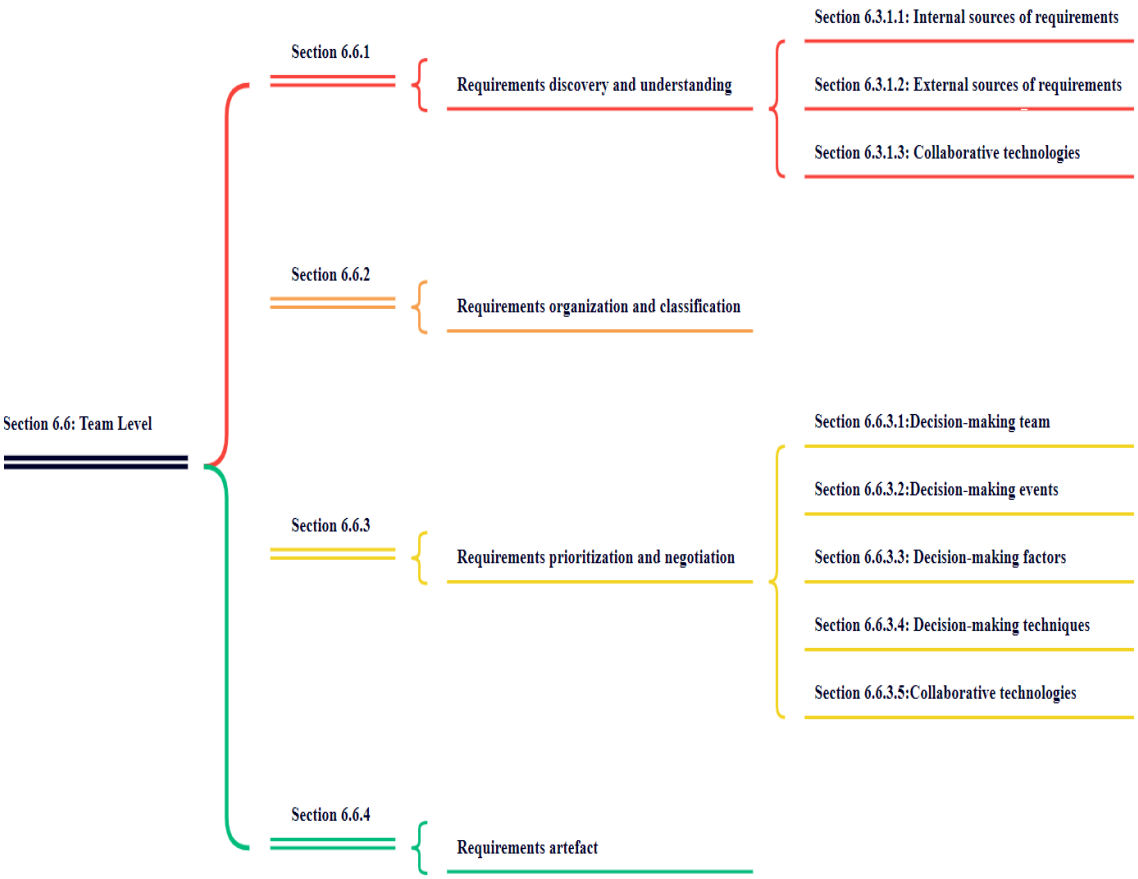
I have two teams including me is 25, so about 12 each. [AKL_P2]

I think one I work with is 10 at the moment otherwise 12 to 13. [AKL_P1]

Figure 6.6 shows the mechanisms that are discussed at the team level in terms of decision-making on the priority of requirements that are structured via employing the initial conceptual framework, as discussed in Chapter 4: Section 4.4.3, in the studied case organization.

Figure 6.6

Snapshot of activities discussed at the team level



6.6.1 Requirements discovery and understanding

The business case and feature description are the two main guiding requirements artefacts acting as boundary spanning artefacts, that are shared with delivery teams in order to explore a particular feature.

> *Business-case that contains epic details and another artefact that they shared with us is feature. These are the two artefacts that we use to understand implementation requirements. [AKL_P8]*

6.6.1.1 Internal sources of requirements

At the team level, engineering team(s) that include PO, BA, SM, developers, and testers, are generally the internal sources of potential requirements.

> *Ideas comes from various places but within the team I would say its PO, BA, scrum master, engineers. [AKL_P2]*

Practices to collate and understand the requirements at the team level (internal sources)

Feature kick-off: At the beginning of the development of a new feature, a *feature kick-off* is typically held in two sessions. First session, which is typically a team triad, that is PO, BA, SM session, generally takes two hours. During this event, PO with support from BA and SM break down a feature(s) into user stories where PO brings business needs, BA acts as a representative of the development team and generally provides technical implementation guidance, SM is the person who facilitates the feature kick-off event.

The input from other members, developers, testers, of the team, is also received in order to breakdown a feature into user stories. The BA often organizes a separate meeting, typically a 30-minute session, with developers and testers to get their input in feature decomposition.

> *feature kick off is basically where we [triad] come together and discuss what we have to do to achieve that feature. We breakdown that feature into user stories. Feature breakdown is actually a collaborative process. team members also give input into that but PO, BA, SM are the primary decision makers. [AKL_P9]*

> *so within that [team] the product owner has the authority to sequence and optimize [user stories]. Yeah, so it's really down to the product owner and support from the Scrum master. BA helps with that, so the scrum master, the product owner look at the velocity and go. What's the best way to break this down, refine it and deliver the piece of work. [AKL_P3]*

> *. So then the team gets to ask questions and get it, you know a shared understanding of the what that is. What that, that whole feature is. so in that session will also get them to post a note like what they think. You know some of the things that they need to do, in order to achieve that feature. […] then they'll get the Devs and the testers in a room which is another 30mins session and they'll sit down, and they'll say. This is what I think should be a story. what you guys think like that type of conversation [AKL_P8]*

6.6.1.2 External sources of requirements

The case study data revealed that senior POs, UX/UI team, support team, sales and marketing team generally collate potential ideas/requirements from the external sources; especially customers. However, for the engineering team typically the PO can also collate potential ideas/requirements from those same sources.

> *They [senior POs] go and test the waters a little bit so they come up with a bunch of ideas and those things could be Electric vehicles, it can be solar charging and they sort of go out and talk to customers and they figure out some idea about what customers actually want. [AKL_P2]*

> *so we have For example, the call centre people are called up- they Record that information. [AKL_P1]*

So if there is something that a customer sees and quite often that will come in throughout our contact centre, then those ideas will bubble their way up as well. [AKL_P5]

We have sales team, we have marketing team, support team who can also communicate with customers to collect their ideas. [AKL_P7]

If needed, POs can also collate ideas from customers. [AKL_P6]

6.6.1.3 Collaborative technologies

Collaborative technologies (CTs)- ADO (Daniels et al., 2015) is a formal place to capture potential requirements that emerge at the portfolio level.

ADO is the tool that we use where all the potential ideas are formally placed. [AKL_P4]

6.6.2 Requirements classification and organization

As discussed in Section 6.2.2, a five level of requirements hierarchy model is used by the case organization.

Theme(s) → Epic(s) → Feature(s) → ***User story(s)*** → ***Task(s)***

At the team level, a feature(s) is broken down into user stories, which is further breakdown into task(s), wherever needed.

Smaller impact requirements, such as requirements that require one or two weeks of work, are usually resolved at the team level. Typically, the requirements such as user stories, defect correction work, technical debt work do not require a lot of ceremony, or portfolio level decision-making. These requirements types, typically demand small amounts of effort, one or two days of work, and are generally sorted at the team level.

So anything, that looks like one or two plus months of work we have to put in a mini business case. And then it will go through the formal process. […] something as small as little, or basically I'll just fix that and no one will see it. […] something small enough that like I said [tech-debt work, enhancement work], it was one week or less, work will just do it and get that prioritized the next Sprint also. [AKL_P2]

6.6.3 Requirements prioritization and negotiation

6.6.3.1 Decision-making team

Team triad (PO, BA, SM), UX/UI wherever needed, works collaboratively to make decisions on the priority of requirements (e.g., user story, defect correction work, and technical debt work).

> *the product owner has the authority to sequence and optimize [user stories]. Yeah, so it's really down to the product owner and support from the Scrum master. BA helps with that, so the scrum master, the product owner look at the velocity and go. What's the best way to break this down, refine it and deliver the piece of work. [AKL_P3]*

> *its PO responsibility but he works with their counterparts BA, scrum master. So they work together and sometime UX/UI input is also required. [AKL_P7]*

6.6.3.2 Decision-making events:

Feature kick-off (discussed in Section 6.6.1), backlog refinement meeting, sprint planning meeting are the key decision-making events that are employed at the team level to make decision on the priority of requirements.

Backlog refinement meeting: Each team has their own backlog that contains a priority list of requirements. Typical requirements include strategic work, typically in the form of user story, technical debt work, defect correction work, spikes, and slipping stories; which is left over work from the previous sprint.

During the development of a feature, each team has an on-going backlog refinement owned by the PO of the team with the support from SM and BA, where requirements are refined. The goal is to set at least two sprints of work which should always be ready in their team backlog. Backlog refinement is typically held on a weekly basis and takes approximate one hour.

> *We have team backlog that contains work items like user stories, tech-debt work, left over work from the previous sprint etc. etc. we have refinement meeting. Actually, we have various refinement meeting that happens at the different-different level. Like refinement at the tribe level, refinement at the team level. [..] at the team level we do this on weekly basis. [AKL_P1]*

> *At the team level, we have weekly refinement meeting to set priorities for next sprint. […] At least 2 sprints of work always ready in our backlog. [AKL_P10]*

Sprint planning: Sprint planning is typically held on a fortnightly basis where capacity planning is done, and engineers split their user stories into concrete development tasks wherever needed, that have to be implemented in the next sprint. The whole team is typically involved in sprint planning.

user stories again prioritized by the product owner and they brought into the sprint by Sprint Planning. […] [sprint planning] Fortnightly or whatever the duration of the sprint is. [AKL_P5]

there [sprint planning] we break user story into tasks. We do capacity planning as well there for the upcoming sprint [AKL_P9]

6.6.3.3 Decision-making factors

Participant AKL_P1 and AKL_P5 asserted that BV is not typically evaluated in order to decide the priority of requirements or user stories, at the team level.

we don't breakdown value into a user story level. [….] we don't typically understand that the value it really comes down to at a user story level. It's about how the technology is best built. So we might actually look at what's feasible to build 1st and then go down from there and workout. It comes down to efficiency at that point and what is the most efficient way to build that the application with the technology to support that feature. [AKL_P5]

we don't have a scoring system at the user story level. [AKL_P1]

6.6.3.4 Decision-making techniques

The following are the techniques employed to make decisions on the priority of requirements at the team level

- relative priority is considered
- team capacity and ability are considered
- dependencies among features (specifically technical dependencies) are considered

how we do this priority- where it doesn't just reflect the kind of the need of the business but reflects the ability of the team to deliver that as well. That should have 50-50 command and control versus ability to do that. [AKL_P4]

BA helps with that, so the scrum master, the product owner look at the velocity. [AKL_P3]

we don't breakdown value into a user story level. [….] we don't typically understand that the value it really comes down to at a user story level. It's about how the technology is best built. So we might actually look at what's feasible to build 1st and then go down from there and workout. It comes down to efficiency at that point and what is the most efficient way to build that the application with the technology to support that feature. [AKL_P5]

We use relative story point technique. [AKL_P9]

6.6.3.5 Collaborative technologies

At the team level, CTs ADO (Daniels et al., 2015) is employed that provide visibility across the entire portfolio and enable tracking the progress of which stage a specific requirement is at. (i.e., It may be discovery/ready to develop/ or in development). The

integration feature of ADO with other CTs such as Confluence, enable traceability and shared understanding of requirements across all the scaling agile levels.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

> *so the teams use Azure DevOps (ADO). Just say it to set up the sprints and to set up the I guess asks everyone just works off that. they also use confluence to documents information. so, a combination of those then I guess yeah, you individually the different people have their own tools like the developers and the designers have their own tools, but those are I guess the workflow ones. AKL_[P10]*

6.6.4 Requirements artefacts

The following requirement artefacts are built in at the team level that are specified via collaborative technologies- Jira, Confluence (Daniels et al., 2015).

User story(s), task(s), spike(s) (electronic template configured via Jira):

User story(s) is the quickest sort of element deliverable that can be implemented within a two-week sprint and by an individual engineer.

> *user stories is the result of feature decomposition. User story is like getting some valuable feedback. It represents small amount of effort like can be implemented in by an individual dev person. [AKL_P8]*

Task(s) that describe in technical terms what are required to be done in order to claim a user story at the end of a two-week sprint.

> *Task represents what needs to be done to complete a user story. [AKL_P8]*

Another requirement is to deal with a spike(s), which is typically a technical investigation that needs to happen in the current sprint to be successful in a future sprint.

> *We have spike as well in our backlog. Spike is like technical investigation. [AKL_P8]*

Team backlog (configured via CTs- Jira): team backlog that contains a priority list of requirements, is built at the team level.

> *We have team backlog that contains workitems like user stories, tech-debt work, left over work from the previous sprint etc. etc. [AKL_P1]*

Confluence: CTs- *confluence* is employed to maintain documentation of requirements (e.g., for detailed discussions and/or analysis on requirements such as recordings of discussion with customers), that helps in building the shared understanding of requirements. The integration feature of *confluence* with *ADO* provides traceability of requirements.

> *ADO is our main tool through which we can track all epics status, feature status, user stories status. Who is working on what […] we have one more tool confluence. All the analysis work like evidences we maintain through our confluence tool. [AKL_P6]*

> *Azure DevOps ADO to visualize work. Confluence for documentation and ADO is linked with Confluence [AKL_P3]*

6.7 Incorporate distributed members of decision-making team

In the studied organization, members of a decision-making team make decisions on the priority of requirements across scaling agile levels and are nationally distributed. This Section describes the mode of communication at the scaling agile decision-making levels and the CTs, that are employed to incorporate with the distributed members of decision-making team.

6.7.1 Mode of communication at the portfolio level

At the portfolio level, ***bigroom planning*** is a formal decision-making event that is organized on a Quarterly basis (i.e., Q1, Q2, Q3, Q4), where priorities on requirements are formally decided and communicated to the delivery teams for the implementation. It has emerged from the data that the whole decision-making team, the Governance forum, is locally distributed. Decision makers are generally located at the head office (Auckland) and Hamilton sites, although the majority of them are located at head office. The case study data revealed that generally the company arranges travel via a shuttle service, for the decision makers to attend the decision-making events. Quarterly bigroom planning is done in-person, which has yielded outcomes that make is easier to reach consensus, better understanding of requirements, and easier to collaborate during decision making. Participants reported that the ***Governance forum*** generally meets ***four times a year,*** where they generally meet in-person in an auditorium, either at the head office or Hamilton site.

> *we have two office- one in Auckland and another one is in Hamilton. At least 2 of GMs would be from Hamilton. [AKL_P4]*

> *some of them [Governance forum] would be distributed because there's some people in Hamilton that would be involved in that [bigroom planning] meeting. [AKL_P8]*

Face to Face- so in most cases the forums that happen where they do the prioritization is done. So, we had ours last week I was there, and it was all of the different GMs. So, they bring their new, you know, new epics, business-cases to get prioritized. So, it's face to face. [AKL_P6]

So, they'll either attend the meeting in person, or they will attend using Teams. We are pretty active using technology to collaborate. But we try to bring all of them at the same location specifically during refinement meeting [bigroom planning]. You know in-person meeting have its own benefits. Like collaboration is easy, better understanding etc etc. [AKL_P5]

Some are actually from Hamilton and we have a shuttle every Tuesday, Wednesday, Thursday, so they do not have to drive up. they jump onto the shuttle, comes over here. [AKL_P2]

6.7.2 Mode of communication at the tribe level

The case study data showed that delivery teams who are the part of a particular tribe either at the same location or are distributed. However, in the majority of the cases, delivery teams are at the same location. Decision making events, like the team of team meeting, are generally organized via an in-person mode event.

Our majority of teams are at Auckland office. Like I am from digital new services tribe, all our teams at Auckland office. [P1]

6.7.3 Mode of communication at the team level

It emerged from the data that delivery team members such as developers and testers, are generally at the same location at the team level. Decision making events, are typically held via face-to-face meetings.

Our all team members are at the same location. [AKL_P8]

We generally meet in-person. Although we have flexibility to work remotely, but when we have refinement meeting. [AKL_P2]

6.7.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection)

During the time of data collection for this study - March 2020, due to the COVID situation, team members were working remotely. Therefore, decision makers typically meet virtually via CTs, which are web conferencing tools discussed in Section 6.7.5. For example, the decision-making event at the portfolio level during the time of data collection, the bigroom planning event was completely virtual.

We prefer to meet in-person specially when we have bigroom planning meeting but now because of COVID we all are working remotely. Technology helping us to work remotely. [AKL_P7]

Now during COVID teams are working remotely. [AKL_P5]

6.7.5 Collaborative technologies

Web conferencing tool: the studied case organization employed a web conferencing tool such as MSTeams as their primary synchronous communication tool, to communicate with cross-site members of decision-making team. The application sharing feature of MSTeams as CTs, enable cross-site members of decision-making teams to allow synchronous editing of requirements artefacts if needed, during decision-making event(s).

In addition to the formal synchronous communication, informal synchronous communication included instant messaging, instant calling, team(s) specific channel to communicate internally within the team, as well as asynchronous communication. These encompass offline chat, tag to notify people, recordings of decision-making events that are also organized via web conferencing tool, to communicate with distributed members wherever needed.

If there were some people who dial-in and we use teams to do that but in most cases the expectation is that people will do it face to face. [AKL_P6]

We have teams that we use for meeting. Sometime we send messages as well via teams. [AKL_P10]

We have our internal channels as well. [AKL_P3]

but it's [decision-making events] also recorded so you can if you'd miss that timeslot, you can go and watch it afterwards [AKL_P10]

We've got lots of lots of digital whiteboards which actually helps us with the meetings, and we can share a white board and work together on that. Yep, sharing screens Wi-Fi is good. Um so I think the technology enabled us to do that. [AKL_P2]

Email: the studied case organization employed 'email' as CTs to communicate with decision-making teams. According to the participants, email is a less frequently used asynchronous communication mechanisms in the studied case organization.

We do communicate via email as well but you know instant messaging, tagging people are the fastest way of getting response. [AKL_P3]

6.8 Challenges and potential strategies

This section describes the challenges that are faced by the studied case organization while making decision on the priority of requirements.

6.8.1 Tension between product and engineering

It emerged from the data, that there is always a tension, which is generally a healthy tension, between the product and engineering in terms of decision making, specifically on the requirements that have smaller impact. These include tech-debt work and/or internal improvement work. In the experience of participants, generally the decision making on smaller impact requirements, requires a lot of negotiation, as value is not clearly articulated in smaller impact requirements. On the other hand, bigger impact requirements like discretionary work, transform work that provide direct value to the customers and business, requires less negotiation due to their direct alignment with the business strategy. One of the steps that is employed to ease this tension is classification of requirements. There are five categories of requirements which are allocated, based on their relevance to produce an efficient priority list of requirements for the delivery teams for implementation.

> *There is also I think, a tension which is generally healthy tension, between what our exec thinks we do in terms of building new products vs the stuffs that's the keeps lights on. [AKL_P4]*

> *So, the problem is often they [internal improvement work, tech-debt work] have -NPV. That's why its bit hard to prioritize them. but we have that categorization [5 categories of requirements]- that I talked you before that at some point helps in managing them. [AKL_P6]*

> *We always wanting to transform and we always wanting to be progressive, but we have to be mindful that if we don't keep on top of the basic foundations that will be hammered anyway. [AKL_P5]*

> *this is where we talked about regulatory and compliance before. We also have another category that is just for keeping the lights on. So we try and prioritize particular things that we know will cause issues if we don't get to them. [AKL_P5]*

6.8.2 Accumulating technical debt

It emerged from the data, that tech-debt is accumulating. Participant AKL_P5 asserted that tech-debt work generally resides in the backlog until the time it hinders the delivery team in their progressive work. The main reason to accumulate tech-debt work, is lack of clearly articulated value associated with tech-debt work.

> *you know technical debt. We know that that if we don't do the technical debt while it won't create value immediately, if will not doing it, it will hinder us later and […] And*

> *quite often it's challenging to get the priority because it won't have a benefit equation that stacks up, and it won't necessarily be for regulation, and so quite often it will just sit there until it stops us from doing something that's progressive. [AKL_P5]*

Some of the strategies that are employed to manage this challenge, include categorization of requirements, and allocating dedicated teams' capacity (20% of teams' capacity) to handle tech-debt work.

we try to set 20% of time to be able to maintain. That's not necessarily feasible, it depends very much on what's happening and what you know how much technical-debt we've acquired. [AKL_P5]

but we have that categorization [5 categories of workitems]- that I talked you before that at some point helps in managing them. [AKL_P6]

6.8.3 Quality of requirements

Participant AKL_P1 asserted that it is challenging to provide the right amount of granularity for the team to work on. They often face a challenge in getting the quality of the work and detail right for the teams. One of the steps that has started in this direction is dedicated capacity (i.e., 10% to 20%) which is allocated for refinement to ensure the right amount of detail is there for the teams to work on.

> *Trying to create a backlog that's we've got the right amount of detail on the right point, yeah. Yeah our biggest challenges I think always been how do we get worked for teams in the right state? a state is sufficiently good quality for the team to work on. […] We probably spend somewhere between 10 and 20% of our capacity as a whole and looking back on just Refining and making sure the quality is there before we actually start working on stuff. [AKL_P1]*

6.8.4 Bias during decision-making

It emerged from the data that a matrix/scoring system is employed in order to make decisions on the priorities of requirements. However, decision making on the requirements is not entirely based on objective data. Participants reported that an organizational 'command and control' culture still exists in terms of decision making on the priority of requirements. Priority decisions are generally biased by the perspectives of senior people, the Governance forum. Participants asserted that potential requirements that are proposed via the Governance forum, generally get more importance as compared to the potential requirements that are proposed by lower-level people like the support team. The following are some of snippets from the participants' interviews, to support the above findings.

> *There's always ideas that an exec team think are great ideas, but that we know are difficult to implement and we know won't get the results that they think they do, but they want it, and so therefore we will deliver it.And there's always ideas that we know*

in our hearts are the right thing to do and we know will have benefit- Will be not financial benefit and that we just can't get prioritized. [AKL_P5]

The other one I mentioned was is still a lot of command and control. We are slowly teaching leadership about how to be more agile. [AKL_P7]

or you would get GMs basically tapping a team on the shoulder and saying, hey you know just to do this bit of work. That's one of the first challenges. [AKL_P4]

it's a long list [work items-epics] and there is, you know, even though we do, you know, sort of a scoring there. It still subjective in terms of actually getting yours prioritized. So for some areas, particularly the epics may be much lower down in terms of the priority, so they might have to wait three to six months or more, you know, because there are other Epic that are deemed to be more important. there are a number of business cases that have come from our service area that are related to you know customer support and they often really don't get prioritized very high so they don't get funding. They don't get the support that potentially You know would be of benefit to [..] in the customer. OK, so there's a lot of people I think who miss out. [AKL_P6]

depending where the idea came from, potentially like if it comes from some of our executives, then it's a bit harder to just drop it. [AKL_P8]

6.8.5 Difficult to measure business value

It emerged from the data, that value, i.e., NPV, plays a prominent role while making decisions on the priority of requirements. However, Participant AKL_P1 asserted it is difficult to assess the concrete value of potential requirements.

how much money it's [idea/epic] gonna make versus a theoretical idea that in five years' time like makes you money. Yeah, it's really hard to compare those two. Yeah, absolutely. And our prioritization system doesn't deal with that technical at the moment. [AKL_P1]

6.8.6 Dependencies across product

Participants asserted that at present (during data collection), the way the tribes are structured, that they have a lot of dependencies across the functions of the business. A team works on between two to five epics at a single time. This dependence often yields conflicting priorities, tension between decision makers, such as tension between GMs in order to get their work priority done, reprioritization, and delays in delivery. Therefore, tribes need to be restructured and mechanisms through which all possible unknowns and dependencies are identified, before starting the implementation of requirements.

because of our dependencies have very high dependencies on things. Each team might be working on anywhere between two and five epics. […] So sometimes the dependencies means that team stops on a particular epic and waits for another team.[….] So the structure of our tribes is a challenge for us. Because the way we're structured residential tribe or Business tribe, Digital and new services. […] most products have to go through more than one of those tribes. So that's challenging. What we would love to do is to restructure our tribe. AKL_[P1]

> *at the moment we work in, I guess fairly artificial constraint. So I work for resi, there is business squad. There is a digital new services squad and the level of dependencies between those; not to mention the kind of the fighting between GMs to get what they want over another GM doesn't make a great deal of sense in terms of that set-up, so we're looking to flex that set up to better reflect the value streams that we delivered to. [AKL_P4]*

> *so there's some challenges, We Got quite far into one feature. We are doing 2 features in parallel. We got quite far into the first one when we realize actually those two needs to be merged. You couldn't actually do one feature without the other, so it didn't make sense to keep them apart. So I suppose the challenges you know for that it's like, well, we will have to rethink how we were doing that feature and slightly change the work and then some information we didn't think we needed tool down here. We need it up front, you know. So changes like that. [AKL_P8]*

The following are some of the mechanisms that are adopted by the case organization to manage dependencies:

Identify dependencies upfront: the case study data showed that practices that yield an early identification of dependencies are employed. However, Participant AKL_P8 asserted that sometimes it is difficult to identify dependencies upfront. The ***Discovery phase*** is a recently adopted practice, that helps in identifying the dependencies upfront before commencing implementation.

> *We Got quite far into one feature. We are doing 2 features in parallel. We got quite far into the first one when we realize actually those two needs to be merged. You couldn't actually do one feature without the other. [AKL_P8]*

> *We have something called increment planning. increment planning as more of a chance to get together and talk about dependencies and share the plan we've already created. [AKL_P3]*

> *We recently started doing discovery activity. We identify all possible information there like resources, dependencies, solution alternatives etc. [AKL_P9]*

Collaborative technologies: ADO is adopted to visualize the dependencies across the functions of the business.

> *we use Microsoft ADO to visualize our work. so we created as an epic in there. [AKL_P4]*

> *We've got the ado boards, which means that all your work is visible and transparent, 'cause it's all on the board. [AKL_P8]*

Continuous collaboration and communication: It emerged from the data, that decision makers, such as the governance forum, have continuous collaboration that helps in identifying and managing the dependencies. Some of the events include ***bigroom planning, sprint planning, and backlog refinement***.

We have backlog refinement. We have sprint planning. These are some of events that helps us to identify them [dependencies]. [AKL_P9]

We have something called bigroom planning. bigroom planning as more of a chance to get together and talk about dependencies and share the plan we've already created. [AKL_P3]

6.8.7 Dealing with quality requirements (QRs)

Participant AKL_P2 asserted that the case organization is facing challenges with QRs in particular, with performance. Mechanisms need to be investigated to handle QRs efficiently. The following are some of the current strategies that are employed to handle QRs:

- in-progress to automate QRs: tools adopted to automate QRs- CI/CD
- discovery phase to identify QRs
- capacity allotment to work on QRs

That is a good that is a good and embarrassing one, and if you know a company that does it well, please please connect me to one our non-functional here is horribly. [AKL_P2]

it's just embedded into the platform of the C/ CD stuff that we're working on at the moment. We don't have it yet. Will start looking at those nonfunctional as well as though we don't need humans to do it. [AKL_P2]

so how we manage them, we reserve some of our engineers' capacity to implement non-functional requirements. [AKL_P5]

6.8.8 Difficulty in estimating effort

Participant AKL_P2 asserted that it is difficult to estimate the effort upfront. Sometimes effort that is needed to implement a requirement(s), is underestimated. The estimation process that is currently followed (during data collection) is totally based on the experiences of decision makers, that is guess estimates. Therefore, a mechanism that provides more objective estimates needs to be investigated.

there are a lot of challenges in their because obviously cost and benefit estimating is just estimating, right? There's probably not enough science behind that. And sometimes things take longer. And actually, a lot of times in Agile, actually a lot of times in any IT project, things take longer. [AKL_P2]

6.8.9 Inadequate requirements analysis

It emerged from the data, that insufficient understanding of the requirements often yields inadequate prioritization. As a consequence of this, often a feature is built that wasn't the appropriate priority.

Therefore, the right amount of effort and collaboration is required in order to make an efficient decision on the priority of requirements. Participant AKL_P4 and AKL_P9 asserted that one of the recently adopted approaches in this direction is the ***discovery phase,*** where potential requirements are evaluated and/or understanding of the potential requirements are developed in depth, before they are committed for delivery.

> *We recently started doing discovery activity. We identify all possible information there like resources, dependencies, solution alternatives etc. [AKL_P9]*

> *We have just started moving towards more of a discovery phase because lots of the work that we would do, we would kick off because it had been decided, probably an exec level that it was a good idea and kind of circumvented some of the process. […] so you can prove that there is an evidence to suggest that if we do this, it has an impact to our customer that should be positive. […] So rather than having to go straight for a full business case then spent months delivering something that, oh, we just tested it customers in groups, so it wasn't what they wanted. We go out and do that discovery phase and hopefully that means that we deliver faster because a lot of the ideas that potentially not so good; disappear from the backlog. [AKL_P4]*

6.9 Summary

This chapter described the prioritization process and the process specific practices that are employed by AKL organization to make decisions on the priority of requirements across scaling agile levels. In addition to this, challenges that are faced by the studied case organization, while making decisions on the priority of requirements and the potential strategies that can be employed to surmount those challenges, are discussed.

The cross-case analysis of studied case organizations is presented in Chapter 7.

7 Cross-case analysis

Introduction: two case organizations are being studied to achieve the objectives of this research study. The previous two chapters (Chapter 5, and Chapter 6) provide in-depth descriptions of the requirements prioritization process of the studied case organizations.

In this Chapter, cross-case analysis is performed which is organized as follows:

First, contexts of the studied case organizations are compared and contrasted by following the guidelines suggested by Petersen and Wohlin (2009), so that other researchers can make comparisons with the findings of this research study.

Second, the findings of the studied case organizations are compared and contrasted based on the initial conceptual framework (discussed in Chapter 4: Section 4.4.3). This involves analysis relating to the requirements discovery and understanding across scaling agile levels (Section 7.2), requirements classification and organization across scaling agile levels (Section 7.3), requirements prioritization and negotiation (Section 7.4), requirements artefacts (Section 7.5).

Incorporating distributed members of decision-making teams, is discussed in Section 7.6. Challenges faced by the studied case organizations during decision-making on the priority of requirements across scaling agile levels and the potential strategies, are discussed in Section 7.7.

In the case organizations, three scaling agile decision-making levels, are studied, where primarily priority of requirements is decided. In the MEL organization, the levels include portfolio level, segment level, and team level. On the other hand, AKL organization termed them as portfolio level, tribe level, and team level. Specifically, the middle decision-making level, that is the segment level in MEL organization, and the tribe level in AKL organization, are tailored as per the studied case organizations' needs. According to the commonly reported scaling agile frameworks in the literature such as DAD, SAFe, the middle level is generally termed as the program level (Ambler, 2018; Leffingwell, 2011). In the studied cases, it is a kind of technical product area, where generalizing specialist teams focus on end-to-end, autonomously solving the assigned business problems. These are termed, features in MEL organization, and epics in AKL organization. In both cases, the middle decision-making level plays a twofold role in terms of making decision on the priority of requirements. Due to the commonalities in

the process at the middle level adopted to make decisions on the priority of requirements (i.e., features in MEL organization and epics in AKL organization), I have termed the middle decision-making level as a ***domain level,*** as a result of cross-case analysis. The in-depth explanation of process that is employed by the studied cases organization at the middle level (new terminology that I adopted is now ***domain level***) is provided in Section 7.4.

The cross-case analysis modifies some of the existing salient categories that had emerged from the individual case analysis. In addition to this, new categories also emerged as an outcome of the analysis (discussed in Section 7.8). Analysis further helps in evolving the conceptual framework of requirements prioritization in SADSD that was initially developed via individual case analysis as well as literature support (see Chapter 4: Section 4.4.3).

7.1 Context comparison of the studied case organizations

The studied case organizations, are distributed large organizations which implement Agile methods at the enterprise level. In order to do this, DAD framework, which is a scaling agile framework, was adopted by the MEL organization. On the other hand, Scrum @ scale was adopted by the AKL organization. MEL organization is an international software vendor, providing IT asset management solutions and has product and delivery teams across seven globally distributed sites. AKL, on the other hand, is one of the largest natural gas and electricity retailers in New Zealand and has nationally distributed teams over two locally distributed sites. Table 7.1 provides a summary of similarities and differences between the two studied case organizations.

Table 7.1

Context comparison of studied case organizations, Adapted from (Petersen & Wohlin, 2009)

Context facet	**Context element**	**MEL case organization**	**AKL case organization**
Product	Domain	IT asset management	Utility
	Product size	Large	Large
	Maturity of product	Long-lived mature product	Long-lived mature product
	Customization	Yes	No (In-house development)
Process	Characteristics of software development process	Scaling agile practices, Adapted DAD framework	Scaling agile practices, Adapted Scrum @ scale framework
	Start of method adoption	2015	2017
	Distributed development	Globally distributed	Nationally distributed
	Team size	Up to 8 members in a team	Up to 13 members in a team
Organization	Size	Large (1000+)	Large (500+)
	Number of development teams	25+ delivery teams	15+ delivery teams
	Hierarchical organizational model	Yes	Yes
	Organizational unit involved in the study	Product and engineering	Product and engineering
People	Roles and their responsibilities	Cross-functional decision-making team	Cross-functional decision-making team
Market	Setting	Business to Business	Enterprise product (in-house development)
	Access to customers or end users	Business customers and key end users	Internal customers (in-house development), key end users

7.2 Requirements discovery and understanding across scaling agile decision-making levels

An open innovation approach that involves internal sources and external sources, is employed by both studied case organizations, where potential ideas/requirements are leveraged across the organizational boundary (Damian et al., 2021).

7.2.1 Sources of requirements at the portfolio level

The following are the internal and external sources of potential ideas/requirements that have emerged at the portfolio level.

7.2.1.1 Internal sources of requirements

At the portfolio level, typically the *portfolio management* as it's termed in the MEL organization, and *Governance forum* in AKL (Ambler, 2018; Ambler & Lines, 2012), is the main internal source of potential ideas/requirements. However, potential requirements are collated from the other levels and/or teams that include the domain level, team level, sales and marketing team, and support team.

7.2.1.2 External sources of requirements

Potential ideas/requirements emerge from the external environment as well, that includes customers, and market competitors. At the portfolio level, typically the portfolio management collate emerging business and customer needs. In both case organizations, techniques that are employed to gather potential ideas/requirements from the external sources are not isolated to the portfolio level only. The same techniques, for example conducting customer interview, can be employed at the other scaling agile levels and/or teams such as domain and team levels, support team, and sales teams. Techniques that are employed by the case organizations to gather potential ideas/requirements from the external sources are discussed in Section 7.2.2.

7.2.2 Sources of requirements at the domain level

The following are internal and external sources of potential ideas/requirements that emerge at the domain level.

7.2.2.1 Internal sources of requirements

The product management, the PMs in MEL organization, and senior POs in AKL, (Ambler, 2018) are dominant internal sources who work with all possible sources - engineering, architecture, UX/UI- to collate and/or understand the potential requirements that can be imposed on products. However, potential ideas/requirements are also collated from the other levels and/or teams, including portfolio and team levels, support team, sales and marketing team.

Practices to collate potential ideas/requirements from the internal sources:

Various practices are employed by the case organizations to collate potential ideas/requirements (shown in Table 7.2). Some of the practices are common in both organizations represented as "Yes" whereas practice(s) that are unique are represented as "No" shown in Table 7.2. This comparison was made by analysing the collected data. The follow up email/discussion was performed with the study participants wherever occurrence of a particular practice was not clear in the studied case organization.

Practices that are employed to collect potential ideas/requirements from the internal sources, are least formal in both studied organizations. However, types of techniques that are employed vary in both organizations. MEL organization employed rich techniques compared to AKL organization.

Table 7.2

Techniques to collate potential requirements from internal sources at the domain level

Techniques	**Short description**	**MEL organization**	**AKL organization**
Engineering and support summit	Annual engineering summit involving around 50 to 60 engineers, (a couple of from each team), generally held for 3 days to collect engineering ideas.	Yes	No
Run hackathons	Run occasionally (typically on a yearly basis), where generally the PMs/POs and engineers collaborate on potential ideas/requirements	Yes	Yes
Reward incentives and recognition	quarterly reward system to encourage staff to bring innovative ideas	Yes	No
Casual conversation with staff	Generate ideas via casual conversation with staff (e.g., engineers)	Yes	Yes
Business strategy	Potential ideas/requirements that are posed on the product generated from the business strategy as well	Yes	Yes
Run workshops	ad-hoc workshops to generate innovative ideas	Yes	No
Champions across the organization	champions who generally collate and represent their teams' members ideas. For example, engineers' ideas are generally communicated via their POs.	Yes	Yes

7.2.2.2 External sources of requirements

Typical external sources include customers, market competitors, and partners. However, customers are the dominant external source of potential requirements/ideas. Customers'

ideas/requirements are typically communicated via the customer services team, sales and marketing team, support team, UX/UI team, and product management. The product management (i.e., PMs in MEL organization, and senior POs in AKL), typically are responsible for collating potential ideas/requirements from the customers. In MEL organization, there is a dedicated amount of time (40% to 50%) allocated for product management to collate and understand potential ideas/requirements from customers. Table 7.3 shows the techniques that are employed by the case organizations to collate potential requirements from the external sources.

Table 7.3

Techniques to collate potential requirements from external sources at the domain level

Techniques	**Short description**	**MEL organization**	**AKL organization**
Market	Competitors analysis	Yes	Yes
	Track current trends/market trends	Yes	Yes
	Interviews with industry analysts	Yes	No
Partners	Potential ideas/requirements are generated from the partners (e.g., development partners- with whom product components are developed)	Yes	No
Customers	Frequent focus group	Yes	No
	Frequent interviews	Yes	Yes
	Customers Advisory board where typically the product management collaborate with a group of customers in order to get and/or understand their needs	Yes	Yes
	Customer portal an online customer community operated via product management tool AHA to get customers' feedback/ideas.	Yes	No
	Quarterly business review with the biggest customers	Yes	No
	UX research studies (e.g., customer journey mapping) to identify customers' pain points and opportunity	Yes	Yes
	Online survey (regular and ad-hoc survey) to get customers' feedback	No	Yes
	App store to get customers' feedback	No	Yes
	feedback pop-up on the company's website through which customers can give their feedback on the services that are provided by the organization	No	Yes

7.2.3 Sources of requirements at the team level

Following are the internal and external sources of potential ideas/requirements that emerge at the team level.

7.2.3.1 Internal sources of requirements

At the team level, engineering team members - PO, AO, TL, engineers in MEL organization; PO, SM, BA, engineers, UX, testers in AKL organization, are typically the main internal source of requirements. However, potential ideas/requirements are also collated from the other levels and/or teams that include portfolio level, domain level, sales and marketing team, and support team. Table 7.4 shows the techniques that are employed at the team level in the studied case organizations.

Table 7.4

Techniques to collate potential requirements from internal sources at the team level

Techniques	**Short description**	**MEL organization**	**AKL organization**
Feature Kick-off	Feature kick-off that is generally organize before starting the implementation of a requirement where engineering team works collaboratively to explore the scope and/or requirements envisioning.	No	Yes
Innovation morning	Fortnightly half a day is generally reserved for engineers for innovative work	Yes	No
Inception phase	Inception phase that generally takes two weeks of time and organize before starting the implementation of a requirement where engineering team members works collaboratively to explore the scope and/or requirements envisioning.	Yes	No

7.2.3.2 External sources of requirements

Market competitors, customers, and partners are generally the external sources of potential ideas/requirements. At the team level, typically the PO collates potential ideas/requirements from the external sources, wherever needed.

7.3 Requirements classification and organization across scaling agile decision-making levels

Requirements classification: A hierarchical model of requirements (Leffingwell, 2011) is employed by the studied cases, to classify and organize the prioritization of requirements against each other that belongs to the same or different categories and/or same abstraction level. In both organizations, requirements that are organized

hierarchically, are not only business specific requirements, that provide direct value to the business, but a mixture of other requirements as well. This includes architectural and maintenance work. As shown in Table 7.5, MEL organization classified the requirements into four categories that include *new feature* work, *defect correction* work, *architectural* work, and *technical debt* work, where each category has a specific capacity allotment. On the other hand, AKL organization also classified the requirements that includes *essential maintenance* work, *growth and efficiency* work, *compliance* work, and *discretionary maintenance*. Requirements' categorization varies in both organizations, but they have some similarities in terms of type of requirements. For instance, in MEL organization, requirements that are needed to implement new opportunities are termed as new feature work' and 'architectural evolution work'. On the other hand, AKL organization termed them as 'growth and efficiency work'. Requirements that belong to each of the categories, span various levels - portfolio level, domain level, team level, in terms of decision making on their priorities (see Table 7.5).

In both case organizations, typically the requirements that require significant investment, required a formal decision-making process. These were analysed and evaluated in the first part of domain level, then submitted to the portfolio level for formal prioritization and then allocated to the engineering teams/delivery teams for implementation (discussed in Section 7.4). On the other hand, requirements that require a small amount of effort (such as one week of work), are typically sorted out at the team level (discussed in Section 7.4).

Requirements abstraction model: The hierarchical requirements model (as shown in Table 7.6) that is employed by the case organizations to organize the requirements, is similar to the Leffingwell (2011) model (epic→feature→user story). However, they have tailored the model as per their need.

In both organizations, strategic theme(s) are the highest abstraction level of requirements that connect with the organization's overall business strategy. Strategic theme(s) belong to the portfolio level in terms of decision making on their priorities.

The next abstraction level varies in both case organizations. In MEL organization, feature is in the second abstraction level of requirements. A feature is something that provides direct value to the customers and business, which can be typically implemented in a single release IR (i.e., three months), and typically required more than one delivery team to implement. All types of requirements - new feature work, architectural evolution work, technical debt work, defect correction work, that required significant effort, typically

come in the form of features and generally span the first part of domain level and portfolio level in terms of decision making. On the other hand, in AKL's organization, Epic is the second abstraction level of requirements, which generally takes more than one release IR (i.e., more than three months) to implement and requires more than one delivery teams to implement.

Table 7.5

Requirements classification across scaling agile decision-making levels

Scaling agile decision-making levels	**MEL organization**		**AKL organization**	
	Requirements categorization	**Short description of the requirement type**	**Requirements categorization**	**Short description of the requirement type**
Portfolio level Domain level Team level	Defect Correction work	Work that is needed to be done to address gaps relative to the work that have attempted in the past. 10% to 20% resources are reserved[15].	Essential Maintenance	Essential Maintenance represents the work that is considered as mandatory so that results stop working something if not resolved.
Portfolio level Domain level Team level	New feature work	Work that is performed for business specifically for customers. 70% to 80% resources are reserved.	Growth and efficiency	Growth and efficiency represent the work that is performed to deal with new products and features. E.g., energy saving tips.
Portfolio level Domain level Team level	Architectural work	Work that is needed to be technically built in the product architecture in order to develop the new features that will be requested by the customers in the future. 5% to 15% resources are reserved.	Compliance work	Compliance represents the work that are considered as mandatory otherwise face breaking the law
Portfolio level Domain level Team level	Tech-debt work	Tech-debt represents the gaps that are needed to be filled in order to have fully solid infrastructure. 25% to 30% resources are reserved.	Discretionary Maintenance	Discretionary Maintenance represents the work that may be considered as optional. For example, tech-debt work, internal improvement work.

[15] Note: This is a tentative resource allocation.

Table 7.6 Requirements hierarchy model employed by the case organizations

Scaling agile decision-making levels	**MEL organization**		**AKL organization**	
	Requirements hierarchy and their short description		**Requirements hierarchy and their short description**	
Portfolio level	Strategic theme(s) *(1st abstraction level of requirements)*	Themes are the business goals that an organization wants to achieve in a certain time -period (e.g., yearly goals).	Strategic theme(s) *(1st abstraction level of requirements*)	Themes are the business goals that an organization wants to achieve in a certain time period (e.g., yearly goals).
	Feature(s) *(2nd abstraction level of requirements)*	Typically defined and evaluated at the 1st part of domain level but formally signed-off at the portfolio level	Epic(s) *(2nd abstraction level of requirements)*	Typically defined and evaluated at the 1st part of domain level but formally signed off at the portfolio level.
1st part of domain level	Feature(s) *(2nd abstraction level of requirements)*	A feature is something that typically provides concreate value to the customers and business, and can be implemented in a single release iteration (i.e., 3 months)	Epic(s) *(2nd abstraction level of requirements)*	An Epic is a large development initiative that realizes the value of strategic themes and can take multiple release iterations to implement (i.e., more than 3 months)
2nd part of domain level	Epic(s)	Convert feature(s) into epic(s) for the delivery teams for implementation	-	No change in the nomenclature.
	User story(s) *(3rd abstraction level of requirements)*	User story(s) is a detailed description of functionality.	Feature(s) (3rd *abstraction level of requirements)*	A feature(s) is something that typically provides concreate value to the customers and business, and typically implement in a single iteration (i.e., 2 weeks).
Team level	User story(s) *(3rd abstraction level of requirements)*	Breakdown User story(s) into further user stories wherever needed.	User story(s) *(4th abstraction level of requirements)*	User story(s) is a detailed description of functionality.
	Task(s) *(4th abstraction level of requirements)*	Task(s) that describe in technical terms what things are required to be done in order to claim a user story at the end of sprint or iteration.	Task(s) *(5th abstraction level of requirements)*	Task(s) that describe in technical terms what things are required to be done in order to claim a user story at the end of sprint or iteration.

The third abstraction also differs in both organizations. In MEL organization, there is no difference between a feature and epic. The main reason to term features as 'epics' is the adoption of the CTs- Jira that is employed by the MEL organization for the engineering teams. In MEL organization, the user story is the third abstraction level of requirements that provides detailed understanding of need/requirements and fit for a team. In MEL organization, a user story typically spans the second part of domain level and the team level in terms of their priority decision-making. On the other hand, in AKL organization, a feature is a third abstraction level which is derived from epic and typically decided at the second part of the domain level. A feature typically needs further decomposition at the team level to provide detailed understanding of requirements to the engineering teams for implementation. A feature gets implemented in a single development iteration (i.e., two weeks), and by an individual delivery team.

In MEL organization, a user story which is the third abstraction level of requirements is decomposed into further user stories if needed at the team level. In AKL organization, user story is the fourth abstraction level of requirements which typically spans team levels in terms of decision making on the priorities. In both studied cases, a user story typically provides detailed understanding of requirements to the delivery teams and is generally delivered by a single delivery team within a single development iteration/sprint (i.e., two weeks). In both organizations, a task is something that is described in technical terms what things are required to be done in order to claim a user story at the end of a development IR. A task is typically an optional abstraction level in both studied case organizations.

7.4 Requirements prioritization and negotiation across scaling agile decision-making levels

In the studied case organizations, typically the classification of requirements that are organized via the requirements hierarchical model serve as an input in terms of deciding the scaling agile decision-making levels of potential requirements. As discussed in Section 7.3, potential requirements that require significant investment and/or effort (e.g., two to three months of work) require a formal decision-making process (i.e., initial evaluation at the domain level and formal decision-making at the portfolio level) in terms of making decision on their priority. On the other hand, potential requirements that require a small amount of effort (e.g., one week of work) can be sorted out at the team level.

In the studied case organizations, decision making on the priority of requirements span various levels (i.e., portfolio level, domain level, and team level) with varying details, decision making practices, decision-making team, decision-making events, decision-making factors. Both organizations have some similarity and differences in terms of decision making on the priority of requirements across scaling agile levels which are discussed in the following sections.

7.4.1 Portfolio level

Portfolio level is the highest decision-making level where primarily the strategic themes (which is the first abstraction level of requirements in both case organizations as per their requirements hierarchy model) that connect with the organization's overall business strategy are decided and prioritized. Table 7.7 shows the similarity and differences between both organizations in terms of decision making on the strategic theme(s).

7.4.1.1 Decision-making team

Portfolio management which is a cross-functional team that consists of senior leadership (Ambler & Lines, 2012) collaboratively makes decisions on strategic theme(s). Portfolio management typically act as a boundary spanning team (Jain et al., 2014) in terms of decision making on the priority of requirements (e.g., strategic themes).

In MEL organization, SVP of product, SVP architecture, SVP engineering, SVP UX/UI, domain leaders (i.e., segment leaders) constitute of portfolio management. In AKL organization, portfolio management which is called Governance forum consists of senior leadership that includes CDO, CCO, domain leaders (i.e., GMs) who collaboratively decide and prioritize the strategic theme(s).

7.4.1.2 Decision-making events

A quarterly event is held where portfolio management formally communicate and represents the strategic theme(s) across the whole organization. This event is termed bigroom planning in the studied case organizations and typically acts as a boundary spanning event (Jain et al., 2014) that provide shared understanding of the organization's strategy(s) across the whole organization. Duration of the event varies in both organizations. In MEL organization, their event is typically held for a week. On the other hand, this event is organized for a day in AKL organization.

Table 7.7

Decision making at the portfolio level in the studied case organizations

Scaling agile decision-making levels	**MEL organization**				**AKL organization**			
	Decision-making team	Decision-making events	Decision-making practices	CTs	Decision-making team	Decision-making events	Decision-making practices	CTs
Portfolio level	Portfolio management Cross-functional team Boundary spanning decision-making team	Quarterly Bigroom planning boundary spanning event Quarterly strategic portfolio sync-up	Quantitative approaches Qualitative approaches	AHA	Governance forum Cross-functional team Boundary spanning decision-making team	Quarterly Bigroom planning boundary spanning event Quarterly strategic portfolio sync-up	Quantitative approaches Qualitative approaches	ADO

Strategic portfolio sync-up is another quarterly event to decide and revise the strategic themes wherever needed before communicating them formally via ***bigroom planning*** to the whole organization.

7.4.1.3 Decision-making factors

In the studied case organizations, business value plays a significant role in terms of making decisions on the strategic theme(s). Strategic theme(s) that provide biggest opportunity to the business and customers are typically selected as a high priority theme(s).

7.4.1.4 Collaborative technologies (CTs)

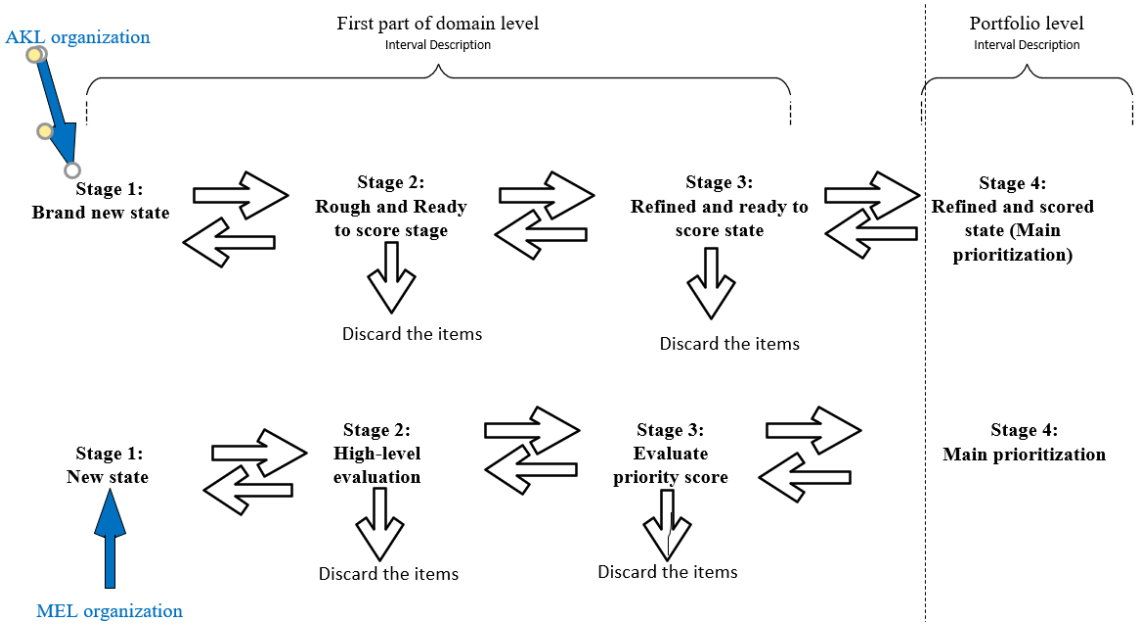
In the studied case organizations, CTs plays a prominent role through which requirements (e.g., strategic themes) are represented, managed and visualized that further helps in getting the shared understanding of requirements across the whole organization. In both organizations, CTs are adapted as per their need. In MEL organization, AHA emerged as primary CTs which is employed synchronously as well as asynchronously to make decisions on the priority of requirements at the portfolio level. On the other hand, ADO emerged as a primary CT in AKL organization.

7.4.2 First part of domain level

All the potential requirements that are posed on products are filtered via various stages that results in a priority list of requirements and allocated to the engineering for implementation. Typically, the requirements that require significant effort (termed as features in MEL organization, epics in AKL organization) span the first part of domain level and portfolio level in terms of decision making on their priority. As an outcome of each stage, either the potential requirement will be accepted or discarded. In the former case, the accepted candidate requirement will either stay in the backlog until the priority is raised, or the accepted candidate requirement will be moved into the next stage. At each of the decision-making stages new potential requirements can emerge that are initially pushed into Stage 1. In addition to this, the potential requirements can be pushed back to the previous stage for further analysis if the provided decision-making information is not sufficient.

Figure 7.1

Decision-making stages in the 1st part of domain level



As depicted in Figure 7.1, in the first part of domain level, potential requirements that are emerging within each domain are first evaluated at the high level (i.e., Stage 2- e.g., alignment of candidate requirement(s) with strategic theme(s)). This high-level evaluation yields filtering the requirements that are valuable to the organization. Candidate requirements that are selected as an outcome of high-level evaluation (i.e., Stage 2) are further evaluated at Stage 3 by employing the whole set of decision-making factors (as discussed in Section 7.4.2.3) to evaluate the priority score of candidate requirements. After evaluating the priority score of candidate requirements (i.e., features in MEL organizations, epics in AKL organization) they are submitted to the portfolio level (i.e., Stage 4) for formal sign-off and prioritization.

7.4.2.1 Decision-making team

Decision making team in the first part of domain level: The decision-making team is typically a cross-functional team who collaboratively makes decisions on the priorities of requirements. These requirements consist of features in MEL organization, and epics in AKL.

In MEL organization, typically a triad (PM, director of engineering, program architecture owner), director of UX/UI wherever needed, perform initial evaluation of potential requirements, or features.

On the other hand, in the AKL organization, the triad consists of a senior PO, tribe lead, solution architect who work collaboratively to perform the initial evaluation and decision-making on the potential requirements, or epics. Input from senior UX/UI is also undertaken, wherever it's required.

Decision making team at the portfolio level: Potential requirements that are posed on product(s) and are evaluated in the first part of the domain level (i.e., Stage 1, Stage 2, Stage 3 as in Figure 7.1) are formally signed off and prioritized at the portfolio level (i.e., Stage 4 as in Figure 7.1) by the portfolio management team. In both organizations, typically the leaders of the domain, the segment leaders in MEL organization, GMs in AKL, generally act as *boundary spanners* (Jain et al., 2014). They typically are responsible for providing all the required facts and data related to the potential requirements that are emerging within their domain during decision making at the portfolio level.

7.4.2.2 Decision-making events

Decision making event(s) at the first domain level: In both organizations, decision-making teams meet on a regular cadence. Decision-making team typically meets on a fortnightly basis where potential new requirements get evaluated, existing requirements are planned, any changes in priority are decided, and dependencies if any are discussed. However, they have weekly check-ins as well.

Decision making event(s) at the portfolio level: Portfolio management typically meets on a ***quarterly basis***, termed ***bigroom planning*** in the studied organizations. Portfolio management typically validate the requirements against the organizational strategies and/or strategic themes, and they may make decisions where there are constraints. For instance, it may be deciding between two important requirements, where they only have capacity to do one. However, recommendations and priorities to assist them, ultimately come from the delivery teams/engineering teams. The ***bigroom planning event*** typically acts as a ***boundary spanning decision-making event*** (Jain et al., 2014)***,*** where requirements across the domains are formally decided and communicated to engineering for implementation. However, this event is tailored as per each organization's needs. In MEL organization, *bigroom planning* takes a week to complete. On the other hand, in the AKL organization, *bigroom planning* is a one-day event. The main reason of this variance typically include size of organization, complexity of the product and their way of working.

Another event is ***portfolio sync-call*** which is typically an hour-long meeting, where portfolio management typically meet for check-ins that provide visibility on the requirements' progress towards achieving the strategic themes, revisit the existing requirements, remove impediments, determine any change in priorities, score and rank the new requirements, and address dependencies. In MEL organization, this event happens on a ***monthly basis***. On the other hand, in AKL organization, it takes place on a ***fortnightly basis***.

Ad-hoc call- in both organizations, an ad-hoc discussion can also be organized wherever needed.

7.4.2.3 Decision-making practices

In the studied cases potential requirements that emerged from each of the domain(s) and/or contribute to the formation of strategic theme(s), are scored and ranked, employing a combination of quantitative objective weights and qualitative approaches requiring human judgement.

7.4.2.3.1 Decision-making factors

In MEL organization, key factors are: business value and urgency, risk reduction and opportunity, and effort. On the other hand, factors that are employed by AKL organization, include NPV and strategic fit.

In both studied case organizations, decision-making factors are evaluated via a numerical scale (1 to 10 in MEL, 1 to 5 in AKL). This is an automated matrix configured via CTs (*AHA* in MEL organization, *ADO* in AKL organization) and is employed where one represents the lowest value item, and the highest number represents the most valuable item.

7.4.2.3.2 Decision-making techniques

Weighted shortest job first approach (WSJF): both studied organizations employed the WSJF approach to evaluate the priority score of potential requirements which are features in MEL organization, epics in AKL, that emerging from each domain(s) and/or contribute to the formation of a theme. However, they have tailored the practice as per their need. In MEL, WSJF is applied in the form of quantitative as well as qualitative practice, whereas AKL organization applied WSJF in the form of qualitative practice.

Perform additional research wherever needed: Participants of both studied cases asserted that sometimes reaching an agreement on the priority of requirements, is

difficult. In such cases, often additional research is performed to collect all the required information to convince decision makers. In addition to this, in such cases, if needed, generally product management takes an ultimate decision on requirement(s) priority.

Learning experience: the priority score of requirements can be adjusted, based on the decision makers' knowledge and experience, especially when more than one item holds the same priority score and/or has the same importance. In such cases, relative priority plays a significant role, generally the benefits such as return on investment, customer impact of one over another are compared.

7.4.2.4 Collaborative technologies

In both organizations, CTs play a prominent role through which requirements are represented, managed and visualized, that further helps in getting the shared understanding of requirements across the whole organization. In both organizations, CTs are adapted as per their need. In MEL, *AHA* emerged as primary CT which are employed synchronously as well as asynchronously to make decisions on the priority of requirements in the first part of domain level. On the other hand, *ADO* emerged as the primary CT in AKL organization.

7.4.3 Second part of domain level

In the second part of the domain level, the priority list of requirements (i.e., features in MEL organization and epics in AKL organization) that is contained in the portfolio backlog, serves as an input for each of the domains. Requirements that are formally signed off by the portfolio management, are allocated to the domain(s), based on their relevance to the domain(s). For each requirement (i.e., features in MEL and epics in AKL), there is a dedicated person who is typically a product management team member. It's a PM in MEL, and a sponsoring PO in AKL. They act as ***boundary spanners*** (Jain et al., 2014) responsible for providing all the relevant information, for the engineering teams for implementation. The boundary spanner often occupies a role of PO in the engineering team who are responsible for the implementation of requirements. Generally, a one-to-one relationship between the product owner and engineering team is maintained. However, one product owner can manage more than one engineering team as well if required.

In the second part of the domain level, in MEL organization, a new name *"epics"* is given to the requirements or features that are formally prioritized at the portfolio level. An Epic is the reflection of a feature. The main reason to give them a new name, is the adoption

of CTs- namely *Jira* that is employed for the engineering teams to manage requirements. The main reason to change in the nomenclature of 'feature' to 'epic' is the tool limitation as Jira does not directly support terminology as 'feature'. However, indirectly it can be possible via an integration of third-party tools. On the other hand, in the AKL organization, there is no change in the nomenclature of requirements or epics. In the second part of the domain level, requirements (i.e., epics/features in MEL organization, epics in AKL organization) that are formally decided at the portfolio level are decomposed into further requirements, such as, user stories in MEL, features in AKL. These are typically implemented via a short development cycle of two weeks. In MEL, decomposed requirements or user stories, are typically detailed enough in terms of providing required information, to the engineering teams. However, they can be decomposed into further requirements if needed at the team level. On the other hand, in AKL, decomposed requirements or features, typically require further decomposition at the team level. Table 7.8 shows decision making practices employed by the studied case organizations at the second part of the domain level.

Table 7.8

Decision making practices employed by the studied case organizations at the second part of domain level

Scaling agile decision-making levels	**MEL organization**				**AKL organization**			
	Decision-making team	**Decision-making events**	**Decision-making practices**	**CTs**	**Decision-making team**	**Decision-making events**	**Decision-making practices**	**CTs**
2nd part of domain level	Cross-functional team Boundary spanning decision-making team	Fortnightly team of team meeting weekly sync-up Boundary spanning event Ad hoc meeting (as and when needed) As an outcome produced detailed requirements	Quantitative approaches Qualitative approaches Spanning boundary in terms of decision-making	Jira	Cross-functional team Boundary spanning decision-making team	Fortnightly team of team meeting Weekly sync-up Boundary spanning event Ad hoc meeting (as and when needed) As an outcome further decomposition needed at the team level	Qualitative approaches No particular scoring system	ADO

7.4.3.1 Decision-making team

In the studied organizations, decision-making is typically a boundary spanning team consisting of a delivery teams' triad with PM/POs, AOs, TLs, in MEL organization; sponsoring PO/POs, BAs, SMs in AKL organization. The delivery teams' triad is a cross-functional decision-making team working collaboratively to breakdown a requirement and make decision on their priorities (i.e., epic/feature in MEL, and an epic in AKL, that breaks down into further requirements).

7.4.3.2 Decision-making events

Decision-making teams meet on a regular cadence, to make decisions on the priorities. A ***Team of team meeting*** which is typically ***a boundary spanning decision-making event*** (Jain et al., 2014) generally occurred on a fortnightly basis, where priorities of requirements - user stories in MEL, and features in AKL across teams, are decided. Decision-making teams have weekly sync-up calls as well for check-ins where in-progress requirements are monitored, existing priorities are revisited, they remove impediments, and change in priorities are discussed. However, decision-making can occur on an ad-hoc basis if needed.

7.4.3.3 Decision-making practices

Decision-making practices greatly vary in the studied case organizations. Decision-making practices are generally a combination of qualitative and quantitative approaches.

Decision-making factors: business value (BV) is the key prioritization driver typically having higher influence, while making decisions on the requirements priorities which are *user stories* in MEL organization, and *features* in AKL, that are decided in the second part of the domain level.

In MEL organization, the BV of requirements which are dedicated for strategic work and typically represented as user stories consists of the impact on customers and degree of urgency. Whereas the BV of requirements that are dealing with technical debt work and enhancement work, are typically evaluated via the severity and scope factor, such as number of customers and/or the impact on the system functionality. BV is typically evaluated via employing quantitative weights on the numerical scale one to five where lower value indicates lower impact requirements and higher value indicates highly valuable requirements.

In AKL organization, the BV of requirements or features, is decided in the second part of the domain level and typically are evaluated based on tacit knowledge. No particular scoring system and/or technique (e.g., MoSCoW) is employed.

In addition to the BV, there are various other aspects that are taken into consideration in the studied cases, such as team ability and capacity, functional dependencies, strategic priorities.

Decision-making techniques:

MoSCoW (must have, should have, could have, would have)**:** MoSCoW is a key qualitative practice which is employed by the MEL organization to decide the priority on requirements or user stories in the second part of domain level. On the other hand, in AKL, there is no indication that the MoSCoW technique emerged from the collected data.

Learning experience: the priority score of requirements can be adjusted, based on the decision makers' knowledge and experience, especially when more than one item holds the same priority score and/or has the same importance.

7.4.3.4 Collaborative technologies

In the studied case organizations, CTs play a prominent role through which requirements are represented, managed and visualized, that further helps in getting the shared understanding of requirements across the whole organization. In both organizations, CTs are adapted as per their needs. In MEL, *Jira* emerged as primary CT which is employed synchronously as well as asynchronously to make decisions on the priority of requirements in the second part of domain level. On the other hand, *ADO* emerged as the primary CT in AKL.

7.4.4 Team level

Agile methods that are commonly employed by the engineering team(s), include *Scrum* (Schwaber, 2004) and *Lean* (Ambler & Lines, 2012). At present, *Scrum* is the dominant method that is employed by the engineering team(s) for implementation in the studied cases. However, the Lean way of working is getting tremendous attention among engineering teams over Scrum, in MEL organization. The main reason behind this is that lean supports continuous prioritization, yielding the result that changes are easy to make in the decided priorities, if required. An example of this is *unplanned work*.

At the team level, each team has their own backlog that typically contains strategic work, typically in the form of a user story, technical debt work, defect correction work, spikes requiring technical investigation, and slipping stories or left-over work from the previous development IR/sprint. In both organizations, within the team, the PO typically acts as a boundary spanner (Jain et al., 2014) who generally provides clarity and understanding about requirements, among team members. On the team level, the main goal of the prioritization is to organize the already-selected requirements into a sufficiently rational implementation order. Table 7.9 shows the decision-making practices employed by the studied cases at the team level.

Table 7.9

Decision making practices employed by the studied case organizations at the team level

Scaling agile decision-making levels	**MEL organization**				**AKL organization**			
	Decision-making team	**Decision-making events**	**Decision-making practices**	**CTs**	**Decision-making team**	**Decision-making events**	**Decision-making practices**	**CTs**
Team level	Team's triad Cross-functional team Boundary spanning decision-making team	Fortnightly IR planning meeting, weekly refinement meeting Look-ahead meeting Inception phase Ad-hoc meeting (as and when needed) As an outcome produced detailed requirements	Quantitative approaches Qualitative approaches Spanning boundary in terms of decision-making	Jira	Team's triad Cross-functional team Boundary spanning decision-making team	Fortnightly sprint planning meeting, weekly refinement meeting Feature kick-off Ad-hoc meeting (as and when needed), As an outcome produced detailed requirements	Qualitative approaches No particular scoring system	ADO

7.4.4.1 Decision-making team

In the studied organizations, the delivery team triad, that is PO, AO, TL in MEL organization; and PO, BA, SM in AKL, works collaboratively to make decisions on the priority of requirements within the team.

7.4.4.2 Decision-making events

In both cases, decision-making events occurred on a regular cadence typically fortnightly, with a weekly sync-up call, and on an ad-hoc basis, as and when necessary.

The following are the decision-making events which are organized to decide the priority of requirements within the teams:

Backlog refinement meeting: In the studied organizations, delivery team(s) have an ongoing backlog refinement, that typically occurred on a weekly basis, where delivery team triad, UX/UI team member if required, collaboratively decides priorities on requirements, such as user stories.

Iteration/Sprint planning meeting: In the studied case organizations, fortnightly meetings are organized where priorities are decided for the upcoming development IR/sprint.

Look-ahead meeting: In MEL organization, a look-ahead meeting (held in the second week of the current IR), is organized to support backlog refinement and IR planning. A look ahead meeting is typically organized in two sessions, for the triad look-ahead and team look ahead. Triad look-ahead is where the next highly valuable requirements that the team need to implement, in the next two IRs are refined. Team look-ahead ensures requirements are clear to the engineers.

7.4.4.3 Decision-making practices

In MEL organization, typically the requirements or user stories, that are decomposed in the second part of domain level are detailed enough for the engineering for implementation. Typically, no further decomposition of requirements happens at the team level. Therefore, decision-making practices, which are generally a combination of qualitative and quantitative approaches are employed in the second part of domain level and are adapted at the team level as well.

On the other hand, in AKL organization, requirements, or features, need to be decomposed into further requirements, to provide detailed understanding of requirements

for implementation by the engineering team. In AKL organization, priority decisions are typically made on tacit knowledge. No specific scoring practices and qualitative approach (e.g., MoSCoW) are employed. Typical factors that are taken into consideration, include strategic priority, feature priority, team ability and capacity, dependencies, and specifically technical dependencies.

7.4.4.4 Collaborative technologies

In both the studied cases, CTs play a prominent role through which requirements are represented, managed and visualized, that further helps in getting the shared understanding of requirements across the whole organization. In both organizations, CTs are adapted as per their need. In MEL organization, *Jira* emerged as primary CT which is employed synchronously as well as asynchronously, to make decisions on the priority of requirements at the team level. On the other hand, *ADO* emerged as the primary CT in AKL organization.

7.5 Requirements artefacts across scaling agile decision-making levels

In both studied case organizations, requirements artefacts are produced in various forms at various levels: portfolio level, domain level, team level, that generally provide information that is needed by the decision makers to make decisions on the priorities. In addition to this, requirements artefacts help in getting the shared understanding of requirements across scaling agile levels.

In each case, requirements artefacts are produced via CTs. However, CTs are adopted as per organizations' needs. In MEL organization, AHA, Jira, Confluence, Spreadsheet, PowerPoint are employed as CTs to specify the requirements. On the other hand, AKL organization employed ADO, Confluence, PowerPoint for building the requirements artefacts.

In both cases, the CTs- *Confluence* is mainly employed to maintain documentation of requirements (e.g., for a detailed discussions and/or analysis on requirements such as recordings of discussion with customers) that helps in building the shared understanding of requirements.

7.5.1 Requirements artefacts at the portfolio level

Strategic theme(s): In the studied case organizations, strategic theme(s) is the key requirements artefact built at the portfolio level and act as a ***boundary spanning requirement artefact*** (Jain et al., 2014) that has influence on the decision-making on the

priorities of requirements, across all scaling agile decision-making levels. An electronic template which is configured via CTs -*AHA* in MEL organization, and *ADO* in AKL, is employed to specify the strategic theme(s). However, CTs *PowerPoints* slides related to the strategic theme(s) are also maintained specifically to use asynchronously during decision-making events, such as bigroom planning, for knowledge sharing.

Portfolio backlog is another boundary spanning requirement artefact which is configured via CTs. AHA is used in MEL organization, and ADO in AKL, to build in at the portfolio level. The portfolio backlog contains the priority list of requirements - themes, features in MEL; themes and epics in AKL, which are allocated to the engineering teams for implementation. In MEL organization, the *spreadsheet* as CT is also employed to build an informal portfolio backlog at the portfolio level, which is specifically used during decision-making event like bigroom planning. The main reason behind the adoption of *Spreadsheet* is that it enables synchronous editing during decision-making events, specifically helpful when the members of decision-making team are distributed.

7.5.2 Requirements artefacts at the first part of domain level

Requirements artefacts that are built in the first part of domain level, seems to be case and company specific. In MEL organization, requirements that require formal decision-making process, are termed 'features'. On the other hand, AKL termed them ‘epics'. An electronic template configured via CTs, *AHA* in MEL, and *ADO* in AKL, is employed to document the requirements. Business-case, an electronic artefact configured via AHA in MEL organization, and ADO in AKL, is a common requirement artefact that contains all the required facts and data needed to make decisions on the priority of requirements. Business-case typically acts as a boundary spanning requirement artefact, typically consumed and/or built during decision making events to make priority decisions.

7.5.3 Requirements artefacts at the second part of domain level

Requirements artefacts that are built in the second part of the domain level, seem to be case and company specific. In MEL organization, a *storyboard* which is typically a boundary spanning requirement artefact (configured via CT Jira), is built in the second part of the domain level. On the other hand, feature(s) which is typically a boundary spanning requirement artefact, a template configured via CT *ADO*, is built in the AKL organization.

Backlog, termed a *work item list* in MEL organization, and *product backlog* in AKL, contains a priority list of requirements which are user stories in MEL, and features in AKL. Backlog typically acts as a boundary spanning requirements artefact which is configured via CTs - Jira in MEL, ADO in AKL.

7.5.4 Requirements artefacts at team level

At the team level, User story(s), spike(s), task(s), team backlog are the requirements artefacts which are configured via CTs (i.e., Jira in MEL organization, ADO in AKL organization).

7.6 Incorporate distributed members of decision-making teams

In the studied organizations, members of the decision-making team who make decisions on the priority of requirements across scaling agile levels are distributed globally in MEL organization and locally distributed in AKL.

This section compares and contrast the practices that are employed by the case organizations to incorporate distributed team members during decision-making on the priority of requirements across scaling agile decision-making levels.

7.6.1 Mode of communication at the portfolio level

At the portfolio level, *bigroom planning* is a formal decision-making event that is organized on a Quarterly basis (Q1, Q2, Q3, Q4), where priorities on requirements such as, themes and features, are decided. In both organizations, portfolio management, who typically make decisions on the priority of requirements at the portfolio level are distributed, globally in MEL organization, and locally distributed in AKL. Both organizations have stressed the importance of ***face-to-face communication*** during decision making that yields outcomes that have been easy to reach through consensus, arriving at better understanding of requirements through collaboration. In order to do that, both organizations have arranged ***travel for the decision makers***, to attend the decision-making event(s) such as bigroom planning. At the AKL organization, where decision makers are locally distributed, they typically have more frequent face to face collaboration, typically every quarter, when planning is face-to-face in an auditorium either at the head office or Hamilton site. This is more frequent than MEL organization. On the other hand, in MEL, where cost constraints hinder the occurrence of face-to-face collaboration, generally meetings are four times a year and in-person at least twice a year.

These are in an auditorium either at the head office or Melbourne site and alternating virtually, every three months via CTs discussed in Section 7.6.5.

7.6.2 Mode of communication at the domain level

In the studied case organizations, decision-making teams who makes decision on the priority of requirements at the domain level, are either at the same location or distributed. Therefore, decision-making events are either organized face-to-face (in-person) or virtually via CTs (discussed in Section 7.6.5).

7.6.3 Mode of communication at the team level

At the team level, in both organizations, all team members are generally at the same location. Typically, the decision-making events- team backlog refinement, sprint/iteration planning meeting - are team specific and are organized face-to-face in the studied organizations.

7.6.4 Current mode of communication across scaling agile levels (i.e., during the time of data collection)

During the time of data collection (September 2020 in MEL organization, March 2020 in AKL organization), due to the COVID-19 pandemic, team members were working remotely. Therefore, all the scaling agile decision-making events were organized virtually via CTs (web conferencing tools discussed below in Section 7.6.5).

7.6.5 Collaborative technologies

Web conferencing tool: The studied cases employed web conferencing tools, such as MSTeams, as their primary synchronous communication tool to communicate with cross-site members of the decision-making team. The application sharing feature of MSTeams as CT, enable cross-site members of decision-making teams, to apply synchronous editing of requirements artefacts if needed during decision-making event(s).

In addition to the formal synchronous communication, informal synchronous communication (e.g., instant messaging, instant calling, team(s) specific channel to communicate internally within the team) as well as asynchronous communication (e.g., offline chat, tag to notify people, recordings of decision-making events) are also organized via web conferencing tool (i.e., MSTeams) to communicate with distributed members wherever needed.

Email as CT is employed as an asynchronous communication mechanism to communicate with decision-making teams in the studied case organizations.

7.7 Challenges and potential strategies

Challenges that are faced by the organizations during decision-making on the priority of requirements across scaling agile decision-making levels, are outlined in Table 7.10. The potential strategies that emerged to overcome those challenges, are outlined in Table 7.11. challenges and potential strategies that were not identified in the studied case organization are represented via a blank value in Table 7.10 and Table 7.11.

Table 7.10

Challenges faced by studied case organizations

Challenges	**MEL organization**	**AKL organization**
Distributed distance (DD)	Yes	Yes
Temporal distance (TD)	Yes	N/A
Tension between product and engineering (TPE)	Yes	Yes
Accumulating technical debt (ATD)	Yes	Yes
Quality of requirements (QoR)	Yes	Yes
Bias during decision making (BDM)	Yes	Yes
Difficult to measure Business Value (DBV)	Yes	Yes
Dealing with quality requirements (QRs)	Yes	Yes
Difficulty in estimating effort (DEE)	Yes	Yes
Inadequate requirements analysis (IRA)	Yes	Yes
Lack of requirements clarity (LRC)	Yes	
Dependencies across domain(s) (DaD)	No	Yes

Table 7.11

Potential strategies to overcome challenges emerged from the studied case organizations

Challenge category	**Potential strategies**	**MEL organization**	**AKL organization**
For DD	CTs (e.g., MSTeams)	Yes	Yes
For TD	CTs (e.g., MSTeams) Organize travel Formal synchronous (e.g., dedicated events- like bigroom planning) and asynchronous communication (e.g., Email) Informal synchronous (e.g., instant messaging) and asynchronous communication (e.g., offline chat) In-person meeting wherever possible Early morning and Late evening meetings	Yes	N/A
For TPE	Optimal mix of requirements	Yes	Yes
For ATD	Implement technical debt together with strategic work (e.g., user story),	Yes	Yes
	Payoff tech-debt as a part of each IR	Yes	Yes
	Categorize the requirements	Yes	Yes
	Encourage teams to bring-in tech-debt	Yes	
	Establish dedicated technical debt team	Yes	
	Plan 1IR to pay off tech-debt when in-between the release	Yes	
For QoR	Dedicated capacity (i.e., 10% to 20%) to ensure the right amount of detail is present in requirements (e.g., user story)	No	Yes
	AC and DOD need to support the lean principles	Yes	
For BDM	Optimal mix of requirements	Yes	Yes
For DBV	Business education required for the decision-making team(s)	Yes	
For QRs	Unambiguously specified the QRs	Yes	Yes
	Inception phase to identify QRs upfront	Yes	
	Discovery phase to identify QRs (specifically performance requirements) upfront	Yes	Yes
	Microservices style architecture	Yes	Yes
	In-Progress: security/to ensure zero attacks via API gateway	Yes	
	Service level objectives dashboard	Yes	
	Automate QRs	Yes	Yes
	Capacity allotment to work on QRs	Yes	
For DEE	No particular solution strategies	Yes	Yes
For IRA	Discovery phase	Yes	Yes
For LRC	Escalation matrix Teams' catch-up	Yes	

Challenge category	Potential strategies	MEL organization	AKL organization
For DaD	Identify dependencies upfront (e.g., discovery phase, Inception phase), CTs to visualize dependencies, Continuous communication and collaboration Microservices architecture	Yes	Yes
	Domain specific requirements	Yes	In-progress

7.8 Salient categories emerged as an outcome of cross-case analysis

The following salient categories guided me in evolving the initial conceptual framework of requirements prioritization in SADSD, that was developed by analysing the individual case organizations' data as well as from literature support (discussed in Chapter 4: Section 4.4.3).

Domain level: As discussed in the introduction section of this chapter, the middle decision-making level, which is termed segment level in MEL case organization, and tribe level in AKL organization, is a technical product area where teams are generalizing specialists who work autonomously end-to-end to solve business problems'. These business problems, termed feature in MEL and epics in AKL typically span portfolio and the middle levels (termed domain level in my new adopted level) in terms of decision making on their priority. Therefore, due to the commonalities in terms of the process adopted by the organizations on the decision-making of their priorities, I have termed them as domain level. A more elaborated description of the process is provided in the Section 7.4 of this chapter.

High-level requirements: In the studied cases, requirements that demand significant effort and/or investment, require a formal decision-making process. This draws initial decision-making at the first part of the domain level, and formal prioritization and communication at the portfolio level. In the studied cases, these requirements are termed as features, in MEL and epics in AKL (discussed in Section 7.4.2, 7.4.1). Therefore, due to the commonalities in terms of processes adopted, I have termed them as high-level requirements (HLRs). The detailed discussion on this category is provided in a discussion in Chapter 8.

Intermediate-level requirements: In the studied case organizations, HLRs are decomposed into further requirements in the second part of the domain level, typically via a *team of teams event* that can be implemented via a short development cycle, like,

two weeks. These requirements are generally termed as user stories in the MEL organization, and features in AKL. Therefore, due to the commonalities in terms of process adopted by the studied case organizations on the decision-making of requirements that are decomposed from the HLRs, I have termed them as intermediate-level requirements (ILRs). The detailed discussion on the ILRs is provided in a discussion in Chapter 8.

Low-level requirements: In the studied case organizations, ILRs are decomposed into further requirements that can provide detailed information to the engineering/delivery team(s) for implementation. These requirements are generally termed as user stories in both studied case organizations (discussed in Section 7.4.4). Therefore, due to the commonalities in terms of processes adopted by the organizations on the decision-making of priority of requirements at the team level, I have termed them as low-level requirements (LLRs). The detailed discussion on the LLRs is provided in a discussion Chapter 8.

Inter-iteration prioritization, Intra-iteration prioritization: The cross-case analysis guided me in linking the requirements classification and organization, decision-making levels (i.e., portfolio level, domain level, team level), and decision-making events, which yield emerging categories called "inter-iteration prioritization" and "intra-iteration" prioritization.

Inter-iteration prioritizations: For example, bigroom planning which is performed at the portfolio level where organization-wide requirements priority decisions, and HLRs that require significant effort and/or investment, are formally decided. They are then communicated to the engineering teams for implementation. This yields an emerging category - *inter-iteration prioritization*.

Intra-iteration prioritizations: For example, development iteration planning or sprint planning, is performed at the team level where team specific requirements are decided and prioritized that can be implemented in short development cycles. They yield an emerging category called *intra-iteration prioritization*.

A detailed discussion on the inter-iteration prioritization and intra-iteration prioritization is provided in a discussion Chapter 8.

Integrated agile RE practices: Classic RE activities as suggested by Sommerville (2009) that consist of requirements discovery and understanding, requirements prioritization and negotiation, requirements artefacts that are initially adopted to structure

the findings of this research study still exist, but typically in the form of integrated agile RE. According to the findings of this research study, these activities are tightly integrated, which further guided me in evolving the initial conceptual framework of requirements prioritization in SADSD (discussed in Chapter 4: Section 4.4.3). The detailed discussion on this category is provided in discussion Chapter 8.

Boundary spanning mechanisms: the cross-case analysis guided me in linking the decision-making events, decision-making teams, requirements artefacts, and decision-making levels which yields the emerging the category of boundary spanning mechanisms (in the form of personnel involved, requirements artefacts, decision-making events). The detailed discussion on this category is provided in discussion Chapter 8.

7.9 Summary

The findings of the studied case organizations are compared and contrasted in this chapter. The existing salient categories are modified, and new salient categories are defined, wherever they are needed as an outcome of cross-case analysis.

Chapter 8 describes the conceptual framework that was built to synthesize and aggregate the findings of this research study.

8 Discussion

Introduction: The requirements prioritization process was analysed to identify and understand the life cycle of requirements prioritization, from an idea to operational in two scaled agile distributed organizations. The synthesized model was developed that depicts the distilled version of core and peripheral requirements prioritization activities, as identified across the studied case organizations. The conceptual framework draws upon mutually supporting multiple data sources and methods which are: interview data, informal discussion with participants, electronic requirements artefacts, close observation of decision-making event(s) employed by the studied cases, in order to make decisions on the requirements priorities. This closeness to practices, enabled me to identify the divergence between practitioners' actions and both the prescribed models in the organizations as well as classical decision-making activities to prioritize the requirements found in the literature.

Analysis was done in multiple phases that guided me to develop a synthesized but abstracted version of the requirements prioritization process from the studied organizations, MEL case organization, and AKL case organization. Following are the three core phases of analysis: individual case analysis - inductive analysis, literature support - deductive analysis, cross-case analysis - inductive and deductive analysis.

In the first phase of analysis, salient categories of the requirements prioritization process emerged from the individual analysis of studied case organizations (MEL case analysis- Chapter 5, AKL case analysis- Chapter 6). As an outcome of the second phase of analysis (discussed in Chapter 4: Section 4.4.3), findings that emerged from the individual case analysis were structured via literature support and guided me to develop an initial conceptual framework of requirements prioritization in SADSD. In the third phase of analysis, i.e., cross-case analysis, new salient categories emerged and some of the existing categories were re-defined. They include segment level in MEL organization, tribe level in AKL organization, that further guided me in evolving the initial conceptual framework of requirements prioritization in SADSD. The framework was developed by analysing the individual case organizations' data as well as via literature support. In addition to this, feedback from key decision-makers of studied case organizations was sought to validate the conceptual framework. An overview of salient categories is provided in Chapter 7: Section 7.8.

Section 8.1 provides an overview of the conceptual framework of requirements prioritization in SADSD, developed by analysing the case organizations. Life cycle activities of the subject, in distributed settings, are discussed in detail in Section 8.2, Section 8.3, and Section 8.4. Next in Section 8.5, CTs and their role in requirements prioritization is discussed. In Section 8.6, challenges that occur while making decisions on those priorities, the settings and potential strategies are discussed.

8.1 Conceptual Framework overview

Figure 8.1 depicts the framework derived for requirements prioritisation in SADSD. Pragmatic notations are employed to draw the conceptual framework instead of using a formal notation such as a process diagram, or data flow diagram (DFD). The main reason to employ pragmatic notation is that I wanted to provide a pictorial representation of the high-level overview of prioritization process that could be difficult to achieve via formal pragmatic notation. However, the pragmatic notations are initially adapted from the DFD. The research study findings indicate that as shown in the conceptual framework (Figure 8.1), there are three decision making levels, including portfolio level, domain level, team level, where primarily the priorities of requirements are decided in scaled agile organizations. Portfolio level and domain level are the levels where typically the inter-iteration prioritization as well as intra-iteration prioritization are performed. On the other side, typically intra-iteration prioritization is performed at the team level.

- Portfolio level (inter-iteration prioritization)
- Domain level (inter-iteration prioritization, intra-iteration prioritization)
- Team level (intra-iteration prioritization)

The occurrence of these decision-making levels varies with requirements classification. Typically, the requirements that require significant investment and effort, such as two to three months of work, are represented as high-level requirements (HLRs) in this research study requires formal decision-making process (i.e., portfolio level, domain level, team level). Their initial decision-making is generally performed in the first part of domain level and then moved to the portfolio level for final confirmation on their priority. At the 2nd part of domain level, the HLRs are then decomposed into further requirements represented as intermediate level requirements (ILRs) and typically form inter-iteration prioritization across teams (project specific). The next level is the team level where ILRs are decomposed into low-level requirements if needed that provide detailed requirements required information to the delivery teams and typically form intra-iteration prioritization.

Boundary spanning is an important component of requirements prioritization in scaled agile development, that emerged from the case analysis. Boundary spanning is important for requirements prioritization, as boundary spanning yields knowledge acquisition, negotiation, consensus building, and conflict resolution which are the key activities typically needed while making decision on the priority of requirements (Cross et al., 2013; Jain et al., 2014). Boundary spanning is the act of bringing together two or more groups of people who are typically separated by location, that is, locally distributed and/or globally distributed., or separated by hierarchy, or a function (Cross et al., 2013; Jain et al., 2014). Three categories of boundary spanning mechanisms are identified in this research study, that includes *boundary spanning event(s), boundary spanning requirement artefact(s), boundary spanner role(s)*.

According to the findings of this research study, boundary spanning mechanisms primarily occur when requirements prioritization decisions, are made at the inter-iteration levels. However, boundary spanning mechanisms partially occur during intra-iteration prioritization as well. According to this research study findings, inter-iteration prioritization decisions are typically formed at the portfolio and domain levels. Scope of the boundary spanning mechanisms that occur to perform inter-iteration prioritization, vary at both levels - portfolio and domain levels. A notable difference is that, for organization- wide requirements prioritization decisions are formed via boundary spanning at the portfolio level. On the other hand, requirements prioritization decisions across teams (project specific), are performed at the domain level.

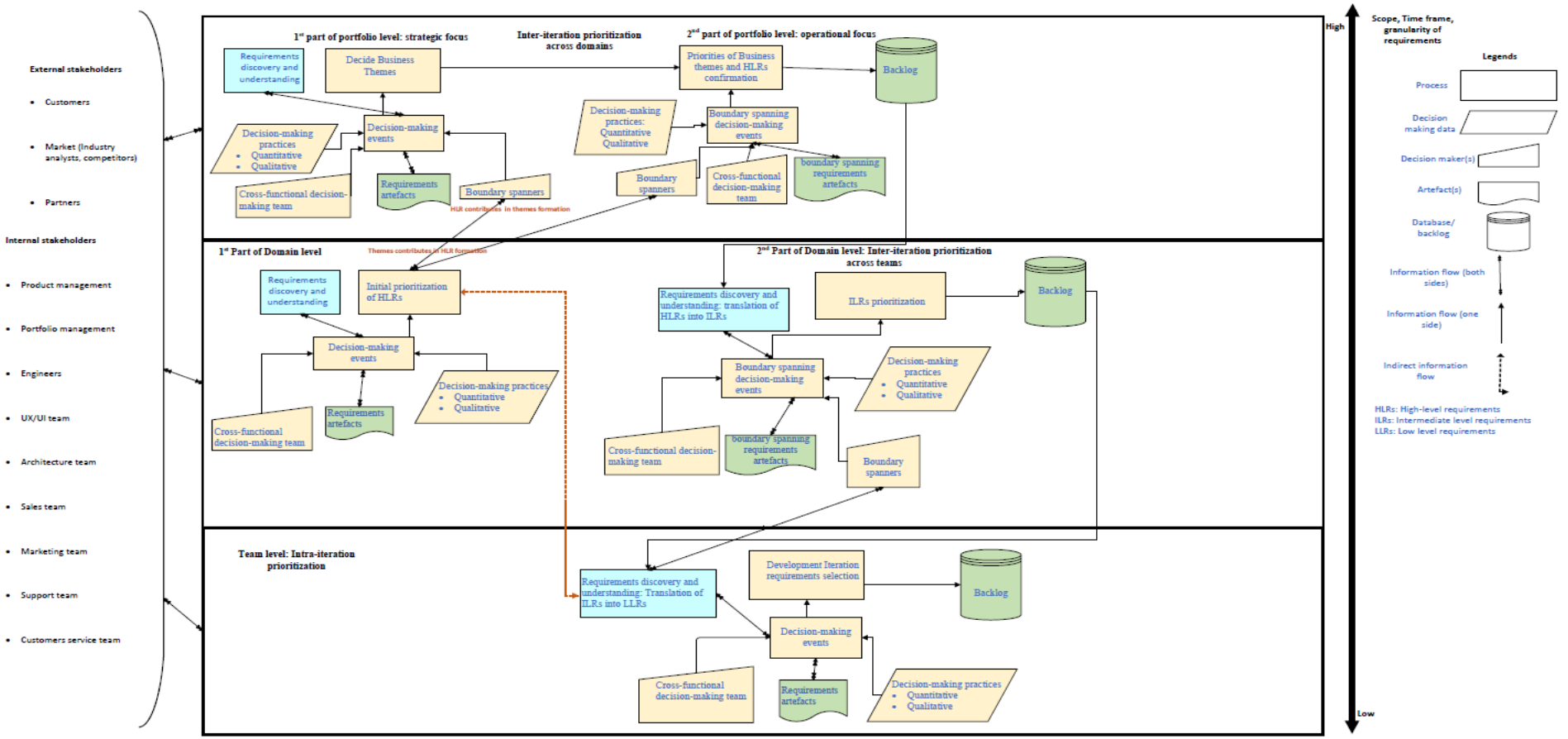
Prioritization of requirements at each of these decision-making levels, portfolio, domain, and team levels, typically consist of requirements discovery and understanding, decision making events, decision making practices, decision making team, requirements artefacts (as shown in Figure 8.1). These activities are influenced by (Sommerville (2009)), classic requirements elicitation and analysis. They include: (i) requirements discovery and understanding (shown by blue colour in the model), (ii) requirements prioritization and negotiation (shown by yellow colour in the model), (iii) requirements artefacts (shown by green colour in the model), (iv) classification and organization of requirements (typically present in the form of decision-making practices shown by yellow colour). According to the research study findings, these activities are not clearly separated and instead, they are highly integrated and occur at various levels with varying details, in order to make

decisions on the priority of requirements. This finding confirms the findings of Sommerville (2005), who reported integrated RE in large scale agile software development.

- Requirements discovery and understanding: Potential requirements are discovered, understood, and classified at each of the decision-making levels (i.e., portfolio level, domain level, team level).
- Decision making events: At each level, decision making events that are typically team specific decision-making events or boundary spanning decision-making events, occur to make decisions on priority requirements.
- Decision making team: At each level, the decision-making team, which is typically a cross-functional decision-making team, works collaboratively and continuously to make decisions on priority requirements. In addition to this, boundary spanners may also be a part of the decision-making team wherever needed.
- Decision making practices: Practices to make decisions on priority requirements, are a combination of quantitative methods or objective weights, as well as qualitative methods or subjective weights, which are typically consumed during decision making events. Scope, timeframe, and hierarchical structure of requirements generally influence the decision-making practices needed to make priority decisions at each of the scaling agile decision-making levels.
- Requirements artefacts: Requirements are generally documented in the form of locally relevant requirement artefacts and/or boundary spanning requirements artefacts, which are typically consumed and/or produced during decision making events to make decisions on priority requirements. Scope, timeframe, and requirements classification, typically influence the type of requirements artefacts produced and/or consumed across scaling agile decision-making levels.

Figure 8.1

Conceptual framework of requirements prioritization in SADSD



8.2 Portfolio level

It is a portfolio level, which is typically the highest decision-making level where primarily the business goals, (represented as *strategic themes* in this research study), that connect with the organization's overall business strategy are decided and prioritized. In addition to this, requirements that demand significant investment and effort, such as two to three months of work, are represented as high-level requirements (HLRs) in this research study. The HLRs typically emerge from each of the domains towards achieving strategic themes, and are formally signed off, communicated, and allocated to the engineering for implementation at the portfolio level.

According to the research study findings, the above two objectives are achieved via *inter-iteration prioritization* which is typically performed through *boundary spanning mechanisms*. Three categories of boundary spanning mechanisms are identified during inter-iteration prioritization, that includes *boundary spanning events, boundary spanning requirement artefacts, boundary spanner(s) role.*

The following are the mechanisms that are performed at the portfolio level to make decisions on the priority of requirements.

8.2.1 Requirements discovery and understanding

According to the findings of this research study, an open innovation approach (Damian et al., 2021) is employed to collate potential requirements from all possible stakeholders. At the portfolio level, portfolio management predominantly collate the potential requirements from all possible stakeholders (internal stakeholders as well as external stakeholders) that are needed to accomplish strategic themes and/or contribute to the themes' formation. Some of the practices that are employed to collate potential requirements include competitors' analysis, track market trends, industry analysts (Damian et al., 2021). However, practices that are employed to collate potential requirements seems to be case and company specific. A detailed discussion of this is provided in Chapter 7: Section 7.2. Possible reasons for this variance, includes the type of business an organization is performing, scaling agile framework and their local adaptation.

8.2.2 Decision making team

Portfolio management, which is a cross-functional decision-making team, consists of most senior enterprise business and technology stakeholders, as well as business owners (Ambler & Lines, 2012; Leffingwell, 2011), called domain leaders in this research study. Domain leaders typically bring business and technology needs that are emerging from their respective domains. Designations the decision makers hold, may vary in organizations and one person may hold multiple responsibilities (Ambler & Lines, 2012; Leffingwell, 2011). A typical reason for this variance is that emerging from this research study, are organizational needs, adoption of scaling agile frameworks and their local adaptations.

In MEL organization, portfolio management consists of SVP engineering, SVP product, SVP architecture, SVP UX, domain leaders where SVP architecture brings strategic technical direction and SVP engineering provides a resourcing perspective. SVP product provides overall business perspectives, SVP UX brings the customer voice, domain leaders provide emerging business and technical needs, within their domain.

In the AKL organization, portfolio management which is called Governance forum consists of CCO bringing the customer voice, CDO bringing technical needs, resources as well as business perspectives, domain leaders providing emerging businesses and technical needs within their domain.

According to this research study findings, *portfolio management* is typically a *boundary spanning decision making team* by location, function, and hierarchy.

As for location, members of portfolio management are distributed in the studied case organizations. In MEL, decision makers are globally distributed, whereas in AKL organization they are nationally distributed.

As for function, each of the members of portfolio management team have their own specific responsibilities during decision-making. For example, in AKL organization, the CCO brings customer voice, CDO brings technical perspectives, business and resource perspectives during decision-making on the priority of requirements.

As for hierarchy, typically the *leaders of domains* play a key role in facilitating the exchange of information, such as clarity of requirements between portfolio and domain levels. At the portfolio level, leaders of domain(s) typically provide all the facts and data

that are needed to make decisions on the potential HLRs and/or themes that are emerging within their domain, and progress of HLRs. At the domain level also, leaders of the domain typically provide clarity of requirements, and communicate any change in priority decisions, on the behalf of portfolio management.

These findings coincide with Eckstein (2014), Grewal and Maurer (2007), Hobbs and Petit (2017) who posit that a hierarchical structure of decision-making team is needed in large scale agile projects, in order to make decisions on the priority of requirements.

8.2.3 Boundary spanning requirement artefact

Requirement artefacts that are typically consumed by various levels are termed as boundary spanning requirement artefacts (Jain et al., 2014). At the portfolio level, a *strategic theme(s)* is the boundary spanning requirement artefact, an electronic artefact configured via CTs, which is typically defined and prioritized at the portfolio level that has influence at all the levels: domain level, and team level, in terms of decision making on requirements priorities.

Another boundary spanning requirement artefact is the business-case, an electronic artefact configured via CTs (discussed in Section 8.5). This is consumed at the portfolio level and typically built at the domain level, that contains all the facts and data such as benefits, costs, and effort that are typically needed to make decisions on the potential HLRs, as well as provide a basis to the engineering team(s) to decompose HLRs into further requirements, termed as ILRs and LLRs in this research study.

Portfolio backlog is another boundary spanning requirement artefact, an electronic artefact configured via CTs (discussed in Section 8.5), which is produced at the portfolio level, that contains strategic themes and the priority list of HLRs that are typically allocated to each of the domains for implementation, based on their relevancy.

8.2.4 Boundary spanning events

Portfolio Sync-up is typically organized via CTs (discussed in Section 8.5). This is for check-ins where typically the portfolio management team members come together to get visibility on the progress of HLRs towards achieving strategic themes, removing impediments, addressing dependencies and any change in the priorities of requirements. This event has a more operational focus than a strategic portfolio review.

Strategic portfolio sync-up event is organized via CTs (discussed in Section 8.5) and is typically focused on achieving and advancing the strategic vision. This event is organized on a quarterly cadence, where portfolio management evaluates, defines and updates the strategic themes that connect with the organization's overall business strategy. This event typically occurs before organizing the "*bigroom planning*" event.

Bigroom planning planning event is organized via CTs (discussed in Section 8.5): this is the event where whole organization is generally invited, but product and engineering are the key participants of the event. This event is usually a quarterly cadence event where strategic themes, HLRs for the upcoming release IR of 3 months, are formally communicated, shared understanding of priority of requirements among product and engineering is achieved, visualizing and negotiating the priorities and dependencies across the domains, are identified and managed.

To sum up, according to the findings of this research study, the boundary spanning event is tailored as per each organizations' needs. Frequency and duration of boundary spanning event(s) may vary between organizations. For example, a *bigroom planning* event, which is a week-long event in MEL organization, in AKL organization, is only a one-day event. This event is typically a SAFe practice (Dalton, 2018; Mutschler et al., 2020) often held for two-days, but studied case organizations modified this practice as per their needs.

8.2.5 Decision making practices (decision-making factors and techniques)

For strategic themes: Strategic themes are very high business level requirements that are generally unchanged for up to one year or can go above a year (Ambler & Lines, 2012). The study findings indicated that business values, which are native to Agile and is mentioned by Agile authors (Berander & Andrews, 2005; Daneva et al., 2013; Ramesh et al., 2010), is a main prioritization driver in terms of deciding strategic themes. A strategic theme(s) provides the biggest opportunity to the business and customers and is typically selected as a high priority theme.

For HLRs: As emerged from the study findings, HLRs typically span boundary at portfolio and domain levels, in terms of decision making on their priority. According to the study findings, usually high-level evaluation is performed in terms of decision making on the priorities of HLRs by the portfolio management, at the portfolio level. Portfolio management may adjust the priority score of HLRs wherever needed, but ultimate recommendation typically comes from the delivery team(s).

In summary, the study findings indicated that in both studied case organizations, processes that are employed at the portfolio level to make decision on the priority of requirements such as, strategic themes and HLRs, as well as the process specific practices, have similarities. However, process specific practices seem to be case and company specific, to some extent (shown in Table 8.1,Table 8.2).

Table 8.1

Process for decision-making at the portfolio level

Process to make decisions on the priority of requirements (i.e., strategic themes, HLRs) in the studied case organizations (MEL organization and AKL organization)
Two main objectives at the portfolio level in terms of decision making: •Business goals that connect with organization's overall business strategy are decided and prioritized •HLRs towards achieving business goals are formally signed off, communicated, and allocated to the engineering for implementation
Inter-iteration prioritization across domains
Portfolio management a cross-functional decision-making team
Boundary spanning mechanisms occurrence at the portfolio level

Table 8.2: Process specific practices at the portfolio level

Process specific practices		**MEL Organization**	**AKL Organization**
Terminology		Business goals represented as strategic themes HLRs terminology discussed in section 8.3.	Business goals represented as strategic themes
Time frame		generally unchanged for up to one year or can go above year HLRs timeframe discussed in section 8.3.	generally unchanged for up to one year or can go above year
Scope		Strategic themes are very high business level requirements that are generally unchanged for up to one year or can go above year. HLRs scope discussed in section 8.3.	Strategic themes are very high business level requirements that are generally unchanged for up to one year or can go above year
Prioritization practices	Main prioritization driver	BV	BV
	Learning experience	Applied during prioritization	Applied during prioritization
	Hierarchical structure of requirements	Applied during prioritization	Applied during prioritization
Decision making events on regular cadence (Organized via CTs, face to face)		quarterly, monthly, ad-hoc	quarterly, fortnightly, ad-hoc
Cross-functional decision-making team		Called Portfolio management	Called Governance forum
Requirements artefacts configured via CTs		AHA, Spreadsheet, Confluence, PowerPoint	ADO, Confluence, PowerPoint

8.3 Domain level

Products are classified into domains which is typically a technical product area (e.g., Residential tribe) where teams are generalizing specialist's teams in a particular domain. According to the findings of this research study, the role of a domain is twofold. In the first part of a domain, intra-iteration prioritization is performed to support inter-iteration prioritization at the portfolio level, whereas in the second part of the domain, inter-iteration prioritization is performed.

> *so at the tribe level, eg. Residential tribe we have meeting fortnightly which is where basically leaders across resi tribe. [AKL_P2]*

8.3.1 Intra-iteration prioritization: first part of Domain level

Potential requirements that require significant effort and/or investment, such as two-three months of work, are termed HLRs and typically need a formal process in terms of decision making on their priorities. This finding coincides with (Leffingwell, 2007), who posits the need for a structured decision-making process for the requirements that need a large effort.

According to the findings of the research study, HLRs typically span a boundary in terms of decision making on their priority. In the first part of a domain, potential HLRs emerge from each of the domains towards achieving strategic themes and/or contributes to the formation of strategic themes. These are initially defined, prioritized and submitted to the portfolio level via domain(s) leaders for formal sign-off. Domain leaders are the actors who own the business and technical need emerging from their respective domain which seems to be an adaption of BOs role, as defined in SAFe framework (Leffingwell, 2011).

The following are the mechanisms that emerged from the research study in order to make decisions on the priority of potential HLRs.

8.3.1.1 Requirements discovery and understanding

According to the research study findings, collating the potential requirements is a continuous exploration process (Damian et al., 2021). The potential HLRs that include new functionality, enhancement in the existing system, architectural work, may locally originate within the domain and/or be served by the portfolio level and team level. According to the findings, an open innovation approach (Damian et al., 2021) is employed to collate potential requirements from all possible stakeholders, internal and external. At the domain level, product management predominantly collate the potential requirements from all possible stakeholders that are needed to accomplish strategic themes and/or contribute to the themes' formation. However, the practices that are employed to collate potential HLRs from all possible stakeholders, seem to be case and company specific (described in Chapter 7). Potential reasons include type of business an organization is performing, adoption of scaling agile framework and their local adaptation.

8.3.1.2 Boundary spanning requirement artefact(s)

According to the research study findings, potential HLRs are specified in the form of a business-case. An electronic business-case template configured via CTs (discussed in Section 8.5), is typically built to collate all the required facts and data. This includes a high-level description of requirements, benefits, effort, risks required to make decisions

on the candidate HLRs (Ambler & Lines, 2012; Heikkilä et al., 2017; Leffingwell, 2011). A Business-case typically acts as a boundary spanning requirement artefact, which is produced at the first part of a domain level for initial decision making on the potential HLR as well as being consumed at the portfolio level by the portfolio management for formal sign-off. In addition to this, a business-case is also consumed at the second part of the domain level as well as at the team level for decomposing the HLRs into further requirements, termed as ILRs and LLRs in this research study, and sets priorities on them.

8.3.1.3 Boundary spanning decision making team

According to the research study findings, middle management which is a cross-functional decision-making team works collaboratively to define and evaluate the priority score of potential HLRs. Designations the decision makers' hold may vary in organizations and one person may hold multiple responsibilities (Daneva et al., 2013; Leffingwell, 2011). Typical reasons of this variance, includes organization needs, adoption of scaling agile frameworks and their local adaptations.

In MEL organization, a decision-making team who makes decisions at the first part of a domain level, typically consists of PM, who brings business perspectives, director of UX/UI bringing the customers' voice wherever needed, program architecture owner who brings solution perspectives, and the director of engineering ensures delivery and resource perspectives. They all work collaboratively to make decisions on the priorities of potential HLRs.

In the AKL organization, the decision-making team typically consists of senior PO, who brings business perspectives, senior UX/UI who brings the customers voice wherever needed, solution architect who brings solution perspectives, and tribe lead who brings overall resource perspectives and ensures delivery.

According to the research study findings, middle management is typically a *boundary spanning decision making team* by location, function, and hierarchy.

As for location, there are distributed members of the decision-making team. In MEL, decision makers are typically globally distributed, while AKL decision-makers are locally distributed.

As for function, each of the members of decision-making team have their own specific responsibilities during decision-making. For example, in MEL organization, the PM brings business perspectives, program architecture owner brings technical perspectives.

As for hierarchy, there are designated leaders of the domains. As emerged from the research study, at the domain level, typically the PMs/senior POs facilitate the exchange of information such as potential HLRs and/or change in requirements from the delivery team side, which is then communicated and represented via leaders of domain(s) to the portfolio level and vice versa. These findings coincide with (Eckstein, 2014; Grewal & Maurer, 2007; Hobbs & Petit, 2017) who posit that a hierarchical structure of a decision-making team is needed in large scale agile projects in order to make decisions on the priority of requirements.

8.3.1.4 Decision making events

Decision making events are typically organized via CTs, and/or a face-to-face event (discussed in Section 8.5). The events that are organized to make an initial decision on the priorities of HLRs generally occur on regular cadences (Heikkilä et al., 2017; Ramesh et al., 2010). Decision-making events are typically held on a fortnightly basis to plan current HLRs, future HLRs, and take ideas through discovery. Weekly sync-up calls are held to check progress of already scheduled HLRs, any changes in priorities decided priority, resolve dependencies if occur any. In addition to the planned events, ad-hoc calls are made, as and when necessary.

8.3.1.5 Decision making practices (decision-making factors and techniques)

According to the study findings, techniques that are consumed to evaluate the priority score of potential HLRs, are a combination of *quantitative methods* (i.e., objective weights), as well as *qualitative* methods (i.e., human judgement) (Berander & Andrews, 2005; Lehtola et al., 2004). In quantitative methods, typically numerical values are assigned to the different aspects of potential HLRs. I have used the aspects and factors interchangeably that are typically considered to evaluate the priority score of potential HLRs.

Decision-making factors: Common factors that are employed which are commonly reported in the literature (Alenljung & Persson, 2008; Berander & Andrews, 2005; Daneva et al., 2013), include business value, risks, urgency, opportunity, effort, NPV, and strategic fit. The salient factors are typically evaluated on a numerical scale, 1 to 10 in MEL organization, and 1 to 5 in AKL. This is an automated matrix, configured via CTs (discussed in Section 8.5), where the lowest number represents the lowest value item and the highest number represent the most valuable item. However, interviewees agreed that these aspects could well be subjective to some extent. For example, effort is typically

estimated based on the tacit knowledge of the participants. These findings coincide with Daneva et al. (2013) who also admitted the role of subjectivity, while evaluating the priority score of requirements in large scale agile projects.

> *but the idea is that we have a weighting system and then it's managed by people, not just by math. [MEL_P4]*

Decision-making techniques: Following are the quantitative and qualitative methods that have emerged from the research study findings, which are commonly reported in the literature as well (Daneva et al., 2013; Herrmann & Daneva, 2008) to evaluate the priority score of high-level requirements.

Weighted shortest job first approach (WSJF): WSJF which is typically a SAFe practice (Leffingwell, 2011) employed by case organizations and tailored as per their need to evaluate the priority score of potential HLRs.

In the original WSJF method, typically the Fibonacci series is employed on factors that are considered to evaluate the priority score of a potential HLR (Leffingwell, 2011). In MEL case numerical scale 1 to 10 is used. On the other hand, in AKL organization, WSJF is typically employed qualitatively instead of employing its proper matrix (i.e., WSJF =User business value +Time criticality + (Risk reduction + Opportunity)/Job size)[16].

Hierarchical structure of requirements: It is typically used as a relative value of a requirement along with the other requirements. and enables the prioritization of requirements against each other that belong to the same or different category and/or same abstraction level (Ktata & Lévesque, 2009)

Learning experience: Adjustments to the priorities of potential HLR are based on the participants' knowledge and/or experience. The priority of potential HLR can be adjusted, especially when more than one item holds the same priority score and/or has the same importance (Daneva et al., 2013; Herrmann & Daneva, 2008; Zowghi & Damian, 2003). In such cases, generally the benefits especially return on investment, and customer impact of one over another are compared, which is typically done based on the participants' knowledge and/or experience.

To sum up, according to the findings, the processes that are adopted to make decision on the priority of HLRs, seem to be similar in both studied case organizations (summarized

[16] https://www.scaledagileframework.com/wsjf/

in Table 8.3). But the process specific practices to make decisions on the HLRs, seems to be case and company specific (summarized in Table 8.4).

- *Timeframe and scope (granularity of requirements):* Timeframe and scope of HLRs vary in the studied case organizations. In MEL organization, HLRs typically fit in a release IR of 3 months. On the other hand, HLRs are broader in scope and timeframe, typically taking multiple releases in the AKL organization than in MEL. In addition to this, HLRs typically span across domains from the implementation point of view in AKL organizations that results in excessive dependencies across domains, whereas the MEL organization manages this by limiting the scope of HLRs to be domain specific.

Table 8.3

Process for decision-making on HLRs at the domain level

Process to make decision on the HLRs in the studied case organization (MEL organization and AKL organization)
Definition of HLR: potential requirement that require significant effort and investment termed as HLRs
Initial decision making on HLRs in the 1st part of domain level
Formal sign-off on HLRs at the portfolio level
Intra iteration prioritization to support inter-iteration prioritization at the portfolio level
Partial occurrence of boundary spanning mechanisms during intra iteration prioritization

- *Terminology:* Terminology that is employed to denote HLRs, seems to be case and company specific. A requirements abstraction model is employed, which is similar to the Leffingwell (2011) model (epic→feature→user story) in the studied case organizations, that have influence on the nomenclature of requirements. However, both organizations tailored requirements by a hierarchy model as per their need. In MEL, feature(s) are typically termed as HLRs, whereas epic(s) typically act as HLRs in AKL. A possible reason for this variance that has emerged from our findings, is the ***CTs adoption*** that is employed to manage the requirements across the scaling agile levels in studied case organizations.

Table 8.4

Process specific practices at the portfolio level

Process specific practices		**MEL Organization**	**AKL Organization**
Terminology for HLRs		Features	Epics
Time frame		Fit in a release iteration (i.e., 3 months)	Take multiple release iterations (more than three months)
Scope		Broad in scope typically require more than one delivery team for implementation, Typically, domain specific	Much broader than MEL organization require more than one delivery team Typically, span multiple domains
Prioritization practices	Numerical scale	Numerical scale 1 to 10	Numerical scale 1 to 5
	Prioritization factors	BV, Risk, Urgency, opportunity, effort	Strategic fit, NPV
	WSJF (SAFe practice)	Adapted as per their need and applied quantitatively	Adapted as per their need and applied qualitatively
	Automated prioritization matrix configured via CTs	AHA as CTs	ADO as CTs
	Requirements classification	Plays a significant role during prioritization. Percentage based resource allocation on each category of requirements (architectural work, tech-debt work, strategic work, defect correction work)	Plays a significant role during prioritization. No fixed percentage, Typically classify them as mandatory and optional (discretionary work, tech-debt work, compliance work)
	Learning experience	Applied during prioritization	Applied during prioritization
	Hierarchical structure of requirements	Applied during prioritization	Applied during prioritization
Decision making events on regular cadence (Organized via CTs, face to face)		Fortnightly, weekly, ad-hoc	Fortnightly, weekly, ad-hoc
Cross-functional decision-making team		PM, engineering manager, program AO, director of UX/UI	Senior PO, tribe lead, solution architect, senior UX/UI
Requirements artefacts configured via CTs		AHA, Confluence	ADO, Confluence

- *Classification of HLRs:* According to the study findings, all potential requirements, that include architectural work, enhancement work, strategic work, and technical debt work, that require significant effort and/or investment are termed as HLRs. However, requirements classification and the resource spending on each of the categories of HLRs seems to be case and company specific. In MEL, typically quantitative weights, where 30% of resources are assigned to architectural work, and 60% are assigned to strategic work, are assigned to each of the categories of HLRs, whereas in AKL, typically qualitive weights such as, compliance work that is essential, tech-debt work is optional, are assigned to each of the categories. Typical reasons for this variation include ***organization context,*** (type of business an organization is performing).
- Factors and the techniques (i.e., quantitative methods, qualitative methods) through which potential HLRs are scored and evaluated, seem to be very case and company specific. This finding coincides with Lehtola et al. (2004). It seems that the type of business an organization is performing, classification of requirements, learning experience (i.e., tacit knowledge) has an effect on the aspects that are employed to evaluate the priority score of potential HLRs.

8.3.2 Inter-iteration prioritization: second part of domain level

In the second part of a domain level, a priority list of HLRs that resides in the portfolio backlog serves as an input and is allocated to each of the domain(s) for implementation, based on their relevance to the domain. As discussed in Section 8.3.1, HLRs are broader in scope and typically need more than one delivery team for implementation. Therefore, further decomposition of HLRs is typically needed that form inter-iteration prioritization across teams (project specific), in order to implement them via short development cycles (e.g., two weeks) and a single delivery/engineering team. In our framework, requirements that are decomposed from the HLRs are termed as ILRs that can be implemented via single development IR/sprint and single delivery team.

As discussed in Section 8.1, inter-iteration prioritization across teams (project specific) is formed in the second part of domain level which is typically employed via boundary spanning mechanisms according to findings of this research study.

8.3.2.1 Requirements discovery and understanding

The research study findings indicated that activities that are performed in the form of requirements discovery and understanding in the second part of domain level typically

yield decomposition of HLRs into further requirements in order to implement them via short development cycles.

8.3.2.2 Boundary spanning decision making events

Boundary spanning events that are generally organized via CTs and/or via face-to-face in-person occurred at regular cadences: a fortnightly event where HLRs typically decomposed into ILRs and set priorities on them. Weekly sync-up calls, providing visibility on ILRs progress, and checking any change in decided priority, resolving dependencies if any occurred. Ad-hoc call, made as and when needed.

Business-case- a boundary spanning requirements artefact is typically consumed during a decision-making event, in order to decompose HLRs into ILRs and set priorities for them.

8.3.2.3 Decision making team

The decision making team typically consists of *boundary spanners* who are typically differs from functions, hierarchy, location (Jain et al., 2014). Boundary spanners relate to particular roles who typically have responsibility on the adjacent levels to facilitate the exchange of information (e.g., requirements clarity). At the domain level, the PM/senior PO (customer representative who generally own the outcome of a particular HLR) typically occupy a role of PO in delivery teams/engineering teams provide clarity, any change in decided priority, and understanding of HLRs among triad of delivery team(s). On the other side, within the teams' triad typically the PO interface as a boundary spanner from the team level side to facilitate the exchange of information to the PM. Similar findings are reported by Hobbs and Petit (2017), who admitted that in large scale organizations, PMs, also known as "chief product owners," have hierarchical connections with the product owners on the development team. On each development team, the latter is the person the product owners turn to for advice and direction on requirements.

According to the research study findings, a Triad of delivery teams (i.e., teams who will be involved in a particular HLR implementation) typically works collaboratively to decompose HLRs into ILRs and set the priority on them. Designations the decision makers hold, may vary between organizations and one person may hold multiple responsibilities (Daneva et al., 2013). A typical reason for this variance includes organizational need, adoption of scaling agile frameworks and their local adaptations.

MEL organization employed the DAD framework where PO, TL, AO are the primary roles that are present in a delivery team (Ambler & Lines, 2012). As a result, the delivery team triad in MEL organization typically consists of PO who provide business perspectives, TL who provide resource and ensure delivery, and AO provide technical perspectives.

AKL organization adapted *Scrum* practices (Sutherland, 2006) at the team level, where a delivery team triad typically consists of PO with business perspectives, SM resource and ensure delivery, technical BA providing technical perspectives who all work collaboratively to decompose the HLRs into ILRs and set priorities for them.

8.3.2.4 Boundary spanning requirement artefact

Requirements are typically specified in the form of ILRs that act as boundary spanning requirements artefact (an electronic artefact configured via CTs discussed in Section 8.5) and usually consumed to plan and perform intra-iteration prioritization at the team level. According to the study findings, terminology to represent ILRs seems to be case and company specific, as represented by user story in MEL, feature in AKL.

In MEL organization, user story which is typically documented via "*storymap"*, an electronic artefact configured via CTs discussed in Section 8.5, acted as a boundary spanning requirement artefact, that is typically consumed at the team level to provide detailed understanding and clarity of requirements to the delivery teams. By comparison, the AKL organization, feature(s) an electronic artefact configured via CTs discussed in Section 8.5 that typically acts as a boundary spanning artefact.

Another boundary spanning artefact that is built at the domain level is backlog (called *product backlog* scrum practice (Schwaber, 2004) in AKL organization, *workitem list* DAD practice in MEL organization (Ambler & Lines, 2012)). The backlog is configured via CTs discussed in Section 8.5, that contain a priority list of ILRs and any other requirements, such as training, where a portion of backlog is consumed by the delivery team(s) at the team level.

8.3.2.5 Decision making practices (decision-making factors and techniques):

Common factors that emerged from the findings to set the priority on ILRs, include *strategic priority, business value, HLR priority, team ability and capacity, dependencies (typically functional dependencies)*. However, ***business value*** (BV), which is native to Agile according to Agile authors (Berander & Andrews, 2005; Daneva et al., 2013; Inayat

et al., 2015), is the main prioritization driver that yields from the study findings as well. But the evaluation of BV seems to be case and company specific as well as varying with requirement classification.

In AKL organization, BV is typically evaluated via tacit knowledge instead of applying objective weights and any standard qualitative practice (e.g., MoSCoW).

In MEL organization, BV where the impact on customers and urgency are the factors evaluated via quantitative weights using numerical scale 1 to 5. This process evaluates the business value of a particular ILR which is dedicated for strategic work. However, interviewees agreed that these aspects could well be subjective to some extent. Typically, ***MoSCoW*** -must have, should have, could have, would have, is a qualitative prioritization practice (Anand & Dinakaran, 2017; Popli et al., 2014) and is employed to set priorities on ILRs. Whereas BV of an ILR that is dealing with technical-debt work, enhancement work is typically evaluated via severity and scope factor, such as number of customers and/or impact on system functionality.

To sum up, the study findings indicated the process that is employed to make decisions on the ILRs, is universal that is, using inter-iteration prioritization across teams (project specific), in both studied case organizations (depicted in Table 8.5). But the process specific practices (shown in Table 8.6) greatly varied and seem to be case and company specific. Possible reasons that emerged from our findings, includes adoption of CTs, type of business an organization is performing, hierarchical structure of requirements, adoption of scaling agile frameworks and their local adaptation.

Table 8.5

Process for decision-making on ILRs at the domain level

Process to make decisions on the ILRs in the studied case organizations
ILRs: Requirements that are decomposed from HLRs termed as ILRs
Teams' triad as a decision-making team (team of team approach)
Inter-iteration prioritization across teams (project specific)
Boundary spanning mechanisms during inter-iteration prioritization

- *From the CTs adoption point of view:* In MEL organization, in the second part of domain level because of the CTs that are employed for the delivery teams for implementation, HLRs that are termed as features are converted into epics (a reflection of features). On the other hand, the terminology to denote HLRs which

is termed as epic remains the same in the second part of domain level in AKL organization.

- *Scope (granularity of requirements):* Granularity of ILRs seems to be case and company specific. In MEL organization, ILRs that are decomposed from HLRs are typically detailed enough to provide required information to the delivery team for implementation. However, further breakdown of the ILRs can happen at the team level if needed. But in AKL organization, ILRs are broader in scope and typically require further decomposition at the team level.
- *From hierarchical structure of requirements point of view*: according to our findings, scope of requirements of upper level, typically influence the scope and/or further decomposition of requirements at the subsequent level. For example, in MEL organization, ILRs are detailed enough that they can be implemented by a single member of a delivery team at the team level, whereas in AKL organization, ILRs are broader in scope and require further decomposition at the team level.
- *Terminology:* According to our findings, terminology to denote ILRs seems to be case and company specific. Requirements abstraction model and the CTs which are adapted to manage the requirements, typically influence the terminology of requirements. In MEL organization, ILRs are represented as user stories, whereas in AKL organization, ILRs are represented as features.
- Practices that are consumed to evaluate the priority score of ILRs are typically a combination of *quantitative methods* (weighting system) as well as *qualitative* or human judgement (Anand & Dinakaran, 2017; Berander & Andrews, 2005) which seems to be case and company specific as well as varying with requirements classification (e.g., strategic work, technical debt work, and enhancement work).

The process specific practices for ILRs are depicted in Table 8.6.

Table 8.6: Process specific practices for ILRs

Process specific practices		**MEL Organization**	**AKL Organization**
Terminology for ILRs		User stories	Features
Time frame		Fit in a development IR (i.e., 2 weeks) and can be implemented via an individual engineer of delivery team	Fit in a development IR (i.e., 2 weeks) and typically require whole delivery team
Scope		Generally detailed enough to provide required information to the delivery teams for implementation. But can be decomposed further at the team level if needed.	broad in scope generally require further decomposition at team level for implementation.
Prioritization practices	Numerical scale	Numerical scale 1 to 5	-
	Prioritization factors	strategic priority, BV, HLR priority, team ability and capacity, dependencies (typically functional dependencies	strategic priority, BV, HLR priority, team ability and capacity, dependencies (typically functional dependencies
	Business value a main prioritization driver	Numerical scale 1 to 5	-
		Scoring system	-
	Qualitative practice	MoSCoW	Based on tacit knowledge
	Requirements classification	Plays a significant role during prioritization. factors consideration for BV evaluation varies with requirements classification	Plays a significant role during prioritization. factors consideration for BV evaluation varies with requirements classification
	Learning experience	Applied during prioritization	Applied during prioritization
	Hierarchical structure of requirements	Applied during prioritization	Applied during prioritization
Decision making events on regular cadence (Organized via CTs, face to face)		Fortnightly, weekly, ad-hoc	Fortnightly, weekly, ad-hoc
Cross-functional decision-making team		POs, TLs, AOs, UX/UI wherever needed	POs, BAs, SMs, UX/UI wherever needed
Requirements artefacts configured via CTs		Jira, Confluence	ADO, Confluence

8.4 Intra-iteration prioritization: Team level

At the team level, delivery team(s) are flexible to use any of the Agile methods such as Scrum, or Kanban, for implementation. However, *Scrum* is the dominant Agile method that is employed by the delivery team for implementation (Hobbs & Petit, 2017; Kasauli et al., 2017). This is the level where requirements are typically represented as low level requirements (LLRs) and are translated into actual code.

8.4.1 Requirements discovery and understanding

The study findings indicated that activities that are performed in the form of requirements discovery and understanding at the team level, are primarily performed to decompose ILRs into LLRs and create shared understanding to the delivery teams before starting the implementation. LLRs are generally detailed enough to provide in-depth information on requirements (e.g., requirements clarity) to the delivery teams for implementation. LLRs are typically represented as user stories in the studied case organizations, and they can be implemented via short development cycle (e.g., two weeks). The main guiding boundary requirements artefacts that emerged from the study findings, which are consumed by the delivery teams to discover and understand the requirements include ***HLRs description*** (i.e., business-case) and ***ILRs description*** (i.e., feature description in AKL organization, storyboard in MEL organization).

The extent of front-end planning of development IRs/sprints in the studied organizations varies greatly from minimal to large amount of effort. This finding is similar to Hobbs and Petit (2017), who also acknowledge the variance in front end planning of sprints. According to the study findings, possible reasons for this variance, is the complexity, the scope and timeframes of level up requirements.

In MEL organization, the ***inception phase*** typically takes one IR, a DAD technique (Ambler & Lines, 2012) organized before starting the implementation of new requirements. During the *inception phase*, the delivery team (PO, AO, TL, engineers) works collaboratively to discover and understand all possible information such as breakdown of a user story into further user stories if needed, technical investigation in the form of spikes before starting the implementation.

In AKL organization, ***feature kick off*** is organized to discover and understand all possible information needed by delivery team, for the implementation. Feature kick-off is similar to inception phase technique but varies in some aspects from MEL organization. For

instance, a feature kick-off, is typically organized in two sessions, first a triad specific event which is an hour-long event, where PO, BA, SM, work collaboratively to decompose ILRs into LLRs. The second session is a 30 min event, where a BA typically provide clarity of requirements and understanding among engineers and breakdown ILRs, into further requirements wherever needed.

In addition to the translation of ILRs into detailed requirements that are done via requirements discovery and understanding, practices also contribute to the formation of potential new HLRs and ILRs. For example, in MEL organization, ***innovation morning*** is typically organized on a fortnightly basis, where half a day is reserved for engineers for sharing innovative ideas and/or potential new requirements.

8.4.2 Decision making team

A decision-making team which is typically a cross-functional team, works collaboratively to make decisions based on the priority of LLRs or, user stories., In a team's triad, typically the PO interfaces through the boundary spanner role, in terms of providing clarity of requirements and communicate any change in decided priority on the behalf of product management. These findings are consistent with those reported by Gat (2006), who posit the role of requirements architect, who is the PO in the studied cases, is embedded in the delivery team to bridge the gap between product management and the development team. The designation the decision makers hold, may vary between organizations and one person may hold multiple responsibilities (Leffingwell, 2011). A typical reason for this variance, includes organization needs, scaling agile frameworks and their local adaptations.

MEL employed the DAD framework where PO, TL, AO are the primary roles that are typically present in delivery team (Ambler & Lines, 2012). As a result, a delivery team triad in MEL organization typically consists of PO (provide business perspectives), TL (provide resource and ensures delivery), AO (provide technical perspectives), UX/UI (provide customer perspectives) if needed.

AKL organization adapted scrum practices at the team level, where a delivery team triad typically consists of PO (provide business perspectives), SM (provide resource and ensures delivery), technical BA (provide technical perspectives), UX/UI (provide customer perspectives) wherever needed works collaboratively to decompose ILRs into LLRs and set priority on them.

8.4.3 Decision-making events

Decision-making events (organized via CTs, face to face events as discussed in section 8.5) that are organized to make a decision on the priority of LLRs occurred on a regular cadence (Bjarnason et al., 2011; Heikkilä et al., 2017). Decision-making events typically occurred on: fortnightly basis, weekly sync-up call, ad-hoc basis (i.e., as and when necessary).

The following are the events that emerged from our findings, commonly reported in the literature as well, to make decision on the priority of LLRs at the team level:

Backlog refinement meeting: Each delivery team has their own backlog that contains a priority list of LLRs such as user stories. Each delivery team has an on-going backlog refinement process, owned by the PO of the team, where LLRs are refined, decomposed into further user stories if required, and are prioritized. The goal is to set at least two IRs/sprints of work which should always be ready in their team backlog (Heikkilä et al., 2017; Hobbs & Petit, 2017). Backlog refinement is typically held on a weekly basis and takes approximately one hour.

Lookahead meeting: In MEL organization, the delivery team conducts a look ahead meeting (Ambler & Lines, 2012), which is typically a DAD practice that helps with *backlog refinement* and *IR planning*. The look ahead meeting is generally held in the second week of the current IR. Look ahead meeting is typically organized in two sessions- *decision making team look ahead* where what will be the next highest business value items that the team needs to tackle in the next two IRs are decided, and *team look ahead* where engineers review the estimates and ensures requirements are clear. Any spikes or investigation that needs to happen in the current IR to be successful in a future IR, are also identified. Moreover, any risks or unknowns which need to be handled or taken into consideration so execution will be smooth, are identified. The overall goal of the team lookahead is when the team goes to IR planning, they should have as much clarity as possible.

*Iteration planning (aka sprint planning) meeting**:*** IR planning (Ambler & Lines, 2012; Leffingwell, 2007) happens at the end of the current IR, after the retrospective of current IR, where user stories are broken down into ***tasks,*** if needed and assigned to the engineers for the next IR.

8.4.4 Requirements artefact

LLRs are typically specified in the form of user stories, the quickest sort of element deliverable through a thin pipeline, that can give some valuable feedback. In addition to this are: a spike (e.g., technical investigation), tasks (e.g., test approach), technical-debt and/or enhancement work also specified as requirements artefact at the team level (configured via CTs as discussed in Section 8.5).

8.4.5 Decision making practices (decision-making factors and techniques)

Decision-making factors that emerged from the study findings to set the priorities on LLRs, include *strategic priority, HLR priority, ILR priority, business value, Symbiotic relationships/dependencies (e.g., B will be easier if we do A first), relative implementation difficulty, team ability and capacity.*

According to the study findings, consideration of these factors and evaluation seems to be case and company specific. In MEL organization, as discussed in Section 8.3.2, ILRs are detailed enough and typically considered as LLRs at the team level. However, ILRs can be broken down into further requirements if needed. In MEL organization as discussed in Section 8.3.2, BV is typically a main prioritization driver which is evaluated via a combination of quantitative weights via a numerical scale as well as qualitative weights e.g., MoSCoW technique (Berander & Andrews, 2005). These are typically employed to evaluate the priority score of LLRs at the team level as well (termed ILRs at the second part of domain level).

In AKL organization, BV is not typically evaluated in order to set the priority on LLRs. Priority of *LLR(s)* is typically based on tacit knowledge, where symbiotic relationships/dependencies (e.g., B will be easier if we do A first), relative implementation difficulty, team ability and capacity are typically considered, while deciding priorities on LLRs.

To sum up, the research findings indicated that the process that is employed for decision making at the team level in both studied organizations, is similar (as shown in Table 8.8). But the process specific practices seem to be case and company specific (as shown in Table 8.7). For example, factors that are evaluated to set priorities on LLRs, are typically evaluated via a combination of *quantitative methods* , a weighting system, as well as *qualitative* using human judgement (Anand & Dinakaran, 2017; Daneva et al., 2013; Popli et al., 2014) These factors seem to be case and company specific. Typical reasons

for this variance, includes type of business an organization is performing, scaling agile framework, requirements abstraction model and their local adaption.

Table 8.7 Process specific practices at the team level

Process specific practices		**MEL Organization**	**AKL Organization**
Terminology for LLRs		User stories	User stories
Time frame		Fit in an iteration (i.e., 2 weeks) and can be implemented via an individual engineer of delivery team	Fit in an iteration (i.e., 2 weeks) and can be implemented via an individual engineer of delivery team
Scope		Generally detailed enough to provide required information to the delivery teams for implementation.	Generally detailed enough to provide required information to the delivery teams for implementation.
Prioritization practices	Numerical scale	Numerical scale 1 to 5	-
	Prioritization factors	strategic priority, BV, HLR priority, team ability and capacity, dependencies (typically technical dependencies	strategic priority, BV, HLRs priority, ILRs priority, team ability and capacity, dependencies (typically technical dependencies
	Business value a main prioritization driver	Numerical scale 1 to 5	-
		Scoring system	-
		factors consideration for BV evaluation varies with requirements classification	
	Qualitative practice	MoSCoW	Based on tacit knowledge
	Requirements classification	Plays a significant role during prioritization.	Plays a significant role during prioritization.
	Learning experience	Applied during prioritization	Applied during prioritization
	Hierarchical structure of requirements	Applied during prioritization	Applied during prioritization
Decision making events on regular cadence (Organized via CTs, face to face)		Fortnightly, weekly, ad-hoc	Fortnightly, weekly, ad-hoc
Cross-functional decision-making team		POs, TLs, AOs, UX/UI wherever needed	POs, BAs, SMs, UX/UI wherever needed
Requirements artefacts configured via CTs		Jira, Confluence	ADO, Confluence

Table 8.8

Process for decision-making on ILRs at the team level

Process to make decision on the LLRs in the studied case organization (MEL organization and AKL organization)
LLRs: detailed requirements provide required information to the delivery teams termed as LLRs
Team's triad as a decision-making team
Intra-iteration prioritization within the team(s)

8.5 Collaborative technologies and their role in requirements prioritization activities

Collaborative technologies (CTs) were considered as a significant and essential tool for collaboration and to support requirements prioritization activities, admitted by both studied cases' research participants. This perceived usefulness of CTs in distributed collaboration over requirements, is coherent with the contemporary wisdom indicating that an adequate use of CT lessens the impact of distance and supports requirements engineering activities such as requirements prioritization in distributed development and is extensively reported in the literature (Gupta et al., 2009; Holmstrom et al., 2006; Hussain, 2018; Lang & Duggan, 2001). CTs in both studied cases typically enable shared understanding of requirements and specify requirements typically in the form of electronic requirements artefacts (e.g., user stories), resolving conflicts, allowing collaboration with distributed members of a decision-making team. These findings are confirmed by Sinha et al. (2006), Lang and Duggan (2001), who also acknowledge the use of CTs in distributed development and specifically for requirements engineering.

The findings of this research study are mainly confirmatory in terms of the supportive role of CTs for requirements prioritization and collaboration with cross-site members of decision-making team(s). The participants of both studied case organizations admitted the value of CTs for decision making on the priorities of requirements, and to collaborate with distributed members of a decision-making team. In both case organizations, mechanisms that are performed as a part of requirements prioritization such as ***requirements discovery and understanding, decision-making events***, ***requirements artefacts, decision-making practices, decision-making team*** are typically supported via CTs.

CTs that emerged from the study findings for supporting the requirements prioritization across scaling agile levels, include ***Spreadsheet, Jira, AHA, ADO, Confluence,***

PowerPoint, MSTeams, Email employed as synchronous and asynchronous mechanisms (see the roles of these CTs as shown in Table 8.9 below) which are extensively reported in the literature as well (Daniels et al., 2015; Gupta et al., 2009; Hussain & Clear, 2014).

Spreadsheet as CTs was identified to have played a crucial role in the MEL organization, to promote collaboration with distributed members over requirements, specifically at the portfolio level and to create the final form of a shared requirement artefact (i.e., formal sign-off on the priority of HLRs). Spreadsheet provided the flexibility to the cross-site members of a decision-making team, to amend and structure the spreadsheet to perform requirements prioritization during synchronous collaboration. This is typically done via synchronous editing, using screen share feature of communication media such as MSTeams. However, a shared requirement artefact such as a portfolio backlog, yielded as an outcome of decision making, is formally maintained in a separate collaborative tool "AHA". As a consequence of this, it can be problematic for the decision-making team to maintain two mechanisms to support requirements prioritization, which often results in duplication of effort. Findings regarding the use of spreadsheet to collaborate with cross site team over requirements coincide with Hussain and Clear (2014) who also admitted the supportive role of spreadsheets in managing the requirements.

PowerPoint slides: PowerPoint as CTs was identified to have played a role in both case organizations to collaborate with distributed members over requirements, specifically for knowledge sharing that yields shared understanding of priority of requirements. During decision making events such as quarterly bigroom planning, PowerPoint slides typically disseminate knowledge sharing via screen share features of synchronous communication media such as MSTeams.

AHA: AHA, which is an online SaaS tool as CTs, was identified to have played a significant role in MEL organization supporting decision making activities specifically performed at the portfolio and domain level:

- AHA tool provides multilevel visibility across an entire portfolio and helps tracking the progress of which stage a specific requirement is at.
- A template is configured in AHA to specify requirements.
- Scoring matrix an automated matrix configured in AHA through which priority score of particular requirements is evaluated.
- An online customer portal configured in AHA where customers collaborate on potential ideas and/or requirements.

ADO is another online SaaS tool which is employed in AKL organization, as CTs provide multilevel visibility across an entire portfolio, tracking the progress of requirements, building requirements artefacts (e.g., portfolio backlog, product backlog), configure template to specify requirements (e.g., template for HLRs, template for ILRs, business-case), scoring matrix configuration to evaluate the priority score of requirements.

Jira: Another significant tool as CTs is Jira which is employed by the MEL organization to provide support for decision making activities, specifically at the engineering level, such as configuring a template to specify requirements, build requirement artefacts, (e.g., user story, team backlog), progress of a particular requirement.

Confluence: confluence as CTs provide a crucial role in both studied case organizations in terms of storing, tracking and managing the requirements documents. These are generally needed for detailed discussions and/or analysis on certain items and typically yields shared understanding among teams.

The integration feature of these tools such as AHA tool integration with Jira, yields multilevel visibility of requirements and influences the studied case organizations as their primary CT tools, to collaborate with distributed members over requirements. This finding is consistent with Hoffmann et al. (2004) who also admitted the need of integration between tools in order to achieve traceability and/or visibility of requirements, between different phases of requirements (e.g., decomposing HLRs into ILRs).

Table 8.9 Collaborative Technologies

CTs	**Role of CTs for requirements prioritization across scaling agile levels**					**Case 1**	**Case 2**
	Requirements discovery and understanding - elicitation of requirements - knowledge sharing- shared understanding of requirements	**Requirements artefacts**	**Decision making practices- priority score matrix**	**Decision making events: collaborate with cross site members over requirements**	**Scaling agile levels**		
Spreadsheet	For knowledge sharing- *shared understanding of requirements*	Spreadsheet as an informal Portfolio backlog		decision making event(s) performed at the portfolio level	Primarily employed at the Portfolio level, Can be used at the other levels as well (i.e., domain level, team level)	Yes	No
PowerPoint	For knowledge sharing- *shared understanding of requirements*			decision making event(s) performed at the portfolio level	Primarily employed at the Portfolio level, Can be used at the other levels as well (i.e., domain level, team level)	Yes	Yes
AHA	Requirements elicitation- *Online customer portal configured via AHA* Knowledge sharing- *shared understanding of requirements*	Backlog (i.e., Portfolio backlog), template configured for HLRs, template configured for strategic themes,	Automated scoring matrix for HLRs configured in AHA	decision making event(s) performed at the portfolio level, 1st part of domain level	Portfolio level, Ist part of domain level	Yes	No

CTs	**Role of CTs for requirements prioritization across scaling agile levels**					**Case 1**	**Case 2**
	Requirements discovery and understanding - elicitation of requirements - knowledge sharing- shared understanding of requirements	**Requirements artefacts**	**Decision making practices- priority score matrix**	**Decision making events: collaborate with cross site members over requirements**	**Scaling agile levels**		
ADO	Knowledge sharing- shared understanding of requirements	Backlog (i.e., Portfolio backlog, work item list/product backlog, team backlog), Template for strategic themes, Template for HLRs, Template for ILRs, Template for LLRs	Automated scoring matrix for HLRs configured in ADO	decision making events that are performed at all the levels	Portfolio level, Domain level, Team level	Yes	No
Jira	Requirements elicitation- *support team requests* Knowledge sharing- *shared understanding of requirements*	Backlog (i.e., work item list/product backlog, team backlog), Storyboard, Template for ILRs		decision making event(s) performed at the 2nd part of domain level and team level	2nd part of domain level, Team level	Yes	No
Confluence	Knowledge sharing- shared understanding of requirements	in-depth information in terms of decision making on the requirements (i.e., Strategic themes, HLRs, ILRs, LLRs) documented via confluence			Portfolio level, Domain level, Team level	Yes	Yes

CTs	**Role of CTs for requirements prioritization across scaling agile levels**					**Case 1**	**Case 2**
	Requirements discovery and understanding - elicitation of requirements - knowledge sharing- shared understanding of requirements	**Requirements artefacts**	**Decision making practices- priority score matrix**	**Decision making events: collaborate with cross site members over requirements**	**Scaling agile levels**		
MSTeams Features of MSTeams *Screen sharing, Instant messaging, Team specific channel, Tag to notify people, recording of decision-making events, Offline chat audio/video call, web conference*	Elicitation of requirements, knowledge sharing- *shared understanding of requirements*			decision making events at all the levels for synchronous and asynchronous communication	Portfolio level, Domain level, Team level	Yes	Yes
E-mail	For knowledge sharing- *shared understanding of requirements*				Least commonly used asynchronous CTs. Employed at Portfolio level, domain level, team level	Yes	Yes

Email: Email is the least frequently used asynchronous communication technology for cross site communication in the studied cases. According to the research participants, emails were seen as tools that are typically used to send formal invitations to the decision-making events such as bigroom planning. Synchronous as well as asynchronous communication via web conference tool by MSTeams, was a preferred communication method in the organizations. These findings coincide with Zowghi and Damian (2003), who suggest using emails along with rich communication media, to collaborate with distributed members of decision making team(s).

Web conference tool: In the cases, web conferencing tools such as MSTeams were utilized primarily as synchronous communication technology to collaborate with distributed members of decision-making teams and organize decision making events to make decision on the priority of requirements. The application sharing facilities of web conferencing tools such as, document and screen sharing, enabled team members to work on shared requirements artefacts during audio/video conferencing sessions.

In addition to this, an informal synchronous (e.g., team specific channel and instant messaging via MSTeams) and asynchronous (e.g., offline chat via MSTeams) communication, is also organized via Web conferencing tools in both organizations to collaborate with cross-site members. The findings related to the use of synchronous communication along with asynchronous modes of communication as well as requirements artefacts, has seen a useful practice in both studied organizations to collaborate with cross-site members of decision-making teams. These findings are consistent with Zowghi and Damian (2003) who reported usefulness of synchronous and asynchronous communication in distributed organizations to manage requirements engineering activities, like requirements prioritization, requirements elicitation.

8.6 Challenges and strategies

This section discusses the challenges that are faced by the studied cases while making decisions on the priority of requirements in SADSD and the potential strategies.

8.6.1 Challenges specifically related to distributed members of decision-making team

In the studied case organizations, members of decision-making team who typically make decisions on the priority of requirements across scaling agile levels are distributed. In AKL organization, members of decision-making teams, are locally distributed and pose no specific challenges in terms of prioritization of requirements. On the other hand, in MEL, organization, members are globally distributed, and collaboration often suffers due to distance, and creates challenges for requirement prioritization activities. Challenges are identified on the basis of three distance dimensions - , temporal distance, geographical distance, socio-cultural distance, which are discussed below (Holmstrom et al., 2006).

8.6.1.1 Challenges due to geographical distance

Geographical distance (Holmstrom et al., 2006) is one of the challenges that limits face-to-face collaboration during decision making events . Participants of MEL organization admitted that adoption of CTs (discussed in section 8.5), enables geographical distance to be manageable (Paasivaara & Lasssenius, 2004). Rich synchronous communication along with an asynchronous mode of communication as well as electronic requirements artefacts (Paasivaara & Lasssenius, 2004), have seen useful practices in MEL organization to allow collaboration with cross-site members of decision-making teams on the priority of requirements.

8.6.1.2 Challenges due to temporal distance

Temporal distance (Holmstrom et al., 2006) was the commonly stated challenge in MEL organization. Managing the time zones is more difficult, especially during day light saving. As a consequence of this, delays in communication and/or response has happened, especially at the weekend, and beginning of the week, when typically, they lose one to two days with some time zones. Late sittings occur due to lack of overlapping time zones that often results in stress and work-life balance problems. These findings are confirmed by Grewal and Maurer (2007), who reported synchronous communication suffered due to time zone differences in large scale globally distributed organizations.

To manage the temporal distance, the company has arranged ***travel*** (Paasivaara & Lasssenius, 2004) for decision makers in order to attend the decision-making event(s) in-person, specifically the "***quarterly bigroom planning event organized at the portfolio level***". This has yielded outcomes that it made it easy to reach a consensus, better understanding of requirements, and easy collaboration during decision making. But due to the cost constraints, this does not happen frequently (Holmstrom et al., 2006). Therefore, decision making teams at the portfolio level typically meet four times a year, where they meet in-person at least twice a year in an auditorium either at the head office or Melbourne site and virtually alternate, every three months.

In addition to this, to reduce the coordination effort during decision making on the priority of requirements at the subsequent level, MEL organization derived the HLRs in such a way that they are generally domain specific where generalizing specialist teams in a particular domain typically at the same location, work collaboratively to perform decision-making activities.

8.6.1.3 Socio-cultural distance

According to the research study findings, no particular challenges occurred due to socio-cultural distance on requirements prioritization activities, which contradicts the challenges that include misunderstanding of requirements, delay in requirements due to language and cultural distance, that are commonly reported in the literature (Holmstrom et al., 2006; Hussain, 2018; Ravishankar et al., 2012; Zowghi & Damian, 2003).

8.6.2 Challenges occurred regardless of distribution

This section describes the challenges that occurred regardless of distribution related to the requirements prioritization in large scale agile.

8.6.2.1 Tension between product and engineering

In both case organizations, there is always a tension, generally a healthy tension, between product and engineering in terms of decision making specifically on smaller impact requirements. This includes tech-debt work and/or internal improvement work. In the experience of studied case participants, generally the decision making on smaller impact requirements requires a lot of negotiation, as value is not clearly articulated with them. On the other hand, bigger impact requirements such as strategic work that provides direct value to customers and business, require less negotiation due to their direct alignment with business strategy. In addition to this, decision makers like PMs, generally give importance to requirements that bring high revenue, which in-case of smaller impact

requirements, is hard to articulate. Tension between product and engineering in terms of decision making on the smaller impact requirements, is reported in literature as well (Hannay & Benestad, 2010; Heikkilä et al., 2010, 2015; Viswanath, 2016). The literature confirms the findings of this research study as both study participants reported that smaller impact requirements often receive less attention due to lack of clearly articulated process associated with them.

In such cases, often additional research is needed to gather all required information/facts/data in order to justify the business problem. As a consequence of conflicting perspectives between product and engineering, it often causes tension, a breakdown in the ability to collaborate, and creates pressure on team members, which coincides with Hannay and Benestad (2010), and Viswanath (2016) findings.

Strategies:

The study findings indicated that decision-making processes should yield an optimal mix of requirements, which coincides with Hannay and Benestad (2010) findings. The potential strategy which is adopted by both case organizations to ease this tension, is that categorizing the requirements, like., strategic work, technical debt work, maintenance work and allocating capacity allotment to each of those categories.

8.6.2.2 Accumulating tech-debt work

The research study participants asserted that, it is impossible to get completely rid of tech-debt, which coincides with existing literature (Gupta et al., 2017; Heikkilä et al., 2015, 2017; Kasauli et al., 2017; Roopa et al., 2017; Viswanath, 2016). In AKL organization, participants asserted that technical debt generally resides in the backlog until the time it hinders the delivery team in their progressive work.

The possible reasons to accumulate tech-debt, include sometimes teams struggle to find a time payoff for tech-debt, lack of clearly articulated value associated with tech-debt work, and as a result of cut scope, the urgency to deliver feature within a specified timeframe and/or business need, which is commonly reported in the literature (Gunyho & Plaza, 2011; Gupta et al., 2017; Kasauli et al., 2017; Roopa et al., 2017; Viswanath, 2016). As a result of not paying off tech-debt, often results in increased turnover, creates frustration among team members, quality issues start to creep in, and impacts delivery (Gupta et al., 2017; Kasauli et al., 2017; Viswanath, 2016).

Therefore, mechanisms need proactive and reactive strategies to manage technical debt. The case organizations, adopted the following strategies to minimize technical debt:

- Implement technical debt together with strategic work like user story,
- Payoff tech-debt as a part of each development IR
- Categorize the requirements - strategic work, technical debt work, maintenance work
- Encourage teams to bring-in tech-debt
- Establish dedicated technical debt team
- Capacity allotment to handle tech-debt such as 10% to 20% of engineering time
- Plan one development IR to pay off tech-debt when in-between the release IR (Alsaqaf et al., 2018; Gupta et al., 2017; Kasauli et al., 2017; Roopa et al., 2017; Viswanath, 2016).

8.6.2.3 Quality of requirements

In both case organizations, participants reported that it is challenging to provide the right amount of granularity for the engineering team to work on. Participants revealed that AC and DoD, in some cases, are ill defined or continue to focus on output rather than customer outcomes, resulting in missed expectations. They often face a challenge in getting the quality of the work and detail right for the teams, which often impacts delivery. This challenge is consistent with the findings reported by Bjarnason et al. (2011), Dikert et al. (2016), Gat (2006), Hussain (2018), Kasauli et al. (2017), who also report that writing a meaningful requirement or user story, is a continuous challenge in large scale distributed organizations.

One of the steps that has started in this direction by AKL organization, is to allocate dedicated capacity, say 10% to 20%, for refinement to ensure the right amount of detail is there for the teams to work on. Participant MEL_P1 of MEL organization, suggests AC and DOD need to support the lean principles of delivering working codes which are fit for purpose, in as short a time as possible. In addition to this, PMs and engineers need to agree on clear AC and DOD, including performance testing, documentation.

8.6.2.4 Bias during decision making

The study findings indicated that the ‘Command and control’ nature still exists in terms of decision making on the priority of requirements. Priority decisions are generally biased by the perspectives of higher-level people such as portfolio management. Potential

requirements that are proposed by such higher-level people typically get higher importance as compared to the requirements/ideas that are proposed by lower-level people, like the support team. As a consequence of this, often tension arises, breakdowns in the ability to collaborate, and creates pressure on team members. This finding coincides with Bjarnason et al. (2011), Gunyho and Plaza (2011) who report innovative ideas from lower levels need to be taken into consideration to produce an optimal list of priority of requirements.

8.6.2.5 Difficult to measure Business Value

Business value which is native to Agile, has higher influence while making decisions on the priority of requirements (Daneva et al., 2013; Heikkilä et al., 2015; Herrmann & Daneva, 2008; Inayat et al., 2015). The study participants asserted that it is difficult to evaluate the business values, as everyone may have a different level of understanding, and sometimes scoring may not align with the importance of the requirements. In such cases, business education for the decision makers suggested by one of the participants of this research study may help them to understand the business value of potential requirements.

8.6.2.6 Dependencies across domains(s)

Participants of AKL organization asserted that they are facing challenges in terms of breaking down the requirements. At present, HLRs cross several domains, where one team works on in between 2 to 5 HLRs at a single time, that often results in unnecessary dependencies across domain(s), creating conflicting priorities, tension between decision makers in order to get their priority done, reprioritization of requirements and delay in delivery. This finding coincides with Hussain (2018), Kasauli et al. (2017), Paasivaara et al. (2018), Paasivaara and Lassenius (2016), who also reported difficulty in breaking down large and complex requirements into smaller requirements. In addition to this, participants stated that sometimes dependencies revealed during implementation, often technical dependencies, also results in reprioritization, and unnecessary delays in delivery. Therefore, mechanisms are needed through which possible dependencies are identified and managed before commencing the implementation. This finding coincides with Scheerer et al. (2015), who also suggested to identify and manage dependencies upfront.

The following are some of the mechanisms that are adopted by the case organizations to manage dependencies:

Domain specific requirements: In MEL organization, HLRs are domain specific where teams are generalizing specialist teams in order to overcome unnecessary dependencies. AKL organization also uses work-in-progress to restructure the domain(s) where requirements will be domain specific.

Identify dependencies upfront: Scheerer et al. (2015) recommended decomposing the requirements into smaller requirements as well as approaches that generate an overview of connections between requirements, to manage dependencies and make them transparent early on. At the domain level, both studied case organizations employed a "*discovery phase*" practice to identify dependencies upfront. At the engineering level-a team level, inception phase in MEL organization, feature kick-off in AKL organization, are the practices that are employed by the studied case organizations to identify the dependencies upfront.

CTs: collaborative technologies support is undertaken by the studied case organizations to visualize the dependencies across domains and/or teams (discussed in section 8.5). This finding is consistent with Scheerer et al. (2015), who recommended adopting tool support in order to create transparency and visualization of requirements across product and engineering teams.

Microservices architecture: Stab API is employed by MEL organization to provide a stable and known set of interfaces to decouple the dependencies. Scheerer et al. (2015) also suggested that the architecture of software plays a strong role in dependency avoidance.

Continuous collaboration and communication: Decision making team has continuous communication and collaboration that also helps in identifying and managing the dependencies (AbdElazim et al., 2020; Berander & Andrews, 2005; Lehtola et al., 2004). Some of these events that are employed in the studied case organizations, include *quarterly bigroon planning, portfolio sync-up (at portfolio level), triads' meeting (at the domain level), IR planning, Look ahead meeting (at the team level)*.

8.6.2.7 Dealing with quality requirements (QRs)

The studied case organizations face challenges with QRs such as security and performance which is a commonly reported challenge in the Agile literature (Alsaqaf et al., 2017; Heikkilä et al., 2015; Inayat et al., 2015; Ramesh et al., 2010). In MEL organization, interviewees asserted that QRs, specifically performance are not met, which

is often revealed in production. Some of the reasons include that QRs are ill defined, urgency to deliver features in a specified time frame, sometimes the effort required for a particular QR implementation, is left out during estimation. These findings coincide with Alsaqaf et al. (2017), Roopa et al. (2017) who also reported challenges with QRs specification, lack of QRs visibility in large agile distributed projects.

Following are some of the strategies that are employed by the case organizations to overcome the QRs problems:

- Unambiguously specified the QRs
- Inception phase to identify QRs upfront
- Discovery phase to identify QRs, specifically performance requirements upfront
- Microservices style architecture
- In-Progress: security/to ensure zero attacks via API gateway
- Service level objectives dashboard
- Automate QRs
- Capacity allotment to work on QRs

8.6.2.8 Difficulty in estimating effort

The studied organizations experience difficulty in estimating the complexity of requirements upfront. Participants revealed that sometimes effort that is required to implement a requirement, is underestimated that often impacts delivery. At present in both case organizations, requirements are typically estimated based on experiences of decision makers such as a, guess estimate. This finding coincides with Dikert et al. (2016), Heikkilä et al. (2015), Usman et al. (2018), who also reported it being challenging to estimate the complexity of requirement. Therefore, mechanisms that provides more objective estimates need to be investigated.

8.6.2.9 Inadequate requirements analysis

Participants of both case organizations admitted that insufficient understanding of a potential requirement such as a business problem, often yields inadequate prioritization. As a consequence of this, often a feature is built where the business value wasn't there or wasn't at the appropriate priority.

Therefore, the right amount of effort and collaboration is required in order to make an efficient decision on requirements' priorities. Mishra and Mishra (2011) also reported

inadequate prioritization often yields unused or rarely used features due to their irrelevance to business.

Both case organizations are working towards in-depth understanding of the business problems and delivering solutions that are going to solve those problems and provide solutions that are easy to consume by business users. One of the recently adopted approaches in this direction, is introducing a ***discovery phase*** where product and engineering work collaboratively, to identify all possible unknowns and/or risks that are associated with a business problem and/or potential requirements before it is committed for delivery.

8.6.2.10 Lack of requirements clarity

In MEL organization, participants asserted that the team often lacks clarity, specifically on requirements that are needed to be implemented to unblock the other team. As a result of this, teams often face difficulty in managing their own work vs work that is required to unblock the other team. This finding coincides with Hussain (2018) who also reported requirements' clarity issues in large scale agile projects.

Therefore, a continuous communication and collaboration mechanism is required to manage this challenge (Hussain, 2018). Some of the strategies that are employed by MEL organization includes:

- **Escalation matrix:** As an example - within the team, it should be the PO's responsibility to provide clarity for the teams.
- **Teams' catch-up:** Participant MEL_P2 of MEL organization explains that all the teams who are involved in a particular HLR implementation, have a common meeting, typically meeting on a fortnightly basis, to establish relationships with other teams. As a consequence of this, one team can directly approach other teams' members if needed, for example to clarify requirements.

8.7 Summary

The conceptual framework of requirements prioritization in SADSD that was built by studying the two scaled organizations, is discussed in this chapter. In addition to this, challenges that have emerged from this research as well potential strategies to surmount requirements prioritization challenges, are discussed.

Chapter 9 concludes the findings of this research study.

9 Conclusion

This chapter concludes the research work that is reported in this research study. It summarizes the work that was undertaken, the main findings that emerged in terms of addressing the research questions that were posed to attain the objectives of this study, key contributions made, and it acknowledges the limitations as well. Future research implications are also acknowledged in this chapter.

9.1 Summary

The main objective of this research study was to understand the requirements prioritization process, from an idea to operation, in scaled agile distributed software development. This was motivated by the scarcity of empirical research on prioritization of requirements process in SADSD environment which was revealed via an MLR study that was conducted as a preliminary study to understand the current state of art of RE in SADSD, (discussed in Chapter 3). Another motivation factor to conduct research on requirements prioritization in SADSD was my interest and previous experience in RE (masters research in Agile RE), as impacted by the relatively novel phenomenon of adoption of Agile methods in large scale distributed software development.

This research study has sought interpretive exploratory multiple case studies to investigate the phenomena of interest (requirements prioritization process in SADSD). Empirical data was collected from the two software development organizations for this research study, one from Australia (Pseudonym: MEL organization) and another one was from New Zealand (Pseudonym: AKL organization). A kick off meeting was held with the managers of both participating organizations (my primary contact in the studied organization) to discuss the objectives of study and to understand the development processes of the participating organizations. This knowledge was then used to perform in-depth investigation, primarily employing semi-structured interviews with both participating case organizations.

9.2 Main findings related to research questions:

Following are the research questions (RQ) that were set to attain the objective of this research study:

RQ1: How is prioritization of requirements done (i.e., from an idea to operation) in scaled agile distributed software development?

RQ1.1: Who are the decision makers and what are they responsible for?

RQ1.2: What prioritization criteria are employed to make decisions on the priority of requirements?

RQ1.3: What are the practices and artefacts employed for decision making on the priority of requirements?

RQ2: What are the challenges related to the requirements prioritization processes?

RQ3: What are the strategies to mitigate the impact of reported challenges?

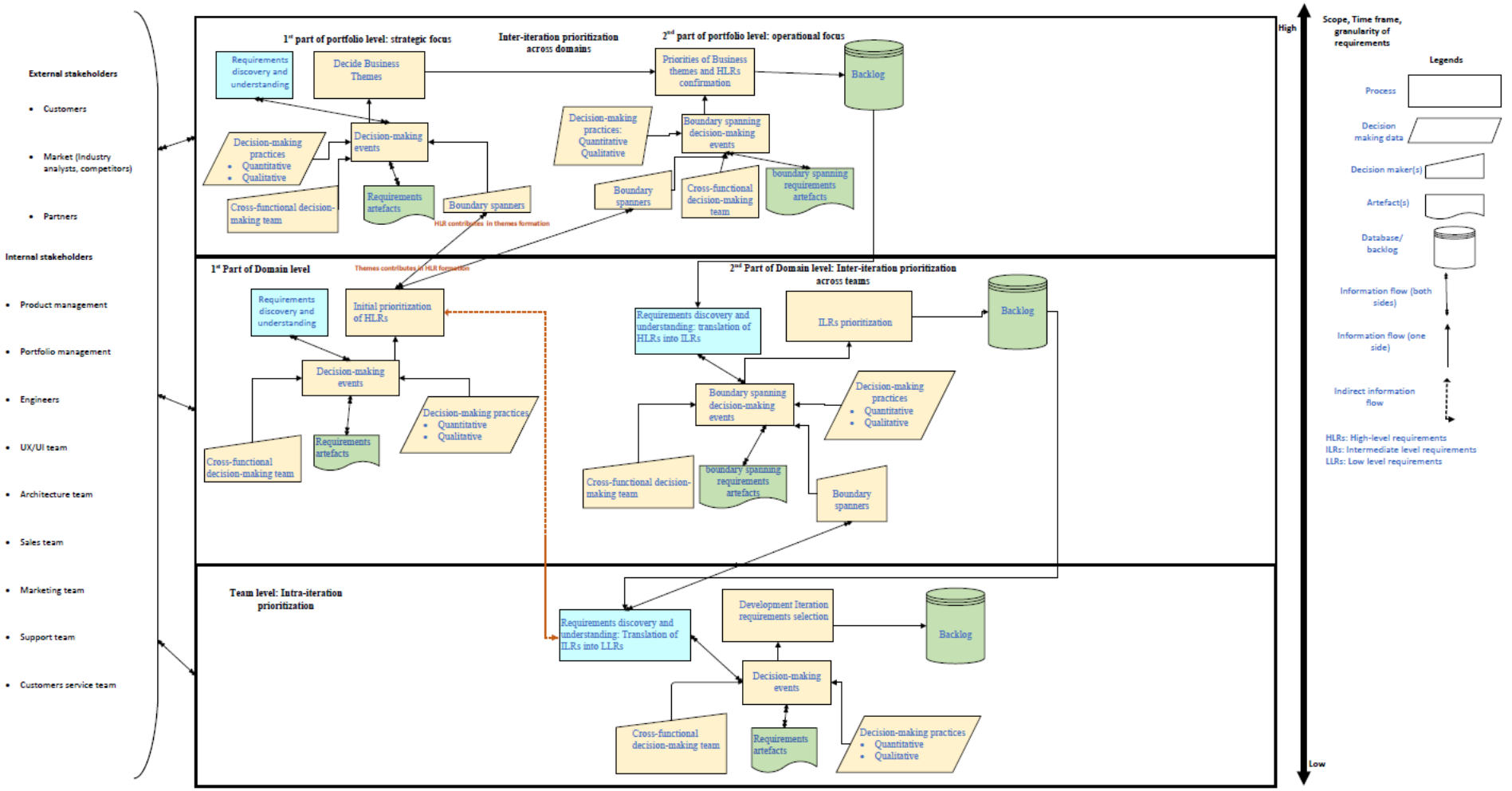
9.2.1 Findings as an outcome of RQ1

In an effort to answer the RQ1 and its sub-research questions (i.e., RQ1.1, RQ1.2, RQ1.3), a conceptual framework is built for requirements prioritization in SADSD, shown in Figure 9.1, by studying the two case organizations. In order to attain that, 28 practitioners were interviewed from the participating case organizations,18 from MEL organizations, 10 from AKL organization. In addition to this, relevant data collected from other sources as well, such as recording of decision-making events, and requirements artefacts. Data analysis was performed through within the case and cross-case analysis. In the first phase of analysis, individual case study data was analysed without any particular theoretical framework in mind, via employing grounded theoretic analysis (Glaser & Strauss, 1967).

In the second phase of analysis, literature support (deductive analysis), was undertaken in order to structure the findings that emerged from the studied case organizations and guided me to develop an initial conceptual framework of requirements prioritization in SADSD. In the third phase of analysis, cross-case analysis was performed to identify commonalties, contrasting or aggregating the findings of studied case organizations, in order to enhance the generalizability, robustness, and applicability of findings. The cross-case analysis guided me in evolving the conceptual framework of requirements prioritization in SADSD, that was initially developed by analysing the individual case organizations' data as well as via literature support; discussed in Chapter 4: Section 4.4.3).

Figure 9.1

Conceptual framework of requirements prioritization in SADSD



This section provides a high-level overview of the conceptual framework in Figure 9.1, (detailed discussion is in Chapter 8), and the key findings that emerged as an outcome RQ1.

The analysis of the studied case organizations, revealed that there are three key decision-making levels, including portfolio level, domain level, team level, where the priorities of requirements are decided in scaled agile software development. These are typically performed in the form of intra-iteration prioritization and inter-iteration prioritization. The scope of intra-iteration and inter-iteration prioritization varies at each of these scaling agile decision-making levels – portfolio, domain and team levels.

At the portfolio level, inter-iteration prioritization is typically performed where organization-wide priority decisions are formed. According to the findings of this research study, portfolio level is the highest decision-making level, where primarily the business goals that connect with an organization's overall business strategy, are decided and prioritized. Requirements are typically very high business level requirements, generally unchanged for up to one year or can go above year. They are represented as ***strategic themes*** in this research study, that are decided at the portfolio level and have influence on the decision-making on the priority of requirements across all the scaling agile decision-making levels. A more elaborate description on strategic themes is provided in Chapter 8. In addition to the decision-making on strategic themes, requirements, particularly ***high-level requirements (HLRs)*** in this study, that are needed to accomplish overreaching strategic themes, are formally signed-off, prioritized and communicated to the delivery team(s) for implementation.

Domain level plays a twofold role in terms of making decisions on the requirements priorities. ***In the first part of domain level,*** intra-iteration prioritization decisions are made, where requirements that require significant effort and/or investment termed ***HLRs*** in this research study, that emerge from each of the domain(s) and/or contributes to the formation of strategic themes are initially defined, prioritized and submitted to the portfolio level for formal decision-making. ***In the second part of domain level***, inter-iteration prioritization of requirements across teams (project specific) is performed. In the second part of domain level, HLRs that are generally broader in scope and typically need more than one delivery teams for implementation, are broken down into further requirements and undergo inter-iteration prioritization across teams in order to implement them via short development cycles (e.g., two weeks) and by a single delivery team. In this

research study, requirements that are decomposed from the HLRs are termed intermediate level requirements (ILRs) that can be implemented via a single development IR/sprint (e.g., 2 weeks) and by a single delivery team.

At the team level, intra-iteration prioritization decisions are performed where team specific requirements are decided and prioritized. At the team level, requirements should be detailed enough to provide required information, such as requirements clarity, to the delivery teams, for implementation and can be achieved via short development cycles of 2 weeks. The ILRs are decomposed into further requirements wherever needed, termed ***low-level requirements*** (LLRs), and through this, in-depth information is provided to the delivery team(s) before the beginning of implementation.

Prioritization of requirements mechanisms at each of the decision-making levels (i.e., portfolio level, domain level, team level) typically consists of requirements discovery and understanding (open innovation approach for requirements discovery), decision making events (which occurred on a regular cadence), decision making practices (a combination of quantitative as well as qualitative techniques), decision making teams (cross-functional decision-making teams), requirements artefacts. This study indicated that these activities are not clearly separated and instead, are highly integrated and occur at various levels with varying details in order to make decisions on the priority of requirements.

Boundary spanning is an important component of requirements prioritization in scaled agile development that emerged from the case analysis. Boundary spanning is important for requirements prioritization as it yields knowledge acquisition, negotiation, consensus building, and conflict resolution, which are the key activities typically needed while making decisions on the priority of requirements. Three categories of boundary spanning mechanisms are identified in this research study, that include *boundary spanning event(s), boundary spanning requirement artefact(s), boundary spanner role(s)* (Jain et al., 2014). According to the findings of this research study, boundary spanning mechanisms primarily occur when requirements prioritization decisions are made at inter-iteration levels - the portfolio level, and second part of domain level. However, boundary spanning mechanisms partially occur during intra-iteration prioritization as well.

Process and process specific practices According to the findings of this research study, the process that is employed to make decisions on the priority of requirements at each of the decision-making levels, seems to be universal, but the process specific practices are greatly varied and tailored as per an organization's need. One such example is HLRs-

scope, timeframe, and nomenclature of HLRs in the studied case organizations greatly varied, but the process that is employed to make decision on the priority of HLRs seems to be universal. In MEL organization, HLRs are generally termed as 'features' which typically require significant effort and/or investment and can be fit in a release IR (i.e., three months' timeframe). On the other hand, HLRs are termed as 'epic'(s) which are broader in scope and timeframe and typically take multiple release IRs, in AKL organization than in MEL organization. A typical reason that emerged behind the variance of scope, timeframe, and nomenclature of requirements, includes adaptation of the requirements abstraction model, requirements classification, collaborative technologies (CTs), and scaling agile framework such as, DAD, Scrum @ scale. But, in terms of a process for decision-making, HLRs are initially evaluated, decided, and prioritized in the first part of domain level and then formal decision-making on HLRs is performed at the portfolio level. The detailed information of process and process specific practices that emerged from this research study and employed at each of the decision-making levels, is provided in Chapter 8.

Collaborative technologies (CTs) - spreadsheet, Jira, AHA, MSTeams - emerged as significant and essential tools typically employed as synchronous and asynchronous mechanisms, to support requirements prioritization activities. These include enabling shared understanding of requirements, specifying requirements typically in the form of electronic requirements artefacts, resolving conflicts, allowing collaboration with distributed members of the decision-making team. According to the findings of this research study, mechanisms that occurred at each of the decision-making levels as a part of requirements prioritization, such as ***requirements discovery and understanding, decision-making events, requirements artefacts, decision-making practices, decision-making team,*** are typically supported via CTs. The detailed explanation of CTs and their supporting role in decision-making on the priority of requirements at each of the decision-making levels that emerged from this research study is provided in Chapter 8.

9.2.2 Findings as an outcome of RQ2 and RQ3

In order to answer the RQ2 and RQ3, questions specifically related to challenges that occurred during requirements prioritization and the potential strategies to surmount those challenges, were asked of the study participants. While asking questions related to challenges, I noticed study participants were not covering challenges that may occur, due to the distributed nature of decision-making team(s). Therefore, to identify the requirements prioritization challenges that may typically arise due to distributed nature

of decision-making teams, three distance dimensions - temporal distance, distributed distance, and socio-cultural distance related questions were asked of the study participants (Holmstrom et al., 2006).

The participants of this research study reported several challenges, that include accumulating technical debt (ATD), bias during decision-making (BDM), tension between product and engineering (TPE), quality of requirements (QoR), difficult to measure business value (DMBV), dealing with quality requirements (QRs). Also revealed were: difficulty in estimating effort (DEE), inadequate requirements analysis (IRA), lack of requirements clarity (LRC), dependencies across domain(s) (DaD), distributed distance (DD), temporal distance (TD). The majority of the identified challenges and the potential strategies to surmount those challenges (shown in Table 9.1), had been noted in the prior studies and were verified by the studied case organizations which are discussed in detail in Chapter 8: Section 8.6.

Table 9.1
Challenges that occur during prioritization of requirements in SADSD and potential strategies

Challenge category	**Potential strategies**
For DD	CTs (e.g., MSTeams)
For TD	CTs (e.g., MSTeams) Organize travel Formal synchronous and asynchronous communication Informal synchronous and asynchronous communication (e.g., offline chat) In-person meeting wherever possible Early morning and Late evening meetings
For TPE	Optimal mix of requirements
For ATD	Implement technical debt together with strategic work (e.g., user story)
	Payoff tech-debt as a part of each development IR
	Categorize the requirements
	Encourage teams to bring-in tech-debt
	Establish dedicated technical debt team
	Plan 1IR to pay off tech-debt when in-between the release
For QoR	Dedicated capacity (i.e., 10% to 20%) to ensure the right amount of detail is present in requirements (e.g., user story)
	AC and DOD need to support the lean principles
For BDM	Optimal mix of requirements
For DBV	Business education required for the decision-making team(s)
	Unambiguously specified the QRs

Challenge category	Potential strategies
For QRs	Inception phase to identify QRs upfront
	Discovery phase to identify QRs (specifically performance requirements) upfront
	Microservices style architecture
	In-Progress: security/to ensure zero attacks via API gateway
	Service level objectives dashboard
	Automate QRs
	Capacity allotment to work on QRs
For DEE	No particular solution strategies
For IRA	Discovery phase
For LRC	Escalation matrix Teams' catch-up
For DaD	Identify dependencies upfront (e.g., discovery phase, Inception phase), CTs to visualize dependencies, Continuous communication and collaboration Microservices architecture
	Domain specific requirements

9.3 Research contributions

This research study has made several contributions to providing understanding of a requirements prioritization process in SADSD, across the substantive and conceptual domains as suggested by McGrath and Brinberg (1983).

Substantive domain:

- This study has been added to the very few case studies in the area of requirements prioritization process in SADSD, as empirical research was limited during the commencement of this research study. It was identified via an MLR study that was conducted as a preliminary study, to understand the current state of art of RE in SADSD (discussed in Chapter 3).
- This study identified challenges related to the requirements prioritization in SADSD and confirmed them with literature.
- This study identified potential strategies for surmounting requirements prioritization challenges in SADSD and confirmed them with literature.
- This study identified the roles of decision makers, who make decisions on the priority of requirements in SADSD.

- This study identified the decision-making events, decision-making practices, requirements artefacts, CTs that are needed to make decisions on the priority of requirements in SADSD.
- This study has profiled a fine-grained in-depth multi-case study that explored the life cycle activities of prioritising requirements, from an idea to operation, in SADSD.

Conceptual domain:

- This study has produced the first known novel conceptual framework that captures life cycle activities of prioritization of requirements, from an idea to operation, that is synthesized from the two scaled agile distributed organizations. The study provides a general framework for prioritizing software requirements for organisations. The conceptual framework depicts decision-making levels, timeframe and scope of requirements at each of the decision-making levels, roles involved in decision-making, requirements artefacts, decision-making practices, boundary spanning mechanisms, and CTs that are needed to make decisions on the priority of requirements in SADSD. The conceptual framework has the potential to be applicable in the contexts similar to the organizations that were studied in this research study. In addition to this, new organizations who are looking to scale their prioritization practices such as basic agile prioritization practices, classic decision-making activities to prioritize the requirements.

9.4 Limitations

This research study had to completed in a specific time-period that posed several limitations, which are discussed in this section.

Limit on generalizability: To achieve the objective of this research study, only two organizations were studied (selection criteria is discussed in Chapter 4: section 4.2.3).One was an Australian based large software vendor and has teams all over the world. The other one was a New Zealand based large electricity retailer and has nationally distributed teams. Theory building of prioritization process could be strengthened by looking at the organizations with different context such as large government organizations.

Inability to attend key decision-making events: I initially planned to attend one week of in-person observation before starting the data collection, as well as key decision-

making events, in order to understand the prioritization process of participating case organizations, which was cancelled due to COVID-19 pandemic.

The understanding of the practices and rapport with the participants, would have been improved via attending their key decision-making events. In order to manage this, before starting the data collection, a kick off meeting was held with the primary contacts of both participating organizations, to discuss the objectives of this research study and to understand the development processes of the participating organizations. In addition to this, I managed to get recordings of some of the key decision-making events.

Diverse perspectives: the phenomenon of interest is investigated by interviewing all possible participants, who are typically in decision-making roles on the priority of requirements in SADSD. In order to do this, the initial list of potential participants who are typically involved in decision making was prepared: product owner, product manager, team leader, chief product owner, chief technology officer, chief customer officer, enterprise architecture, UX/UI, developers, testers, architecture owner. This was assisted via studying the literature and some of the popular scaling agile frameworks (e.g., DAD, Scrum @ Scale). This list was then discussed with the primary contact of participating organizations and revised as per their feedback, as both participating case organizations had distributed teams. Therefore, potential participants from the distributed sites also were interviewed to get perspectives of distributed members of decision-making teams wherever possible. However, findings of this research study specifically in terms of distributed perspectives such as socio-cultural impact on prioritization, might have been hindered, due to my limited contact with the distributed sites. In AKL organization, the majority of the participants were interviewed from the head-office (Auckland) and only two participants were interviewed from another site (Hamilton). In MEL organization, although they have seven global sites, including Melbourne (Australia), Cheshire, Belfast (England), Itasca (USA), Bangalore (India), Hamburg (Germany), China, but only key participants (i.e., the person who were typically involved in the decision making on requirements priorities such as PO, SVP architecture) from the Australia and USA sites were interviewed.

Lack of direct access of CTs: Due to the privacy policy of the participating organizations, I could not get direct access to their CTs such as Jira, ADO, and Confluence that are employed as synchronous and asynchronous mechanism to make decision on the priority of requirements. The direct access of their CTs would have

resulted in enhanced data collection and deeper analysis of the requirements prioritization process and their practices. However, I managed to get some of the documents such as requirements prioritization criteria, business case via follow-up with participants.

9.5 Future directions

This research study has several implications for the future research.

The key findings of this research study involve decision-making levels (i.e., portfolio level, domain level, team level), decision making in the form of inter-iteration and intra-iteration prioritization, occurrence of boundary spanning mechanisms (e.g., boundary spanner role, boundary spanning event), supporting role of CTs, process and process specific practices to perform prioritization of requirements at each of the decision-making levels. But this is still an under-researched phenomenon, that requires validation with more varied case organizations to generalize the findings that emerged from this research study.

The two large organizations that were studied were mainly located in the western countries, AKL organization New Zealand based, and MEL organization Australian based. The MEL organization had teams in India, but lack of direct contact and the time-boxed nature of this research study, restricted me to get perspectives from the Indian site. Therefore, organizations specifically from Asian countries such as India could be investigated in a future study, as cultural values typically vary between Western and Asian countries (Brockmann & Thaumuller, 2009; Ravishankar et al., 2012) that could provide improved understanding and further refinement of the findings of this research study.

This study provides a thorough understanding of the requirements prioritization process in SADSD. In order to attain that, a novel conceptual framework has been developed that captures the life cycle activities of prioritization of requirements, from an idea to operation. It is synthesized from the two scaled agile distributed organizations via employing interpretive exploratory multi case study research. The conceptual framework may therefore be applicable in a similar context to those which were studied in this research. However, there is still a possibility that results might concur in different contexts such as large government organizations that abide by much stricter regulations than private organizations (Mishra & Weistroffer, 2008). Therefore, further research could be undertaken to validate and enhance the conceptual framework of requirements

prioritization in SADSD into different contexts, organizations, and assess its applicability (e.g., positivist case study research).

Most of the available literature is mainly devoted to applying Agile practices at the team level. This research study aimed to fill this gap via providing understanding of functional set-up and units, artefacts, processes, practices, techniques and roles specifically needed beyond the team level to perform prioritization of requirements. The conceptual framework that has been developed as an outcome of this research study, could be employed to extend and/or merge existing requirements prioritization frameworks that are specifically built for applying at the team level.

9.6 Final statement

The research reported in this research study has provided understanding of the requirements prioritization process in SADSD, which is an under-researched area as impacted by the relatively novel phenomenon of adoption of Agile methods in large scale distributed software development. It is hoped that contributions provided by this research study would help guide the practitioners in terms of functional set-up and units, artefacts, processes, practices, techniques and roles specifically needed beyond the team level to perform prioritization of requirements in SADSD. Research recommendations for the researchers have also been given to replicate and/or enhance the findings of this research study, by conducting further research in the similar context and/or different contexts.

References

AbdElazim, K., Moawad, R., & Elfakharany, E. (2020). A Framework for Requirements Prioritization Process in Agile Software Development Experimental validation of predicted cancer genes using FRET A Framework for Requirements Prioritization Process in Agile Software Development. *Journal of Physics: Conference Series*, *1454*, 1–11. https://doi.org/10.1088/1742-6596/1454/1/012001

Adolph, S., Hall, W., & Kruchten, P. (2011). Using grounded theory to study the experience of software development. *Empirical Software Engineering*, *16*(4), pages487–513. https://doi.org/10.1007/s10664-010-9152-6

Alberto Espinosa, J., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, *24*(1), 135–169. https://doi.org/10.2753/MIS0742-1222240104

Alenljung, B., & Persson, A. (2008). Portraying the practice of decision-making in requirements engineering: A case of large scale bespoke development. *Requirements Engineering*, *13*(4), 257–279. https://doi.org/10.1007/S00766-008-0068-2/FIGURES/6

Alqudah, M., & Razali, R. (2016). A Review of Scaling Agile Methods in Large Software Development. *International Journal on Advanced Science, Engineering and Information Technology*, *6*, 828–837. https://doi.org/10.18517/ijaseit.6.6.1374

Alsaqaf, W., Daneva, M., & Wieringa, R. (2017). Quality Requirements in Large-Scale Distributed Agile Projects:A Systematic Literature Review. *Requirements Engineering: Foundation for Software Quality*, 219–234. https://doi.org/10.1007/978-3-319-54045-0_17

Alsaqaf, W., Daneva, M., & Wieringa, R. (2018). Understanding challenging situations in agile quality requirements engineering and their solution strategies: insights from a case study. *26th International Requirements Engineering Conference (RE)*, 274–285. https://doi.org/10.1109/RE.2018.00035

Ambler, S. (2013). *11 strategies for dealing with technical debt*. http://disciplinedagiledelivery.com/technical-debt/

Ambler, S. (2014). *Managing requirements dependencies between agile teams*. https://disciplinedagiledelivery.com/managing-requirements-dependencies-between-agile-teams/

Ambler, S. (2018). *Disciplined agile program management The product owner team*. http://disciplinedagiledelivery.com/disciplined-agile-program-management-the-product-owner-team/

Ambler, S. W., & Lines, M. (2012). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press.

Anand, R. V., & Dinakaran, M. (2017). Handling stakeholder conflict by agile requirement prioritization using Apriori technique. *Computers & Electrical Engineering*, *61*, 126–136. https://doi.org/10.1016/J.COMPELECENG.2017.06.022

Aurum, A., & Wohlin, C. (2005). Requirements Engineering: Setting the Context. In *Engineering and Managing Software Requirements* (pp. 1–15). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-28244-0_1

Badampudi, D., Fricker, S. A., & Moreno, A. M. (2013). Perspectives on Productivity and Delays in Large-Scale Agile Projects. *Agile Processes in Software Engineering and Extreme Programming*, 180–194. https://doi.org/10.1007/978-3-642-38314-4_13

Barbour, R. S. (2001). Checklists for improving rigour in qualitative research: a case of the tail wagging the dog? *BMJ*, *322*(7294), 1115–1117. https://doi.org/10.1136/BMJ.322.7294.1115

Bass, J. M. (2015). How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering*, *20*(6), 1525–1557. https://doi.org/10.1007/s10664-014-9322-z

Bass, J. M. (2016). Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology*, *75*, 1–16. https://doi.org/10.1016/j.infsof.2016.03.001

Batra, & Dinesh. (2020). Research Challenges and Opportunities in Conducting Quantitative Studies on Large-Scale Agile Methodology. *Journal of Database*

Management, *31*(2), 64–73. https://doi.org/10.4018/JDM.2020040104

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Berander, P., & Andrews, A. (2005). Requirements Prioritization. In *Engineering and Managing Software Requirements* (pp. 69–94). Springer. https://doi.org/10.1007/3-540-28244-0_4

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). A case study on benefits and side-effects of agile practices in large-scale requirements engineering. *AREW '11: Proceedings of the 1st Workshop on Agile Requirements Engineering*, 1–5. https://doi.org/10.1145/2068783.2068786

Boehm, B. (2006). Some Future Trends and Implications for Systems and Software Engineering Processes. *Systems Engineering*, *9*(1), 1–19. https://doi.org/10.1002/sys.20044

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, *21*(5), 61–72. https://doi.org/10.1109/2.59

Brockmann, P. S., & Thaumuller, T. (2009). Cultural Aspects of Global Requirements Engineering: An Empirical Chinese-German Case Study. *IEEE International Conference on Global Software Engineering*, 353–357. https://doi.org/10.1109/ICGSE.2009.55

Chua, W. F. (1986). Radical Developments in Accounting Thought. *The Accounting Review*, *61*(4), 601–632.

Clear, T., & MacDonell, S. G. (2011). Understanding technology use in global virtual teams: Research methodologies and methods. *Information and Software Technology*, *53*(9), 994–1011. https://doi.org/10.1016/J.INFSOF.2011.01.011

Conboy, K., & Carroll, N. (2019). Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. *IEEE Software*, *36*(2), 44–50. https://doi.org/10.1109/MS.2018.2884865

Creswell & Dana, J. W., & Miller, L. L. (2000). Determining Validity in Qualitative Inquiry. *Theory Into Practice*, *39*(3), 124–130. https://doi.org/10.1207/s15430421tip3903_2

Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications.

Cross, R., Ernst, C., & Pasmore, B. (2013). A bridge too far? How boundary spanning networks drive organizational change and effectiveness. *Organizational Dynamics*, *42*(2), 81–91. https://doi.org/10.1016/j.orgdyn.2013.03.001

Dalton, J. (2018). Big Room Planning / Release Zero. In *Great Big Agile: An OS for Agile Leaders* (pp. 135–137). Apress. https://doi.org/10.1007/978-1-4842-4206-3_17

Damian, D., Linåker, J., Johnson, D., Clear, T., & Blincoe, K. (2021). Challenges and Strategies for Managing Requirements Selection in Software Ecosystems. *IEEE Software*, *38*(6), 76–87. https://doi.org/10.1109/MS.2021.3105044

Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., & Wieringa, R. (2013). Agile requirements prioritization in large-scale outsourced system. *Journal of Systems and Software*, *86*(5), 1333–1353. https://doi.org/10.1016/J.JSS.2012.12.046

Daniels, M., Cajander, Å., Clear, T., & McDermott, R. (2015). Collaborative technologies in global engineering : New competencies and challenges. *International Journal of Engineering Education*, *31*(1), 267–281. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-240569

Dawson, R., Bones, P., Oates, B. J., Brereton, P., Azuma, M., & Jackson, M. Lou. (2004). Empirical Methodologies in Software Engineering. *Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04)*, 52–58. https://doi.org/10.1109/STEP.2003.9

De Lucia, A., & Qusef, A. (2010). Requirements Engineering in Agile Software Development. *Journal of Emerging Technologies in Web Intelligence*, *2*(3), 212–220. https://doi.org/10.4304/jetwi.2.3.212-220

Digital.ai. (2021). *15th State Of Agile Report*. https://digital.ai/resource-center/analyst-reports/state-of-agile-report

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87–108.

Dingsøyr, T., & Moe, N. B. (2014). Towards Principles of Large-Scale Agile Development. *International Conference on Agile Software Development*, 1–8. https://doi.org/10.1007/978-3-319-14358-3_1

Dingsøyr, T., Moe, N. B., Fægri, T. E., & Seim, E. A. (2018). Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*, *23*(1), 490–520. https://doi.org/10.1007/s10664-017-9524-2

Dumitriu, F., Mesnita, G., & Radu, L.-D. (2019). Challenges and Solutions of Applying Large-Scale Agile at Organizational Level. *Informatica Economică*, *23*(3), 61–71. https://doi.org/: 10.12948/issn14531305/23.3.2019.06

Dybå, T., & Dingsøyr, T. (2008). *Empirical studies of agile software development: A systematic review*. https://doi.org/10.1016/j.infsof.2008.01.006

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In *Guide to Advanced Empirical Software Engineering* (pp. 285–311). Springer, London. https://doi.org/10.1007/978-1-84800-044-5_11

Eckstein, J. (2014). Architecture in Large Scale Agile Development. *International Conference on Agile Software Development*, 21–29. https://doi.org/10.1007/978-3-319-14358-3_3

Elghariani, K., & Kama, N. (2016). Review on Agile Requirements Engineering Challenges. *3rd International Conference On Computer And Information Sciences (ICCOINS)*, 507–512. https://doi.org/978-1-5090-2549-7/16

Evbota, F., Knauss, E., & Sandberg, A. (2016). Scaling up the planning game: Collaboration challenges in large-scale agile product development. *International Conference on Agile Software Development*, 28–38.

Fitzgerald, B., & Howcroft, D. (1998). Competing Dichotomies in IS Research and Possible Strategies for Resolution. *ICIS 1998 Proceedings*, 155–164.

http://aisel.aisnet.org/icis1998/14

Flyvbjerg, B. (2006). Five misunderstandings about case-study research. *Qualitative Inquiry*, *12*(2), 219–245. https://doi.org/10.1177/1077800405284363

Fossey, E., Harvey, C., McDermott, F., & Davidson, L. (2016). Understanding and Evaluating Qualitative Research∗. *Australian & New Zealand Journal of Psychiatry*, *36*(6), 717–732. https://doi.org/10.1046/J.1440-1614.2002.01100.X

Gable, G. G. (1994). Integrating case study and survey research methods: an example in information systems. *European Journal of Information Systems*, *3*(2), 112–126. https://doi.org/10.1057/EJIS.1994.12

Garousi, V., Felderer, M., & Mäntylä, M. V. (2016). The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 1–6. https://doi.org/10.1145/2915970.2916008

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, *106*, 101–121. https://doi.org/10.1016/j.infsof.2018.09.006

Gat, I. (2006). How BMC is scaling agile development. *AGILE 2006 (AGILE'06)*, 315–320. https://doi.org/10.1109/AGILE.2006.33

Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory*. Mill Valley.

Glass, R. L., Vessey, I., & Ramesh, V. (2002). Research in software engineering: an analysis of the literature. *Information and Software Technology*, *44*(8), 491–506. https://doi.org/10.1016/S0950-5849(02)00049-6

Glinz, M. (2017). *A glossary of requirements engineering terminology. Version 1.7. International Requirements Engineering Board (IREB)*. https://expleoacademy.com/dach/wp-content/uploads/sites/4/2020/11/ireb_cpre_glossary_17.pdf

Green, J. L., Camilli, G., & Elmore, P. B. (2012). *Handbook of complementary methods*

in education research. Routledge.

Grewal, H., & Maurer, F. (2007). Scaling Agile Methodologies for Developing a Production Accounting System for the Oil Gas Industry. *Agile 2007 (AGILE 2007)*, 309–315. https://doi.org/10.1109/AGILE.2007.50

Gunyho, G., & Plaza, J. G. (2011). Evolution of Longer-Term Planning in a Large Scale Agile Project F-Secure's Experience. *Agile Processes in Software Engineering and Extreme Programming*, 306–315. https://doi.org/10.1007/978-3-642-20677-1_22

Gupta, A., Mattarelli, E., Seshasai, S., & Broschak, J. (2009). Use of collaborative technologies and knowledge sharing in co-located and distributed teams: Towards the 24-h knowledge factory. *The Journal of Strategic Information Systems*, *18*(3), 147–161. https://doi.org/10.1016/J.JSIS.2009.07.001

Gupta, R. K., Manikreddy, P., & Arya, K. C. (2017). Pragmatic Scrum Transformation: Challenges, Practices & Impacts During the Journey A case study in a multi-location legacy software product development team. *Proceedings of the 10th Innovations in Software Engineering Conference*, 147–156. https://doi.org/10.1145/3021460.3021478

Gustavsson, T. (2017). Assigned roles for Inter-team coordination in Large-Scale Agile Development: a literature review. *Proceedings of the XP2017 Scientific Workshops*, 1–5. https://doi.org/10.1145/3120459.3120475

Guyot, C. (n.d.). *SAFe case study: Kantar Retail Virtual Reality*. Retrieved December 1, 2018, from https://www.scaledagileframework.com/kantar-retail-case-study/

Hannay, J. E., & Benestad, H. C. (2010). Perceived productivity threats in large agile development. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–10. https://doi.org/10.1145/1852786.1852806

Heikkilä, V., Rautiainen, K., & Jansen, S. (2010). A Revelatory Case Study on Scaling Agile Release Planning. *36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 289–296. https://doi.org/10.1109/SEAA.2010.37

Heikkilä, V. T., Damian, D., Lassenius, C., & Paasivaara, M. (2015). A Mapping Study on Requirements Engineering in Agile Software Development. *41st Euromicro*

Conference on Software Engineering and Advanced Applications, 199–207. https://doi.org/10.1109/SEAA.2015.70

Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering*, *22*(6), 2892–2936. https://doi.org/10.1007/s10664-016-9491-z

Herrmann, A., & Daneva, M. (2008). Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. *16th IEEE International Requirements Engineering Conference*, 125–134. https://doi.org/10.1109/RE.2008.48

Hobbs, B., & Petit, Y. (2017). Agile Methods on Large Projects in Large Organizations. *Project Management Journal*, *48*(3), 3–19. https://doi.org/10.1177/875697281704800301

Hoda, R. (2011). *Self-Organizing Agile Teams: A Grounded Theory* [Victoria University of Wellington]. http://hdl.handle.net/10063/1617

Hoffmann, M., Kühn, N., Weber, M., & Bittner, M. (2004). Requirements for Requirements Management Tools. *12th IEEE International Requirements Engineering Conference (RE'04)*, 301–308. https://doi.org/10.1109/ICRE.2004.1335687

Holmstrom, H., Conchuir, E. O., Agerfalk, P. J., & Agerfalk, P. J. (2006). Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance. *International Conference on Global Software Engineering (ICGSE'06)*, 3–11. https://doi.org/10.1109/ICGSE.2006.261210

Hussain, N. (2018). *Requirements engineering for globally distributed teams using scaled agile framework.* [Aalto University, Finland]. https://aaltodoc.aalto.fi/handle/123456789/32500

Hussain, W., & Clear, T. (2014). Spreadsheets as Collaborative Technologies in Global Requirements Change Management. *9th International Conference on Global Software Engineering*, 74–83. https://doi.org/10.1109/ICGSE.2014.25

Hussey, J., & Hussey, R. (1997). *Business Research: A Practical Guide for*

Undergraduate and Postgraduate Students. Macmillan Business.

Hutchisona, A. J., Johnstonb, L. H., & Breckona, J. D. (2009). Using QSR-NVivo to facilitate the development of a grounded theory project: an account of a worked example. *International Journal of Social Research Methodology*, *13*(4), 283–302. https://doi.org/10.1080/13645570902996301

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929. https://doi.org/10.1016/j.chb.2014.10.046

Jain, R., Mohan, K., & Ramesh, B. (2014). Situated Boundary Spanning: An Empirical Investigation of Requirements Engineering Practices in Product Family Development. *ACM Transactions on Management Information System*, *5*(3), 1–29. https://doi.org/10.1145/2629395

Jha, M. M., Vilardell, R. M. F., & Jai, N. (2016). Scaling agile scrum software development: providing agility and quality to platform development by reducing time to market. *11th International Conference on Global Software Engineering (ICGSE)*, 84–88. https://doi.org/10.1109/ICGSE.2016.24

Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, *30*(10), 1–24. https://doi.org/10.1002/smr.1954

Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., & Kanagwa, B. (2017). Requirements Engineering Challenges in Large-Scale Agile System Development. *25th International Requirements Engineering Conference (RE)*, 352–361. https://doi.org/10.1109/RE.2017.60

Khalid, H., Ahmed, M., Sameer, A., Arif, F., & others. (2015). Systematic literature review of agile scalability for large scale projects. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *6*(9), 63–75.

Kim, N., & Cho, H. (2018). *Scaling at Food-tech Startup: Transformation challenges, lessons learned and growth*. https://www.agilealliance.org/resources/experience-reports/scaling-at-food-tech-startup-transformation-challenges-lessons-learned-

and-growth/

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*. https://doi.org/10.1.1.117.471

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly: Management Information Systems*, *23*(1), 67–94. https://doi.org/10.2307/249410

Kotonya, G., & Sommerville, I. (1998). *Requirements engineering – processes and techniques*. John Wiley & Sons.

Ktata, O., & Lévesque, G. (2009). Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering*, 59–66. https://doi.org/10.1145/1557626.1557636

Lal, R., & Clear, T. (2017). Scaling Agile at the Program Level in an Australian Software Vendor Environment: A Case Study. *ACIS 2017 Proceedings*, 1–12. https://doi.org/aisel.aisnet.org/acis2017/81

Lal, R., & Clear, T. (2018). Enhancing product and service capability through scaling agility in a global software vendor environment. *Proceedings of the 13th International Conference on Global Software Engineering*, 54–63. https://doi.org/10.1145/3196369.3196378

Lang, M., & Duggan, J. (2001). A Tool to Support Collaborative Software Requirements Management. *Requirements Engineering*, *6*, 161–172. https://doi.org/10.1007/s007660170002

Larsson, S. (2009). A pluralist view of generalization in qualitative research. *International Journal of Research & Method in Education*, *32*(1), 25–38. https://doi.org/10.1080/17437270902759931

Lázaro, M., & Marcos, E. (2006). An Approach to the Integration of Qualitative and Quantitative Research Methods in Software Engineering Research. *Philisophiocal Foundations on Information Systems Engineering*, *240*, 757–764.

Leffingwell, D. (2007). Mastering the iteration: An agile white paper. *Rally Software*

Development Corporation, 3–14.

Leffingwell, D. (2011). *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley.

Lehtola, L., Kauppinen, M., & Kujala, S. (2004). Requirements Prioritization Challenges in Practice. In *Product Focused Software Process Improvement* (pp. 497–508). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-24659-6_36

Lethbridge, T. C., Sim, S. E., & Singer, J. (2005). Studying Software Engineers: Data Collection Techniques for Software Field Studies. *Empirical Software Engineering*, *10*(3), 311–341. https://doi.org/10.1007/s10664-005-1290-x

Macaulay, L. A. (1996). *Requirements Engineering*. Springer.

Maxwell, J. A. (2012). *Qualitative Research Design: An Interactive Approach*. SAGE.

McGrath, J. E., & Brinberg, D. (1983). External Validity and the Research Process: A Comment on the Calder/Lynch Dialogue. *Journal of Consumer Research*, *10*(1), 115–124. https://www.jstor.org/stable/2488862

Medeiros, J. D. R. V., Alves, D. C. P., Vasconcelos, A., Carla, & Wanderley, S. E. (2015). Requirements Engineering in Agile Projects: A Systematic Mapping based in Evidences of Industry. *CibSE*, 1–15.

Mishra, D., & Mishra, A. (2011). Complex software project development: agile methods adoption. *Journal of Software Maintenance and Evolution: Research and Practice*, *23*(8), 549–564. https://doi.org/10.1002/smr.528

Mishra, S., & Weistroffer, H. R. (2008). Issues with Incorporating Regulatory Compliance into Agile Development: A Critical Analysis. *SAIS 2008 Proceedings*, 6.

Moe, N. B., & Dingsøyr, T. (2017). Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at XP2017. *Proceedings of the XP2017 Scientific Workshops*, 1–4. https://doi.org/10.1145/3120459.3120474

Mutschler, R., Trost, O., & Crepin, J. (2020). Agile Methodologies in the Development of Automotive Embedded Software. *ATZelectronics Worldwide*, *15*(7), 44–49.

https://doi.org/10.1007/s38314-020-0225-z

Myers, M. D. (2009). *Qualitative research in business and management*. SAGE.

Ngo-The, A., & Ruhe, G. (2005). Decision Support in Requirements Engineering. In *Engineering and Managing Software Requirements* (pp. 267--286). Springer. https://doi.org/10.1007/3-540-28244-0_12

Orlikowski, W. J., & Baroudi, J. J. (1991). Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research*, *2*(1), 1–28. https://doi.org/10.1287/ISRE.2.1.1

Paasivaara, M. (2017). Adopting SAFe to scale agile in a globally distributed organization. *12th International Conference on Global Software Engineering (ICGSE)*, 36–40. https://doi.org/10.1109/ICGSE.2017.15

Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2014). Towards Rapid Releases in Large-Scale XaaS Development at Ericsson: A Case Study. *9th International Conference on Global Software Engineering*, 16–25. https://doi.org/10.1109/ICGSE.2014.22

Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2018). Large-scale agile transformation at Ericsson: a case study. *Empirical Software Engineering*, *23*(5), 2550–2596. https://doi.org/10.1007/s10664-017-9555-8

Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a Large Globally Distributed Organization: A Case Study. *11th International Conference on Global Software Engineering (ICGSE)*, 74–83. https://doi.org/10.1109/ICGSE.2016.34

Paasivaara, M., & Lasssenius, C. (2004). Collaboration Practices inGlobal Inter-organizationalSoftware DevelopmentProjects. *Software Process: Improvement and Practice*, *8*(4), 183–199. https://doi.org/10.1002/spip.187

Paetsch, F., Eberlein, D. A., & Maurer, D. F. (2003). Requirements Engineering and Agile Software Development. *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)*, 1–6. https://doi.org/10.1109/ENABL.2003.1231428

Pavlichenko, I. (2018). *Eliminate dependencies, don't manage them*.

https://www.scrum.org/resources/blog/eliminate-dependencies-dont-manage-them

Petersen, K., & Wohlin, C. (2009). Context in industrial software engineering research. *Third International Symposiumm on Empirical Software Engineering and Measurement*, 401–404. https://doi.org/10.1109/ESEM.2009.5316010

Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. *Empirical Software Engineering*, *15*(6), 654–693. https://doi.org/10.1007/s10664-010-9136-6

Popli, R., Chauhan, N., & Sharma, H. (2014). Prioritising user stories in agile environment. *International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 515–519. https://doi.org/10.1109/ICICICT.2014.6781336

Putta, A., Paasivaara, M., & Lassenius, C. (2018). Adopting scaled agile framework (SAFe): a multivocal literature review. *Proceedings of the 19th International Conference on Agile Software Development: Companion*, 1–4. https://doi.org/10.1145/3234152.3234164

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, *20*, 449–480. https://doi.org/10.1111/j.1365-2575.2007.00259.x

Rautiainen, K., von Schantz, J., & Vähäniitty, J. (2011). Supporting scaling agile with portfolio management: Case paf. com. *2011 44th Hawaii International Conference on System Sciences*, 1–10. https://doi.org/10.1109/HICSS.2011.390

Ravishankar, M. N., Pan, S. L., & Myers, M. D. (2012). Information technology offshoring in India: a postcolonial perspective. *European Journal of Information Systems*, *22*(4), 387–402. https://doi.org/10.1057/ejis.2012.32

Razavi, A. M., & Ahmad, R. (2014). Agile development in large and distributed environments: A systematic literature review on organizational, managerial and cultural aspects. *8th Malaysian Software Engineering Conference (MySEC)*, 216–221. https://doi.org/10.1109/MySec.2014.6986017

Reinikainen, R. (n.d.). *SAFe case study: Telia Finland*. Retrieved December 1, 2018,

from https://www.scaledagileframework.com/case-study-telia-finland/

Rolland, K. H. (2015). “Desperately” seeking research on agile requirements in the context of large-scale agile projects. *Scientific Workshop Proceedings of the XP2015*, 1–6. https://doi.org/10.1145/2764979.2764984

Roman, G. de C. (2016). *Characterizing the presence of agility in large-scale agile software development* [Pontifical Catholic University of Rio de Janeiro, Brazil]. https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US20100183655.pdf

Roopa, M. S., Mani, V. S., & Sankarasubbiah, C. (2017). Usable Software at the End of Each Takt. A Milestone in the Lean Transformation of a Globally Distributed Software Development Team. *12th International Conference on Global Software Engineering (ICGSE)*, 116–120. https://doi.org/10.1109/ICGSE.2017.9

Rosen, M. (1991). COMING TO TERMS WITH THE FIELD: UNDERSTANDING AND DOING ORGANIZATIONAL ETHNOGRAPHY*. *Journal of Management Studies*, *28*(1), 1–24. https://doi.org/10.1111/J.1467-6486.1991.TB00268.X

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Scheerer, A., Bick, S., Hildenbrand, T., & Heinzl, A. (2015). The effects of team backlog dependencies on agile multiteam systems: A graph theoretical approach. *28th Hawaii International Conference on System Sciences*, 5124–5132. https://doi.org/10.1109/HICSS.2015.606

Schnitter, J., & Mackert, O. (2010). Large-scale agile software development at SAP AG. *Evaluation of Novel Approaches to Software Engineering*, 209–220. https://doi.org/10.1007/978-3-642-23391-3_15

Schön, E. M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards & Interfaces*, *49*, 79–91. https://doi.org/10.1016/J.CSI.2016.08.011

Schwaber, K. (2004). *Agile Project Management with Scrum*. Microsoft Press.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Trans. Software Eng*, *25*(4), 557–572. https://doi.org/10.1109/32.799955

Sekitoleko, N., Evbota, F., Knauss, E., Sandberg, A., Chaudron, M., & Olsson, H. H. (2014). Technical Dependency Challenges in Large-Scale Agile Software Development. *Agile Processes in Software Engineering and Extreme Programming*, 46–61. https://doi.org/10.1007/978-3-319-06862-6_4

Shrivastava, S. V., & Rathod, U. (2017). A risk management framework for distributed agile projects. *Information and Software Technology*, *85*, 1–15. https://doi.org/10.1016/j.infsof.2016.12.005

Sinha, V., Sengupta, B., & Chandra, S. (2006). Enabling Collaboration in Distributed Requirements Management. *IEEE Software*, *23*(5), 52–61. https://doi.org/10.1109/MS.2006.123

Sommerville, I. (2005). Integrated requirements engineering: a tutorial. *IEEE Software*, *22*(1), 16–23. https://doi.org/10.1109/MS.2005.13

Sommerville, I. (2009). *Software Engineering*. Pearson.

Sutherland, J. (2006). *The Definitive Guide to the Scrum@Scale Framework*. https://www.scrumatscale.com/scrum-at-scale-guide-online/

Tripathi, N., Rodriguez, P., Ahmad, M. O., & Oivo, M. (2015). Scaling Kanban for software development in a multisite organization: Challenges and potential solutions. *Agile Processes in Software Engineering and Extreme Programming*, 178–190. https://doi.org/10.1007/978-3-319-18612-2_15

Usman, M., Britto, R., Damm, L.-O., & Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, *99*, 21–40. https://doi.org/10.1016/J.INFSOF.2018.02.009

Van der Heijden, A., Broasca, C., & Serebrenik, A. (2018). An empirical perspective on security challenges in large-scale agile software development. *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–4. https://doi.org/10.1145/3239235.3267426

VersionOne. (2017). *12th state of agile report*. https://www.qagile.pl/wp-

content/uploads/2018/04/versionone-12th-annual-state-of-agile-report.pdfeport/473508

Viswanath, U. (2016). Lean Transformation: Adapting to the change, factors for success and lessons learnt during the journey: A case study in a multi location software product development team. *Proceedings of the 9th India Software Engineering Conference*, 156–162. https://doi.org/10.1145/2856636.2856657

Vlietland, J., & Van Vliet, H. (2015). Towards a governance framework for chains of Scrum teams. *Information and Software Technology*, *57*, 52–65. https://doi.org/10.1016/J.INFSOF.2014.08.008

Wildt, D., & Prikladnicki, R. (2010). Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report. In *Agility Across Time and Space* (pp. 31–46). Springer. https://doi.org/10.1007/978-3-642-12442-6_3

Yin, R. K. (2003). *Case Study Research: Design and Methods*. SAGE.

Yin, R. K. (2009). *Case Study Research: Design and Methods*. SAGE.

Zamudio, L., Aguilar, J. A., Tripp, C., & Misra, S. (2017). A Requirements Engineering Techniques Review in Agile Software Development Methods. *International Conference on Computational Science and Its Applications*, 683–698. https://doi.org/10.1007/978-3-319-62404-4

Zowghi, D., & Damian, D. E. (2003). Requirements Engineering Challenges in Multi-site Soware Development Organizations. *Requirements Engineering*, *8*(1), 149–160.

Glossary

Strategic themes: requirements are typically very high business level requirements, generally unchanged for up to one year or can go above year. Decision-making on strategic themes are typically performed at the portfolio level. Scope, timeframe, and nomenclature as ***strategic theme*** typically coincides in the studied cases, in literature such as the Leffingwell (2011) hierarchical requirements model, and in the scaling agile frameworks such as SAFe, DAD.

High-level requirements (HLRs): requirements that typically require significant effort and/or investment such as 2-3 months of work are termed as HLRs. HLRs typically require a formal decision-making process i.e., initial decision making at the domain level, and final validation and communication at the portfolio level.

Scope, timeframe, and nomenclature of HLRs is greatly varied in the studied cases, ***epics*** in MEL, ***features*** in AKL as well as in the literature such as the Leffingwell (2011) hierarchical requirements model and in scaling agile frameworks such as SAFe and DAD, but due to the commonality in the process to make decisions on their priority in the studied case organizations I have termed them as HLRs.

Intermediate-level requirements (ILRs): requirements that are typically decomposed from the HLRs to implement them via a short development cycle of 2 weeks and by a single delivery team are termed as ILRs. Decision-making on ILRs is typically performed at the domain level.

Scope, timeframe, nomenclature of ILRs is greatly varied in the studied cases, ***user stories*** in MEL, ***features*** in AKL as well as in the literature such as the Leffingwell (2011) hierarchical requirements model and in scaling agile frameworks such as SAFe and DAD, but due to the commonality in the process to make decisions on their priority in the studied case organizations I have termed them as ILRs.

Low-level requirements (LLRs): requirements that are typically detailed enough and can be implemented via a single member of a delivery team are termed as LLRs. Decision-making on LLRs is typically performed at the team level. Scope, timeframe, and nomenclature of LLRs as a ***user story*** typically coincides in the studied cases, in literature such as the Leffingwell (2011) hierarchical requirements model and in scaling agile frameworks such as SAFe, DAD.

Backlog: backlog contains a priority list of requirements that occur at various levels with varying details. The notion of backlog greatly varied across the levels such as:

A backlog that is formed at the portfolio level is termed as ***Portfolio backlog*** in the studied case organizations. Also, this is commonly mentioned as a ***Portfolio backlog*** in the literature (Heikkila, 2017) as well as in the scaling agile frameworks such as SAFe, DAD.

A backlog that is formed at the domain level varied in the studied case organization, a ***Workitems*** **list** in MEL which is an adaptation from the DAD framework, and a ***Product backlog*** in AKL which is an adaptation of basic Agile methods such as in the Scrum framework.

A backlog that is constructed at the team level is termed as a ***Team backlog*** in the studied case organizations. Also, commonly mentioned as a ***Team backlog*** in the literature as well as in the scaling agile frameworks such as SAFe.

Portfolio level: portfolio level is the highest decision-making level, where primarily the business goals that connect with an organization's overall business strategy, are decided and prioritized. Portfolio level is typically a ***steering and validation level*** in terms of making decisions on the priority of requirements such as strategic themes and HLRs.

Domain level: Domain level is typically a ***core decision making level*** in terms of making decisions on the priority of requirements in SADSD. It is typically a technical product area where teams are generalizing specialists who work autonomously end-to-end to solve business problems termed as ***HLRs*** in this research study.

Domain level is commonly termed as the Program level in the literature and in the scaling agile frameworks such as DAD and SAFe. The terminology is locally adapted in the studied organizations regardless of the scaling agile frameworks that are adopted by the studied case organizations, *segment* in MEL, *tribe* in AKL (which is an adaptation of the Spotify framework). But I have adopted the terminology "Domain level" due to the commonality in the process employed by the studied case organizations to make decisions on the priority of requirements at this level.

Team level: team level is the lowest decision-making level where requirements represented as LLRs, that typically require a small amount of effort such as 2 weeks, are decided and prioritized. Team level is also termed as project level in the literature as well as in the scaling agile frameworks such as DAD.

Inter-iteration prioritization: where typically the abstract level requirements such as enterprise-wide requirements prioritization decisions are formed across the span of more than one iteration. However, inter-iteration prioritization partially occurred during decision-making on the action level requirements such as team level.

Intra-iteration prioritization: where typically the action level requirements such as team specific requirements prioritization decisions are formed within the span of an iteration. However, intra-iteration prioritization occurred during decision-making on the abstract level requirements.

Boundary spanning: Boundary spanning is the act of bringing together two or more groups of people who are typically separated by location, that is, locally distributed and/or globally distributed, or separated by hierarchy, or a function.

Portfolio management/Governance forum: a cross-functional decision-making team, that typically consists of most senior enterprise business and technology stakeholders. Designations of the members who constitute the ***portfolio management/governance forum*** greatly varied in the studied cases (e.g., SVP architecture owns strategic technical needs in MEL, CTO owns strategic technical needs in AKL), as well as in the literature (in an empirical study conducted by Heikkila (2017), node architect owns strategic technical needs), and in the scaling agile frameworks such as SAFe, DAD (titled as enterprise architect).

Product manager/Senior PO: a person who is typically a member of product management owns the responsibility of evaluating and identifying the potential HLRs. Their designation in the studied cases varied, *product manager* in MEL, *senior PO* in AKL as well as in the scaling agile frameworks such as Scrum @scale (where they own the title of chief PO), DAD (where they own the title of product manager).

Product Owner (PO): a person who owns the responsibility of requirements prioritization within the delivery team typically owns the title of PO in the studied cases, as well as being a commonly identified role in the literature, and in the scaling agile frameworks such as DAD, Scrum @scale, SAFe.

Domain owner: a person who typically represents the HLRs that are emerging from their respective domain(s) during the formal decision-making discussions i.e., at the portfolio level. The designation greatly varied in the studied cases, ***Segment leader*** in MEL, ***General managers*** in AKL as well as in scaling agile frameworks such as ***business***

owners in SAFe, but due to the commonality in their responsibility to make decision on the requirements priority I have titled them as ***domain owners***.

Architecture Owner (AO): a person who is typically a member of architecture team and owns the responsibility of technical needs within the delivery teams. The presence and title of AO within the delivery team varied in the studied cases (embedded as AO in MEL, technical business analyst in AKL) and some of the scaling agile frameworks such as SAFe (appointed as solution architect on a need basis), DAD (embedded as AO within the delivery team).

Program AO/Solution Architect: a person who is typically a member of architecture team and owns the responsibility of technical needs within the domain. Their designation within the domain greatly varied in the studied cases (program AO in MEL, solution architect in AKL) as well as in the scaling agile frameworks such as Scrum @scale, DAD (where they own the title of chief AO), SAFe (where they own the title of solution architect).

Team leader (TL)/scrum master (SM): a person who owns the responsibility of resources and delivery within the delivery team, TL in MEL, SM in AKL which is an adaptation of the role defined in DAD, and Scrum @ scale framework respectively.

Tribe lead/Director of Engineering: a person who typically owns the responsibility of resources and delivery within the domain. Their designation within the domain greatly varied in the studied cases (*director of engineering* in MEL, *tribe lead* in AKL) and some of the scaling agile frameworks such as Scrum @scale (where they own the title of S*crum of Scrums Master - SOSM*).

Appendices

Appendix A: Permission to Access Sheet (Employer)

Date Information Sheet Produced: 14 October 2019

Project Title: Requirements Prioritization in Global Scaled Agile Software Development

An Invitation

Dear employer,

My name is Priyanka Antil and I am a doctoral student in the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology, under the supervision of Assoc Professor Tony Clear and Dr. Ramesh Lal. I am conducting research investigating the requirements prioritization process in global scaled agile software development.

I would like to invite your organization to take part in my doctoral research. I would like to learn from your organization. As expert practitioners in this area, you are invited to share your perspective and contribute to the body of knowledge in this area.

Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. None of the participants is identified nor will the information gathered be used to hamper, hinder or harm your career. Moreover, this research does not aim to collect commercially sensitive information of the organisation and its members.

The following questions and answers are intended to address the most common questions that the participants may ask about this particular research project. If you need further information, feel free to contact the researcher, Priyanka Antil. My contact details can be found at the end of this document. It is recommended that you use e-mail to reach me.

Project Description

Current trends for developing software include the need to scale development processes to suit global market needs with distributed and agile development being more prevalent. Increasing impact from emerging technologies such as cloud computing further enables

immediate global delivery of software products and services. These scaling processes move software development more strongly into alignment with the current business strategy and planning processes. The demand for both global and local market expertise plus the ability to deliver at volume and scale, require development at multiple global sites, driven by a common culture and engineering processes. Thus, the nature of requirements engineering (RE) in a software development environment has to evolve in response to these changes.

The purpose of this proposed research is to develop an in-depth understanding of the prioritization of requirements in global scaled agile software development. This study will identify the roles (decision makers) that make decisions on the priority of requirements across scaling agile levels (e.g. Portfolio level, Program level, Project level). This study will be a fine-grained study of the processes and artefacts for the prioritization of requirements across scaling agile levels. Moreover, this study will identify the challenges in requirements prioritization process in global scaled agile software development environment, the innate tensions faced, and create an effective set of strategies for surmounting requirements prioritization challenges and confirm them with literature and focus group(s).

Furthermore, a conceptual framework will be constructed during this study that will create knowledge on the underlying philosophy and the related structures (functional set-up and units, artefacts, processes, practices, techniques and roles) driving the requirements prioritization process in a global scaled agile software development environment. Therefore, the goal of this study is to provide an understanding of the requirements prioritization process across scaling agile levels (e.g. portfolio level, program level, and project level).

Another purpose of this research is to contribute to the completion of my PhD degree. We are intending to present the findings from this research in academic conferences and journal papers. In doing so, we aggregate the findings so that participants' observations or comments will not be linked to them personally. They will have the option to verify the accuracy of any transcripts made. Neither the organization nor the participants will be identified.

How was I identified and why am I being invited to participate in this research?

Your organization has been identified either from the research Supervisor's personal network of industry practitioners or from the networking relationship built through Agile Auckland meetup or through the professional social networking media such as LinkedIn. Your organisation is invited to participate in this research as you have the adequate experience and expertise in global scaled agile software development.

How do I agree to participate in this research?

With your assistance, at this stage, we would like to recruit practitioners from your organization based on their roles and overall experience in global scaled agile software development. We anticipate approximately 15 or so members from your organization involved in the requirements prioritization process.

Your participation and feedback will be of great value in helping us to develop an in-depth understanding of the requirements prioritization process in global scaled agile software development. If you are willing for your organization to participate in this research, please return the enclosed permission to access- consent form within one week of receiving this invitation.

What will happen in this research?

Selected participants will be interviewed by the researcher to discuss their experience in requirements prioritization across scaling agile levels.

- Data collection

For this study, several different sources of information (observation, artefacts, and recorded interviews) will be used to gather data.

- Interviews

For each participant we envisage an interview session (on two visits around six month apart) of one hour followed up by a further session, after a group of interviews have been completed, to confirm the understandings and check the transcripts (it is estimated that this follow up session will be less than one hour in duration).

- Analysis

The data collection will be followed by an analysis phase from which we hope to develop insights into challenges faced by the practitioners, and their practices that contribute to effective requirements prioritization across scaling agile levels.

What are the discomforts and risks?

Based on the highly focused nature of the research I have a view that the potential discomforts and risks are minimal. However;

- Your organization may feel uncomfortable concerning the sharing of confidential or commercially sensitive information.

- Your employees may feel uncomfortable because their employer may know who is participating in the study and who has elected not to take up the invitation.

How will these discomforts and risks be alleviated?

In order to alleviate the first area of discomfort both the signing of mutually acceptable non-disclosure agreements and the careful management of sensitive data guided by AUT's formal ethics approval process will be adopted.

Participants will be reminded of our assurance of the voluntary nature of participation to both them and your company at regular intervals throughout the fieldwork.

Your employees' participation will not be divulged to their employer. They will also have the opportunity to review the transcripts or draft before the research results are published, for confirmation to make sure they do not contain any content, which might reveal their identity. Where their identity is not readily kept anonymous, e.g. through the unique nature of their role, or it is logical and desirable to name the organization, role or views, then their and your comfort with such communication by review of any resulting draft publications will be sought. The procedure described below in the section 'how privacy will be protected' will be employed to further guard participants from any potential risk or discomfort.

What are the benefits?

The insights gained from this research will be available for your organization, the participating individuals and their colleagues. It is expected that this research will help in

self-assessment for the organizations and the participants and aid in the adoption of effective practices, processes, and strategies for prioritization of requirements in global scaled agile software development. The research aligns with the agile philosophy of regular reflective review of practices and continuous improvement and learning, now scaled across the organization. The participating organizations can align their existing processes with a guiding set of good practices (provided by this research) that is theoretically informed, practically validated, based on expert insight and suited to global software development.

How will my privacy be protected?

The participants or the organization will not be identified by name; rather a hypothetical name will be used whenever a referral is made to the organization or the participants in subsequent publications (report, thesis, journal or articles etc.). Exceptions to this may be made where appropriate and agreed.

Privacy and confidentiality will be respected and protected with diligence and by all available means. The identity of participants will be protected at all stages of a project to the extent possible. This will be done by coding of data, restricting of access to transcripts and removal of identifying material from published documentation. Furthermore where possible the participants will be addressed by their group roles rather than individual roles to de-identify them and as far as possible prevent communication of their identities. The participant will be provided with the transcripts for their confirmation to avoid any conflict of interest and make sure the transcript does not contain any content which might unnecessarily reveal their identity.

Since your employees have accepted an open invitation to participate in this research, their participation will remain anonymised to their employer. But given the unique nature of individual roles, their contribution may be inferred, although reported by role as anonymously as possible, or in grouped roles where that is appropriate. Participation in this research is voluntary.

What are the costs of participating in this research?

The research will provide the opportunity for the participating organization to benefit from its results without a major cost (in time and resources) and to have an in depth investigation of a critical area of global software development practice with the potential to contribute to future development effectiveness.

What opportunity do I have to consider this invitation?

We would appreciate a response within one week of receiving the invitation.

Will I receive feedback on the results of this research?

The research findings will be disseminated via the standard academic channels, published PhD thesis, related seminars, and conference and journal papers. All willing participants will receive a summary of the research findings either as an extract of the thesis or as developed in academic publications.

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Research Supervisor, Tony Clear; tony.clear@aut.ac.nz; 09 921 9999 ext. 5329.

Concerns regarding the conduct of the research should be notified to the Executive Secretary of AUTEC, Dr Carina Meares, ethics@aut.ac.nz , 09 921 9999 ext. 6038.

Whom do I contact for further information about this research?

Please keep this Information Sheet and a copy of the Consent Form for your future reference. You are also able to contact the research team as follows:

Researcher Contact Details:

Priyanka Antil

Software Engineering Research Lab (SERL)

School of Engineering, Computer and Mathematical Sciences

Auckland University of Technology

Private Bag 92006

Auckland 1142

New Zealand

Phone: + 64 9 921 9999 x 6376

Email: priyanka.antil@aut.ac.nz

Project Supervisor Contact Details:

Assoc Professor Tony Clear

School of Engineering, Computer and Mathematical Sciences

Auckland University of Technology

Private Bag 92006

Auckland 1142

New Zealand

Phone: + 64 9 921 9999 x 5329

Email: tony.clear@aut.ac.nz

Approved by the Auckland University of Technology Ethics Committee on type the date final ethics approval was granted, AUTEC Reference number type the reference number.

Appendix B: Participant Information Sheet

Date Information Sheet Produced: 14 October 2019

Project Title: Requirements Prioritization in Global Scaled Agile Software Development

Research Introduction

An Invitation

My name is Priyanka Antil and I am a doctoral student in the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology, under the supervision of Assoc Professor Tony Clear and Dr. Ramesh Lal. I am conducting research investigating requirements prioritization process in global scaled agile software development.

Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. None of the participants are identified nor will the information gathered be used to hamper, hinder or harm your career.

The following questions and answers are intended to address the most common questions that the participant may ask about this particular research project. If you need further information, feel free to contact the researcher, Priyanka Antil. My contact details can be found at the end of this document. It is recommended that you use e-mail to reach me.

Project Description and invitation

This research will create knowledge on the underlying philosophy and the related structures (functional set-up and units, artefacts, processes, practices, techniques and roles) driving the requirements prioritization process in a global scaled agile software development environment. Therefore, the goal of this study is to provide an understanding of the requirements prioritization process across scaling agile levels (e.g. portfolio level, program level, and project level)..

I would like to invite you to participate in my research into the area of Requirements Prioritization in Global Scaled Agile Software development. Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. None of the participants is identified

nor will the information gathered be used to hamper, hinder or harm your career. Moreover, this research will not aim to collect commercially sensitive information of the organisation and its members.

How was I identified and why am I being invited to participate in this research?

Your organization has been invited to participate in this research because it has used a scaling agile method for product development.

You are invited to participate in this research because you have responsibilities or are (or have been) part of a team who decides work flow priorities across scaling agile levels and have an adequate work experience.

The organization's key contact for the researchers has advised a set of suitably knowledgeable contacts for participation in the research, of the research project and the invitation to participate in the study.

How do I agree to participate in this research?

As an identified suitable participant. You will be invited in turn to contact the researcher and confirm your willingness to participate in the study. Your participation in this research is voluntary (it is your choice) and whether or not you choose to participate will neither advantage nor disadvantage you. You are able to withdraw from the study at any time. If you choose to withdraw from the study, then you will be offered the choice between having any data that is identifiable as belonging to you removed or allowing it to continue to be used. However, once the findings have been produced, removal of your data may not be possible.

What will happen in this research?

If you accept this invitation to participate, you will take part in an interview to discuss your role and your experience of the requirements prioritization processes, practices in within your team and other distributed teams, as well as challenges and tensions you have encountered. This may be face to face interview, at the workplace or offsite to suit the participant. Remote interviews using skype or similar video conferencing technologies may be employed.

For each participant we envisage an interview session (on two visits around six month apart) of one hour followed up by a further session, after a group of interviews have been

completed, to confirm the understandings and check the transcripts (it is estimated that this follow up session will be less than one hour in duration).

The data collection through various means (e.g. artefacts, tools, and interviews) would be followed up by an analysis phase from which we hope to develop insights into challenges faced by the practitioners, and their practices which contribute to effective workflow prioritization across scaling agile levels.

What are the discomforts and risks?

Based on the highly focused nature of the research I have a view that the potential discomforts and risks are minimal. However;

You may feel uncomfortable on the grounds that your employer may know who is participating in the study and who has elected not to take up the invitation.

You may feel uncomfortable having transcripts of the interview(s) recorded.

You may feel uncomfortable having notes taken.

You may feel uncomfortable with me in an observer role

How will these discomforts and risks be alleviated?

In order to alleviate the first area of discomfort you will be reminded of our assurance of the voluntary nature of participation to both you and your company at regular intervals throughout the fieldwork. Your participation will not be divulged to your employer. You will also have the opportunity to review the transcripts or draft before the research results are published, for confirmation to make sure they don't contain any content which might reveal your identity. Where your identity is not readily kept anonymous, e.g. through the unique nature of your role, or it is logical and desirable to name your organisation, role or views, then your comfort with such communication by review of any resulting draft publications will be sought.

You may withdraw yourself and your data from the study prior to publication of the findings.

The second possible area of discomfort will be addressed by emphasising that the data being recorded will be held confidential between the researcher and the participant,

transcripts will be made available for review, and will be selectively reported and discussed should any sensitive matters arise.

The third possible area of discomfort will be addressed by emphasizing that the data being collected through observation and artefacts and notes taken relate to information drawn from the current practices in general, rather than an individual's activity. Personal performance will not be judged.

The researcher will be vigilant to the fact that the anonymity of the participants is maintained to the extent possible at all stages of the research and the participants do not experience any discomfort or pressure of any type. The procedure described below in the section 'how privacy will be protected' will be employed to further guard participants from any potential risk or discomfort.

What are the benefits?

As well as adding to the body of knowledge and influencing practice in this general area, the insights gained from this study will be made available to yourself and your colleagues and it is hoped that the knowledge gained will be useful for improving the practice in your organization. The research aligns with the agile philosophy of regular reflective review of practices and continuous improvement and learning, now scaled across the organization. It is hoped the research will help you to reflect upon and improve your practices.

How will my privacy be protected?

Since you have accepted an open invitation to participate in this research, your participation will remain anonymised to your employer. But given the unique nature of your role, your contribution may be inferred, although reported by role as anonymously as possible, or in grouped roles where that is appropriate. Participation in this research is voluntary.

Privacy and confidentiality will be respected and protected with diligence and by all available means. The identity of participants will be protected at all stages of a project to the extent possible. This will be done by coding of data, restricting of access to transcripts and removal of identifying material from published documentation. Furthermore where possible the participants will be addressed by their group roles rather than individual roles to de-identify them and as far as possible prevent communication of their identities. The

participant will be provided with the transcripts for their confirmation to avoid any conflict of interest and make sure the transcript does not contain any content which might unnecessarily reveal their identity.

What are the costs of participating in this research?

Time is the only cost to you. The interview will take around one hour of your time.

What opportunity do I have to consider this invitation?

We would appreciate a response within 48 hours of receiving this invitation.

Will I receive feedback on the results of this research?

The research findings will be disseminated via the standard academic channels, any published PhD thesis, related seminars, conference, and journal papers. All willing participants will receive a summary of the research findings either as an extract of the thesis or as developed in academic publications

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Research Supervisor, Tony Clear; tony.clear@aut.ac.nz; 09 921 9999 ext. 5329.

Concerns regarding the conduct of the research should be notified to the Executive Secretary of AUTEC, Dr Carina Meares, ethics@aut.ac.nz , 09 921 9999 ext. 6038.

Whom do I contact for further information about this research?

Please keep this Information Sheet and a copy of the Consent Form for your future reference. You are also able to contact the research team as follows:

Researcher Contact Details:

Priyanka Antil

Software Engineering Research Lab (SERL)

School of Engineering, Computer and Mathematical Sciences

Auckland University of Technology

Private Bag 92006

Auckland 1142

New Zealand

Phone: + 64 9 921 9999 x 6376

Email: priyanka.antil@aut.ac.nz

Project Supervisor Contact Details:

Assoc Professor Tony Clear

School of Engineering, Computer and Mathematical Sciences

Auckland University of Technology

Private Bag 92006

Auckland 1142

New Zealand

Phone: + 64 9 921 9999 x 5329

Email: tony.clear@aut.ac.nz

Approved by the Auckland University of Technology Ethics Committee on *type the date final ethics approval was granted*, AUTEC Reference number *type the reference number*.

Appendix C: Consent Form

For use when interviews are involved.

Project title: Requirements Prioritization in Global Scaled Agile Software Development

Project Supervisor: Tony Clear, and Ramesh Lal

Researcher: Priyanka Antil

□ I have read and understood the information provided about this research project in the Information Sheet dated 14 October 2019.

□ I have had an opportunity to ask questions and to have them answered.

□ I understand that notes will be taken during the interviews and that they will also be audio-taped and transcribed.

□ I understand that contributions made in interviews and observations augmented by development artefacts, associated electronic communications and research diary notes taken by the researcher, will be analyzed in order to better understand the collaboration process along with the related practices.

□ I understand that taking part in this study is voluntary (my choice) and that I may withdraw from the study at any time without being disadvantaged in any way.

□ I understand that while information gathered and reported will generally aim to preserve my confidentiality and privacy, in some cases that will be limited, so such cases will be agreed with me by the researcher before any such information is divulged.

□ I understand that if I withdraw from the study then I will be offered the choice between having any data that is identifiable as belonging to me removed or allowing it to continue to be used. However, once the findings have been produced, removal of my data may not be possible.

□ I agree to take part in this research.

□ I wish to receive a summary of the research findings (please tick one): Yes□ No□

Participant's signature:

.....……
……

Participant's name:

.....……
……

Participant's Contact Details (if appropriate):

……..

……..

……..

……..

Date:

Approved by the Auckland University of Technology Ethics Committee on type the date on which the final approval was granted AUTEC Reference number type the AUTEC reference number

Note: The Participant should retain a copy of this form.

Appendix D: Interview protocol

Project title: **Understanding requirements prioritization process in global scaled agile software development**

Project Supervisor: **Tony Clear, Ramesh Lal**

Researcher: **Priyanka Antil**

This protocol is designed to support the conduct of interviews with participants from large organizations who are employing scaling agile methods for product development and teams are globally distributed.

---

1. Participant Briefing

Before I get started on the interview questions, I would like to remind you the key protocols of how this research will be conducted, in accordance to the details found in the participant information sheet. This will include:

- Purpose of the research (*This research is studying some of the challenges and practices for prioritizing the new ideas and how they get implemented by teams. We are mainly interested in the process and who is involved and some of the techniques that worked and what was more challenging. So we have developed some questions we would like to find out about how you do things here.)*
- Getting your signed consent form
- Notes will be taken during the interview
- Interview will be audio recorded if consent is given
- Do you have any questions about what we are doing or how we are going to run this meeting?

The first set of questions are a bit of background about how your company develops software and how you fit into this. Then we will discuss the process from capturing new ideas, to prioritizing them, assigning streams of work to teams and how they prioritize what they do.

2. Background information

- Participant background
    - Participant's name
    - Gender
    - Nationality
    - Education and formal experience
    - Time at company
    - Time on current project (if applicable)
    - Current position
    - Position reported to
    - Current location (primary location or secondary location)
    - Previous position & company
    - Total development experience
    - Total domain experience
- General organization background
    - Type of development does your organization undertake (e.g., in-house development, products for external customers etc.)
    - Type of scaling agile method used (e.g., DAD, SAFe, Scrum @scale etc.)
    - No of sites and no of staff involved in product/software development

3. Understanding requirements prioritization process in participant's organisation

Innovative ideas

- Where new or innovative ideas come from and how they are captured?

Process:

How is the priority decided on ideas/goals?

- What information is required?
  Where does it come from; what format?
- What prioritization criteria are used?

(How do you get that information? Where does it come from?

e.g., business value- what is the business value and how do you measure that, who is responsible for business value assessment)

- How does the shared understanding happen?
- How is consensus reached?

Role:

- Who are involved in the prioritization of the ideas/goals?
- What are their roles and responsibilities?
- How your role does involves in prioritization?
- Who has authority to decide the priority?

Timing:

- How often is prioritization done?
- How long it takes?

Global aspects:

- Are any people involved in the prioritization that are globally/locally distributed?
- If yes, how do the distance people collaborate in prioritization?
- What tools- synchronous (when + why)
    - Asynchronous (why)
- What are the main challenges you have encountered related to the involvement of the people at distance in prioritization?
- How are these challenges addressed?
- What are the other challenges that you are currently facing related to the involvement of the people at distance in prioritization? Do you have any idea how these challenges should be overcome?

Work distribution to teams or team of teams

- How these ideas (new or innovative ideas) turned into actual work for teams or teams of teams to do in certain order?
- How is work described/broken into parts? Who does that?
- How is this information communicated to the teams?
- How do they prioritize their work? What information do they required to prioritize their work?
- How decide who (team?) is doing what? How break the work up?

Changes (reprioritization)

- When are changes made to decided priority and why and how and by whom? How often?

- How is progress monitored?

Challenges

- What are the main challenges you have encountered while prioritizing the work?

  (Types of problems, sources of problem, reason for occurrence)
- How are these problems addressed?
- What are the other challenges that you are currently facing? Do you have any idea how these challenges should be overcome?

Tensions/conflicts

- Are there conflicts in perceptions on the importance of work items across the actors (of what needs to be implemented and how)?? Between which roles most conflicts occur?
- What are the impacts of these conflicts in terms of product, people etc.
- How are these conflicts addressed?

Recommendation

- Is there something that you would like to change to make the prioritization process better?
  - *Any practice, technology, or activity?*
  - *How?*
- Anything else to add?

Appendix E: Ethics Approval letter

16 March 2020

Tony Clear

Faculty of Design and Creative Technologies

Dear Tony

Re Ethics Application: 19/448 Requirements prioritization in global scaled agile software development

Thank you for providing evidence as requested, which satisfies the points raised by the Auckland University of Technology Ethics Committee (AUTEC).

Your ethics application has been approved for three years until 16 March 2023.

Non-Standard Conditions of Approval

1. The non-disclosure for each company is to be submitted to AUTEC in the from the company uses once the organisations have agreed to participant.

Non-standard conditions must be completed before commencing your study. Non-standard conditions do not need to be submitted to or reviewed by AUTEC before commencing your study.

Standard Conditions of Approval

1. The research is to be undertaken in accordance with the Auckland University of Technology Code of Conduct for Research and as approved by AUTEC in this application.

2. A progress report is due annually on the anniversary of the approval date, using the EA2 form.

3. A final report is due at the expiration of the approval period, or, upon completion of project, using the EA3 form.

4. Any amendments to the project must be approved by AUTEC prior to being implemented. Amendments can be requested using the EA2 form.

5. Any serious or unexpected adverse events must be reported to AUTEC Secretariat as a matter of priority.

6. Any unforeseen events that might affect continued ethical acceptability of the project should also be reported to the AUTEC Secretariat as a matter of priority.

7. It is your responsibility to ensure that the spelling and grammar of documents being provided to participants or external organisations is of a high standard.

AUTEC grants ethical approval only. You are responsible for obtaining management approval for access for your research from any institution or organisation at which your research is being conducted. When the research is undertaken outside New Zealand, you need to meet all ethical, legal, and locality obligations or requirements for those jurisdictions.

Please quote the application number and title on all future correspondence related to this project.

For any enquiries please contact ethics@aut.ac.nz. The forms mentioned above are available online through http://www.aut.ac.nz/research/researchethics

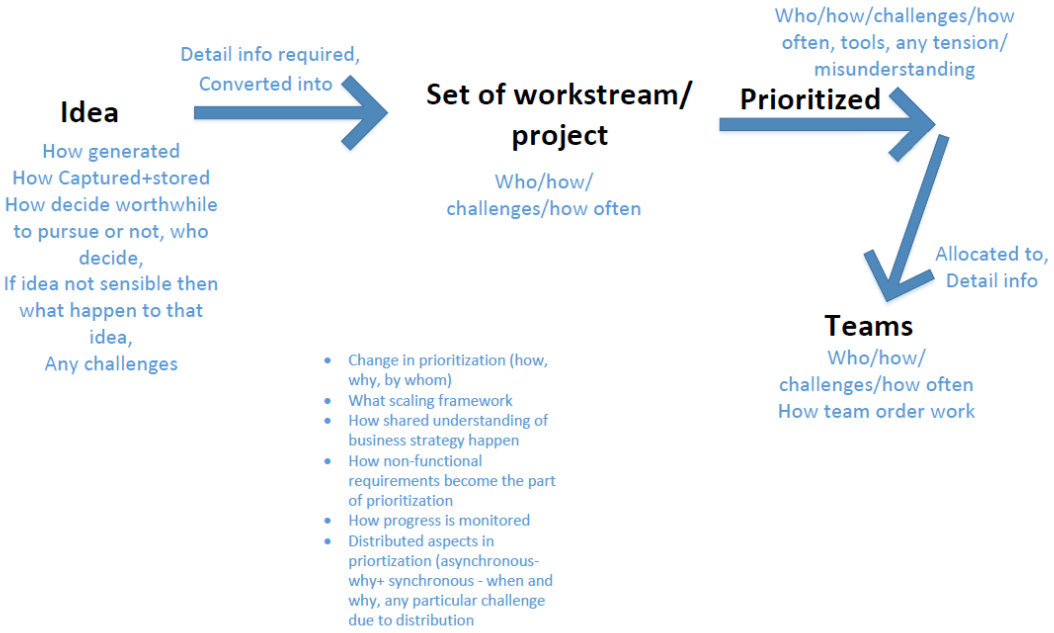
(This is a computer-generated letter for which no signature is required)

The AUTEC Secretariat

Auckland University of Technology Ethics Committee

Cc: priyanka.antil@aut.ac.nz; Ramesh Lal

Appendix F: High-level overview of research study for the interviewees



Appendix G: List of white literature (WL)

Identifier	Document Title
WL1	Daneva, M., van der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., … Wieringa, R. (2013). Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software*, *86*(5), 1333–1353. https://doi.org/10.1016/J.JSS.2012.12.046
WL2	Shrivastava, S. V., & Rathod, U. (2017). A risk management framework for distributed agile projects. Information and Software Technology, 85, 1–15. https://doi.org/10.1016/j.infsof.2016.12.005
WL3	Bass, J. M. (2016). Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology*, *75*, 1–16. https://doi.org/10.1016/j.infsof.2016.03.001
WL4	Vlietland, J., & Vliet, H. van. (2015). Towards a governance framework for chains of Scrum teams. Information and Software Technology, 57, 52–65. https://doi.org/10.1016/J.INFSOF.2014.08.008
WL5	Korkala, M., & Maurer, F. (2014). Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software*, *95*, 122–140. https://doi.org/10.1016/j.jss.2014.03.080
WL6	Moe, N. B., Šmite, D., Šāblis, A., Börjesson, A.-L., & Andréasson, P. (2014). Networking in a Large-Scale Distributed Agile Project. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: ACM. https://doi.org/10.1145/2652524.2652584
WL7	Ktata, O., & Lévesque, G. (2009). Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering* (pp. 59–66). ACM. https://doi.org/10.1145/1557626.1557636
WL8	Spohrer, K., Bick, S., & Scheerer, A. (2016). Inter-Team Coordination in Large Agile Software Development Settings: Five Ways of Practicing Agile at Scale. In *Proceedings of the Scientific Workshop Proceedings of XP2016*. ACM. https://doi.org/10.1145/2962695.2962699
WL9	Šāblis, A., Smite, D., & Šmite, D. (2016). Agile Teams in Large-Scale Distributed Context – Isolated or Connected? In *Proceedings of the Scientific Workshop Proceedings of XP2016*. ACM. https://doi.org/10.1145/2962695.2962705

WL10	Heijden, A. van der, Broasca, C., & Serebrenik, A. (2018). An Empirical Perspective on Security Challenges in Large-Scale Agile Software Development. In International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '18). https://doi.org/10.1145/3239235.3267426
WL11	Rolland, K. H. (2015). "Desperately" seeking research on agile requirements in the context of large-scale agile projects. In *Scientific Workshop Proceedings of the XP2015*. New York, NY, USA: ACM. https://doi.org/10.1145/2764979.2764984
WL12	Paasivaara, M. (2017). Adopting SAFe to Scale Agile in a Globally Distributed Organization. In *Proceedings of the 12th International Conference on Global Software Engineering (ICGSE)* (pp. 36–40). IEEE. https://doi.org/10.1109/ICGSE.2017.15
WL13	Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a Large Globally Distributed Organization: A Case Study. In *Proceedings of the 11th International Conference on Global Software Engineering (ICGSE)* (pp. 74–83). IEEE. https://doi.org/ 10.1109/ICGSE.2016.34
WL14	Paasivaara, M., Heikkila, V. T., & Lassenius, C. (2012). Experiences in Scaling the Product Owner Role in Large-Scale Globally Distributed Scrum. In *Proceedings of the Seventh International Conference on Global Software Engineering* (pp. 174–178). IEEE. https://doi.org/10.1109/ICGSE.2012.41
WL15	Moore, E., & Spens, J. (2008). Scaling Agile: Finding your Agile Tribe. In *Agile 2008 Conference* (pp. 121–124). IEEE. https://doi.org/10.1109/Agile.2008.43
WL16	Gat, I. (2006). How BMC is scaling agile development. In *AGILE 2006 (AGILE'06)* (pp. 315–320). IEEE. https://doi.org/10.1109/AGILE.2006.33
WL17	Grewal, H., & Maurer, F. (2007). Scaling Agile Methodologies for Developing a Production Accounting System for the Oil & Gas Industry. In *Agile 2007 Conference*. IEEE. https://doi.org/10.1109/AGILE.2007.50
WL18	Fitzgerald, B., Stol, K.-J., O'Sullivan, R., & O'Brien, D. (2013). Scaling agile methods to regulated environments: an industry case study. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)* (pp. 863–872). IEEE. https://doi.org/10.1109/ICSE.2013.6606635
WL19	Rautiainen, K., Schantz, J. von, & Vähäniitty, J. (2011). Supporting Scaling Agile with Portfolio Management: Case Paf.com. In *Proceedings of the 44th Hawaii International Conference on System Sciences*. IEEE. https://doi.org/10.1109/HICSS.2011.390

WL20	Heikkila, V., Rautiainen, K., & Jansen, S. (2010). A Revelatory Case Study on Scaling Agile Release Planning. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE. https://doi.org/10.1109/SEAA.2010.37
WL21	Jha, M. M., & Vilardell, R. M. F. (2016). Scaling Agile Scrum Software Development: Providing Agility and Quality to Platform Development by Reducing Time to Market. In *Proceedings of the 11th international conference on global software engineering (ICGSE)* (pp. 84–88). IEEE. https://doi.org/10.1109/ICGSE.2016.24
WL22	Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2014). Towards Rapid Releases in Large-Scale XaaS Development at Ericsson: A Case Study. In *Proceedings of the 9th International Conference on Global Software Engineering* (pp. 16–25). IEEE. https://doi.org/10.1109/ICGSE.2014.22
WL23	Scheerer, A., Bick, S., Hildenbrand, T., & Heinzl, A. (2015). The Effects of Team Backlog Dependencies on Agile Multiteam Systems: A Graph Theoretical Approach. In *Proceedings of the 48th Hawaii International Conference on System Sciences* (pp. 5124–5132). IEEE. https://doi.org/10.1109/HICSS.2015.606
WL24	Evbota, F., Knauss, E., & Sandberg, A. (2016). Scaling up the Planning Game: Collaboration Challenges in Large-Scale Agile Product Development. In *Proceedings of the 17th International Conference, XP 2016* (pp. 28–38). Springer, Cham. https://doi.org/10.1007/978-3-319-33515-5_3
WL25	Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinz, A. (2017). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering*, *44*(10), 932–950. https://doi.org/10.1109/TSE.2017.2730870
WL26	Tripathi, N., Rodríguez, P., Ahmad, M. O., & Oivo, M. (2015). Scaling Kanban for software development in a multisite organization: Challenges and potential solutions. In *International Conference on Agile Software Development* (pp. 178–190). Springer, Cham. https://doi.org/10.1007/978-3-319-18612-2_15
WL27	Alsaqaf, W., Daneva, M., & Wieringa, R. (2018). Understanding challenging situations in agile quality requirements engineering and their solution strategies: insights from a case study. In *Proceedings of the 26th International Requirements Engineering Conference (RE)* (pp. 274–285). IEEE. https://doi.org/10.1109/RE.2018.00035
WL28	Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., & Kanagwa, B. (2017). Requirements Engineering Challenges in Large-Scale Agile System Development.

	In *Proceedings of the 25th International Requirements Engineering Conference* (pp. 352–361). IEEE. https://doi.org/ 10.1109/RE.2017.60
WL29	Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering, 22(6), 2892–2936. https://doi.org/10.1007/s10664-016-9491-z
WL30	Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2018). Large-scale agile transformation at Ericsson: a case study. *Empirical Software Engineering*, *23*(5), 2550–2596. https://doi.org/10.1007/s10664-017-9555-8
WL31	Bjarnason, E., Wnuk, K., & Regnell, B. (2011). A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*. ACM. https://doi.org/10.1145/2068783.2068786
WL32	Sekitoleko, N., Evbota, F., Knauss, E., Sandberg, A., Chaudron, M., & Olsson, H. H. (2014). Technical Dependency Challenges in Large-Scale Agile Software Development. In *International Conference on Agile Software Development* (pp. 46–61). Springer. https://doi.org/10.1007/978-3-319-06862-6_4
WL33	Schnitter, J., & Mackert, O. (2010). Large-scale agile software development at SAP AG. In *International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 209–220). ACM. https://doi.org/10.1007/978-3-642-23391-3_15
WL34	Dingsøyr, T., Brede Moe, N., Erlend Faegri, T., Amdahl Seim, E., & Cortellessa, V. (2018). Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*, *23*(1), 490–520. https://doi.org/10.1007/s10664-017-9524-2
WL35	Gunyho, G., & Gutiérrez Plaza, J. (2011). Evolution of Longer-Term Planning in a Large Scale Agile Project – F-Secure's Experience. In *International Conference on Agile Software Development* (pp. 306–315). https://doi.org/10.1007/978-3-642-20677-1_22
WL36	Badampudi, D., Fricker, S. A., & Moreno, A. M. (2013). Perspectives on Productivity and Delays in Large-Scale Agile Projects. In *International Conference on Agile Software Development* (pp. 180–194). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38314-4_13

WL37	Bass, J. M. (2015). How product owner teams scale agile methods to large distributed enterprises. Empirical Software Engineering, 20(6), 1525–1557. https://doi.org/10.1007/s10664-014-9322-z
WL38	Kausar, M., & Al-Yasiri, A. (2017). Using Distributed Agile Patterns for Supporting the Requirements Engineering Process. *Requirements Engineering for Service and Cloud Computing*, 291–316. https://doi.org/10.1007/978-3-319-51310-2_13
WL39	Wildt, D., & Prikladnicki, R. (2010). Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report. In *Agility Across Time and Space* (pp. 31–46). Springer. https://doi.org/10.1007/978-3-642-12442-6_3
WL40	Avritzer, A., Bronsard, F., & Matos, G. (2010). Improving Global Development Using Agile How Agile Processes Can Improve Productivity in Large Distributed Projects. In *Agility Across Time and Space* (pp. 133–148). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-12442-6_9
WL41	Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. Empirical Software Engineering, 15(6), 654–693. https://doi.org/10.1007/s10664-010-9136-6
WL42	Tureček, T., Šmiřák, R., Malík, T., & Boháček, P. (2010). Energy Project Story: From Waterfall to Distributed Agile. In International Conference on Agile Software Development (pp. 362–371). Springer. https://doi.org/10.1007/978-3-642-13054-0_39
WL43	Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036
WL44	Usman, M., Britto, R., Damm, L.-O., & Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, *99*, 21–40. https://doi.org/10.1016/J.INFSOF.2018.02.009
WL45	Eckstein, J. (2014). Architecture in Large Scale Agile Development. In *International Conference on Agile Software Development* (pp. 21–29). Springer. https://doi.org/10.1007/978-3-319-14358-3_3
WL46	Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. Journal of Software: Evolution and Process, 1–24. https://doi.org/10.1002/smr.1954

WL47	Ali, M., & Lero, B. (2009). An Exploratory Study of Architectural Practices and Challenges in Using Agile Software Development Approaches. In *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture* (pp. 81–90). IEEE. https://doi.org/10.1109/WICSA.2009.5290794
WL48	Roopa, M. S., Sankarasubbiah, C., & Mani, V. S. (2017). Usable Software at the End of Each Takt. A Milestone in the Lean Transformation of a Globally Distributed Software Development Team. In *Proceedings of the 12th International Conference on Global Software Engineering (ICGSE)* (pp. 116–120). IEEE. https://doi.org/10.1109/ICGSE.2017.9
WL49	Hannay, J. E., & Benestad, H. C. (2010). Perceived Productivity Threats in Large Agile Development Projects. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. https://doi.org/10.1145/1852786.1852806

Appendix H: List of grey literature (GL)

Identifier	Document Title
GL1	Nori, R. (n.d.). SAFe Case Study: SproutLoud. Retrieved December 17, 2018, from https://www.scaledagileframework.com/case-study-sproutloud/
GL2	Sferlazza, F. (n.d.). LeSS adoption at Italte. Retrieved December 11, 2018, from https://less.works/case-studies/italtel.html
GL3	Reinikainen, R. (n.d.). SAFe case study: Telia Finland. Retrieved December 1, 2018, from https://www.scaledagileframework.com/case-study-telia-finland/
GL4	Guyot, C. (n.d.). SAFe case study: Kantar Retail Virtual Reality. Retrieved December 1, 2018, from https://www.scaledagileframework.com/kantar-retail-case-study/
GL5	Gusch, L., & Herbai, P. (n.d.). Case study: Elekta. Retrieved December 1, 2018, from https://www.scaledagileframework.com/elekta-case-study/
GL6	Ambler, S. (2014b). Managing requirements dependencies between agile teams. Retrieved December 17, 2018, from https://disciplinedagiledelivery.com/managing-requirements-dependencies-between-agile-teams/
GL7	Ambler, S. (2018a). Disciplined agile program management – The product owner team. Retrieved December 17, 2018, from http://disciplinedagiledelivery.com/disciplined-agile-program-management-the-product-owner-team/
GL8	Ambler, S. (2018b). Strategies for capturing quality requirements. Retrieved December 28, 2018, from http://disciplinedagiledelivery.com/capturing-non-functional-requirements/
GL9	Ambler, S. (2014a). If the requirements aren't changing your team is likely in trouble. Retrieved December 17, 2018, from http://disciplinedagiledelivery.com/changingrequirements/
GL10	Ambler, S. (2013). 11 strategies for dealing with technical debt. Retrieved December 17, 2018, from http://disciplinedagiledelivery.com/technical-debt/
GL11	Ambler, S. (2015). Large agile teams. Retrieved December 17, 2018, from http://disciplinedagiledelivery.com/agility-at-scale/large-agile-teams/
GL12	Hussain, N. (2018). *Requirements engineering for globally distributed teams using scaled agile framework.* Aalto University, Finland. Retrieved from https://aaltodoc.aalto.fi/handle/123456789/32500

GL13	Ehrnberg, F., & Blide, G. (2018). *Incremental requirements engineering in a large-scale agile and safety-critical context : A case study*. Chalmers Uiniversity of Technology, Gothenburg, Sweden. Retrieved from http://publications.lib.chalmers.se/records/fulltext/255469/255469.pdf
GL14	Hinterberg, L., & Hoffman, F. (2018). *Exploring the scaled agile frame-work in a virtual team setting*. Lund University, Sweden. Retrieved from http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8950400&fileOId=8950409
GL15	Roman, G. de C. (2016). *Characterizing the presence of agility in large-scale agile software development*. Pontifical Catholic University of Rio de Janeiro, Brazil. Retrieved from https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US20100183655.pdf
GL16	Scrum.org. (n.d.). Cathay Pacific Airways Nexus case study. Retrieved December 31, 2018, from https://www.scrum.org/resources/cathay-pacific-airways-makes-nexus-their-official-scaling-framework-it
GL17	Scrum.org. (2017). Major asian airlines uses Nexus framework. Retrieved December 31, 2018, from https://www.scrum.org/resources/major-asian-airline-scales-scrum-nexus
GL18	Kim, N., & Cho, H. (2018). Scaling at Food-tech Startup: Transformation challenges, lessons learned and growth. Retrieved December 31, 2018, from https://www.agilealliance.org/resources/experience-reports/scaling-at-food-tech-startup-transformation-challenges-lessons-learned-and-growth/
GL19	Pavlichenko, I. (2018). Eliminate dependencies, don't manage them. Retrieved December 31, 2018, from https://www.scrum.org/resources/blog/eliminate-dependencies-dont-manage-them
GL20	Sofer, E. (n.d.). LeSS adoption for a safety & security management product. Retrieved December 10, 2018, from https://less.works/case-studies/vesecurity.html
GL21	Smet, J. De. (n.d.). Agfa Healthcare. Retrieved December 11, 2018, from https://less.works/case-studies/agfa-healthcare.html
GL22	Lv, Y. (2017). Huawei - LeSS without Scrum. Retrieved December 11, 2018, from https://less.works/case-studies/huawei.html

Appendix I: MLR Study

Requirements Engineering in Global Scaled Agile Software Development: A Multi-Vocal Literature Review

Priyanka Antil[a,*], Tony Clear[a], Ramesh Lal[a]

[a] *Auckland University of Technology, Private Bag 92006, Auckland 1142, New Zealand*

Abstract

Context: Recent trends for developing software to suit global market needs indicate the need to scale development processes with distributed Agile development being more prevalent. Thus, it is vital to evolve the nature of requirements engineering in response to these changes.
Objective: To identify the challenges and mitigating strategies for requirements engineering in global scaled agile software development.
Research Method: Since the agile approach has emerged from the industry and now the evidence suggests that it has become mainstream for software development, a multi-vocal literature review method was adopted to investigate the requirements engineering challenges and to identify the strategies to mitigate them in global scaled agile software development. The grey literature (adoption case studies, thesis, and consultants reports, etc.) together with white literature (e.g. journal article) were reviewed.
Result: The study identified 26 requirements engineering challenges in globally distributed large scale agile software development environment. The study also reported strategies (25 strategies) to mitigate the requirements engineering challenges. These challenges and strategies were then grouped into core groupings, respectively, of four and seven categories.
Conclusion: The findings of this multi-vocal literature review study provide a structured synthesis of the current state of knowledge on requirements engineering in globally distributed large scale agile software development as well as opportunities for future research.

Keywords: Requirements engineering, Global scaled agile software development, Challenges, Strategies, Multi-vocal literature review

1. Introduction

Current trends for developing software include the need to scale development processes to suit global market needs with distributed agile development being more prevalent [1]. The increasing impact from emerging technologies such as cloud computing, further enables the immediate global delivery of software products and services. These scaling processes move software development more strongly into alignment with the current business strategy and planning processes [2]. The demand for both global and local market expertise plus the ability to deliver at volume and scale, require development at multiple global sites, driven by a common culture and engineering processes [2]. Thus, the nature of requirements engineering (RE) in a software development environment has to evolve in response to these changes. These issues are challenging practitioners, with timely, predictable, and high-quality delivery of software proving a major challenge [1]. Research focusing specifically on RE in global scaled agile software development (GSASD) is scarce [3]. This is further evidenced in examining the systematic literature reviews (SLR), multivocal literature reviews (MLR), and mapping studies published until 2018, as discussed in Section 2 (Related work section).This was supported by a supplementary search to include studies published between Jan 2019 to June 2020 as discussed in Section 4.1.3. An MLR study on RE in GSASD was conducted to address this gap. Ample publi-

*Corresponding author
Email address: priyanka.antil@aut.ac.nz (Priyanka Antil)

cations were found (white literature) on agile software development, several on distributed (scaled) and global software development based on structured setups, but limited on GSASD. "However", there was an abundance of grey literature (GL) and more emerging every day on agile and scaled agile methods, worth investigating [4].
Following are the research questions that were set to attain the aim of this research study:
Research Question 1: What are the challenges for requirements engineering in global scaled agile software development?
Research Question 2: What are the strategies for surmounting challenges in requirements engineering activities in global scaled agile software development?
The rest of this MLR study is designed as follows: background, and related work are provided in Section 2, and the research design of the study is presented in Section 3. The findings of the MLR are presented in Section 4. Summary of the findings is presented in Section 5, and limitations of the study are discussed in Section 6. Finally, conclusion and future research are discussed in Section 7.

2. Related work

This section commences with background information on GSASD. As a next step, RE in GSASD is discussed. Lastly, previous secondary studies on GSASD are discussed.

2.1. Global Scaled Agile Software Development

A substantial growth in the adoption of agile methods in software development has been noticed over a decade [5]. Unlike traditional methods driven via documentation and upfront design, agile methods (key agile methods- Scrum and Extreme Programming [6]) ought to be customer-driven, with team collaboration, and evolutionary delivery. Agile methods are best suited for small and co-located development teams [7]. The expected benefits of agile methods are reduced time to market and ability to deliver high quality software, having the ability to manage the requirements which tend to be dynamic in nature [8]. Thus, agile methods have been adopted in large scale software projects where teams are globally distributed [9].
Scaled agile Definition: Participants of the "XP"workshops 2013 - 2016 on the large scale agile approach suggested that "scaling is about more than the size (number of people and teams), but also about scaling the organization and the distribution (number of sites)" [10]. For example, scaled agile software development might involve having "50 or more people or at least six teams" [3] or "more than three sites" [9]. Therefore, building on the definition of "scaled agile"proposed by [3], as "software development organizations with 50 or more people or at least six teams", we include the additional stipulation to incorporate a distributed focus, that these people or teams must work across sites located in at least two different locations. When the organization has people or teams distributed globally across country boundaries, it is defined as "global scaled agile".
Although adoption of agile methods in large scale software projects has gained significant attention in the recent years from the researchers [3, 10] as well as from the practitioners [11]. There are many challenges that needs to be investigated to maximize their successful adoption.
Various scaling agile frameworks have been proposed by the practitioners and consultants that include Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD), Scrum at scale, Spotify and Large Scale Scrum (LeSS), to specific practices from various agile methods which could be adopted for scaling up software development [11]. "However", an empirical evaluation of these frameworks is lacking [9]. The 2017 XP conference also highlighted that research is needed on GSASD. As such, an empirical investigation on the activities relating to planning, analysis, design, implementation (development, testing and documentation) and deployment with GSASD must be explored [10].

2.2. Requirements Engineering in Global Scaled Agile Software Development

RE is the branch of software engineering through which user requirements are collected, specified, and understood for developing software products [12].
In agile software development, RE is performed iteratively, and typically a product backlog is formed that stores the list of prioritized requirements, which is continuously validated through customer feedback [7].The RE activities are not clearly separated in agile. Therefore, agile RE engineering activities can take place in parallel with other activities (i.e., development, testing etc.). Also, the RE activities in agile are carried out on a "just-in-

time"basis with little design upfront [8]. This shows the ad-hoc nature of RE in agile environments [7]. A substantial amount of work has been done related to agile RE (i.e., agile RE practices, RE challenges etc.) [13, 8]. "However", work specifically related to agile RE in large scale globally distributed software projects is lacking [14]. A systematic literature review by Inayat et al. [13] and a mapping study by Heikkila et al. [15] clearly state that research is required on agile RE, specifically on the globally distributed large scale agile software development.

2.3. Secondary studies of global scaled agile software development

There appears to be limited literature on large scale agile globally distributed software development. Only six literature reviews were found related to large scale agile that are summarized below.

Dikert et al. [3] performed a systematic literature review study on the scaled agile transformation. That study identified 35 challenges and 29 success factors grouped into nine and eleven categories, respectively. The noticeable success factors were selecting and customizing the agile model, mind-set and alignment, management support, and training and education. The common challenges identified were change resistance, integrating non-development functions, agile being difficult to implement, and requirements engineering challenges [3]. In addition, Dikert et al. [3] findings revealed that more empirical research is needed on scaled agile.

Khalid et al. [16] performed a systematic literature review study on agile scalability and adoptability. Khalid et al. [16] described and identified the limitations of agile practices for scaled projects. Documentation, time period, human resources related problems, team coordination and communication, and distributed teams were identified as limitations for scaling using agile practices in projects.

Gustavsson et al. [17] investigated the literature on roles required for team coordination in the scaled agile software development environment. They discovered that additional roles are required for coordination. Their investigation also found that only a few organizations have adopted the required coordination responsibility within the teams based on the scaled agile frameworks (e.g., SAFe, LeSS). In addition, the organizations which had coordination roles mainly focused on vertical coordination. This resulted in managers developing plans and accordingly allocating work by themselves. "However", in a scaled agile setup the self-organizing principle ought to be a key practice whereby the entire team are responsible for planning and task allocation (horizontal coordination).

Razavi et al. [18] in their systematic literature review study concluded that an effective set of tools are needed for coordination and communication in any distributed scaled agile development environment. They also claimed that the next complex challenge will be to produce quality products.

Alsaqaf et al. [19] investigated the literature on quality requirements (QRs), focusing on the requirements relating to the quality in scaled agile distributed projects. Alsaqaf et al. [19] discovered twelve QRs related challenges which were mostly as a result of the adopted agile practices. These QRs related challenges were mostly as a result of inability of the product owners (PO) to handle QRs. They also identified strategies to overcome the challenges relating to QRs at a proposal level, while pointing out more research is required in order to understand the QRs in distributed scaled agile software development.

Putta et al. [20] conducted a multi-vocal literature review (MLR) study on adopting SAFe, identifying benefits and challenges. The most common benefits include alignment, productivity, quality, collaboration, and time to market. Challenges that received most attention include moving away from agile, global distributed challenges, agile release train challenges, program increment challenges, backlog management challenges, change resistance, and staffing product owner challenges. In addition, the authors suggested that more primary research is required on transformation steps, implementation, and challenges of the SAFe framework.

To sum up, the existing literature reviews demonstrated that GSASD is a growing area of interest to researchers and practitioners, but little is known about agile RE in GSASD. Only one literature review was found, which was conducted by Alsaqaf et al. [19]. "However", the main focus of the Alsaqaf et al. [19] study was to identify the challenges relating to QRs in large agile distributed projects and strategies to mitigate them. Other critical aspects of RE such as requirements prioritization are missing. Furthermore, the Alsaqaf et al. [19] study focused on the white literature (e.g., journal articles) studies only, despite scaled agile mostly being driven by the practitioners and consultants in the

field.
Hence, an MLR study is needed to provide information and understanding on RE in GSASD as well as to identify research gaps that need addressing. This motivated us to undertake an MLR study on RE in GSASD.

3. Research Methodology

The Kitchenham et al. [21] and Garousi et al. [22] guidelines were followed to conduct this MLR study. The MLR process that consists of planning phase, pilot study, conducting the review, and reporting the review is elaborated in the following subsections.

3.1. Planning phase

The planning phase was comprised of research objectives and research questions, search strings, data sources, and inclusion/exclusion criteria, which are described in the following subsections.

3.1.1. Research Questions

The main purpose of this MLR was to identify the gaps about the software product RE process in GSASD. This was motivated by an interest in RE, as impacted by the relatively novel phenomenon of adoption of agile methods in large scale global settings. The search was limited papers from 2001 as the year in which the agile manifesto was established, and agile methods started gaining attention among the researchers and practitioners [23].
The main objectives of this MLR study were: (i) Identify the challenges in RE in GSASD, and (ii) Identify effective strategies for surmounting challenges in RE activities in GSASD.
There are two research questions (RQs) that were set to achieve the objectives of this study:
RQ 1: What are the challenges for requirements engineering in global scaled agile software development?
RQ 2: What are the strategies for surmounting challenges in requirements engineering activities in global scaled agile software development?

3.1.2. Search strings/constructing the search strings

Step 1: used the Population, Intervention, Comparison and Outcomes (PICO) framework to identify main search terms (as shown in Section 4.1 of review protocol [24];
Step 2: Used synonyms for major terms;
Step 3: Found keywords through relevant papers;
Step 4: If database allows, used Boolean operators OR and AND for synonyms and to link the main search terms respectively.
According to steps as mentioned above:
Main search terms: Requirements engineering, global software development, scaled agile.
The search string for this MLR is made up of three substrings: S1, S2, and S3, defined as follows:
Substring S1 consisted of keywords related to requirements engineering practices: "requirements engineering", "requirements development", "requirements prioritization", "user story", "features", "portfolio management", "backlog management"
Substring S2 made up of keywords related to global software development/engineering: "global software development", "Distributed software development"
Substring S3 was developed by using keywords related to scaled agile software development methods: "scaled agile framework", "Large-Scale Scrum", "Disciplined Agile Delivery".
Eq. (1) represents the main search string,
S1 AND S2 AND S3 (1)
Following shows a sample search conducted in the digital databases [1]:
("requirements engineering" OR "requirements development" OR "requirements prioritization" OR "user story" OR "features" OR "portfolio management" OR "backlog management") AND ("distributed software development" OR global software*) AND ("large-scale scrum" OR "scaling agile" OR "disciplined agile delivery" OR "scrum-of-scrum")[2]

3.1.3. Data sources

In order to retrieve relevant primary studies from white literature (WL) and grey literature (GL), search was performed on digital databases, online web search, key conference proceedings, and methods' creator websites, which are described below.
Resources to be searched for WL: Three search strategies were followed to retrieve relevant WL studies: (i) Automatic search (Digital databases) (ii) Manual search (Key conferences) (iii) Snowballing [21].

[1] Some initial searches were performed to fine-tune the search string (shown in Appendix A of review protocol [24]
[2] * for truncation

(i) Digital databases: An advanced search was performed on the broad range of digital databases that include IEEE, ACM, Scopus, ScienceDirect, and Springer Link[3].
The main reason to include these digital databases was the possibility of accessing their contents (services offered by our institution). Moreover, the impact rate of journals and conference proceeding that are provided by these databases is higher, and comprehensively cover the software engineering field in general [21]. "However", other notable digital databases that include Web of Science and Wiley InterScience were excluded in order to minimize duplicate studies.
(ii) Manual search: An additional search was performed on key conference proceedings to minimize the possibility of missing potential studies. The reason to include these conferences was because of their standing within the field and specific focus on the key areas of requirements engineering, scaled agile, and global software development.
(iii) Snowballing: A snowballing (backward and forward snowballing) was performed on the studies that were selected through an automatic and manual search in order to cover all possible primary studies related to the phenomena of interest [21]. Backward snowballing was performed iteratively by reviewing the references of primary studies. This procedure was repeated until no new relevant study was retrieved. On the other hand, Google Scholar's citation index help was undertaken to perform forward snowballing on the primary studies by looking at the new papers who had cited those studies.
Resources to be searched for GL: According to the MLR guidelines which are proposed by Garousi et al. [22], two search strategies were followed for GL: (i) Manual search- Methods' creator websites, (ii) Automatic search- General web search engine, 'ProQuest Dissertations and Thesis Global' database
(i) Methods' creator websites: As recommended by Garousi et al. [22], consulting firms and scaling agile frameworks have been identified which have accompanying support resources on their websites (e.g., case studies, training materials, and guides). The practitioners' homepages were hand-searched to retrieve potentially relevant studies.

[3]search string that is used in each of these databases is available for download via https://github.com/priya020/MLR-Data in the form of an Excel Spreadsheet

Methods' creator websites that were hand-searched to retrieve potential relevant studies include SAFe, DAD, nexus, LeSS, and Scrum-of-Scrum. The main reason to search these frameworks' websites was, these are the frameworks that were commonly discussed in selected WL primary studies [25, 26]. Moreover, these scaling agile methods are the predominant frameworks adopted by large organizations that are validated through relevant literature reviews as well [27, 20, 18]. Also, the 12th agile state report that was produced by VersionOne in 2017 confirmed that these are the predominant frameworks adopted by large organizations developing software [11].
(ii) An advanced search was performed on 'ProQuest Dissertations and Thesis Global' in order to retrieve potential PhD/Master theses [21]. The main reason to include this database was its global coverage and the possibility of accessing its contents (services offered by our institution). Moreover, an advanced search was also performed on the regular Google search engine ((https://www.google.com) to retrieve relevant masters/PhD thesis.

3.1.4. Study Inclusion/Exclusion criteria

Studies relevant to the search terms were included (as shown in Section 3.1.2). The secondary studies, short papers, panel discussions, PowerPoint presentations, posters and rejected manuscripts were excluded[4] [21].

3.2. Pilot study

By following the Kitchenham et al. [21] guidelines, a pilot study was conducted to check the feasibility of the review protocol that included search strategy, data extraction process, and data synthesis.
The procedure adopted to select primary studies for the pilot study: Ten percent of the final studies were selected to test out the MLR protocol. Five percent of studies were selected from the top of the final studies due to the likelihood of the primary studies on the top[5], while five percent of studies were selected from the remaining studies [28].
A total of 20 studies were selected for the pilot study; 14 from the WL and 6 from the GL. As

[4]Detailed checklist of study Inclusion/exclusion can be found in Section 4.4 of review protocol [24]

[5]The first five studies were selected from the list of studies that were displayed after applying the search string on ACM digital library.

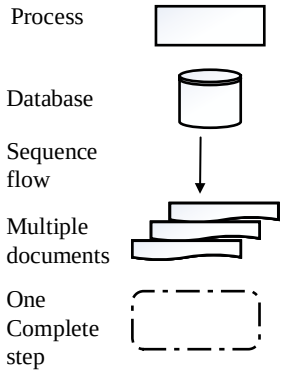


Figure 1: Legend for the screening process

a result of the pilot study, a few amendments were made to the MLR protocol. First, for the main study, experience reports and blogs that come under GL were retrieved from the reputed consultants'/methods' creator websites to ensure the retrieved source is valid and free of undue bias. Second, to retrieve the potential Master/PhD thesis for the main study, a separate search was also performed on the google search engine in addition to 'ProQuest Dissertations and Thesis Global' database. Third, it was noticed that in some cases, masters/PhD thesis and masters/PhD work as a journal article were both available. In these cases, it was decided that (after consultation with the second and third researcher) masters/PhD work as a journal article was given higher preference than the masters/PhD thesis for the main study for evidencing the quality of the work and minimizing the bias. Lastly, the quality assessment checklist was developed to evaluate the quality of the selected primary studies.

3.3. Conducting a review

The overview of the study process and tally of selected papers that were determined by analysing the primary studies are described in this section. Figure 1 shows the legend that is used in Figure 2 (i.e., WL screening process) and Figure 3 (i.e., GL screening process).

3.3.1. WL screening process

Figure 2 shows the screening process of WL studies which is made up of five steps. The same number of steps were followed in the automatic and manual search.

WL-Step1 : In the first step, the search string that was developed in Section 3.1.2 was applied to all selected digital databases, which were resulted 338 studies. In the manual search, each set of selected conference proceedings were examined separately.

WL-Step2: The title and abstract of the possible primary studies were reviewed against study inclusion/exclusion criteria (as shown in Section 3.1.4) to discard the irrelevant studies. If there was any doubt whether a study should be included or not, that was discussed with the secondary and third researchers in order to make an optimal decision. As a result of WL-Step2, a total of 154 candidate studies were retrieved from the automatic search and 61 studies from the manual search.

WL-Step3: In WL-Step3, the pre-selected studies were examined based on the full text. As a result of WL-Step3, a total of 53 studies were selected from the automatic search and 11 studies from the manual search.

WL-Step4: Based on the result of automatic and manual searches, snowballing was performed in WL-Step4 to retrieve all relevant studies. As a result of WL-Step4, four primary studies were retrieved.

WL-Step5: Primary studies that were retrieved as a result of WL-Step3 and WL-Step4 were combined, and the duplicate studies were removed. Therefore, final WL studies selection ended up with 49 studies.

3.3.2. GL screening process

The GL studies' screening process consisted of four steps as shown in Figure 3. The automatic search (represented by Search2 in Figure 3) and the manual search (represented by Search 1 in Figure 3) that were performed to select the WL studies were also carried out to retrieve the relevant GL studies.

WL-Step1 through WL-Step3 of screening the WL studies were also repeated in the screening of GL studies. In the automatic search, 21 studies were selected based on Title/Keywords/Abstract and seven studies on the basis of full text by applying the study inclusion/exclusion criteria (shown in Section 3.1.4) whereas in the manual search, 143 studies were retrieved on the basis of title/Keywords, and 20 studies were retrieved based on full text.

In GL-Step4, studies that were retrieved as a result of automatic and manual search were combined, and the duplicate studies' were removed. Finally, GL studies selection consisted of 26 primary studies.

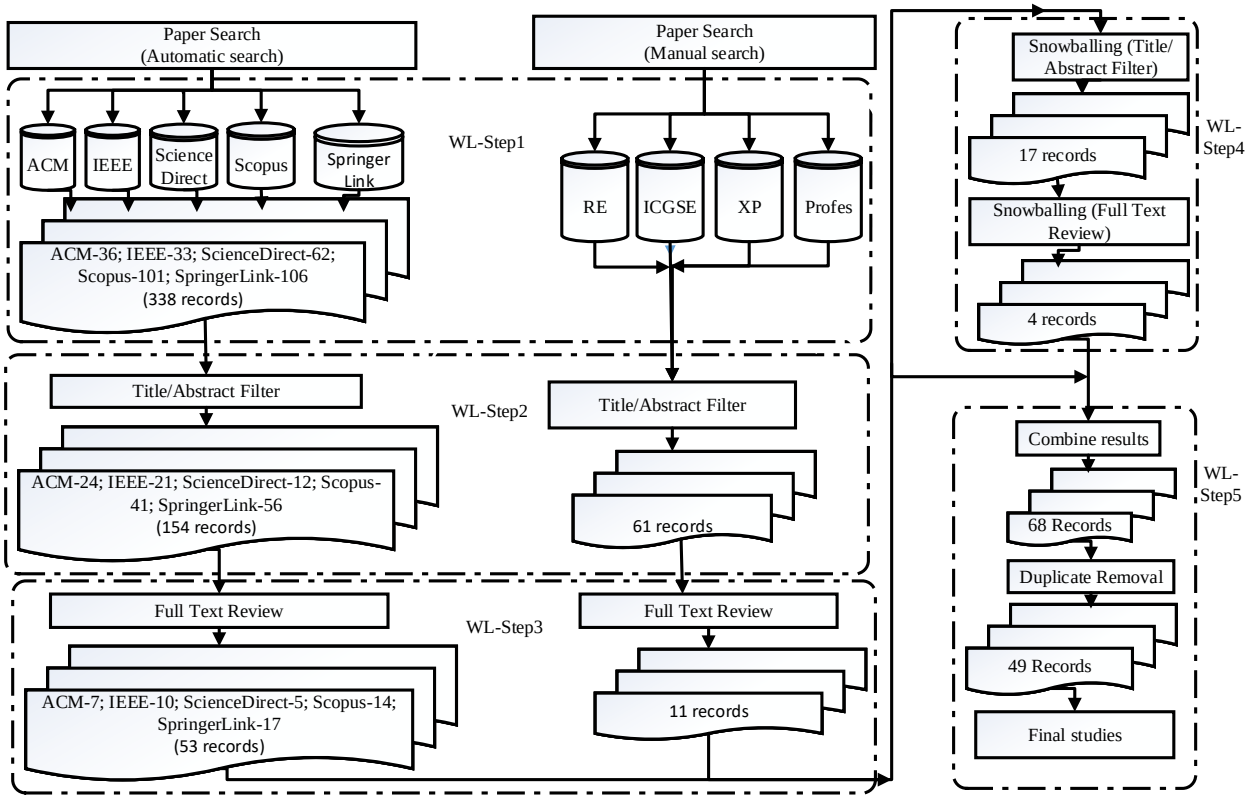


Figure 2: WL screening process

3.3.3. Combining final WL studies and GL studies

In this step, final studies that were retrieved as a result of WL-Step5 in screening the WL studies and GL-Step4 in screening the GL studies were combined, and the duplicates were removed. the study selection ended up with 71 primary studies, that contained 49 WL studies and 22 GL studies[6].

3.3.4. Quality assessment of the studies

The quality assessment of each primary study was conducted after the data extraction. The quality assessment questions were defined by adopting the Garousi et al. [22] guidelines as shown in Table 1[7]. Each category of quality assessment, which is presented in Table 1, was evaluated on a scale from zero to one except the literature type. Literature type was evaluated on the scale of zero to four as this was the category where the precedence of the

Table 1: Quality assessment adopted from [22]

Q1	Is the publishing organization/Author is reputable?
Q2	Has the author cited often by others?
Q3	Has the author published other work in the field?
Q4	Does the source have clearly stated aim/objectives?
Q5	Is the focus of source on RE in global scaled agile setting?
Q6	Are any limits clearly stated?
Q7	Is the source supported by documented references?
Q8	Are the conclusions justified by result?
Q9	Does the source have a clearly stated date?
Q10	Does the source enrich the current research?
Q11	Literature type?

[6]The complete list of WL papers and GL studies is available for download via https://github.com/priya020/MLR-Data in the form of an Excel Spreadsheet

[7]More detail on quality assessment criteria can be found in Section 4.6 of review protocol [24]

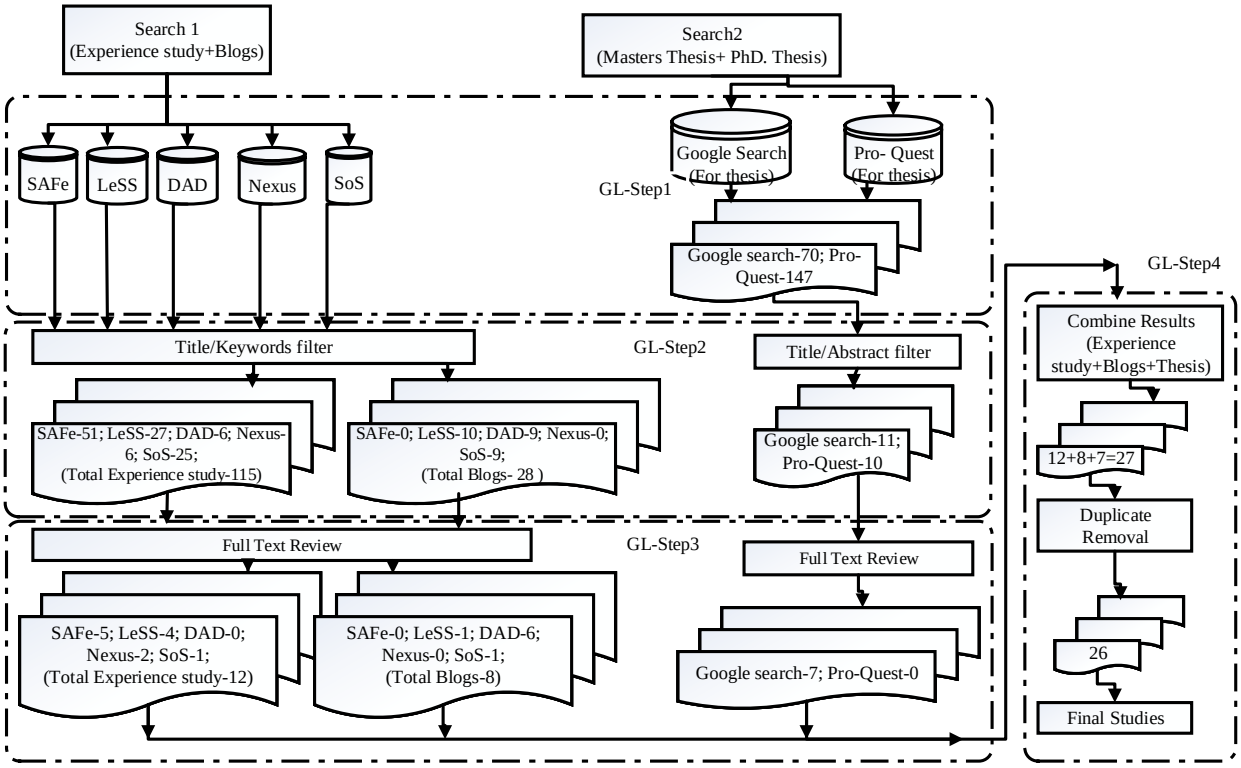


Figure 3: GL screening process

literature was defined[8].

Table 2 and Table 3 showed the quality assessment of each primary study that was done after the data extraction. The quality score of the selected primary studies varied a great deal from four to 14. According to the quality evaluation as shown in Table 2 and Table 3, the majority of the selected studies (60 studies out of 71 (where 49 WL studies as shown in Table 2, 22 GL studies as shown in Table 3) received a quality score from nine to 14, and only 11 primary studies had a quality score from four to eight. Even though the quality score of some studies was low, they were involved in the review since they were yielding valuable insight (e.g., a case study [GL3][9] retrieved from the methods' creator websites contributed more valuable information than a high scoring paper [WL10].

[8]Academic peer-reviewed-4, PhD/Master thesis-3, Peer-reviewed experience reports-2; Methods' creator case studies, blogs-1; Commentary/opinion-0

[9]In this study, primary grey literature (GL) studies are represented as GL1, GL2,—-, GL22 and white literature (WL) studies are represented as WL1, WL2,—-, WL49

3.3.5. Data Extraction

The relevant information from the primary studies was extracted by using a standard data extraction form which was created in an excel spreadsheet (data extraction form can be found in Appendix B of review protocol [24]) by following the Cruzes and Dyba [29] guidelines. The following information were extracted from each included primary WL and GL study:

Information of publication: Paper ID, Authors, Year of Publication, Title, Venue (where the source was published), Quality score.

Context descriptions: Study Type (according to [30] classification[10] i.e., evaluative study, validation study, solution proposal, philosophical papers, opinion papers, experience papers), Settings (country/location of the analysis).

Findings: Relevance to the theme, (i.e., RE challenges, and RE strategies)

[10]Description of Wieringa et al. (2006) study classification can be found in Appendix C of review protocol [24]

Table 2: Quality evaluation of selected WL studies

Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Total
WL1	1	1	1	1	1	1	1	1	1	1	4	14
WL2	1	1	1	1	0	1	1	1	1	1	4	13
WL3	1	1	1	1	0	1	1	1	1	1	4	13
WL4	1	1	1	1	0	1	1	1	1	1	4	13
WL5	1	1	1	1	0	1	1	1	1	1	4	13
WL6	1	1	1	1	0	1	1	1	1	1	4	13
WL7	1	1	1	1	0	0	1	1	1	1	2	10
WL8	1	1	1	1	0	1	1	1	1	1	4	13
WL9	1	1	1	1	0	1	1	1	1	1	4	13
WL10	1	1	1	1	1	1	1	1	1	1	4	14
WL11	1	1	1	1	1	0	1	1	1	1	2	11
WL12	1	1	1	1	0	1	1	1	1	1	4	13
WL13	1	1	1	1	0	1	1	1	1	1	4	13
WL14	1	1	1	1	0	1	1	1	1	1	4	13
WL15	1	1	1	1	0	0	1	1	1	1	2	10
WL16	1	1	1	1	0	0	0	1	1	1	2	9
WL17	1	1	1	1	0	0	1	1	1	1	2	10
WL18	1	1	1	1	0	1	1	1	1	1	4	13
WL19	1	1	1	1	0	1	1	1	1	1	4	13
WL20	1	1	1	1	0	1	1	1	1	1	4	13
WL21	1	1	1	1	0	0	1	1	1	1	2	10
WL22	1	1	1	1	0	1	1	1	1	1	4	13
WL23	1	1	1	1	0	1	1	1	1	1	2	11
WL24	1	1	1	1	0	1	1	1	1	1	4	13
WL25	1	1	1	1	0	1	1	1	1	1	4	13
WL26	1	1	1	1	0	1	1	1	1	1	4	13
WL27	1	1	1	1	1	1	1	1	1	1	4	14
WL28	1	1	1	1	1	1	1	1	1	1	4	14
WL29	1	1	1	1	1	1	1	1	1	1	4	14
WL30	1	1	1	1	0	1	1	1	1	1	4	13
WL31	1	1	1	1	1	1	1	1	1	1	4	14
WL32	1	1	1	1	0	1	1	1	1	1	4	13
WL33	1	1	0	1	0	0	1	1	1	1	2	9
WL34	1	1	1	1	0	1	1	1	1	1	4	13
WL35	1	0	1	1	0	0	1	1	1	1	2	9
WL36	1	1	1	1	0	1	1	1	1	1	4	13
WL37	1	1	1	1	0	1	1	1	1	1	4	13
WL38	1	1	1	1	1	1	1	1	1	1	4	14
WL39	1	1	0	1	0	0	1	1	1	1	2	9
WL40	1	1	0	1	0	0	1	1	1	1	2	9
WL41	1	1	1	1	0	0	1	1	1	1	4	12
WL42	1	1	0	1	0	0	1	1	1	1	2	9
WL43	1	1	1	1	0	0	1	1	1	1	4	12
WL44	1	1	1	1	0	0	1	1	1	1	4	12
WL45	1	1	1	1	0	0	1	1	1	1	2	10
WL46	1	1	1	1	0	1	1	1	1	1	4	13
WL47	1	1	1	1	0	1	1	1	1	1	4	13
WL48	1	1	1	1	0	0	1	1	1	1	2	10
WL49	1	1	1	1	0	1	1	1	1	1	4	13

Table 3: Quality evaluation of selected GL studies

Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Total
GL1	1	0	0	0	0	0	0	1	0	1	1	4
GL2	1	0	0	0	0	0	0	1	0	1	1	4
GL3	1	0	0	0	0	0	0	1	0	1	1	4
GL4	1	0	0	0	0	0	0	1	0	1	1	4
GL5	1	0	0	0	0	0	0	1	0	1	1	4
GL6	1	1	1	1	1	0	1	1	1	1	1	10
GL7	1	1	1	1	0	0	0	1	1	1	1	8
GL8	1	1	1	1	1	0	1	1	1	1	1	10
GL9	1	1	1	1	1	0	0	1	1	1	1	9
GL10	1	1	1	1	0	0	1	0	1	1	1	8
GL11	1	1	1	1	0	0	1	1	1	1	1	9
GL12	1	0	0	1	1	1	1	1	1	1	3	11
GL13	1	0	0	1	1	1	1	1	1	1	3	11
GL14	1	0	0	1	0	1	1	1	1	1	3	10
GL15	1	1	0	1	0	1	1	1	1	1	3	11
GL16	1	0	0	0	0	0	0	1	1	1	1	5
GL17	1	0	0	0	0	0	0	1	1	1	1	5
GL18	1	0	0	1	0	0	1	1	1	1	1	7
GL19	1	0	1	1	1	0	0	1	1	1	1	8
GL20	1	0	0	1	0	0	0	1	0	1	1	5
GL21	1	0	0	1	0	0	1	1	0	1	1	6
GL22	1	0	0	1	0	0	0	1	1	1	1	6

3.3.6. Data Synthesis

Extracted data from Section 3.3.5 was analysed by following the thematic synthesis steps recommended by [29]. The synthesis technique that was used for this study was inductive and data-driven, i.e., Themes were purely from the extracted data [29]. Following are the steps that were undertaken to synthesize the extracted data:

Step 1: Extract data: All included primary studies were read several times to extract the relevant information and extracted the data using a data extraction form.

Step 2: Code data: Codes were generated from the segment of text extracted from the primary studies. We have obtained 186 codes as challenges and 127 codes as strategies from the analysis of WL. On the other hand, we have found 40 codes as challenges and 30 codes as strategies from the GL studies[11].

Step 3: Translate codes into themes: themes were constructed from a comprehensive review of the Codes extracted from Step 2. Next, extracted themes were combined on the basis of their relationship with each other, yielding sub-themes under main theme.

Step 4: Reviewing themes: extracted themes were reviewed in two iterations. First, re- read each sub-theme that was placed under the main theme in order to check the sub-theme was still relevant to the proposed theme or is the sub-theme need to move under another theme. Second, a discussion about the themes was performed with the secondary and third researcher which resulted in the significant re-wording of the themes' names and themes' re- organization.

As a result of these steps, several challenges and suggested strategies regarding RE in GSASD have been identified which are discussed in Section 4.

4. Results

This section presented the findings of the study. In the first subsection, an overview of the selected studies is provided. The next subsection presents the findings related to the research questions.

4.1. General findings

This sub-section presents the distribution of studies, and research types of the primary studies.

[11]All raw codes related to WL studies and GL studies are available for download via https://github.com/priya020/MLR-Data in the form of an Excel Spreadsheet. The list of selected studies is not included here due to space limitation

4.1.1. Distribution of studies according to publication categories

The 71 primary studies were selected from a variety of venues as shown in Table 4. As displayed in Table 4, conference proceedings were the most dominant venue (28 studies). Methods' creator websites yielded 18 studies, and 14 studies from the journals. In terms of thesis and workshop proceedings, four studies were produced from each venue. Regarding studies as book chapters, only three studies were included in this MLR study.

Table 4: Distribution of studies based on publication venues.

Type	Venues (number of studies)
Journals	JSS(3), IST(4), IEEE Transections on Soft. Eng.(1), EmpiSE(5), J. of Software: Evolution and Process(1)
Conference Proceedings	ESEM(3), C3S2E(1), ICGSE (6), Agile Conference (3), ICSE(1), HICSS(2), SEAA(1), XP(7), RE(2), ENASE(1), ECSA(1)
Workshop Proceedings	XP(3), AREW(1)
Book Chapters	Agility across time and space(2), RE for service and cloud computing(1)
Methods creator websites	DAD(6), LeSS(5), SAFe(4), Nexus(2), Scrum-of-Scrum(1)
Thesis	Masters' thesis(4)

4.1.2. Findings for Research Type

The classification scheme[12] of Wieringa et al. [30] was adopted to determine the research type of the selected studies: evaluative research (e.g., case study), validation research (e.g., laboratory experiments), solution proposal, philosophical papers (e.g., conceptual framework), opinion papers, experience papers.

Table 5 displayed the classification of primary studies over research type. As shown in Table 5, evaluative was the most common research type in our MLR (32 out of 71 studies, 45%). This result indicated that largest proportion of the primary studies explored the topic of this research study in a real world context (e.g., particular organization). 35% of studies were classified as experience papers. Other types of research were: opinion papers (10%), philosophical papers (10%), validation research (0%), and solution proposal (0%).

Table 5: Classification based on research type

Research type	Number of studies	Percentage
Evaluative Research	32	45%
Validation Research	0	0%
Solution Research	0	0%
Philosophical Research	7	10%
Opinion Research	7	10%
Experience Research	25	35%

4.1.3. Supplementary search

To enhance the accuracy and currency of the work, a supplementary search was performed on Google Scholar (GS) to include studies published between Jan 2019 to June 2020. The main reason to perform a supplementary search on GS was that it searches the relevant studies on a wide range of sources (e.g. conference proceedings, online repositories) [21] which yielded 164 studies. The studies were screened by following the same procedure that were applied to retrieve WL primary studies as mentioned in Section 3.3.1. As a result of this, seven relevant primary studies were retrieved. "However", by analysing the full text of the selected studies, no significant themes emerged that would amend the existing findings of the study.

4.2. Findings related to the RE challenges in global scaled agile software development

The analysis of the literature discovered that agile methods were posing various challenges related to RE in large software development organizations where teams are globally distributed. In the following subsections, challenges are grouped and summarized using thematic synthesis. The four main themes related to RE challenges and the related papers are summarized in Table 6, and subsequently elaborated in more depth.

[12]Description of Wieringa et al. [30] can be found in Appendix C of review protocol [24]

Table 6: RE challenges themes and related papers

Challenge Theme	WL studies	GL studies
Difficulties in coordinating and communicating the requirements across teams	WL5, WL7, WL11, WL17, WL18, WL24, WL26, WL28, WL34, WL35, WL36, WL38, WL48, WL49	GL1, GL12, GL5, GL13, GL15, GL16, GL20
Over-scoping of requirements across scaling agile levels	WL18, WL16, WL24, WL26, WL28, WL29, WL30, WL31, WL36, WL38, WL39, WL46, WL44, WL48, WL49	GL3, GL12, GL13, GL15, GL18, GL2
Negligence of QRs in the early stages of requirements	WL1, WL3, WL10, WL27, WL31, WL34, WL36, WL45, WL47, WL48, WL17, WL33, WL31, WL2, WL18, WL16, WL41, WL30	GL8, GL10, GL20, GL14, GL21, GL2
Difficulty in aligning the priorities of requirements across scaling agile levels	WL1, WL3, WL4, WL7, WL8, WL11, WL13, WL14, WL19, WL20, WL23, WL24, WL25, WL28, WL30, WL32, WL37, WL43,WL46, WL33, WL42, WL29, WL22, WL30, WL48	GL12, GL13, GL14, GL15, GL5, GL1, GL6, GL19

4.2.1. Difficulty in coordinating and communicating the requirements across teams

In GSASD, it is difficult to coordinate and communicate the requirements across teams. The following are the factors that indicate difficulty in coordinating and communicating the requirements across teams:

(i) Due to language constraints:

As discussed by Badampudi et al. [31], Hussain [32], Kausar et al. [33], and Roopa et al. [34], in GSASD, there might be misunderstanding about work items when converting the requirements from one language to another as a result of language constraints. For example, online translating tools lead to requirements' misunderstanding which can create a significant ambiguity among the engineering teams.

(ii) Lack of suitable communication tools:

As discussed by Badampudi et al. [31], and Tripathi et al. [35], in case of globally distributed teams, follow up of the discussions might be difficult. Gusch et al. [36], Hussain [32], and Tripathi et al. [35] noticed that team members who are located in distributed locations may not be interested in participating in the requirements planning meetings due to the lack of good online conferencing equipment.

(iii) Lack of proper requirement traceability mechanism:

Agile methods indicate the use of the program board which is a physical display of work items dependencies across the teams. "However", in large scale agile projects, often multiple teams (usually the globally distributed teams) are involved in the implementation of a single feature. Therefore, this process (i.e., Program board) is not efficient when the teams are globally distributed and not able to attend in-person meeting [31, 37, 32, 38].

(iv) Due to global distance:

In agile software development, there should be a frequent collaboration between the product owner (PO) and the engineering teams [8]. However, in large scale agile, teams are often globally distributed [25] that may impede frequent collaboration and limits in-depth probing with PO. The lack of frequent collaboration between the PO and the engineering in a globally distributed setup may result lack of shared understanding on the work item list and will often result in extended effort required to deliver software as expected [39, 40, 25, 31, 41, 32, 14, 42, 43, 44, 35, 45].

(v) Lack of understanding on vital scaled agile practices:

Some of the basic agile methods practices for defining a work item list such as theme, epic, feature, user stories and definition of done (DoD) etc. are now critical practices with scaled agile frameworks (e.g., SAFe, DAD, Scrum @ scale). These practices are common organizational practices, regardless of geographical location of the teams, in order to have strategic releases of market quality (bug free) software. Therefore, lack of understanding of common organizational practices may result in poorly defined work items specifically capturing the right level of detail and granular-

ity [31, 37, 46, 47, 32, 48, 35]
Impact: Organizational stress may be created due to poorly defined work items [49].

4.2.2. Over-scoping of requirements across scaling agile levels

In GSASD, the goal at the program level is to make work easier at the project level- a lot of analysis and design work is done at the program level which feeds into projects. "However", often work items/projects which are allocated to the engineering teams are beyond their capacity [46, 50]. The following are the factors that indicates over-scoping of work items in GSASD.
(i) Large-sized work items (High-level abstract requirements-high uncertainty-do not know well):
As discussed by Evbota et al. [46], and Usman et al. [50], upfront plans that includes schedules and duration of large-sized work items are challenging in all circumstances. Therefore, ability to adapt and flexibility is required in large scale agile projects.
(ii) Lack of expertise for developing achievable plans:
As discovered by Evbota et al. [46], Usman et al. [50], and Wildt et al. [51] , lack of involvement of suitable roles introduces unreliable plans. Therefore, this requires adaptation through appropriate organizational structures (functional units, roles and practices) for a collective effort for developing reliable plans.
(iii) Lack of awareness of team maturity:
In GSASD, it may be difficult to know the development teams capacity at the enterprise level especially in case of newly joined team members which may result over-scoping of work items [31, 35, 50].
(iv) Volatile and late requirements changes:
As discussed by Bjarnason et al. [52] , Evbota et al. [46], Hannay et al. [45], Roman [42], and Usman et al. [50], issues relating to requirements such as late changes in requirements results in unreliable estimates being provided during planning sessions which later impacts the delivery of short development cycles.
(v) Distributed team:
In a distributed development setting, wherein development teams are often located in various geographical locations, distribution exacerbates the underestimation bias over that of the co-located teams [33, 50].
(vi) Business Pressure:
As noted by Heikkila et al. [53], and Tripathi et al. [35], the specified development team capacity may be overlooked by the product management due to the business pressure to get the release ready which may later impact teams productivity.
(vii) Uncertainty about feature size:
In GSASD, large end-to-end features which usually requires several engineering teams may obstruct the development pipeline. Therefore, breaking the features into manageable size, while keeping the minimum dependencies needs to be resolved [52, 32, 54, 55, 35, 50].
(viii) Balancing planning effort:
Creating reliable plans (i.e., product backlog, sprint/iteration and daily) with sufficient details within the expected time-frame for development to start is often challenging and requires a balancing approach in GSASD [37, 47, 56, 53, 32, 48, 57, 58, 35].

4.2.3. Negligence of QRs in the early stages of requirements

In GSASD, an adequate attention is not given to the QRs in the early stages (i.e., analysis and design) of the requirement elicitation process [59, 60, 34, 43]. As noted by Alsaqaf et al. [61], Badampudi et al. [31], Bjarnason et al. [52], and Dingsoyr et al. [40], most of the QRs are identified in the later stages of a project.
Impact: Discovering QRs during or after sprint/iteration implementation (development and testing) makes it complex to implement them [61]. Moreover, software architecture may become inappropriate due to lack of upfront analysis [62, 31, 63, 64].
The following are the factors that indicate QRs are not given appropriate attention in the requirement elicitation process.
(i) Lack of clear conceptual definition of QRs:
As discussed by Alsaqaf et al. [61], and Daneva et al. [65], a lack of clear conceptual definition of QRs may results different interpretations of QRs and create confusion among team members that can lead to wrong implementation . Therefore, standard practices are needed to document the QRs in GSASD in order to create a common understanding of QRs among team members.
(ii) Stakeholders viewpoints in QRs:
Alsaqaf et al. [61], Badampudi et al. [31], Bjarnason et al. [52], Roopa et al. [34], and van der Heijden et al. [66], have identified that lack of involvement of all relevant of stakeholders in the requirements elicitation may omit their viewpoints in QRs or result in missed important requirements.

(iii) Lack of QRs visibility:
As discussed by Alsaqaf et al. [61], an internal QRs (e.g., maintainability, modifiability, and extensibility) ought to be visible to the stakeholders as their late detection may yield an expensive maintenance cost.
(iv) Appropriateness of QRs specification approaches:
Appropriateness of agile practices (e.g., user stories) to document the QRs and dependencies between QRs needs to be investigated in terms of GSASD [61, 65].
(v) Lack of testing competence within teams:
In agile methods usually the teams are cross- functional, but in case of GSASD, dedicated testing skills may be needed to test complex modules [52, 67].
(vi) Lack of testing tool support and practices:
In GSASD, an appropriate testing tool support and practices (e.g., unit testing, test driven development (TDD), behaviour driven development (BDD) etc.) are needed in order to avoid an unnecessary delay in development pipeline and to produce the quality software (bug free) [47, 56, 68, 69, 54, 42, 44, 55, 70].
(vii) Inability to adapt:
In GSASD, evaluation of team capacity might be affected due to the lack of team maturity. As discussed by Grewal et al. [41], and Schnitter et al. [71], the team decided to complete the work using overtime instead of cutting the scope of the current sprint. As a result of this, the test scenarios were not mapped into acceptance criteria. Therefore, the lack of team maturity caused the team to generate rework and impact the quality of the product.

4.2.4. Prioritization misalignment across scaling agile levels

Large scale globally distributed agile projects have an inherent problem with prioritizing and organizing a work items list due to various organizational levels and structures involved with product planning and development [14, 72].
Following are the factors that cause prioritization misalignment across the various scaling levels driven by agile approach.
(i) Prioritization mismatch among stakeholders:
In GSASD, pressure on the team from the product management (external pressure) and within their team (internal pressure) to deliver the features may impede the communication in a team [73, 25, 71, 74]. As noted by Kalenda et al. [73], Paasivaara et al. [25], and Schnitter et al. [71], the internal and external pressure to deliver the features often lead to development teams skipping or postponing their team meetings.
Impact: Lack of communication between the product and engineering, and within the engineering teams can cause misunderstanding in requirements which may cause productivity issues [73, 25, 39, 75, 76].
(ii) Insufficient competence across scaling levels:
In agile projects, product management usually set work items priorities based on their business perspectives (i.e., business value) which the engineering teams usually lacked. On the other hand, an engineering team usually gives more importance to the technical items which the product management usually lacked [77, 78] that makes reaching a consensus challenging. Therefore, sequencing of work items requires rules or governance (e.g., allocate dedicated resources to handle technical items). Rules around prioritization provides clarity around scheduling work in short development cycles [78].
(iii) Work items dependencies:
In GSASD, an inadequate decomposition of work items may result unnecessary dependency across the development teams that can also impact the work items priorities [26, 25, 78]. Scheerer et al. [79] and Pavlichenko [80] discovered that the greater the work items dependencies, the lower the degree of freedom the Product management have to prioritize the work items. Therefore, proper planning is required and mechanisms need to be investigated to minimize the work items dependencies in GSASD.
(iv) Retention level of a shared vision:
In GSASD, usually a flat structure but with several POs, coordination could be a problem between teams, within the program and with product management. There is the potential to develop into a hierarchical setup, creating several issues- decisions could be delayed and take time to make, causing productivity issues [46, 54, 81].
(v) User story limitation due to its structure:
In agile methods, user story is an artefact that is typically used to document the specific business requirements and other information. "However", the structure of a user story might not be sufficient to capture the necessary details in case of GSASD that are needed to make an optimal decision on work items priorities. Therefore, appropriateness of user stories needs to be investigated regarding large scale globally distributed projects [63, 65, 48].

(vi) Quantity vs. Quality focus:
In GSASD, product management often focusing on implementing and releasing as many features as possible means less attention is given to the QRs. As a consequence of this, technical debt is accumulating in GSASD [53, 82, 54, 34]. Therefore, mechanisms need to be investigated to balance this trade-off.

4.3. Strategies for surmounting the RE challenges in global scaled agile software development

From the analysis of the literature, the strategies that can be used for surmounting the RE challenges have been discovered. Strategies are grouped into seven categories using thematic synthesis which are elaborated in the following subsections.

4.3.1. Encourage communication, collaboration and transparency

(i) Introduce knowledge transfer sessions

- Community of practice (CoP) where a group of people who have a common interest in a specific business or technical domain regularly collaborates to enhance their knowledge [36, 32, 34, 83, 43].
- Periodic visits: to improve the shared understanding of the work items among team members, offshore team members should be visited by the onshore team members and vice versa [33].

(ii) Adapt an appropriate tool set (Digitize the program board)
This practice can be used to manage and maintain the work items dependencies across geographically distributed multiple teams.
Benefit: The digital program board helps in tracing the work items dependencies, and also solves most of the challenges that distributed teams face (e.g., visibility of work) [84, 63, 56, 36, 32, 85, 33, 57, 35].
(iii) Work and improve collaboratively
Following are the practices that can be used to work and improve collaboratively in GSASD.

- As discussed by Roopa et al. [34] and Sablis et al. [86], define an escalation matrix as this matrix helps in identifying where to contact to resolve a particular issue (e.g., team members could contact the PO if they are lacking clarity related to work items).
- Product management/steering committee can help the POs by offering them appropriate tools or expertise in leading the software development effort [39, 86, 35].
- PO should be located close to the development teams in order to develop a shared understanding of the work items [38].
- All the relevant stakeholders should be involved in the development process (e.g., sprint planning) in order to provide necessary information related to work items (e.g., clarity of work items) to the development teams [38].
- Organize modelling workshops (e.g. requirements and design workshops) to create a shared understanding of the work items among stakeholders [45, 25, 71].

4.3.2. Ensure Training and Education

(i) Applying modelling techniques
Use modelling techniques (e.g., UML and ER diagrams) along with epics, features, and user stories to represent the work items.
Benefit: Modelling techniques helps in improving the understanding of the work items among team members specifically for globally distributed team members due to its ability in breaking the language barrier. Moreover, modelling techniques are beneficial for newly joined team members as these techniques can provide them some good system or domain knowledge [32].
(ii) Learning the RE process of scaling agile methods
Training and education on scaling agile processes and practices should be organized for the teams to acquire quality results in the RE activities [37, 32]. Consulting services, expert coaches, and training materials are highly beneficial for the team members by providing them help in resolving the RE issues [87, 54, 34].
(iii) Provide Training
Hire an experienced consultant to train and educate the team members [65, 88, 80, 58].
(iv) Training personnel on CI
Organize "CI Road Show" for the whole organization. This practice helps in training the personnel on CI, as well as build "CI mind-set" across the entire development organization [41, 73, 54].
(v) Improve team maturity (Pairing)
Team knowledge can be improved by pairing and is a recommended practice for enhancing the team

maturity. For example, a coding dojo can be used to improve the testing skills of a team [89, 90, 54, 42, 70].

4.3.3. Provide management support

(i) Involve expert judgment

Take into consideration that: mature teams provide more accurate estimates due to their knowledge and expertise (i.e. tacit knowledge) [31, 46, 50].

(ii) Mentoring support for less mature teams

Take into consideration that: mentoring support should be provided to the newly on-boarded teams in order to avoid productivity issues [50].

(iii) Monitoring planning session

It is important to monitor the planning sessions in order to prevent extensive discussions without making any progress [35, 50].

4.3.4. Governance Adopted

(i) Skill set for POs

PO must have a RE skill set in a scaled agile set up (e.g., work items need to be broken down by the PO into an efficient manner that can be completed in a short development cycle (i.e., sprint/iteration)) [56].

(ii) Well defined Product Owner team (cross-functional team)

This team ought to consist of senior members that include senior information analysts, business representatives, and senior software architects etc. Their goal would be to define the software architecture up-front, building the product backlog and assigning the backlog items to the development teams based on their skill-set [61, 59, 46, 45, 50, 51].

(iii) Establish QRs specialists team

In GSASD, a team must have an extensive knowledge to take the ownership of a particular QR. For example, a security specialist's team will be responsible for implementing the security requirements across teams. Moreover, an ad-hoc QRs specialist's team might be set to resolve a particular QR related issue [61].

(iv) Establish an independent test team

In large scale agile projects, to produce the higher quality code, an independent test team should be established because of the possibility of reproducing logical errors by the developers in the implemented product [91, 92, 77, 93].

(v) Work collaboratively to manage work items dependencies

Product management and an engineering team should work collaboratively to identify and manage the work items dependencies especially technical dependencies in large scale globally distributed agile projects. An involvement of engineering team will help the product management to understand and covering the potential implications of the technology driven functional dependencies [84, 79].

4.3.5. Provide supporting tools

CI and test automation: To support distributed teams and avoid integration problems, use automated tests and practices like CI and test automation [91, 94, 47, 56, 95, 85, 73, 89, 90, 49, 82, 69, 42, 34, 51].

4.3.6. Solid/appropriate engineering practices

(i) Adopt stop starting and start finishing mantra

- Set the tolerance level: team should focus on quality if the defects are above the tolerance level [63, 85, 73].
- Make it mandatory that code should have passed all unit tests, all the planned automated tests, fulfil the defined completion criteria, and all known defects related to the work items should be fixed [34].
- Reserve resources to implement important QRs (e.g., reserve part of the sprint for QRs) [61].
- Review and prioritize the bugs/defects continuously [63].

(ii) Apply work items prioritization criteria

In large scale agile projects, instead of setting the work items priorities based on the business value, an additional factor such as risks should also be included [65].

(iii) Delivery story as an extension of user story

Delivery stories are the extension of user stories with additional information (i.e., architectural design implications, effort estimation, test scenarios, and risk) and could be an approach to scale agile practices (e.g., user story) in large scale projects [65].

(iv) Maintain an assumption wiki page

The agile teams can maintain an assumption wiki page when a clear description of QRs is missing from the PO and implement the assumptions with functional requirements in different sprints [61].

(v) Incorporate innovation and planning (IP)
This iteration (usually equal to one sprint) can be performed by the agile teams to work on activities that are not visibly delivering business value [61].
(vi) Splitting work items into smaller chunks
Work items should be broken down into smaller pieces and use of approaches such as user story mapping or walking skeleton to establish the connections between work items, are the mechanisms that enable work items dependencies to become transparent early on [84, 79].
(vii) Adopt Creativity techniques in upfront software development stages
In large scale agile projects, work item dependencies need to be not only considered at the intra-team level, but they need to be identified at the inter-team level as well to enhance intra-team work. Ideally, there should be a low degree of dependencies at the inter-team level and the higher degree of dependencies may occur at the intra-team level in large scale agile globally distributed projects [78, 79]. Applying creativity techniques (e.g., design thinking) together with user story mapping in upfront software development stages to create a solid product vision. The other suggested approaches to avoid work item dependencies include plug-in architecture, backlog management, and tracking tools [79].

4.3.7. Use of strategic decision-making strategies

The priority of work items must be matched at the strategic level to enable matching work items priority at the operational level. Some of the strategic decision-making strategies include decision analysis tools, acquire expertise, debasing judgments, analogical reasoning, taking an outside view, and improve the understanding of biases of peers [39, 76].

5. Discussion

As an answer to RQ1: "What are the challenges of requirements engineering in global scaled agile software development?", a total of 26 challenges (discussed in Section 4.2.1) were identified by analysing the primary studies (total 71 primary studies - 49 WL and 22 GL) and grouped them into four categories: difficulties in coordinating and communicating the requirements across teams (14 WL studies and seven GL studies), over-scoping of requirements across scaling agile levels (15 WL studies and six GL studies), negligence of QRs in the early of stages requirements (18 WL studies and six GL studies), difficulties in aligning the priorities of requirements across scaling agile levels (25 WL studies and eight GL studies).
The challenges classification indicates that, there is a diverse set of RE challenges in GSASD, but the misalignment of requirements prioritization across scaling agile levels is the most commonly occurring challenge in WL (25 studies out of 49) as well as in GL (8 studies out of 22). This challenge was also mentioned in the "Systematic Literature Review on Challenges and Success Factors for Large-scale Agile Transformations" conducted by Dikert et al. [3]. The findings revealed that the main context of the selected studies that discussed challenges of RE in GSASD, was implementing agile methods in globally distributed large organizations and discussed challenges as a part of their implementation. Only eleven studies out of 71 were particularly focusing on RE in GSASD, and out of those eleven studies, there was only one study which was conducted by Daneva et al. [65] that was directly related to the most commonly occurring RE challenge of this MLR study i.e., misalignment of requirements prioritization across scaling agile levels, but the complete life cycle of prioritization of requirements (i.e., from an idea to operational level) was lacking in their study. Moreover, the Daneva et al. [65] study was notably devoted to software vendors only. Therefore, other organizations that include software-intensive organizations and large enterprises might not get full benefits from their study.
Another significant observation is that the majority of the included studies discussed RE challenges in GSASD basically at the project level/program level. Portfolio related evidence in terms of RE challenges is largely missing in the literature. It is also important to note that WL studies strike a good balance between challenges and benefits of GSASD, while very few GL studies (e.g., experience reports from methods creator websites) reported challenges. Experience reports that were found through the consultants' websites mainly discussed positive outcomes such as benefits that they achieved after implementing the scaling agile methods. This suggested positive bias and the unlikelihood of negative findings being published in the GL. This positive bias was the reason, this MLR study could not find ample GL studies (only 22 studies out of 71) related to the challenges of RE in GSASD.
To answer RQ2: "What are the strategies to overcome requirements engineering challenges in

global scaled agile software development?", this study identified 25 strategies (discussed in Section 4.3) grouped into seven categories: encourage communication, collaboration and transparency, ensure training and education, provide management support, governance strategies adopted, provide supporting tools, apply solid/appropriate engineering practices, and implementation of strategic decision-making strategies. The RE strategies that were extracted and synthesized from the literature were mapped with the RE challenges of GSASD.
Strategy mapping w.r.t challenges are shown in Table 7 (shows strategy mapping w.r.t to sub-challenges related to misaligning of priority of requirements across scaling agile levels challenge) ,Table 8 (shows strategy mapping w.r.t to sub-challenges related to Negligence of QRs challenge) ,Table 9 (shows strategy mapping w.r.t to sub-challenges related to Difficulty in coordinating and communicating the requirements across teams challenge), Table 10 (shows strategy mapping w.r.t to sub-challenges related to Over-scoping of requirements challenge).

Table 7: Misaligning priority of requirements across scaling agile levels

Sub-Challenges	**Strategies**
Prioritization mismatch among stakeholders	Introduce knowledge transfer sessions, work and improve collaboratively, learning RE process of scaling agile methods
Insufficient competence across scaling levels	Apply prioritization criteria, well defined decision making team
Work items dependencies	Well defined decision making team, work collaboratively, provide training, adopt creativity techniques in upfront software development stages
Retention level of shared vision	Well defined decision making team, apply prioritization criteria
User story limitation due to its structure	Delivery story an extension of user story
Quantity vs. Quality focus	CI and test automation, adopt stop starting and start finishing mantra

To sum up, This MLR study discovered that many of the strategies that were reported in the literature

Table 8: Negligence of QRs

Sub-Challenges	**Strategies**
Lack of clear conceptual definition of QRs	Maintain an assumption wiki-pages
Lack of stakeholders viewpoints in QRs	Establish QRs specialist team, incorporate an IP iteration, well defined decision making team
Lack of awareness of team maturity	Involve expert judgement, mentoring support for less mature teams
Appropriateness of agile practices to document QRs	Maintain an assumption wiki-pages, incorporate an IP iteration
Distributed team	Introduce knowledge transfer sessions, work and improve collaboratively
Lack of QRs visibility	Establish QRs specialist team, maintain an assumption wiki-pages
Inability to adapt	CI and test automation, improve team maturity
Lack of testing tool support and practices	CI and test automation
Inefficient testing competence within cross-functional teams	Establish an independent test team, CI and test automation

for surmounting the RE challenges in GSASD were at the proposal level, empirical evaluation of those strategies was weak. Our findings about the lack of empirically evaluated RE methods in GSASD concur with the results of the SLR performed by Inayat et al. [13] and the systematic mapping study conducted by Heikkila et al. [15].
Another significant observation is that most of the strategies that were suggested in the literature to overcome the RE challenges in GSASD were the combination of plan-driven approaches and agile practices which requires further empirical investigation to prove this claim.

6. Threats to validity

The Ampatzoglou et al. [96] guidelines were followed to address the threats to validity of this MLR study. Ampatzoglou et al. [96] classified threats to

Table 9: Difficulty in coordinating and communicating the requirements across teams

Sub-Challenges	**Strategies**
Lack of suitable communication tools	Introduce knowledge transfer sessions, applying modelling techniques, work and improve collaboratively, digitize program board
Due to language constraints	Introduce knowledge transfer sessions, applying modelling techniques, work and improve collaboratively
Lack of proper requirements tracebility mechanism	Introduce knowledge transfer sessions, applying modelling techniques, work and improve collaboratively, digitize program board, learning RE process of scaling agile methods
Global distance	Introduce knowledge transfer sessions, applying modelling techniques, work and improve collaboratively, digitize program board, learning RE process of scaling agile methods
Lack of understanding on scaling agile practices	Introduce knowledge transfer sessions, provide training, work and improve collaboratively

Table 10: Over-scoping of requirements

Sub-Challenges	**Strategies**
Large-sized work-items	Learning RE process of scaling agile methods, provide training, work and improve collaboratively, involve expert judgement
Lack of expertise for developing achievable plans	Involve expert judgement, well defined decision making team, work and improve collaboratively
Lack of awareness of team maturity	Involve expert judgement, mentoring support for less mature teams
Volatile and late requirements changes	Involve expert judgement
Distributed team	Introduce knowledge transfer sessions, work and improve collaboratively
Business Pressure	Well defined decision making team
Uncertainty about feature size	Well defined decision making team, skill-set for PO
Balancing planning effort	Well defined PO team, work and improve collaboratively, learning RE process of scaling agile methods, provide training

validity of SE secondary studies into three useful categories, which were followed in this study: study selection validity, data validity, and research validity.

6.1. Study selection validity

Threats involved in this category can be discovered in the planning phase of the review which may lead to either inclusion of irrelevant studies or the exclusion of relevant ones [96].

Following are the potential study selection validity threats that were addressed in this MLR study.

6.1.1. Adequately identified all relevant studies

An imperfect search process may lead to missing relevant primary studies. Following are the strategies that were followed to limit this threat:

(i) A review protocol was defined [24] which comprises RQs, search strategies, inclusion/exclusion criteria, selection strategies, and data synthesis methods by following the guidelines provided by Kitchenham and Charters [21], and Garousi et al. [22]. Firstly, the primary researcher designed the review protocol and then the protocol was reviewed by the secondary and third researchers as they have experience in empirical SE, including secondary studies.

(ii) Regarding the WL, well-known digital databases were selected to perform an automatic search (as shown in Section 3.1.3) containing publications of highly reputed conferences and journals [21]. Moreover, manual search was also performed on key conferences (as shown in Section 3.1.3). Furthermore, snowballing was also per-

formed on the selected primary studies. On the other hand for GL, methods' creator websites (as shown in Section 3.1.3) were the main source. Case studies and blogs that were published there were included, to retrieve all possible studies related to the phenomena of interest.
(iii) The search string was systematically constructed by following the PICO framework (as shown in Section 4.1 of the review protocol [24]). Moreover, a trial search has been performed as well (as shown in Appendix A of the review protocol) to train the search string [24].

6.1.2. Study inclusion/exclusion criteria

The potential studies may be missed due to inadequate study inclusion/exclusion criteria. The risk of losing relevant sources was mitigated by following the strategies as discussed below:
(i) Inclusion/exclusion criteria were defined in the Section 4.4 of review protocol [24].
(ii) The potential primary studies were evaluated against inclusion/exclusion criteria by the primary researcher. After completing this process, a set of randomly selected primary studies were screened by the secondary and third researcher in order to validate the selection based on inclusion criteria.

6.1.3. Inclusion/exclusion of grey literature

The GL is included as the aim of this research study was to gain all relevant information on the phenomena of interest from academic researchers as well as from the practitioners.

6.1.4. Managed duplicate studies

A consistent strategy was developed to remove the duplicate studies. Studies that appeared more than once were found and removed by using an automated procedure (Mendeley referencing tool). Furthermore, as the inclusion of GL, in some cases, masters/PhD thesis and masters/PhD work as a journal paper were both available. In those cases, it was decided that (after consultation with the secondary and third researcher) masters/PhD work as a journal paper was given higher preference than the masters/PhD thesis for the main study for maximizing the study quality.
This study did not suffer from the risk of "papers published in a limited number of journals and conferences" [96] as the search was performed in five commonly used digital libraries for empirical software engineering, including SCOPUS which indexes papers from several publishers. In addition, GL search was also performed to retrieve all relevant studies on the phenomena of interest.
Also, the "missing non-English papers" threat was not applicable in this study as the aim of this study was to retrieve studies that were published in English only. Finally, no papers were missed due to lack of access as full access database facility was provided by our research institute.

6.2. Data validity

Threats involved in this category can be found in the conducting and reporting phases of the review that may lead to unreliable findings and conclusions [96].
Following are the potential data validity threats that were addressed in this MLR study.

6.2.1. Data extraction bias

In this MLR study, the primary researcher extracted and recorded manually all relevant data. "However", this procedure inserted some subjectivity that was mitigated by defining the review protocol based on strict guidelines [22, 21]. Also, a random selection of papers was reviewed by the secondary and third researcher, which were previously reviewed by the primary researcher to cross check the extracted data.

6.2.2. Publication bias

In this MLR study, WL as well as GL were searched to identify relevant information on the phenomena of interest but GL studies were more likely to report positive results [96, 22]. During this MLR study, it was also noticed the presence of publication bias in the GL which tends not to report negative results. On the other hand, WL studies strike a good balance between the positive and negative results which can mitigate the publication bias in GL.

6.2.3. Low quality of primary studies

Regarding WL studies, quality was not a problem as they were retrieved from top software engineering venues. On the other hand, in case of GL studies, quality is always a significant concern [22]. Therefore, to limit this threat GL studies were retrieved only from the recognised methods' creator websites as GL studies (e.g., case studies) that were available on methods' creator websites and had therefore gone through some validation process before being published on their websites [97, 98]. Moreover,

a quality assessment checklist (shown in Section 3.3.4) was developed to evaluate the quality of primary studies by following the guidelines provided by Kitchenham and Charters [21], and Garousi et al. [22].

6.2.4. Bias of classification schema

To limit this threat, a carefully staged process to extract themes as recommended by Cruzes and Dyba [29] was followed as shown in Section 3.3.6 and extracted themes were cross checked with the secondary and third researchers.

6.2.5. Researcher's bias in data interpretation and analysis

Pilot data analysis was performed to mitigate this threat and extracted data were cross checked with the secondary and third researchers.

6.2.6. Selection of variables to be extracted

Selection of variables were tightly linked with the RQs and discussed with secondary and third researchers as well. Some variables were selected by following the RE taxonomy suggested by Wieringa et al. [30].

6.2.7. Lack of relationship

Relationships among themes were drawn by applying the Cruzes and Dyba [29].

Finally, following are threats that were not applicable in our MLR study:

(i) Small sample size: this study retrieved 72 primary studies that offer a considerable amount of data.

(ii) Lack the use of statistical analysis: this study was not intending to do quantitative data analysis.

6.3. Research validity

In all phases of the review (i.e. planning, conducting, and reporting), research validity threats can be found [96].

Following are the potential research validity threats that were addressed in this MLR study.

6.3.1. Repeatability

To limit this threat, a detailed protocol was designed and progressively updated to reflect the actuality of the study, by the primary researcher based on the Kitchenham and Charters [21] and Garousi et al. [22] guidelines and then reviewed by the secondary and third researcher. Furthermore, all relevant data of this study are publicly available including extracted data from primary studies, search process history (search strings and their results), raw codes to ensure other researchers can replicate this study.

6.3.2. Research methods bias

The review protocol [24] and discussions among the researchers were involved in this MLR study to mitigate the research method bias threat.

6.3.3. Experience of the author team

Although, the primary researcher was new to the research area, the secondary and third researcher were experienced in the field of empirical software engineering, including secondary studies.

6.3.4. Generalizability

In order to generalize the findings of this study, a wide range of studies have been examined related to requirements engineering in scaled agile global software development, without any focus on particular scaling agile methods (e.g., SAFe, DAD, LeSS etc.) and software organization. Findings are therefore considered broadly applicable to the distributed agile setting, in contexts such as software vendors, software intensive organizations, and enterprise contexts.

6.3.5. Lack of comparable studies

Requirements engineering in global scaled agile setting was a new area of focus. The closest comparable study was "quality requirements in large scale distributed agile projects- a systematic literature review" which was conducted by Alsaqaf et al. [19]. Researchers' of this MLR study were informed by that, but because of this study intended to investigate the new phenomena using a different approach, specific challenges were found related to RE in GSASD and strategies to overcome those challenges as discussed in Section 4 that were classified by following the process as suggested by Cruzes and Dyba [29].

7. Conclusion and future work

The findings of a multi-vocal literature review study on RE in globally distributed large scale agile software development are presented in this paper. Kitchenham and Charters [21], and Garousi et al. [22] guidelines were followed to perform this

MLR study. A total of 71 studies were identified through this MLR study. Out of 71 studies, 49 were WL papers (e.g., journal articles, conference papers) and 22 were GL papers (e.g., blog posts, white papers) describing challenges of RE in GSASD and strategies to mitigate those challenges.

Although publications related to agile practices in GSASD are increasing, the researchers of this study observed that a clear conceptual definition of large scale agile is missing in the literature which limits the understanding of the phenomenon. Further it was noticed that the majority of the GL studies focused on the benefits of the scaling agile frameworks (e.g., SAFe, DAD, LeSS) and rarely discussed the challenges. On the other hand, WL studies presented a good balance between benefits and challenges.

This review discovered a diverse set of RE challenges in globally distributed large scale agile software development that include difficulty in coordinating and communicating the requirements across teams, difficulty in aligning priorities of requirements across scaling agile levels, over-scoping of requirements, and negligence of non-functional requirements in the early of stages requirements. But the challenge that received the most mentions is misaligning the priorities of requirements across scaling agile levels.

This review identified strategies as well that include encourage communication, collaboration and transparency, provide supporting tools, provide management support, appropriate/solid engineering practices for surmounting the RE challenges in GSASD. In addition to this, most of the identified strategies were the mix of plan-driven approaches and agile practices but an empirical research study is required to evaluate these strategies' scalability to a real-life context (e.g., a particular organization).

This review also revealed that empirical studies that provide an in-depth understanding of the RE process, RE challenges, and effective RE practices in globally distributed large scale agile software development are lacking and present an area that needs attention for future research.

In the future work, interviews will be conducted in two large globally distributed organizations employing scaling agile methods (e.g., SAFe, DAD) to validate the findings of this MLR study.

References

[1] R. Lal, T. Clear, Enhancing product and service capability through scaling agility in a global software vendor environment, in: ICGSE '18: Proceedings of the 13th International Conference on Global Software Engineering, ACM/IEEE, 2018, pp. 54–63. doi:10.1145/3196369.3196378.

[2] R. Lal, T. Clear, Scaling agile at the program level in an australian software vendor environment: A case study, in: ACIS 2017 Proceedings, 2017, pp. 1–12. doi:aisel.aisnet.org/acis2017/81.

[3] K. Dikert, M. Paasivaara, C. Lassenius, Challenges and success factors for large-scale agile transformations: A systematic literature review, Journal of Systems and Software 119 (2016) 87–108.

[4] V. Garousi, M. Felderer, M. V. Mäntylä, The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature, in: Proceedings of the 20th international conference on evaluation and assessment in software engineering, 2016, pp. 1–6. doi:10.1145/2915970.2916008.

[5] M. Al-Zewairi, M. Biltawi, W. Etaiwi, A. Shaout, Agile Software Development Methodologies: Survey of Surveys, Journal of Computer and Communications 5 (2017) 74–97. doi:10.4236/jcc.2017.55007.

[6] A. M. M. Hamed, H. Abushama, Popular agile approaches in software development: Review and analysis, in: 2013 International Conference on Computing, Electrical and Electronic Engineering (ICCEEE), IEEE, 2013, pp. 160–166. doi:10.1109/ICCEEE.2013.6633925.

[7] K. Schwaber, Agile Project Management with Scrum, Microsoft Press, Redmond, United States, 2004.

[8] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Information Systems Journal 20 (2010) 449–480. doi:10.1111/j.1365-2575.2007.00259.x.

[9] T. Dingsøyr, N. B. Moe, Towards principles of large-scale agile development, in: Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation, Springer International Publishing, 2014, pp. 1–8. doi:10.1007/978-3-319-14358-3_1.

[10] N. B. Moe, T. Dingsøyr, Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at XP2017, in: Proceedings of the XP2017 Scientific Workshops, 2017, pp. 1–4. doi:10.1145/3120459.3120474.

[11] VersionOne, 12th stage of agile report (2017). URL https://www.stateofagile.com/{#}ufh-i-423641583-12th-annual-state-of-agile-report/473508

[12] P. Zave, Classification of Research Efforts in Requirements Engineering, ACM Computing Surveys (CSUR) 29 (4) (1997) 315–321. doi:10.1145/267580.267581.

[13] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, S. Shamshirband, A systematic literature review on agile requirements engineering practices and challenges, Computers in Human Behavior 51 (2015) 915–929. doi:10.1016/j.chb.2014.10.046.

[14] K. H. Rolland, 'Desperately' seeking research on agile requirements in the context of large-scale agile projects,

in: Scientific Workshop Proceedings of the XP2015, ACM, 2015, pp. 1–6. doi:10.1145/2764979.2764984.

[15] V. T. Heikkilä, D. Damian, C. Lassenius, M. Paasivaara, A Mapping Study on Requirements Engineering in Agile Software Development, in: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications, IEEE, 2015, pp. 199–207. doi:10.1109/SEAA.2015.70.

[16] H. Khalid, M. Ahmed, A. Sameer, F. Arif, et al., Systematic literature review of agile scalability for large scale projects, International Journal of Advanced Computer Science and Applications (IJACSA) 6 (9) (2015) 63–75.

[17] T. Gustavsson, Assigned roles for inter-team coordination in large-scale agile development: a literature review, in: Proceedings of the XP2017 Scientific Workshops, 2017, pp. 1–5. doi:10.1145/3120459.3120475.

[18] A. M. Razavi, R. Ahmad, Agile development in large and distributed environments: A systematic literature review on organizational, managerial and cultural aspects, in: 2014 8th. Malaysian Software Engineering Conference (MySEC), IEEE, 2014, pp. 216–221. doi:10.1109/MySec.2014.6986017.

[19] W. Alsaqaf, M. Daneva, R. Wieringa, Quality requirements in large-scale distributed agile projects:a systematic literature review, in: Requirements Engineering: Foundation for Software Quality, Springer International Publishing, 2017, pp. 219–234. doi:10.1007/978-3-319-54045-0_17.

[20] A. Putta, M. Paasivaara, C. Lassenius, Adopting scaled agile framework (SAFe): a multivocal literature review, in: Proceedings of the 19th International Conference on Agile Software Development: Companion, ACM, 2018, pp. 1–4. doi:10.1145/3234152.3234164.

[21] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical report EBSE-2007-01, Keele University (2007). doi:10.1.1.117.471.

[22] V. Garousi, M. Felderer, M. V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology 106 (2019) 101–121. doi:10.1016/j.infsof.2018.09.006.

[23] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, Manifesto for Agile Software Development (2001).
URL https://agilemanifesto.org/

[24] P. Antil, Requirements engineering in global scaled agile software development environment: a multi-vocal literature review protocol, arXiv preprint arXiv:2004.12647 (2020) 1–20arXiv:arXiv:2004.12647.

[25] M. Paasivaara, C. Lassenius, Scaling Scrum in a Large Globally Distributed Organization: A Case Study, in: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), IEEE, 2016, pp. 74–83. doi:10.1109/ICGSE.2016.34.

[26] V. Heikkilä, K. Rautiainen, S. Jansen, A Revelatory Case Study on Scaling Agile Release Planning, in: 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, 2010, pp. 289–296. doi:10.1109/SEAA.2010.37.

[27] M. Alqudah, R. Razali, A Review of Scaling Agile Methods in Large Software Development, International Journal on Advanced Science, Engineering and Information Technology 6 (2016) 828–837. doi:10.18517/ijaseit.6.6.1374.

[28] N. R. Lackey, A. L. Wingate, The pilot study: one key to research success, The Kansas nurse 61 (11) (1986) 6–7.
URL http://europepmc.org/abstract/MED/3642076

[29] D. S. Cruzes, T. Dyba, Recommended Steps for Thematic Synthesis in Software Engineering, in: 2011 International Symposium on Empirical Software Engineering and Measurement, IEEE, 2011, pp. 275–284. doi:10.1109/ESEM.2011.36.

[30] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requirements engineering 11 (1) (2006) 102–107. doi:10.1007/s00766-005-0021-6.

[31] D. Badampudi, S. A. Fricker, A. M. Moreno, Perspectives on Productivity and Delays in Large-Scale Agile Projects, in: Agile Processes in Software Engineering and Extreme Programming, Springer, Berlin, Heidelberg, 2013, pp. 180–194. doi:10.1007/978-3-642-38314-4_13.

[32] N. Hussain, Requirements engineering for globally distributed teams using scaled agile framework., Ph.D. thesis, Aalto University, Finland (2018).
URL https://aaltodoc.aalto.fi/handle/123456789/32500

[33] M. Kausar, A. Al-Yasiri, Using distributed agile patterns for supporting the requirements engineering process, in: Requirements Engineering for Service and Cloud Computing, Springer, 2017, pp. 291–316. doi:10.1007/978-3-319-51310-2_13.

[34] M. Roopa, V. Mani, C. Sankarasubbiah, Usable Software at the End of Each Takt. A Milestone in the Lean Transformation of a Globally Distributed Software Development Team, in: 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE), IEEE, 2017, pp. 116–120. doi:10.1109/ICGSE.2017.9.

[35] N. Tripathi, P. Rodríguez, M. O. Ahmad, M. Oivo, Scaling Kanban for software development in a multi-site organization: Challenges and potential solutions, in: Agile Processes in Software Engineering and Extreme Programming, Springer, Cham, 2015, pp. 178–190. doi:10.1007/978-3-319-18612-2_15.

[36] L. Gusch, P. Herbai, Case study: Elekta.
URL https://www.scaledagileframework.com/elekta-case-study/

[37] F. Ehrnberg, G. Blide, Incremental requirements engineering in a large-scale agile and safety-critical context : A case study, Ph.D. thesis, Chalmers Uiniversity of Technology, Gothenburg, Sweden (2018).
URL http://publications.lib.chalmers.se/records/fulltext/255469/255469.pdf

[38] M. Korkala, F. Maurer, Waste identification as the means for improving communication in globally distributed agile software development, Journal of System and Software 95 (2014) 122–140. doi:10.1016/j.jss.2014.03.080.

[39] O. Ktata, G. Lévesque, Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects, in: proceedings of the 2nd Canadian conference on computer science and software engineering, 2009, pp. 59–66. doi:10.1145/1557626.1557636.

[40] T. Dingsøyr, N. B. Moe, T. E. Fægri, E. A. Seim, Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation, Empirical Software Engineering 23 (1) (2018) 490–520. doi:10.1007/s10664-017-9524-2.

[41] H. Grewal, F. Maurer, Scaling agile methodologies for developing a production accounting system for the oil gas industry, in: Agile 2007 (AGILE 2007), IEEE, 2007, pp. 309–315. doi:10.1109/AGILE.2007.50.

[42] G. d. C. Roman, Characterizing the presence of agility in large-scale agile software development, Ph.D. thesis, Pontifical Catholic University of Rio de Janeiro, Brazil (2016).
URL https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US20100183655.pdf

[43] E. Sofer, LeSS adoption for a safety & security management product.
URL https://less.works/case-studies/vesecurity.html

[44] Scrum.org, Cathay Pacific Airways Nexus case study.
URL https://www.scrum.org/resources/cathay-pacific-airways-makes-nexus-their-official-scaling-framework-1

[45] J. E. Hannay, H. C. Benestad, Perceived productivity threats in large agile development, in: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2010, pp. 1–10. doi:10.1145/1852786.1852806.

[46] F. Evbota, E. Knauss, A. Sandberg, Scaling up the planning game: Collaboration challenges in large-scale agile product development, in: International Conference on Agile Software Development, Springer, Cham, 2016, pp. 28–38.

[47] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, D. O'Brien, Scaling agile methods to regulated environments: An industry case study, in: 2013 35th International Conference on Software Engineering (ICSE), IEEE, 2013, pp. 863–872. doi:10.1109/ICSE.2013.6606635.

[48] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, B. Kanagwa, Requirements Engineering Challenges in Large-Scale Agile System Development, in: 2017 IEEE 25th International Requirements Engineering Conference (RE), IEEE, 2017, pp. 352–361. doi:10.1109/RE.2017.60.

[49] R. Nori, SAFe Case Study: SproutLoud.
URL https://www.scaledagileframework.com/case-study-sproutloud/

[50] M. Usman, R. Britto, L.-O. Damm, J. Börstler, Effort estimation in large-scale software development: An industrial case study, Information and Software technology 99 (2018) 21–40. doi:10.1016/J.INFSOF.2018.02.009.

[51] D. Wildt, R. Prikladnicki, Transitioning from distributed and traditional to distributed and agile: An experience report, in: Agility Across Time and Space, Springer, 2010, pp. 31–46. doi:10.1007/978-3-642-12442-6_3.

[52] E. Bjarnason, K. Wnuk, B. Regnell, A case study on benefits and side-effects of agile practices in large-scale requirements engineering, in: AREW '11: Proceedings of the 1st Workshop on Agile Requirements Engineering, ACM, 2011, pp. 1–5. doi:10.1145/2068783.2068786.

[53] V. T. Heikkilä, M. Paasivaara, C. Lasssenius, D. Damian, C. Engblom, Managing the requirements flow from strategy to release in large-scale agile development: a case study at ericsson, Empirical Software Engineering 22 (6) (2017) 2892–2936. doi:10.1007/s10664-016-9491-z.

[54] M. Paasivaara, B. Behm, C. Lassenius, M. Hallikainen, Large-scale agile transformation at Ericsson: a case study, Empirical Software Engineering 23 (5) (2018) 2550–2596. doi:10.1007/s10664-017-9555-8.

[55] F. Sferlazza, LeSS adoption at Italte.
URL https://less.works/case-studies/italtel.html

[56] I. Gat, How BMC is scaling agile development, in: AGILE 2006 (AGILE'06), IEEE, 2006, pp. 315–320. doi:10.1109/AGILE.2006.33.

[57] N. Kim, H. Cho, Scaling at Food-tech Startup: Transformation challenges, lessons learned and growth (2018).
URL https://www.agilealliance.org/resources/experience-reports/scaling-at-food-tech-startup-transformation-challenges-lessons-

[58] R. Reinikainen, SAFe case study: Telia Finland.
URL https://www.scaledagileframework.com/case-study-telia-finland/

[59] S. Ambler, 11 strategies for dealing with technical debt (2013).
URL http://disciplinedagiledelivery.com/technical-debt/

[60] S. Ambler, Strategies for capturing quality requirements (2018).
URL http://disciplinedagiledelivery.com/capturing-non-functional-requirements/

[61] W. Alsaqaf, M. Daneva, R. Wieringa, Understanding challenging situations in agile quality requirements engineering and their solution strategies: insights from a case study, in: 2018 IEEE 26th International Requirements Engineering Conference (RE), IEEE, 2018, pp. 274–285. doi:10.1109/RE.2018.00035.

[62] M. A. Babar, An Exploratory Study of Architectural Practices and Challenges in Using Agile Software Development Approaches, in: 2009 Joint Working IEEE/IFIP Conference on Software Architecture and European, Conference on Software Architecture, IEEE, 2009, pp. 81–90. doi:10.1109/WICSA.2009.5290794.

[63] J. M. Bass, Artefacts and agile method tailoring in large-scale offshore software development programmes, Information and software technology 75 (2016) 1–16. doi:10.1016/j.infsof.2016.03.001.

[64] J. Eckstein, Architecture in Large Scale Agile Development, in: Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation, Springer, 2014, pp. 21–29. doi:10.1007/978-3-319-14358-3_3.

[65] M. Daneva, E. Van Der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, R. Wieringa, Agile requirements prioritization in large-scale outsourced system, Journal of Systems and Software 86 (5) (2013) 1333–1353. doi:10.1016/J.JSS.2012.12.046.

[66] A. van der Heijden, C. Broasca, A. Serebrenik, An empirical perspective on security challenges in large-scale agile software development, in: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2018, pp. 1–4. doi:10.1145/3239235.3267426.

[67] S. V. Shrivastava, U. Rathod, A risk management

framework for distributed agile projects, Information and software technology 85 (2017) 1–15. doi:10.1016/j.infsof.2016.12.005.

[68] L. Hinterberg, F. Hoffman, Exploring the scaled agile frame-work in a virtual team setting, Ph.D. thesis, Lund University, Sweden (2018).
URL http://lup.lub.lu.se/luur/download?func=downloadFile{&}recordOId=8950400{&}fileOId=8950409

[69] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study, Empirical Software Engineering 15 (6) (2010) 654–693. doi:10.1007/s10664-010-9136-6.

[70] J. D. Smet, Agfa Healthcare.
URL https://less.works/case-studies/agfa-healthcare.html

[71] J. Schnitter, O. Mackert, Large-scale agile software development at SAP AG, in: Evaluation of Novel Approaches to Software Engineering, Springer Berlin Heidelberg, 2010, pp. 209–220. doi:10.1007/978-3-642-23391-3_15.

[72] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, Journal of systems and software 82 (9) (2009) 1479–1490. doi:10.1016/j.jss.2009.03.036.

[73] M. Kalenda, P. Hyna, B. Rossi, Scaling agile in large organizations: Practices, challenges, and success factors, Journal of Software: Evolution and Process 30 (10) (2018) 1–24. doi:10.1002/smr.1954.

[74] T. Tureček, R. Šmiřák, T. Malík, P. Boháček, Energy Project Story: From Waterfall to Distributed Agile, in: Agile Processes in Software Engineering and Extreme Programming, Springer, 2010, pp. 362–371. doi:10.1007/978-3-642-13054-0_39.

[75] M. Paasivaara, V. T. Heikkilä, C. Lassenius, Experiences in Scaling the Product Owner Role in Large-Scale Globally Distributed Scrum, in: 2012 IEEE Seventh International Conference on Global Software Engineering, IEEE, 2012, pp. 174–178. doi:10.1109/ICGSE.2012.41.

[76] J. Vlietland, H. van Vliet, Towards a governance framework for chains of Scrum teams, Information and Software Technology 57 (2015) 52–65. doi:10.1016/J.INFSOF.2014.08.008.

[77] J. M. Bass, How product owner teams scale agile methods to large distributed enterprises, Empirical Software Engineering 20 (6) (2015) 1525–1557. doi:10.1007/s10664-014-9322-z.

[78] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, A. Heinzl, Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings, IEEE Transactions on Software Engineering 44 (10) (2017) 932–950. doi:10.1109/TSE.2017.2730870.

[79] A. Scheerer, S. Bick, T. Hildenbrand, A. Heinzl, The effects of team backlog dependencies on agile multiteam systems: A graph theoretical approach, in: 2015 48th Hawaii International Conference on System Sciences, IEEE, 2015, pp. 5124–5132. doi:10.1109/HICSS.2015.606.

[80] I. Pavlichenko, Eliminate dependencies, don't manage them (2018).
URL https://www.scrum.org/resources/blog/eliminate-dependencies-dont-manage-them

[81] K. Rautiainen, J. von Schantz, J. Vähäniitty, Supporting scaling agile with portfolio management: Case paf.com, in: 2011 44th Hawaii International Conference on System Sciences, 2011, pp. 1–10. doi:10.1109/HICSS.2011.390.

[82] M. Paasivaara, B. Behm, C. Lassenius, M. Hallikainen, Towards Rapid Releases in Large-Scale XaaS Development at Ericsson: A Case Study, in: 2014 IEEE 9th International Conference on Global Software Engineering, IEEE, 2014, pp. 16–25. doi:10.1109/ICGSE.2014.22.

[83] N. Sekitoleko, F. Evbota, E. Knauss, A. Sandberg, M. Chaudron, H. H. Olsson, Technical Dependency Challenges in Large-Scale Agile Software Development, in: Agile Processes in Software Engineering and Extreme Programming, Springer, 2014, pp. 46–61. doi:10.1007/978-3-319-06862-6_4.

[84] S. Ambler, Managing requirements dependencies between agile teams (2014).
URL https://disciplinedagiledelivery.com/managing-requirements-dependencies-between-agile-teams/

[85] M. M. Jha, R. M. F. Vilardell, N. Jai, Scaling agile scrum software development: providing agility and quality to platform development by reducing time to market, in: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), IEEE, 2016, pp. 84–88. doi:10.1109/ICGSE.2016.24.

[86] A. Šāblis, D. Šmite, Agile teams in large-scale distributed context: Isolated or connected?, in: Proceedings of the Scientific Workshop Proceedings of XP2016, ACM, 2016, pp. 1–5. doi:10.1145/2962695.2962705.

[87] M. Paasivaara, Adopting SAFe to scale agile in a globally distributed organization, in: 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE), IEEE, 2017, pp. 36–40. doi:10.1109/ICGSE.2017.15.

[88] C. Guyot, SAFe case study: Kantar Retail Virtual Reality.
URL https://www.scaledagileframework.com/kantar-retail-case-study/

[89] Y. Lv, Huawei - LeSS without Scrum (2017).
URL https://less.works/case-studies/huawei.html

[90] E. Moore, J. Spens, Scaling Agile: Finding your Agile Tribe, in: Agile 2008 Conference, IEEE, 2008, pp. 121–124. doi:10.1109/Agile.2008.43.

[91] S. Ambler, Large Agile Teams (2015).
URL http://disciplinedagiledelivery.com/agility-at-scale/large-agile-teams/

[92] S. Ambler, Disciplined agile program management The product owner team (2018).
URL http://disciplinedagiledelivery.com/disciplined-agile-program-management-the-product-owner-team/

[93] Scrum.org, Major asian airlines uses Nexus framework (2017).
URL https://www.scrum.org/resources/major-asian-airline-scales-scrum-nexus

[94] A. Avritzer, F. Bronsard, G. Matos, Improving global development using agile, in: Agility Across Time and Space, Springer, 2010, pp. 133–148. doi:10.1007/978-3-642-12442-6_9.

[95] G. Gunyho, J. G. Plaza, Evolution of Longer-Term Planning in a Large Scale Agile Project F-Secure's Experience, in: Agile Processes in Software Engineering and Extreme Programming, Springer Berlin Heidelberg, 2011, pp. 306–315. doi:10.1007/978-3-642-20677-1_22.

[96] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, A. Chatzigeorgiou, Identifying, categorizing and mitigating threats to validity in software engineering secondary studies, Information and Software Technology 106 (2019) 201–230. doi:10.1016/j.infsof.2018.10.006.

[97] R. Cleveland, Scaled Agile Inc.: Review process.
URL https://www.scaledagile.com/case-study-faqs/

[98] R. Cleveland, Scaled Agile, Inc: Questionnaire for safe adopters.
URL https://www.scaledagile.com/resources/submit-a-case-study/